On a Fast Algorithm for Computing the Laplacian Eigenpairs via Commuting Integral Operators

By

Xiaodong Xue
B.S. Mathematics (Nanjing University, China) 2001

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

APPLIED MATHEMATICS

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA,

DAVIS

Approved:

_____

_____

_____

Committee in Charge

2007

On a Fast Algorithm for Computing the Laplacian Eigenpairs via Commuting Integral Operators

Copyright 2007

by

Xiaodong Xue

Contents

Abstract

N. Saito recently proposed a new method to analyze and represent data recorded on a domain Ω of general shape in $\mathbb{R}^d$ by expanding the data into a set of the eigenfunctions of Laplacian defined over Ω. Instead of directly solving the Laplacian eigenvalue problem on Ω via the Helmholtz equation (which can be quite complicated and costly), he found an integral operator commuting with the Laplacian, which shares the eigenfunctions with the Laplacian.

After discretization, the eigenvalue problem for this integral operator is converted to that of a symmetric kernel matrix. A conventional approach of computing selected eigenvalues of such a matrix is, for instance, an implicitly re-started Lanczos method (IRLM), which costs $O(kN^2)$ floating point operations where k is related to the number of the selected eigenvalues and the distribution of the eigenvalues. This approach is prohibitively expensive for a large kernel matrix that is often generated by a dense sampling of the domain.

The kernel function of this integral operator for a domain in $\mathbb{R}^2$, however, is of the special form $K(\boldsymbol{x}, \boldsymbol{y}) = -\frac{1}{2\pi} \log \|\boldsymbol{x} - \boldsymbol{y}\|$, i.e., the fundamental solution of the Laplacian. This kernel function represents the potential at point $\boldsymbol{x}$ due to a charge located at point $\boldsymbol{y}$. This fact allows us to apply the ideas of Fast Multipole Method (FMM) and Hierarchical Semi-Separable (HSS) representation to exploit the structure of the kernel matrix. In this thesis, we investigate a hierarchical matrix-splitting scheme and the low rank property of the submatrices generated by the splitting scheme. By doing so, we design a hierarchical matrix decomposition method that leads to a fast algorithm of matrix-vector multiplication which costs $O(N)$. By supplying our fast matrix-vector multiplication routine to IRLM, we obtain an eigenvalue solver that costs $O(kN)$.

Acknowledgments

It is a pleasure to thank the many people who made this thesis possible.

I would like to express my deep and sincere gratitude to my supervisor, Professor Naoki Saito. His broad knowledge and his logical way of thinking have been of great value for me. His overly enthusiasm and integral view on research and his mission for providing "only high-quality work and not less", has made a deep impression on me. I owe him lots of gratitude for having me shown this way of research.

I want to thank the other two thesis committee members: Professor Zhaojun Bai and Professor Thomas Strohmer for their help and important suggestions and corrections.

My fellow students in Math Department all gave me the feeling of being at home at work. Arthur Cheng, Peng Li, Yung-Ta Li, Ben-Shan Liao, Shuang Liu, Smith Noel, Shaowu Tian, Ernest Woei, Jiadong Xu, Mike Yan, Wei Yu, Zhihua Zhang.

This research was supported and funded by NSF grant DMS-0410406 and ONR grants N00014-00-1-0469, N00014-06-1-0615, and N00014-07-1-0166. I am also grateful for the Department of Mathematics in the University of California at Davis for providing me an excellent work environment during the past years and Celia Davis, Richard Edmiston, Jessica Potts and Diana Coombe for their cheerful assistance.

Finally, I would like to express my profound gratitude to my beloved parents, mother-in-law, and my wife for their love, support, and absolute confidence in me.

Chapter 1

Introduction

There is an increasing need to analyze high dimensional data sampled on irregular grids (e.g., meteorological data sampled at weather stations) or objects defined on a domain of general shape (e.g., cells in histological images). The related applications include the numerical solution of partial differential equations, kernel methods in machine learning, surface reconstruction, and terrain modeling (see e.g., [32] and the references therein).

Most of the currently available signal and image processing tools were designed and developed for signals and images that are sampled on regular/uniform grids and supported on a rectangular or cubic domain. For example, the conventional Fourier analysis using complex exponentials, sines and/or cosines, have been the crown jewels for analyzing such data. Unfortunately, these conventional tools cannot efficiently handle the data and objects that are irregularly sampled on a general shape of domain.

Recently, Saito proposed a new technique that can analyze spatial frequency information of such data and objects, filter the frequency contents if one wishes, and synthesize the data and objects at one's disposal [25]. This is a direct generalization of the conventional Fourier analysis and synthesis. His new tool explicitly incorporates geometric configuration of the domain or spatial location of the sensors.

Let us consider a bounded domain of general shape $\Omega \subset \mathbb{R}^d$, where typically $d = 2$ or 3. Assume that the boundary $\Gamma = \partial\Omega$ consists of piecewise C^2 surfaces (although one may be able

to weaken this assumption by more subtle arguments). We want to analyze the spatial frequency information *inside* of the object (i.e., the data measured in Ω) without the annoying interference of the Gibbs phenomenon due to the boundary Γ. We also want to represent the object compactly for analysis, interpretation, discrimination, and so on, by expanding it into a basis that generates fast decaying expansion coefficients.

There are at least two approaches to this problem. One is to extend (or extrapolate) a general shape object smoothly to its outside, cut it by a circumscribed rectangle, and use the conventional tools to analyze the extended object on this rectangle [27], [35, Chap. 4]. Although such approach can analyze the spatial frequency contents of the object without being bothered by the boundary and the Gibbs phenomenon, the resulting analysis is still affected by the extended part which is smooth (in fact, harmonic) regardless of the smoothness of the object inside the original domain Ω.

Saito [25], [26] proposed a method to use a genuine orthonormal basis tailored to a domain of general shape. To do so, the eigenfunctions of the Laplacian defined on the domain were used. In order to avoid the difficulties of computing the Laplacian eigenfunctions defined on a domain of general shape, he used an integral operator that commutes with the Laplacian and therefore shares the eigenfunctions with the Laplacian. The computation of the eigenfunctions of this integral operator can be discretized into the eigenvalue computation of a kernel matrix.

In practice, to capture the data defined on a domain of large size or with complicated shape, a considerable amount of the sampling data points are needed. The kernel matrix could be so large that the eigenvalue computation is prohibitive. In this thesis, we design a fast algorithm to compute the eigenpairs for large kernel matrices.

Motivated by the original ideas from fast multipole method (FMM, [24], [14], [13]) and hierarchical semi-separable representation (HSS, [4]), we explore the special structure of the kernel matrix of this integral operator and further investigate the hierarchical matrix-splitting scheme. We design a hierarchical matrix decomposition to rapidly compute the matrix-vector multiplication, which is the key to the fast eigenvalue computation using implicitly re-started Lanczos method [20].

The organization of the rest of this thesis is as follows.

In Chapter 2, we briefly review the Laplacian eigenvalue problem and the current computational methods.

In Chapter 3, we review the new tool developed by Saito to compute the eigenpairs of the Laplacian on a domain of general shape [25], [26]. We will discuss an integral operator commuting with the Laplacian, which shares the eigenfunctions with the Laplacian. The new eigenfunctions must satisfy a non-local boundary condition, which is neither the Dirichlet, nor the Neumann boundary conditions. In Section 3.1, we discuss the properties and computation of these new Laplacian eigenfunctions. In Section 3.2, we present two examples, in which we explicitly compute the forms of our Laplacian eigenpairs for a unit interval (1D Example) and a unit disk (2D Example). In Section 3.3, we show a promising application of our Laplacian eigenfunctions to the image approximation on a domain of general shape.

In Chapter 4, we design a fast eigenvalue computation algorithm tailored for the special integral operator in $\mathbb{R}^2$. The task is to compute the eigenpairs of the kernel matrix constructed by the sampling and discretizing the underlying domain. The kernel matrix is highly structured and can be splitted into submatrices with low rank properties. The contribution of this thesis is to design a hierarchical matrix decomposition method which leads to a fast matrix-vector multiplication.

In Section 4.1, we briefly review the important issues involved with FMM. In Section 4.2, we investigate the matrix representations of FMM. After introducing the indexing method and matrix-splitting scheme, we explore the low rank property of the submatrices in Section 4.2.4. In Section 4.2.5, we discuss an approach to handle these low rank submatrices by reviewing the famous low rank approximation problem. In Section 4.3, we present our fast matrix-vector multiplication based on our hierarchical decomposition of the original kernel matrix. In Section 4.3.1 and Section 4.3.2, some important properties of the low rank submatrices are investigated. In Section 4.3.3, we present the algorithm of matrix-vector multiplication. In Section 4.3.4, we present the algorithm of hierarchical decomposition. In Section 4.4, we describe the eigenvalue computation using our fast matrix-vector multiplication. In Section 4.5, we present some numerical examples of the eigenvalue computation.

In Chapter 5, we summarize this thesis in Section 5.1. Finally in Section 5.2, we discuss some topics for the future study.

Chapter 2

The Laplacian Eigenvalues and Eigenfunctions

2.1 Introduction

The **Laplace operator** or **Laplacian** is perhaps the most important of all partial differential operators. The Laplace operator acting on a function $u(\boldsymbol{x}) = u(x_1, x_2, \ldots, x_d)$ of class C^2 in a domain $\Omega \subset \mathbb{R}^d$ is defined by

$$\Delta u = \nabla \cdot \nabla u = \sum_{k=1}^{d} \frac{\partial^2 u}{\partial {x_k}^2}. \tag{2.1}$$

In physics, it is used in modeling of heat flow and wave propagation, where the Laplacian forms the spatial component of the heat operator $\partial_t - \Delta$ and the wave operator $\partial_t^2 - \Delta$ respectively (see e.g., [29]). Also, the Laplacian is central in electrostatics, Helmholtz equation, Laplace's equation and Poisson's equation (see e.g., [9]). One of the important properties of Laplacian is that it commutes with translations and rotations [9, p.67]. For any physical process whose underlying physics is homogeneous (independent of position) and isotropic (independent of direction), the Laplacian is likely to turn up. There are many interesting and important issues about the Laplacian. In this chapter, we are going to review a few of them, which are related to the contents of the following chapters.

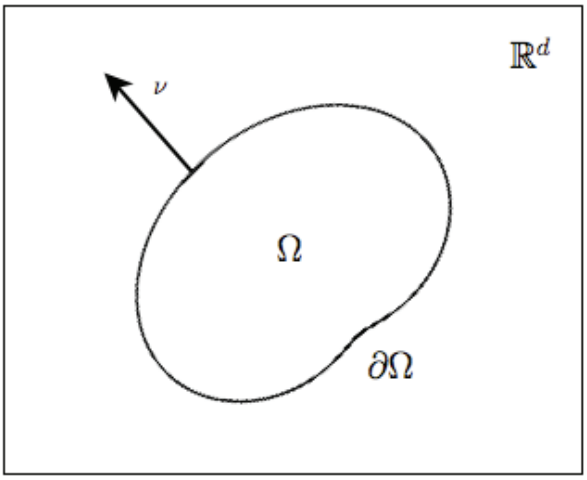


Figure 2.1: $\Omega \subset \mathbb{R}^d$, with ν is the unit normal vector.

2.2 Laplacian Eigenvalue Problem

Consider a domain $\Omega \subset \mathbb{R}^d$ of finite volume. Figure 2.1 displays a simple example of such Ω. **Laplacian Eigenvalue Problem** on Ω is to find the pairs of (u, λ) satisfying the equation:

$$-\Delta u = \lambda u \quad \text{in } \Omega, \tag{2.2}$$

with one of the following boundary conditions (BCs):

$$\begin{array}{lll} (i) & \text{Dirichlet BC:} & u = 0, \quad \text{on } \partial\Omega; \\ (ii) & \text{Neumann BC:} & \dfrac{\partial u}{\partial \nu} = \nu \cdot \nabla u = 0, \quad \text{on } \partial\Omega; \\ (iii) & \text{Robin (Mixed) BC:} & \dfrac{\partial u}{\partial \nu} + au = 0, a \in \mathbb{R} \quad \text{on } \partial\Omega. \end{array} \tag{2.3}$$

where ν is a unit normal vector and $\partial/\partial\nu$ indicates differentiation in the direction of the exterior normal to $\partial\Omega$.

If $u \neq 0$ in Ω satisfies the Laplacian eigenvalue problem with either of the above BCs, then u is called an **eigenfunction** and the corresponding λ is called the **eigenvalue**.

Laplacian eigenvalues and eigenfunctions allow us to perform numerous analysis with the

given domain Ω. We will see that the eigenvalues of (2.2) reflect geometric information about Ω. Also, the eigenfunctions can be used for spectral analysis of data defined (or living) on Ω.

Example 2.2.1 (Wave Equation and Heat Equation)**.** Consider a bounded domain $\Omega \subset \mathbb{R}^d, \quad d \geq 2,\ d \in \mathbb{N}$. Let us first review the wave equation with both boundary conditions and initial conditions

$$u_{tt} = c^2 \Delta u \quad \text{in } \Omega, \tag{2.4}$$

with one of the three boundary conditions in (2.3), and with the initial conditions:

$$u(\boldsymbol{x}, 0) = f(\boldsymbol{x}), \qquad u_t(\boldsymbol{x}, 0) = g(\boldsymbol{x}). \tag{2.5}$$

The heat equation is of the form

$$u_t = k \Delta u \quad \text{in } \Omega, \tag{2.6}$$

with one of the three boundary conditions in (2.3) and the initial conditions in (2.5).

Use the method of separation of variables and set $u(\boldsymbol{x}, t) = T(t)v(\boldsymbol{x})$, which leads to the following equations

$$\begin{aligned} &\text{From wave equation:} \quad \frac{T''}{c^2 T} = \frac{\Delta v}{v} = -\lambda. \\ &\text{From heat equation:} \quad \frac{T'}{kT} = \frac{\Delta v}{v} = -\lambda. \end{aligned} \tag{2.7}$$

Later in this thesis we will show that $\lambda \geq 0$, for at least either (D), (N), or (R) in (2.3) is satisfied. Regardless of whether we consider the heat or the wave equation, we reach a **Laplacian eigenvalue problem**:

$$\begin{gathered} -\Delta v = \lambda v \qquad \text{in } \Omega \\ \text{where } v \text{ satisfies either (D), (N), or (R).} \end{gathered} \tag{2.8}$$

Lots of mathematics are involved to prove that the set of λ satisfying (2.8) is discrete, i.e., $\lambda_1, \lambda_2, \cdots$, and there exist the corresponding eigenfunctions $\varphi_1, \varphi_2, \cdots$ that are mutually orthogonal. We will discuss about them later. But at this point, assuming the existence of the eigenpairs

$\{(\lambda_n, \varphi_n)\}_{n=1}^{\infty}$, we can write the solutions for (2.4) and (2.6) as

$$\begin{aligned} &\text{wave equation:} \quad u(\boldsymbol{x},t) = \textstyle\sum_{n=1}^{\infty} \left[A_n \cos(\sqrt{\lambda_n}ct) + B_n \sin(\sqrt{\lambda_n}ct)\right]\varphi_n(\boldsymbol{x}) \\ &\text{heat equation:} \quad u(\boldsymbol{x},t) = \textstyle\sum_{n=1}^{\infty} A_n \mathrm{e}^{-\lambda_n k t}\varphi_n(\boldsymbol{x}) \end{aligned} \tag{2.9}$$

where A_n and B_n are appropriate constants.

Before further discussion, let us review the famous **Green's Identities.** For $u, v \in C^2(\overline{\Omega})$, we have Green's identities:

$$\int_\Omega v\Delta u \, \mathrm{d}\boldsymbol{x} = -\int_\Omega \nabla v \cdot \nabla u \, \mathrm{d}\boldsymbol{x} + \int_{\partial\Omega} v\frac{\partial u}{\partial \nu} \, \mathrm{d}S \qquad (G1)$$

$$\int_\Omega v\Delta u \, \mathrm{d}\boldsymbol{x} = \int_\Omega u\Delta v \, \mathrm{d}\boldsymbol{x} + \int_{\partial\Omega} \left(v\frac{\partial u}{\partial \nu} - u\frac{\partial v}{\partial \nu}\right) \, \mathrm{d}S, \qquad (G2)$$

where $\mathrm{d}\boldsymbol{x} = \mathrm{d}x_1 \, \mathrm{d}x_2 \ldots \mathrm{d}x_d$ and $\mathrm{d}S$ is a surface measure on $\partial\Omega$. Now, let us briefly review some important properties of the Laplacian eigenvalues and eigenfunctions.

Theorem 2.2.2 ([29, Sec. 10.1])**.** *As in* (2.8)*, let λ_k be the Dirichlet-Laplacian eigenvalues, let ν_k be the Neumann-Laplacian eigenvalues, and let ρ_k be the Robin-Laplacian eigenvalues, where $k \in \mathbb{N}$. Then*

$$\lambda_k > 0, \quad \nu_k \geq 0, \quad \textit{and} \;\; \rho_k \geq 0, \; \textit{if } a \geq 0.$$

Proof. Let u and v are corresponding eigenfunctions. Use Green's first identity (G1):

$$\int_\Omega u\Delta v \, \mathrm{d}\boldsymbol{x} + \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\boldsymbol{x} = \int_{\partial\Omega} u\frac{\partial v}{\partial \nu} \, \mathrm{d}S,$$

Set $v = u$,

$$\int_\Omega u\Delta u \, \mathrm{d}\boldsymbol{x} + \int_\Omega |\nabla u|^2 \, \mathrm{d}\boldsymbol{x} = \int_{\partial\Omega} u\frac{\partial u}{\partial \nu} \, \mathrm{d}S,$$

For the boundary condition (D),

$$\int_\Omega u(-\lambda u) \, \mathrm{d}\boldsymbol{x} + \int_\Omega |\nabla u|^2 \, \mathrm{d}\boldsymbol{x} = 0 \;\; \Rightarrow \;\; \lambda = \frac{\int_\Omega |\nabla u|^2 \, \mathrm{d}\boldsymbol{x}}{\int_\Omega u^2 \, \mathrm{d}\boldsymbol{x}} \geq 0.$$

But $|\nabla u|^2 \neq 0$. Since if so, $u = \text{const}$, then $u \equiv 0$, which conflicts with the fact that u is eigenfunction. Therefore, $\lambda > 0$.

For the boundary condition (N),

$$\nu = \frac{\int_\Omega |\nabla u|^2 \,\mathrm{d}\boldsymbol{x}}{\int_\Omega u^2 \,\mathrm{d}\boldsymbol{x}} \geq 0.$$

Here $|\nabla u|^2 = 0$ is acceptable, i.e., $u \equiv \text{const} \neq 0$. Hence $\nu \geq 0$, where $\nu = 0$ corresponds to the eigenfunction $\phi_0(x) \equiv \text{const} \neq 0$.

For the boundary condition (R), we have

$$-\rho \int_\Omega |u|^2 \,\mathrm{d}\boldsymbol{x} + \int_\Omega |\nabla u|^2 \,\mathrm{d}\boldsymbol{x} = \int_{\partial\Omega} u(-au) \,\mathrm{d}S$$

$$\Rightarrow \rho = \frac{a\int_{\partial\Omega} |u|^2 \,\mathrm{d}S + \int_\Omega |\nabla u|^2 \,\mathrm{d}\boldsymbol{x}}{\int_\Omega |u|^2 \,\mathrm{d}\boldsymbol{x}} \geq 0, \qquad \text{if } a \geq 0.$$

□

To consider the eigenfunctions, we are going to use the fact that for any $u, v \in C^2(\overline{\Omega})$, satisfying either one of the boundary conditions, i.e., (D), (N), or (R), we have

$$\langle u, \Delta v\rangle = \langle \Delta u, v\rangle \,,$$

where the inner product $\langle f, g\rangle \triangleq \int_\Omega f(\boldsymbol{x})\overline{g(\boldsymbol{x})} \,\mathrm{d}\boldsymbol{x}$, where $\Omega \in \mathbb{R}^d$. Then, we have the orthogonality of eigenfunctions.

Theorem 2.2.3 ([29, Sec. 10.1])**.** *Suppose both u, v are real eigenfunctions satisfying*

$$-\Delta u = \lambda_1 u, \qquad -\Delta v = \lambda_2 v$$

and satisfying either (D), (N), or (R). Then λ_1, λ_2 are reals, and if $\lambda_1 \neq \lambda_2$, then $\langle u, v\rangle = 0$.

Proof.

$$\begin{aligned}\lambda_1\langle u,u\rangle &= \langle \lambda_1 u,u\rangle \\ &= \langle -\Delta u,u\rangle \\ &= \langle u,-\Delta u\rangle \\ &= \langle u,\lambda_1 u\rangle \\ &= \overline{\lambda_1}\langle u,u\rangle,\end{aligned}$$

which implies $(\lambda_1-\overline{\lambda_1})\|u\|_2^2=0$. Since $\|u\|_2\neq 0$, $\lambda_1=\overline{\lambda_1}\Leftrightarrow\lambda_1\in\mathbb{R}$. Similarly,

$$\begin{aligned}\lambda_1\langle u,v\rangle-\lambda_2\langle u,v\rangle &= \langle \lambda_1 u,v\rangle-\langle u,\lambda_2 v\rangle \\ &= \langle -\Delta u,v\rangle-\langle u,-\Delta v\rangle \\ &= \langle u,-\Delta v\rangle-\langle u,-\Delta v\rangle \\ &= 0\end{aligned}$$

which implies $(\lambda_1-\lambda_2)\langle u,v\rangle=0$. Since $\lambda_1\neq\lambda_2$, $\langle u,v\rangle=0$. □

More over, the Laplacian eigenfunctions with boundary conditions (D) or (N) are complete in L^2 sense.

Theorem 2.2.4 ([29, Chapter 11])**.** *Both the Dirichlet-Laplacian (DL) and the Neumann-Laplacian (NL) eigenfunctions are* **complete** *in the L^2 sense, i.e., $\forall\ f\in L^2(\Omega)$,*

$$\begin{aligned}\left\|f-\textstyle\sum_{n=1}^{N}c_n\varphi_n\right\|^2\to 0 \text{ as } N\to\infty, \quad \text{where } c_n=\frac{\langle f,\varphi_n\rangle}{\langle\varphi_n,\varphi_n\rangle}.\\ \left\|f-\textstyle\sum_{n=1}^{N}d_n\psi_n\right\|^2\to 0 \text{ as } N\to\infty, \quad \text{where } d_n=\frac{\langle f,\psi_n\rangle}{\langle\psi_n,\psi_n\rangle}.\end{aligned} \tag{2.10}$$

or $\langle f,\varphi_n\rangle=0,\ \forall n\in\mathbb{N}\ \Leftrightarrow\ f\equiv 0$, a.e., and $\langle f,\psi_n\rangle=0,\ \forall n\in\mathbb{N}\ \Leftrightarrow\ f\equiv 0$, a.e..

For the proof, the related arguments and discussions can be found in [29, Sec. 11.3, 11.5], [34, Chap. 11] and [8, Sec. 3.3]. For more advanced treatments, see [7, Sec. 6.5].

Remark 2.2.5. This theorem is important since $\{\varphi_n\}$, $\{\psi_n\}$ are not useful if they are not complete. In other words, if $\exists f \in L^2(\Omega)$, such that $\|f - \sum_{n=1}^{\infty} c_n\varphi_n\| > 0$, then $\{\varphi_n\}$ spans only a subspace of $L^2(\Omega)$.

2.3 Computational Methods of the Laplacian Eigenpairs

2.3.1 Finite Difference Scheme [17].

Its main idea is to discretize the equation $-\Delta u = \lambda u$ by the finite difference approximation

$$-\frac{1}{h^2}\left[\, u_{i+1,j} + u_{i,j+1} + u_{i-1,j} + u_{i,j-1} - 4u_{ij} \,\right] = \lambda u_{ij}.$$

Here the domain $\Omega \subset \mathbb{R}^2$ is cut into squares of side h, u_{ij} is the value of the eigenfunction corresponding to λ at the lattice point (ih, jh). The convergence rate of this method is $O(h^2)$. Moreover, it is slow and inaccurate if Ω is of a complicated shape with a large number of grid points.

2.3.2 Finite Element Method [23].

Many people use this in practice. It is better for general Ω than the finite difference scheme. But it is still very cumbersome. Here we briefly describe the idea of the finite element method.

From Equation $-\Delta u = \lambda u$, it follows that for any function $\varphi \in C^2(\Omega)$ with $\varphi = 0$ on the boundary of Ω,

$$-\iint_{\Omega} \varphi\, \Delta u \,\mathrm{d}\boldsymbol{x} = \lambda \iint_{\Omega} \varphi\, u \,\mathrm{d}\boldsymbol{x}.$$

By Green's first identity (G1), this implies that

$$\iint_{\Omega} \nabla\varphi \cdot \nabla u \,\mathrm{d}\boldsymbol{x} = \lambda \iint_{\Omega} \varphi\, u \,\mathrm{d}\boldsymbol{x}. \tag{2.11}$$

We choose n linearly independent functions $F_1, ..., F_n$ which are the linear, quadratic or cubic polynomials on every tetrahedral element of Ω and satisfy $F_k = 0$ on the boundary of Ω. Take

their linear combination as approximation of the solution u, i.e.,

$$u \approx \sum_{i=1}^{n} U_i F_i, \quad U_i \in \mathbb{R}. \tag{2.12}$$

Again, let $\varphi = F_k$. So

$$\iint_\Omega \nabla\varphi \cdot \nabla u \, \mathrm{d}\boldsymbol{x} = \sum_{i=1}^{n} a_{ki} U_i, \qquad k = 1, ..., n,$$

where

$$a_{ki} = \iint_\Omega \sum_{j=1}^{n} \left(\frac{\partial F_k}{\partial x_j}\right)\left(\frac{\partial F_i}{\partial x_j}\right) \mathrm{d}\boldsymbol{x}.$$

On the other hand,

$$\iint_\Omega \varphi \, u \, \mathrm{d}\boldsymbol{x} = \sum_{i=1}^{n} b_{ki} U_i, \quad k = 1, ..., n,$$

where

$$b_{ki} = \iint_\Omega F_k F_i \, \mathrm{d}\boldsymbol{x}.$$

From this and (2.11), one obtains that

$$\sum_{i=1}^{n} a_{ki} U_i = \lambda \sum_{i=1}^{n} b_{ki} U_i, \quad k = 1, ..., n.$$

Define the matrix $A = (a_{ki})$ and $B = (b_{ki})$. Then (2.11) can be written as

$$A\boldsymbol{U} = \lambda B\boldsymbol{U}, \quad \text{where} \quad \boldsymbol{U} = (U_1, ..., U_n)^T.$$

So λ and $\boldsymbol{U}$ can be calculated. Furthermore, the eigenfunction u is obtained by (2.12).

2.3.3 Integral Operator and Green's Function

As we all know, directly dealing with a partial differential operator $\mathcal{L}$ is more difficult than considering its inverse $\mathcal{L}^{-1}$, which is an integral operator and whose kernel is called **Green's function**. Let us review some basics about the Green's function of the Laplacian Δ (see e.g., [15], [9] and

[29]).

First of all, we need to review the contents about **fundamental solutions** of the Laplacian. We call $K(\boldsymbol{x}, \boldsymbol{y})$ a fundamental solution for the Laplace operator Δ, if $K(\boldsymbol{x}, \boldsymbol{y})$ satisfies

$$\Delta K(\boldsymbol{x}, \boldsymbol{y}) = \delta(\boldsymbol{x} - \boldsymbol{y}), \quad \boldsymbol{y} \text{ is a point in } \Omega.$$

Here $\delta(\cdot)$ is the Dirac delta function. By using the symmetric property of the Laplacian and writing $K(\boldsymbol{x}, \boldsymbol{y})$ as $K(r)$ with $r \triangleq |\boldsymbol{x} - \boldsymbol{y}|$, it is not difficult to derive the form of fundamental solutions of the Laplacian in any dimensional space (see e.g., [15, Chap. 4]):

$$K(r) \triangleq \begin{cases} -\dfrac{1}{2}|x - y| & \text{if } d = 1, \\ -\dfrac{1}{2\pi} \log |\boldsymbol{x} - \boldsymbol{y}| & \text{if } d = 2, \\ \dfrac{|\boldsymbol{x} - \boldsymbol{y}|^{2-d}}{(d-2)\omega_d} & \text{if } d > 2, \end{cases} \tag{2.13}$$

where $\omega_d \triangleq \frac{2\pi^{d/2}}{\Gamma(d/2)}$ is the surface area of the unit sphere in $\mathbb{R}^d$, and $|\cdot|$ is the standard Euclidean norm. In this thesis, the notation $\log$ means the natural logarithm.

For a bounded domain Ω, the Green's function $G(\boldsymbol{x}, \boldsymbol{y})$ for the Laplacian with Dirichlet boundary condition is determined by the following properties:

1. $G(\boldsymbol{x}, \boldsymbol{y}) = K(\boldsymbol{x}, \boldsymbol{y}) + v(\boldsymbol{x}, \boldsymbol{y})$, for $\boldsymbol{x} \in \overline{\Omega}$, $\boldsymbol{y} \in \Omega$, $\boldsymbol{x} \neq \boldsymbol{y}$, with K defined by (2.13), where $v(\boldsymbol{x}, \boldsymbol{y}) \in C^2(\overline{\Omega})$ is harmonic, i.e., $\Delta_{\boldsymbol{x}} v = 0$.

2. $G(\boldsymbol{x}, \boldsymbol{y}) = 0$, for $\boldsymbol{x} \in \partial\Omega$, $\boldsymbol{y} \in \Omega$. In other words, $v = -K$ on $\partial\Omega$.

Assume we are given a particular Green's function G for the underlying domain Ω. Then we can integrate both sides of the Laplacian eigenvalue problem (2.2) as follows

$$\int_\Omega G(\boldsymbol{x}, \boldsymbol{y}) \cdot \Delta u \, \mathrm{d}\boldsymbol{y} = -\lambda \int_\Omega G(\boldsymbol{x}, \boldsymbol{y}) \cdot u(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{y},$$

By applying the Green's second identity and the properties of G on the righthand side of the

above equation, we have

$$\int_\Omega G(\boldsymbol{x},\boldsymbol{y})\cdot \Delta u \ \mathrm{d}\boldsymbol{y} = \int_\Omega \Delta G(\boldsymbol{x},\boldsymbol{y})\cdot u \ \mathrm{d}\boldsymbol{y} = u(\boldsymbol{x}).$$

Therefore, we have

$$\int_\Omega G(\boldsymbol{x},\boldsymbol{y})\cdot u(\boldsymbol{y}) \ \mathrm{d}\boldsymbol{y} = -\frac{1}{\lambda}u(\boldsymbol{x}),$$

which can be solved by discretization of the integral. So, we can solve the Laplacian eigenvalue problem in terms of the matrix by discretizing kernel function $G(\boldsymbol{x},\boldsymbol{y})$ in Ω.

To construct G in general we have to find a harmonic v with $v = -K$ on $\partial\Omega$, which is again a Dirichlet problem. In some cases G can be produced explicitly, e.g., when Ω is a halfspace or a ball. But it could be extremely difficult when Ω has a complicated shape. In Chapter 3, we are going to avoid such difficulties by introducing an integral operator commuting with the Laplacian.

Chapter 3

The Integral Operator commuting with the Laplacian

3.1 Introduction

Recently, Saito [25] proposed a new method to compute Laplacian eigenfunctions using an easily constructible integral operator commuting with the Laplacian. This chapter reviews this method. The contents of this chapter mainly comes from his two paper [25], [26].

3.2 Properties of Laplacian Eigenfunctions and Their Computation

Consider an operator $\mathcal{L} = -\Delta = -\frac{\partial^2}{\partial {x_1}^2} - \cdots \frac{\partial^2}{\partial {x_d}^2}$ in $L^2(\Omega)$ with *appropriate* boundary condition (we will be more specific about it later). The direct analysis of $\mathcal{L}$ is difficult due to the unboundedness (see e.g., [18]). A much better approach is to analyze its inverse $\mathcal{L}^{-1}$, which is referred to as the Green's operator because it is a *compact* and *self-adjoint* operator and consequently we can have a good grip of its spectral properties. In fact, $\mathcal{L}^{-1}$ for a reasonable regular boundary Γ has discrete spectrum (i.e., a countable number of eigenvalues with finite multiplicity) except 0 spectrum [6, Chap. 6,7]. Moreover, thanks to this spectral property, $\mathcal{L}$ has a complete orthonormal basis of $L^2(\Omega)$, and this allows us to do *eigenfunction expansion* in $L^2(\Omega)$ [6, 22].

The key difficulty is to compute such eigenfunctions. Directly solving the Helmholtz equation on a general domain, i.e., finding non-trivial solutions of $\Delta\phi = \lambda\phi$ that satisfy $\mathcal{B}\phi = 0$ (where $\mathcal{B}$ is an operator specifying the boundary condition) is quite tough. Unfortunately, computing the Green's function for a general Ω satisfying the usual boundary condition such as the Dirichlet or the Neumann condition is also very difficult.

3.2.1 Integral Operators Commuting with Laplacian

Our idea to avoid those difficulties is to find an integral operator *commuting* with the Laplacian without imposing the strict boundary condition a priori. Then, from the following well-known theorem (see [10, pp.63-67]), we know that the eigenfunctions of the Laplacian is the same as those of the commuting integral operator that is much easier to deal with.

Theorem 3.2.1. *Let $\mathcal{K}$ and $\mathcal{L}$ be operator acting on $L^2(\Omega)$. Suppose $\mathcal{K}$ and $\mathcal{L}$ commute and one of them has an eigenvalue with finite multiplicity. Then, $\mathcal{K}$ and $\mathcal{L}$ share the same eigenfunction corresponding to that eigenvalue, i.e., there exists a function $\phi \in L^2(\Omega)$ such that $\mathcal{K}\phi = \mu\phi$ and $\mathcal{L}\phi = \lambda\phi$.*

Here is the key step in our development. Let us replace the Green's function $G(\boldsymbol{x}, \boldsymbol{y})$ (the kernel of Green's operator) by the *fundamental solution of the Laplacian* or the *harmonic kernel* defined in (2.13). The price we pay for this replacement is to have rather implicit, non-local boundary condition (which we will discuss shortly) although we do not have to deal with this boundary condition directly. Let $\mathcal{K}$ be the integral operator with its kernel $K(\boldsymbol{x}, \boldsymbol{y})$:

$$\mathcal{K}f(\boldsymbol{x}) \triangleq \int_\Omega K(\boldsymbol{x}, \boldsymbol{y}) f(\boldsymbol{y})\, \mathrm{d}\boldsymbol{y}, \quad f \in L^2(\Omega). \tag{3.1}$$

We now have the following theorem.

Theorem 3.2.2 (Saito [25])**.** *The integral operator $\mathcal{K}$ commutes with the Laplacian $\mathcal{L} = -\Delta$ with the following non-local boundary condition:*

$$\int_\Gamma K(\boldsymbol{x}, \boldsymbol{y}) \frac{\partial \phi}{\partial \nu_{\boldsymbol{y}}}(\boldsymbol{y})\, \mathrm{d}s(\boldsymbol{y}) = -\frac{1}{2}\phi(\boldsymbol{x}) + pv \int_\Gamma \frac{\partial K(\boldsymbol{x}, \boldsymbol{y})}{\partial \nu_{\boldsymbol{y}}} \phi(\boldsymbol{y})\, \mathrm{d}s(\boldsymbol{y}), \tag{3.2}$$

for all $\boldsymbol{x} \in \Gamma$*, where* $\partial/\partial\nu_{\boldsymbol{y}}$ *is the normal derivative operator at the point* $\boldsymbol{y} \in \Gamma$ *and* $\mathrm{d}s(\boldsymbol{y})$ *is the surface measure on* Γ.

Proof. Let $\mathcal{L} = -\Delta$ and $\mathcal{K}$ be defined as (3.1). Then, for $f \in C^2(\Omega) \cup C^1(\overline{\Omega})$, we have

$$\mathcal{L}\mathcal{K}f(\boldsymbol{x}) = -\Delta_{\boldsymbol{x}}\mathcal{K}f(\boldsymbol{x}) = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega,$$

which is referred to as "Poisson's formula" [15, p.99]. On the other hand, using the Green's second identity (see (G2) in Chap. 2), we have

$$\begin{aligned} \mathcal{K}\mathcal{L}f(\boldsymbol{x}) &= -\int_\Omega K(\boldsymbol{x},\boldsymbol{y})\Delta_{\boldsymbol{y}} f(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y} \\ &= f(\boldsymbol{x}) - \int_\Gamma K(\boldsymbol{x},\boldsymbol{y})\frac{\partial f}{\partial \nu_{\boldsymbol{y}}}(\boldsymbol{y})\,\mathrm{d}s(\boldsymbol{y}) + \int_\Gamma \frac{\partial K}{\partial \nu_{\boldsymbol{y}}}(\boldsymbol{x},\boldsymbol{y}) f(\boldsymbol{y})\,\mathrm{d}s(\boldsymbol{y}). \end{aligned}$$

Thus, $\mathcal{K}$ and $\mathcal{L}$ commute if and only if

$$\int_\Gamma K(\boldsymbol{x},\boldsymbol{y})\frac{\partial f}{\partial \nu_{\boldsymbol{y}}}(\boldsymbol{y})\,\mathrm{d}s(\boldsymbol{y}) = \int_\Gamma \frac{\partial K}{\partial \nu_{\boldsymbol{y}}}(\boldsymbol{x},\boldsymbol{y}) f(\boldsymbol{y})\,\mathrm{d}s(\boldsymbol{y}), \quad \boldsymbol{x} \in \Omega. \tag{3.3}$$

We now would like to move $\boldsymbol{x} \in \Omega$ to the boundary Γ in Eq. (3.3). While we do not have any problem in the left-hand side, we must treat the right-hand side carefully following Folland [9, Chap. 2]. Let us consider the right-hand side at $\boldsymbol{x} + t\nu_{\boldsymbol{x}} \in \Omega$ for $\boldsymbol{x} \in \Gamma$ with a sufficient small $t < 0$ instead of $\boldsymbol{x} \in \Omega$. Thus, for $\boldsymbol{x} \in \Gamma$, we have

$$\begin{aligned} \int_\Gamma \frac{\partial K}{\partial \nu_{\boldsymbol{y}}}(\boldsymbol{x} + t\nu_{\boldsymbol{x}},\boldsymbol{y}) f(\boldsymbol{y})\,\mathrm{d}s(\boldsymbol{y}) = f(\boldsymbol{x})\int_\Gamma \frac{\partial K}{\partial \nu_{\boldsymbol{y}}}(\boldsymbol{x} + t\nu_{\boldsymbol{x}},\boldsymbol{y})\,\mathrm{d}s(\boldsymbol{y}) \\ + \int_\Gamma \frac{\partial K}{\partial \nu_{\boldsymbol{y}}}(\boldsymbol{x} + t\nu_{\boldsymbol{x}},\boldsymbol{y})(f(\boldsymbol{y}) - f(\boldsymbol{x}))\,\mathrm{d}s(\boldsymbol{y}). \end{aligned} \tag{3.4}$$

The first term in the right-hand side is $-f(\boldsymbol{x})$ thanks to the following

Lemma 3.2.3 (a variant of Lemma (3.19) in [9])**.**

$$\int_\Gamma \frac{\partial K}{\partial \nu_{\boldsymbol{y}}}(\boldsymbol{x},\boldsymbol{y})\,\mathrm{d}s(y) = \begin{cases} -1 & \text{if } \boldsymbol{x}\in\Omega; \\ -\dfrac{1}{2} & \text{if } \boldsymbol{x}\in\Gamma; \\ 0 & \text{if } \boldsymbol{x}\notin\overline{\Omega}. \end{cases}$$

As for the second integral in Eq. (3.4), because $\psi(\boldsymbol{y}) \triangleq f(\boldsymbol{y}) - f(\boldsymbol{x})$ is continuous and $\psi(\boldsymbol{x}) = 0$ for $\boldsymbol{x}\in\Gamma$, we can use Lemma (3.21) of Folland [9, p. 127] to conclude that as $t\to 0$ the second integral in (3.4) approach to

$$\begin{aligned}\int_\Gamma \frac{\partial K}{\partial \nu_{\boldsymbol{y}}}(\boldsymbol{x},\boldsymbol{y})f(\boldsymbol{y})\,\mathrm{d}s(\boldsymbol{y}) - f(\boldsymbol{x})\int_\Gamma \frac{\partial K}{\partial \nu_{\boldsymbol{y}}}(\boldsymbol{x},\boldsymbol{y})f(\boldsymbol{y})\,\mathrm{d}s(\boldsymbol{y}) \\ = \int_\Gamma \frac{\partial K}{\partial \nu_{\boldsymbol{y}}}(\boldsymbol{x},\boldsymbol{y})f(\boldsymbol{y})\,\mathrm{d}s(\boldsymbol{y}) + \frac{1}{2}f(\boldsymbol{x}), \quad \boldsymbol{x}\in\Gamma, \quad (3.5)\end{aligned}$$

where we used Lemma 3.2.3 again in the last equality. Therefore, we can finally have

$$\int_\Gamma K(\boldsymbol{x},\boldsymbol{y})\frac{\partial f}{\partial \nu_{\boldsymbol{y}}}(\boldsymbol{y})\,\mathrm{d}s(\boldsymbol{y}) = -\frac{1}{2}f(\boldsymbol{x}) + \mathrm{pv}\int_\Gamma \frac{\partial K}{\partial \nu_{\boldsymbol{y}}}(\boldsymbol{x},\boldsymbol{y})f(\boldsymbol{y})\,\mathrm{d}s(\boldsymbol{y}), \quad \boldsymbol{x}\in\Gamma,$$

which is the same as (3.2). This completes the proof. □

Consequently, we also have the following theorem (see e.g., [22, Sec. 4.5]).

Theorem 3.2.4. *The integral operator $\mathcal{K}$ is compact and self-adjoint on $L^2(\Omega)$. Thus, the kernel $K(\boldsymbol{x},\boldsymbol{y})$ has the following eigenfunction expansion (in the sense of mean convergence):*

$$K(\boldsymbol{x},\boldsymbol{y}) \sim \sum_{j=1}^{\infty} \mu_j \phi_j(\boldsymbol{x})\overline{\phi_j(\boldsymbol{y})}, \quad (3.6)$$

and $\{\phi_j\}_{j\in\mathbb{N}}$ forms an orthonormal basis of $L^2(\Omega)$.

We will use the basis $\{\phi_j\}_{j\in\mathbb{N}}$ to expand and represent the data supported on Ω.

3.2.2 Discretization of the Eigenvalue Problem

We want to analyze an object of general shape defined on a digitized image or 3D data set using the Laplacian eigenfunctions. Therefore, we must discretize our eigenvalue problem in order to compute the eigenfunctions and analyze such an object. In this subsection, we describe our discretization strategy and assumptions on the data set.

Let us first assume that the whole data set consists of a collection of data sampled on a regular grid, and that each sampling cell is a box of size $\prod_{i=1}^{d} \Delta x_i$. Let us also assume that an object of our interest consists of a subset of these sampled values and the shape of the object Ω is defined as a collection of the corresponding boxes (or pixels in 2D or voxels in 3D). Hence, let Ω be a collection of boxes whose centers are $\{\boldsymbol{x}_i\}$, $i = 1, \ldots, N$. Under these assumptions, we can approximate the integral eigenvalue problem $\mathcal{K}\phi = \mu\phi$, where $\mathcal{K}$ is defined as (3.1), with a simple quadrature rule with node-weight pairs $(\boldsymbol{x}_j, w_j)$ as follows

$$\sum_{j=1}^{N} w_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)\phi(\boldsymbol{x}_j) = \mu\phi(\boldsymbol{x}_i), \quad i = 1, \ldots, N, \tag{3.7}$$

where $w_j = \prod_{i=1}^{d} \Delta x_i$. Let $K_{i,j} \triangleq w_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $\phi_i \triangleq \phi(\boldsymbol{x}_i)$, and $\boldsymbol{\phi} \triangleq (\phi_1, \ldots, \phi_N)^T \in \mathbb{R}^N$. Then, the equation (3.7) can be written in a matrix-vector format as:

$$K\boldsymbol{\phi} = \mu\boldsymbol{\phi}, \quad \text{where } K = (K_{ij}) \in \mathbb{R}^{N\times N}. \tag{3.8}$$

Under our assumption, the weight w_j does not depend on j, which makes K *symmetric*. Once the matrix version of the eigenvalue problem, $K\boldsymbol{\phi} = \mu\boldsymbol{\phi}$, is obtained, we can compute its eigenvalues and the corresponding eigenvectors. In the numerical experiments, we can use the conventional technique to compute the eigenvalues and eigenvectors of such a matrix, i.e., a slow algorithm of $O(N^3)$, where N is the number of samples in the discretization. This amount of computation cost could be prohibitive when N is huge. Fortunately, we can considerably speed up the eigenvalue/eigenvector computation up to $O(N^2)$ using a hierarchical matrix decomposition based on Fast Multipole Method (FMM, [24], [14], [13]) which will be presented in Chapter

4. In this chapter, all of the examples uses sufficient small size of data sets, where the $O(N^3)$ eigenvalue computation is feasible.

3.3 Examples

In this section, we will show a few analytic examples to contrast our eigenfunctions with the conventional basis functions to deepen our understanding of those eigenfunction-based representation.

3.3.1 1D Example

Consider a unit interval $\Omega = (0, 1)$. Then, the kernel of the commuting operator $\mathcal{K}$ becomes $K(x, y) = -|x-y|/2$, and we can obtain the Laplacian eigenfunctions explicitly in the following corollary of Theorem 3.2.1.

Corollary 3.3.1 (Saito [26])**.** *The eigenfunctions of the integral operator $\mathcal{K}$ for the unit interval $\Omega = (0, 1)$ satisfy the following Laplacian eigenvalue problem:*

$$-\phi'' = \lambda\phi, \quad x \in (0, 1);$$

$$\phi(0) + \phi(1) = -\phi'(0) = \phi'(1), \tag{3.9}$$

which can be solved explicitly as follows.

- $\lambda_0 \approx -5.756915$ *is a solution of the secular equation:*

$$\tanh \frac{\sqrt{-\lambda_0}}{2} = \frac{2}{\sqrt{-\lambda_0}}, \tag{3.10}$$

  and the corresponding eigenfunction is:

$$\phi_0(x) = A_0 \cosh \sqrt{-\lambda_0}(x - \frac{1}{2}), \tag{3.11}$$

where $A_0 = \sqrt{2}\left(1 + \dfrac{\sinh\sqrt{-\lambda_0}}{\sqrt{-\lambda_0}}\right)^{-1/2} \approx 0.7812598$ *is a normalization constant to have* $\|\phi_0\|_{L^2(\Omega)} = 1$.

- $\lambda_{2m-1} = (2m-1)^2\pi^2$, $m = 1, 2, \ldots$, *and the corresponding eigenfunction is:*

$$\phi_{2m-1}(x) = \sqrt{2}\cos(2m-1)\pi x; \tag{3.12}$$

These are canonical cosines with odd modes.

- λ_{2m}, $m = 1, 2, \ldots$, *is a solution of the secular equation:*

$$\tan\frac{\sqrt{\lambda_{2m}}}{2} = -\frac{2}{\sqrt{\lambda_{2m}}}, \tag{3.13}$$

and the corresponding eigenfunction is:

$$\phi_{2m}(x) = A_{2m}\cos\sqrt{\lambda_{2m}}\left(x - \frac{1}{2}\right), \tag{3.14}$$

where $A_{2m} = \sqrt{2}\left(1 + \dfrac{\sin\sqrt{\lambda_{2m}}}{\sqrt{\lambda_{2m}}}\right)^{-1/2}$ *is a normalization constant.*

Proof. Let us directly derive the Laplacian eigenvalue problem from the integral eigenvalue problem, $\mathcal{K}\phi = \mu\phi$. We have

$$\begin{aligned}
\mathcal{K}\phi(x) &= -\frac{1}{2}\int_0^1 |x-y|\phi(y)\,\mathrm{d}y \\
&= -\frac{1}{2}\left(\int_0^x (x-y)\phi(y)\,\mathrm{d}y + \int_x^1 (y-x)\phi(y)\,\mathrm{d}y\right) \\
&= -\frac{1}{2}\left(x\int_0^x \phi(y)\,\mathrm{d}y - \int_0^x y\phi(y)\,\mathrm{d}y + \int_x^1 y\phi(y)\,\mathrm{d}y - x\int_x^1 \phi(y)\,\mathrm{d}y\right) \\
&= \mu\phi(x).
\end{aligned}$$

Differentiating both side with respect to x, we get

$$\int_0^x \phi(y)\,\mathrm{d}y - \int_x^1 \phi(y)\,\mathrm{d}y = -2\mu\phi'(x). \tag{3.15}$$

We can get the Laplacian eigenvalue problem by differentiating both sides of (3.15) with respect to x again:

$$2\phi(x) = -2\mu\phi''(x) \iff -\phi''(x) = \lambda\phi(x),\ \lambda = \frac{1}{\mu}.$$

To derive the boundary condition as (3.9), let us set $x = 0$ and $x = 1$ in (3.15), we have

$$\phi'(0) = -\phi'(1) = \frac{1}{2\mu}\int_0^1 \phi(y)\,\mathrm{d}y. \tag{3.16}$$

Now, evaluating $\mathcal{K}\phi(x) = \mu\phi(x)$ at $x = 0, 1$ yields

$$\phi(0) = -\frac{1}{2\mu}\int_0^1 y\phi(y)\,\mathrm{d}y,$$

$$\phi(1) = -\frac{1}{2\mu}\int_0^1 (1-y)\phi(y)\,\mathrm{d}y = -\frac{1}{2\mu}\left(\int_0^1 \phi(y)\,\mathrm{d}y - \int_0^1 y\phi(y)\,\mathrm{d}y\right).$$

Adding these two equations and use (3.16), we obtain

$$\phi(0) + \phi(1) = -\frac{1}{2\mu}\int_0^1 \phi(y)\,\mathrm{d}y = -\phi'(0) = \phi'(1).$$

Let us now compute the solutions to this Laplacian eigenvalue problem with the boundary condition (3.9). The characteristic equation for $\phi'' + \lambda\phi = 0$ is $r^2 + \lambda = 0$. Therefore, we need to consider the following three cases:

Case I $\lambda < 0$: The eigenfunction is of the form $\phi(x) = A\cosh\sqrt{-\lambda}x + B\sinh\sqrt{-\lambda}x$ where A, B are some constants. Apply the boundary condition (3.9), we have

$$\begin{aligned}\phi(0) + \phi(1) &= A\left(1 + \cosh\sqrt{-\lambda}\right) + B\sinh\sqrt{-\lambda} \\ &= -\phi'(0) = -\sqrt{-\lambda}B \\ &= \phi'(1) = \sqrt{-\lambda}\left(A\sinh\sqrt{-\lambda} + B\cosh\sqrt{-\lambda}\right).\end{aligned}$$

From these equalities, we have the following 2×2 linear system for A and B:

$$\begin{pmatrix} 1+\cosh\sqrt{-\lambda} & \sqrt{-\lambda}+\sinh\sqrt{-\lambda} \\ \sinh\sqrt{-\lambda} & 1+\cosh\sqrt{-\lambda} \end{pmatrix} \cdot \begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Thus, in order to have nontrivial eigenfunctions, the determination of this equation must be zero. This leads to

$$\begin{aligned} 0 &= \left(1+\cosh\sqrt{-\lambda}\right)^2 - \sinh\sqrt{-\lambda}\left(\sqrt{-\lambda}+\sinh\sqrt{-\lambda}\right) \\ &= 2 + 2\cosh\sqrt{-\lambda} - \sqrt{-\lambda}\sinh\sqrt{-\lambda} \\ &= 4\cosh^2\frac{\sqrt{-\lambda}}{2} - 2\sqrt{-\lambda}\sinh\frac{\sqrt{-\lambda}}{2}\cosh\frac{\sqrt{-\lambda}}{2} \\ &= 2\cosh^2\frac{\sqrt{-\lambda}}{2}\left(2-\sqrt{-\lambda}\tanh\frac{\sqrt{-\lambda}}{2}\right). \end{aligned}$$

The last equality is justified because $\cosh\left(\sqrt{-\lambda}/2\right) \neq 0$. Therefore the second factor must be zero, i.e., $\tanh\left(\sqrt{-\lambda}/2\right) = 2/\sqrt{-\lambda}$, which is exactly (3.10). Let λ_0 be the solution of this secular equation, which can be found numerically as $\lambda_0 \approx -5.756915$. For this λ_0, the relationship between the constants A and B above must have:

$$A\sinh\sqrt{-\lambda_0} + B(1+\cosh\sqrt{-\lambda_0}) = 0 \Longleftrightarrow B = \frac{\sinh\sqrt{-\lambda_0}}{1+\cosh\sqrt{-\lambda_0}}A.$$

Thus, we have

$$\begin{aligned} \phi_0(x) &= A\left(\cosh\sqrt{-\lambda_0}x - \frac{\sinh\sqrt{-\lambda_0}}{1+\cosh\sqrt{-\lambda_0}}\sinh\sqrt{-\lambda_0}x\right) \\ &= A'\left(\cosh\sqrt{-\lambda_0}x + \cosh\sqrt{-\lambda_0}\cosh\sqrt{-\lambda_0}x - \sinh\sqrt{-\lambda_0}\sinh\sqrt{-\lambda_0}x\right) \\ &= A'\left(\cosh\sqrt{-\lambda_0}x + \cosh\sqrt{-\lambda_0}(1-x)\right) \\ &= A_0\cosh\sqrt{-\lambda_0}\left(x-\frac{1}{2}\right), \end{aligned}$$

which is exactly (3.11). The constant A_0 is the normalization constant so that $\|\phi_0\|_{L^2(\Omega)} = 1$.

Thus,

$$\begin{aligned} A_0 &= \left(\int_0^1 \left(\cosh \sqrt{-\lambda_0}(x - \frac{1}{2}) \right)^2 \, \mathrm{d}x \right)^{-1/2} \\ &= \left(\frac{1}{2} + \frac{1}{2} \int_0^1 \cosh 2\sqrt{-\lambda_0}(x - \frac{1}{2}) \, \mathrm{d}x \right)^{-1/2} \\ &= \sqrt{2} \left(1 + \frac{\sinh \sqrt{-\lambda_0}}{\sqrt{-\lambda_0}} \right)^{-1/2} \\ &\approx 0.7812598. \end{aligned}$$

Case II $\lambda = 0$: we have $\phi''(x) = 0$. Thus $\phi(x) = Ax + B$. But the boundary condition (3.9) leads to $2A + B = -A = A$, i.e., $A = B = 0$. Therefore, $\lambda = 0$ is not an eigenvalue for this problem.

Case III $\lambda > 0$: The eigenfunction is of the form $\phi(x) = A \cos \sqrt{\lambda} x + B \sin \sqrt{\lambda} x$. Similarly to Cast I, using the boundary condition (3.9), we have

$$\begin{aligned} \phi(0) + \phi(1) &= A(1 + \cos \sqrt{\lambda}) + B \sin \sqrt{\lambda} \\ &= -\phi'(0) = -\sqrt{\lambda} B \\ &= \phi'(1) = \sqrt{\lambda} \left(-A \sin \sqrt{\lambda} + B \cos \sqrt{\lambda} \right) \end{aligned}$$

From these equalities, we have the following 2×2 linear system for A and B:

$$\begin{pmatrix} 1 + \cos \sqrt{\lambda} & \sqrt{\lambda} + \sin \sqrt{\lambda} \\ -\sin \sqrt{\lambda} & 1 + \cos \sqrt{\lambda} \end{pmatrix} \cdot \begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Again, the vanishing determinant of this equation leads to

$$\begin{aligned} 0 &= \left(1 + \cos \sqrt{\lambda} \right) + \sin \sqrt{\lambda} \left(\sqrt{\lambda} + \sin \sqrt{\lambda} \right) \\ &= 2 \cos \frac{\sqrt{\lambda}}{2} \left(2 \cos \frac{\sqrt{\lambda}}{2} + \sqrt{\lambda} \sin \frac{\sqrt{\lambda}}{2} \right) \end{aligned}$$

Thus, we have $\cos(\sqrt{\lambda}/2) = 0$ or $2\cos(\sqrt{\lambda}/2) + \sqrt{\lambda}\sin(\sqrt{\lambda}/2) = 0$. From the former, we can easily get the eigenvalues $\lambda = \lambda_{2m-1} = (2m-1)^2\pi^2$ and the corresponding eigenfunctions $\phi_{2m-1}(x) = \sqrt{2}\cos(2m-1)\pi x$, $m = 1, 2, \ldots$, which is (3.12). From the latter, we get the secular equation, $\tan(\sqrt{\lambda}/2) = -2/\sqrt{\lambda}$, which is exactly (3.13). Considering the graph of periodic asymptotes of $\tan(\sqrt{\lambda}/2)$ at $\lambda = (2m-1)^2\pi^2$, we observe that the eigenvalues satisfying (3.13) and λ_{2m-1} are interlacing. Thus, we naturally denote the eigenvalues satisfying (3.13) by λ_{2m}, $m = 1, 2, \ldots$, which must be computed numerically. The corresponding eigenfunctions are

$$\begin{aligned}
\phi_{2m}(x) &= A\left(\cos\sqrt{\lambda_{2m}}x + \frac{\sin\sqrt{\lambda_{2m}}}{1+\cos\sqrt{\lambda_{2m}}}\sin\sqrt{\lambda_{2m}}x\right) \\
&= A'\left(\cos\sqrt{\lambda_{2m}}x + \cos\sqrt{\lambda_{2m}}\cos\sqrt{\lambda_{2m}}x + \sin\sqrt{\lambda_{2m}}\sin\sqrt{\lambda_{2m}}x\right) \\
&= A'\left(\cos\sqrt{\lambda_{2m}}x + \cos\sqrt{\lambda_{2m}}(1-x)\right) \\
&= A_{2m}\cos\sqrt{\lambda_{2m}}\left(x - \frac{1}{2}\right),
\end{aligned}$$

which is exactly (3.14). The constant A_{2m} is the normalization constant so that $\|\phi_{2m}\|_{L^2(\Omega)} = 1$. Thus,

$$A_{2m} = \left(\int_0^1 \left(\cos\sqrt{\lambda_{2m}}(x-\frac{1}{2})\right)^2 \,\mathrm{d}x\right)^{-1/2} = \sqrt{2}\left(1 + \frac{\sin\sqrt{\lambda_{2m}}}{\sqrt{\lambda_{2m}}}\right)^{-1/2},$$

which completes the proof. □

Fig. 3.1 shows these Laplacian eigenfunctions of the lowest five frequencies.

3.3.2 2D Example

Let us now consider the unit disk Ω in $\mathbb{R}^2$, where the kernel of our integral operator $\mathcal{K}$ becomes $K(\boldsymbol{x}, \boldsymbol{y}) = -\frac{1}{2\pi}\log\|\boldsymbol{x} - \boldsymbol{y}\|_2$ for $\boldsymbol{x}, \boldsymbol{y} \in \overline{\Omega}$. Tailoring Theorem 3.2.2 for the unit disk, we have the following corollary:

Corollary 3.3.2 (Saito [26])**.** *The eigenfunctions of the integral operator $\mathcal{K}$ for the unit disk in $\mathbb{R}^2$*

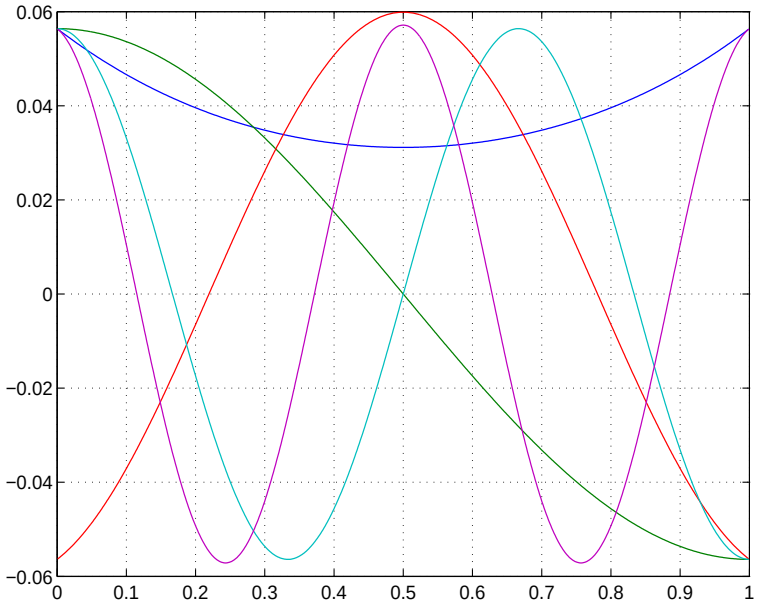


Figure 3.1: First five eigenfunctions of the Laplacian on the interval $[0, 1]$ with the non-local boundary condition (3.9).

satisfies the following Laplacian eigenvalue problem:

$$-\Delta\phi = \lambda\phi, \quad \textit{in } \Omega;$$

$$\left.\frac{\partial\phi}{\partial\nu}\right|_{\Gamma} = \left.\frac{\partial\phi}{\partial r}\right|_{\Gamma} = -\left.\frac{\partial\mathcal{H}\phi}{\partial\theta}\right|_{\Gamma}, \tag{3.17}$$

where $\mathcal{H}$ is the Hilbert transform for the circle, i.e.,

$$\mathcal{H}f(\theta) \triangleq \frac{1}{2} pv \int_{-\pi}^{\pi} f(\eta) \cot\left(\frac{\theta-\eta}{2}\right) \, \mathrm{d}\eta, \quad \theta \in [-\pi, \pi].$$

Moreover, these eigenfunctions are of the form:

$$\phi_{m,n}(r,\theta) = \begin{cases} J_m(\beta_{m-1,n} r) \begin{pmatrix} \cos \\ \sin \end{pmatrix} (m\theta) & \textit{if } m = 1, 2, \ldots, \ n = 1, 2, \ldots, \\ J_0(\beta_{0,n} r) & \textit{if } m = 0, \ n = 1, 2, \ldots, \end{cases}$$

where $\beta_{k,\ell}$ is the ℓth zero of the Bessel function of order k, i.e., $J_k(\beta_{k,\ell}) = 0$. The corresponding

eigenvalues are

$$\lambda_{m,n} = \begin{cases} \beta_{m-1,n}^2 & \text{if } m = 1, \ldots, \ n = 1, 2, \ldots, \\ \beta_{0,n}^2 & \text{if } m = 0, \ n = 1, 2, \ldots. \end{cases} \tag{3.18}$$

Proof. In $\mathbb{R}^2$, we have $K(\boldsymbol{x}, \boldsymbol{y}) = -\frac{1}{2\pi} \log \|\boldsymbol{x} - \boldsymbol{y}\|_2$. Thus,

$$\nabla_{\boldsymbol{y}} K(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2\pi} \frac{\boldsymbol{x} - \boldsymbol{y}}{\|\boldsymbol{x} - \boldsymbol{y}\|_2^2}.$$

Now, the normal derivative of $K(\boldsymbol{x}, \boldsymbol{y})$ at $\boldsymbol{y} \in \Gamma$ can be computed as

$$\frac{\partial K}{\partial \nu_{\boldsymbol{y}}}(\boldsymbol{x}, \boldsymbol{y}) = \nu_{\boldsymbol{y}} \cdot \nabla_{\boldsymbol{y}} K(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2\pi} \frac{(\boldsymbol{x} - \boldsymbol{y}) \cdot \nu_{\boldsymbol{y}}}{\|\boldsymbol{x} - \boldsymbol{y}\|_2^2}.$$

When Ω is the unit disk in $\mathbb{R}^2$. Let $\boldsymbol{x} = e^{i\theta}$, $\boldsymbol{y} = e^{i\eta}$ be any two boundary points. Then, it is easy to show that

$$|\boldsymbol{x} - \boldsymbol{y}|^2 = 4 \sin^2 \left(\frac{\theta - \eta}{2} \right), \quad (\boldsymbol{x} - \boldsymbol{y}) \cdot \nu_{\boldsymbol{y}} = (\boldsymbol{x} - \boldsymbol{y}) \cdot \boldsymbol{y} = -2 \sin^2 \left(\frac{\theta - \eta}{2} \right),$$

which lead to

$$\frac{\partial K}{\partial \nu_{\boldsymbol{y}}}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2\pi} \frac{(\boldsymbol{x} - \boldsymbol{y}) \cdot \nu_{\boldsymbol{y}}}{\|\boldsymbol{x} - \boldsymbol{y}\|_2^2} = -\frac{1}{4\pi}. \tag{3.19}$$

Recall the non-local boundary condition (3.3) as follows:

$$\int_\Gamma K(\boldsymbol{x}, \boldsymbol{y}) \frac{\partial \phi}{\partial \nu_{\boldsymbol{y}}}(\boldsymbol{y}) \, \mathrm{d}s(\boldsymbol{y}) = -\frac{1}{2}\phi(\boldsymbol{x}) + \mathrm{pv} \int_\Gamma \frac{\partial K(\boldsymbol{x}, \boldsymbol{y})}{\partial \nu_{\boldsymbol{y}}} \phi(\boldsymbol{y}) \, \mathrm{d}s(\boldsymbol{y}) \tag{3.20}$$

Let $\phi \in C^2(\Omega) \cap C^1(\overline{\Omega})$ be represented in the polar coordinates as $\phi(r, \theta)$. Plugging (3.19) and the 2D kernel above into (3.20) and multiplying 2 on both sides, we get

$$-\frac{1}{\pi} \int_{-\pi}^{\pi} \log \left| 2 \sin \left(\frac{\theta - \eta}{2} \right) \right| \frac{\partial \phi}{\partial r}(1, \eta) \ \mathrm{d}\eta = -\phi(1, \theta) - \frac{1}{2\pi} \int_{-\pi}^{\pi} \phi(1, \eta) \ \mathrm{d}\eta. \tag{3.21}$$

Note that the second term in the righthand side is a constant. Differentiating both sides in θ

leads to

$$\frac{\partial\phi}{\partial\theta}(1,\theta)=\frac{1}{2\pi}\int_{-\pi}^{\pi}\frac{\partial\phi}{\partial r}(1,\eta)\cot\left(\frac{\theta-\eta}{2}\right)\,\mathrm{d}\eta=\mathcal{H}\frac{\partial\phi}{\partial r}(1,\theta).$$

Note that $\mathcal{H}^2=-Id$ where Id is the identity operator. Thus, we have

$$\frac{\partial\phi}{\partial r}(1,\theta)=-\mathcal{H}\frac{\partial\phi}{\partial\theta}(1,\theta)=-\frac{\partial\mathcal{H}\phi}{\partial\theta}(1,\theta), \tag{3.22}$$

which is exactly (3.17).

Finally, let us compute the eigenfunctions and the corresponding eigenvalues satisfying (3.17). Using the separation of variables, the eigenfunction is of the form:

$$\phi(r,\theta)=J_m(\sqrt{\lambda}r)\begin{pmatrix}\cos\\ \sin\end{pmatrix}(m\theta),\quad m=0,1,\dots. \tag{3.23}$$

If the boundary condition were the standard Dirichlet condition, then the eigenvalue λ could be obtained by the condition $J_m(\sqrt{\lambda})=0$. However, our boundary condition is quite different from the Dirichlet case: it must satisfy (3.17). Let us consider the following two cases.

Case I $m>0$: By plugging (3.23) into (3.17), we have

$$\begin{aligned}\frac{\partial\phi}{\partial r}(1,\theta)&=\sqrt{\lambda}J_m'\big(\sqrt{\lambda}\big)\begin{pmatrix}\cos\\ \sin\end{pmatrix}(m\theta)=-\frac{\partial\mathcal{H}\phi}{\partial\theta}(1,\theta)\\ &=-J_m\big(\sqrt{\lambda}\big)\frac{\partial}{\partial\theta}\begin{pmatrix}\sin\\ -\cos\end{pmatrix}(m\theta)=-mJ_m\big(\sqrt{\lambda}\big)\begin{pmatrix}\cos\\ \sin\end{pmatrix}(m\theta).\end{aligned}$$

From these, we have

$$\sqrt{\lambda}J_m'\big(\sqrt{\lambda}\big)=-mJ_m\big(\sqrt{\lambda}\big),\quad m>0. \tag{3.24}$$

Now using the standard recursion formulas (see e.g., [1, Formula 9.1.27]):

For $z\in\mathbb{C}$,

(i) $2J_m'(z)=J_{m-1}(z)-J_{m+1}(z)$

(ii) $J_m'(z)=J_{m-1}(z)-\dfrac{m}{z}J_m(z)$

By (ii) with $z = \sqrt{\lambda}$, we get

$$2J'_m(\sqrt{\lambda}) = 2J_{m-1}(\sqrt{\lambda}) - \frac{2m}{\sqrt{\lambda}} J_m(\sqrt{\lambda}) \tag{3.25}$$

From (3.24), we get

$$2J'_m(\sqrt{\lambda}) = -\frac{2m}{\sqrt{\lambda}} J_m(\sqrt{\lambda}) \tag{3.26}$$

Therefore, combine (3.25) and (3.26), we find

$$J_{m-1}(\sqrt{\lambda}) = 0, \quad m = 1, 2, \ldots,$$

which tells us that

$$\lambda_{m,n} = \beta^2_{m-1,n}, \quad \text{where } \beta_{m-1,n} \text{ are zeros of } J_{m-1}(x),\ m, n = 1, 2, \ldots.$$

Case II $m = 0$: The eigenfunction is of the form $\phi(r, \theta) = J_0\big(\sqrt{\lambda} r\big)$, i.e., a radial function. In this case, we need to go back to (3.21) because (3.17) simply says $0 = 0$ and does not give rise to a constraint. Now, substituting $\phi(r, \theta) = J_0\big(\sqrt{\lambda} r\big)$ into (3.21), we have

$$-\frac{1}{\pi}\int_{-\pi}^{\pi} \left|2 \sin \frac{\theta - \eta}{2}\right| \sqrt{\lambda} J_0\big(\sqrt{\lambda}\big)\ \mathrm{d}\eta = -J_0\big(\sqrt{\lambda}\big) - \frac{1}{2\pi}\int_{-\pi}^{\pi} J_0\big(\sqrt{\lambda}\big)\ \mathrm{d}\eta = -2J_0\big(\sqrt{\lambda}\big). \tag{3.27}$$

Since the integral

$$\begin{aligned}
\int_{-\pi}^{\pi} \log \left|2 \sin \frac{\theta - \eta}{2}\right|\ \mathrm{d}\eta &= 2\pi \log 2 + \int_{-\pi}^{\pi} \log \left|\sin \frac{\theta - \eta}{2}\right|\ \mathrm{d}\eta \\
&= 2\pi \log 2 + 2\int_{\frac{\theta-\pi}{2}}^{\frac{\theta+\pi}{2}} \log |\sin u|\ \mathrm{d}u \\
&= 2\pi \log 2 + \int_{-\pi/2}^{\pi/2} \log |\sin u|\ \mathrm{d}u,
\end{aligned}$$

where the last equality comes from the periodicity of the integrand.

We also know that $\int_0^{\pi/2} \log \sin x\ \mathrm{d}x = -\frac{\pi}{2} \log 2$ via [12, Forumla 4.224.3], which immedi-

ately gives us

$$\int_{-\pi/2}^{\pi/2} \log|\sin u| \ \mathrm{d}u = -\pi \log 2.$$

Therefore,

$$\int_{-\pi}^{\pi} \log\left|2\sin\frac{\theta-\eta}{2}\right| \ \mathrm{d}\eta = 0 \quad \text{for any } \theta \in [-\pi, \pi].$$

So, (3.27) requires $J_0\big(\sqrt{\lambda}\big) = 0$. Thus, we have $\lambda_{0,n} = \beta_{0,n}^2$, $n = 1, 2, \ldots$. This completes the proof.

□

Remark 3.3.3. This corollary suggests that out of the Laplacian eigenfunctions computed with our formulation, those corresponding to J_0, i.e., the radially symmetric eigenfunctions satisfy the Dirichlet boundary condition, but the other eigenfunctions do not. Also, we note that there are three eigenfunctions corresponding to each $\beta_{0,n}$, namely, $J_0(\beta_{0,n}r)$, $J_0(\beta_{0,n}r)\cos\theta$ and $J_0(\beta_{0,n})\sin\theta$.

It is also interesting to compare (3.18) with the eigenvalues $\lambda_{m,n}^D$ of the Dirichlet Laplacian and $\lambda_{m,n}^N$ of the Neumann-Laplacian:

$$\lambda_{m,n}^D = \beta_{m,n}^2, \quad \lambda_{m,n}^N = \alpha_{m,n}^2, \quad \text{for } m = 0, 1, \ldots, \ n = 1, 2, \ldots,$$

where $J_m'(\alpha_{m,n}) = 0$.

Fig. 3.2 shows our Laplacian eigenfunctions of the lowest 25 frequencies (or the smallest 25 eigenvalues λ's). Fig. 3.3 shows the Dirichlet Laplacian eigenfunctions (by separation of variables) of the lowest 25 frequencies using the codes in [30]. These eigenfunctions can be viewed as "modes" of the vibration of the domain if the domain is interpreted as a "drum" although our eigenfunctions do not satisfy the Dirichlet boundary condition.

3.4 Application to Image Approximation

In this section, we discuss the image approximation capability of our Laplacian eigenfunctions given image data on a domain with irregular shape. We will compare the performance with that of

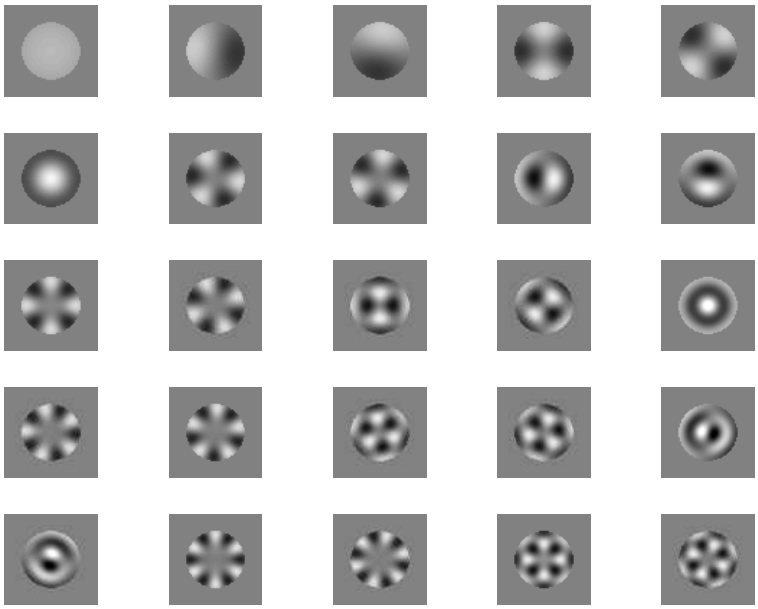


Figure 3.2: First 25 eigenfunctions of the Laplacian on the unit disk with the non-local boundary condition (3.20).

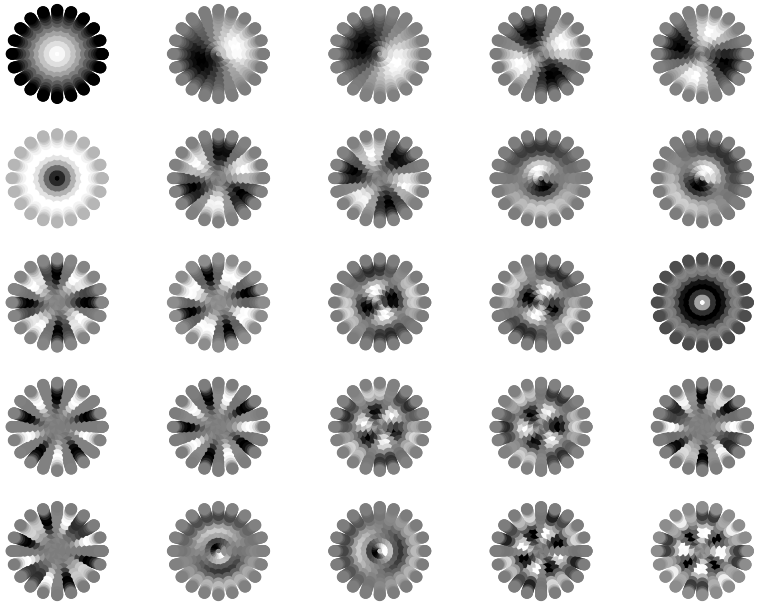
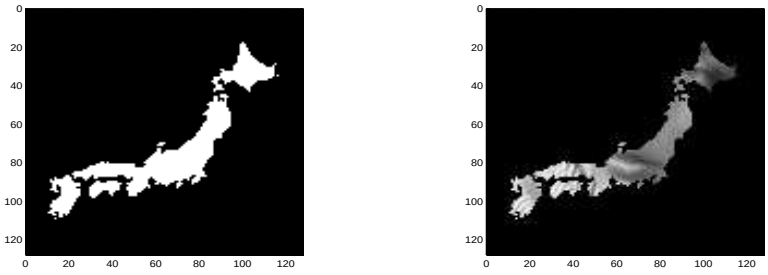


Figure 3.3: First 25 eigenfunctions of the Dirichlet Laplacian on the unit disk.



(a) χ_Ω (b) $\chi_\Omega \cdot$ Barbara

Figure 3.4: The characteristic function of the Japanese Islands (a) and the Barbara image overlaid over the islands.

the standard wavelet-based methods.

We use a coarsely digitized image of the island of Japan, as shown in Fig. 3.4(a), which was obtained by the google image search. We then define the characteristic function $\chi_\Omega(\boldsymbol{x})$ to indicate the shape of the islands. As for the data living on this domain Ω, we multiply the standard Barbara image with χ_Ω, which is shown in Fig. 3.4(b). The number of sample forming the data on the islands is 1625. We compute the Laplacian eigenfunctions defined on Ω from the kernel matrix of 1625×1625. We display the first 25 eigenfunctions in Fig. 3.5.

We then compute the 1625 expansion coefficients relative to this Laplacian eigenbasis, sort them in terms of their magnitudes, and approximate the data using the top 200 coordinates. In other words, we performed 200-term nonlinear approximation using the Laplacian eigenfunctions. The result is shown in Fig. 3.6(a) and the reconstruction error (or the residual) is shown in Fig. 3.6(b). Note that the display dynamic range of these two figures is different. We note that the scarf region of the Barbara image was not captured by these 100 terms of the Laplacian eigenfunctions. To capture the high frequency features, we need more terms.

We also approximate the same image using the top 200 coefficients computed by the standard 2D wavelet basis called "Symmlet8" [5]. Note that this comparison is not really fair in the sense that the input image to the 2D wavelet transform is whole rectangular image shown in Fig. 3.4(b), i.e., not only the islands, but also the outer ocean part. The approximation and its residual error are shown in Fig. 3.7. In this case, most of the top 200 wavelet coordinates are used to capture the

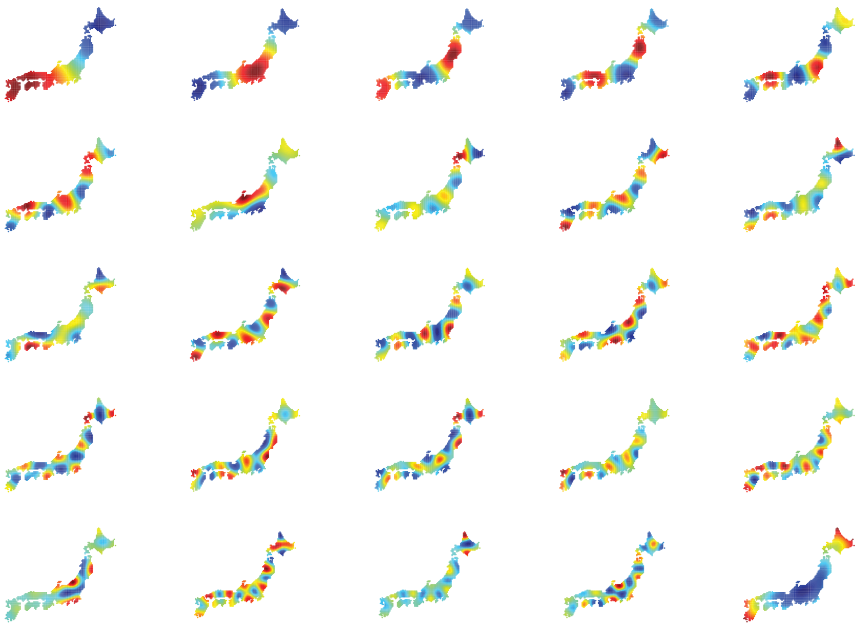
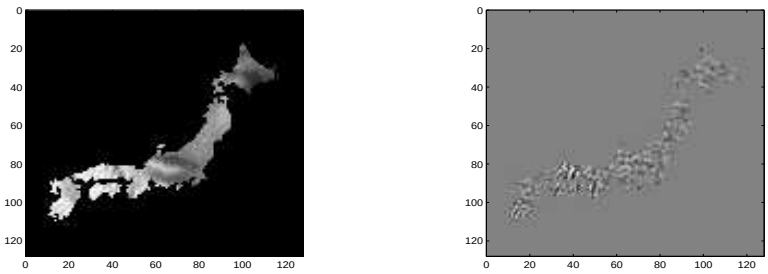
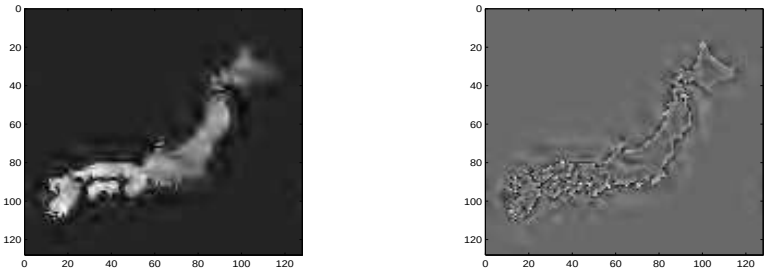


Figure 3.5: The first 25 Laplacian eigenfunctions over the islands of Japan.



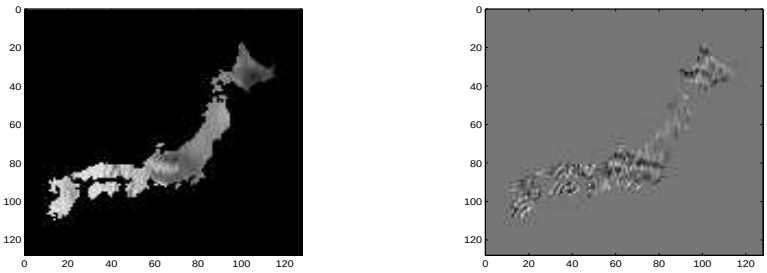
(a) 200-term Approx (b) Error

Figure 3.6: The 200-term approximation and the residual error using the Laplacian eigenfunctions.



(a) 200-term Approx (b) Error

Figure 3.7: The 200-term approximation and the residual error using the standard 2D Wavelets (Symmlet 8).



(a) 200-term Approx (b) Error

Figure 3.8: The 200-term approximation and the residual error using the 1D Wavelets (Symmlet 8).

boundary of the island and cannot afford to capture the internal structure within the domain. Note that the 200 wavelet coefficients are simply not enough to capture the boundary of the domain precisely for this image.

To be fairer, we organize these 1625 data points in a one-dimensional array in a column scanning order in Fig. 3.4(b), and apply the 1D wavelet transform using the Symmlet 8 filter. Then, the 200-term approximation and its residual error are computed. The results are shown in Fig. 3.8. In this case, there is not blurred boundary shape because we explicitly use the geometric information. However, observe the stripe-shape artifacts in the approximation. This is because we deconstructed the 2D spatial coherency of the original data by putting them into the 1D array.

Chapter 4

Fast Multipole Method and Eigenvalue Computation

4.1 Introduction

Fast Multipole Method (FMM) was introduced by V. Rokhlin [24] as an $O(N)$ numerical method for solving an integral equation for 2D Laplace's equation. This method was further developed and made famous by Rokhlin and Greengard as they applied FMM to N-body problems (see e.g., [13, 14].

The algorithm allows the product of a specially structured dense matrix of size $N \times N$ with a vector of length N to be evaluated approximately in $O(N)$ or $O(N \log N)$ operations, when direct multiplication requires $O(N^2)$ operations. For extremely large problems, the gain in efficiency and memory can be very significant, and enables the use of more sophisticated modeling approaches that may have been discarded as computationally infeasible in the past.

Originally this method was developed for the fast summation of the potential fields generated by a large number of particles (point charges), such as those arising in gravitational or electrostatic potential problems. First, we are going to review the theoretical background of FMM in a finite 2D domain, which is the fundamental motivation of our fast eigenpairs computation.

4.1.1 Potential and Multipole Expansion

We are interested in a two-dimensional physical model which consists of a set of N charged particles with potential obtained as the sum of pariwise interactions from Coulomb's law. Suppose that a point charge of unit strength is located at point $(x_0, y_0) = \boldsymbol{x}_0 \in \mathbb{R}^2$. Then, for any $\boldsymbol{x} = (x, y) \in \mathbb{R}^2$ with $\boldsymbol{x} \neq \boldsymbol{x}_0$, the potential due to this charge is described by

$$\phi_{\boldsymbol{x}_0}(x, y) \stackrel{\Delta}{=} -\log \|\boldsymbol{x} - \boldsymbol{x}_0\|. \tag{4.1}$$

It is well known that $\phi_{\boldsymbol{x}_0}$ is harmonic in any region not containing the point $\boldsymbol{x}_0$. Moreover, for every harmonic function u, there exists an analytic function $w : \mathbb{C} \to \mathbb{C}$ such that $u(x, y) = \mathrm{Re}(w(x, y))$. In the following, we will work with analytic functions in complex domain. Let $x_0 + \mathrm{i}y_0 = z_0 \in \mathbb{C}$ and $x + \mathrm{i}y = z \in \mathbb{C}$, we have

$$\phi_{\boldsymbol{x}_0}(\boldsymbol{x}) = \mathrm{Re}(-\log(z - z_0)).$$

We will refer to the analytic function $\log(z)$ as the potential due to the unit charge located in the origin.

Lemma 4.1.1 ([13])**.** *Let a point charge of intensity q be located at z_0. Then for any z such that $|z| > |z_0|$,*

$$\phi_{z_0}(z) = q \log(z - z_0) = q \left(\log z - \sum_{k=1}^{\infty} \frac{1}{k} \left(\frac{z_0}{z} \right)^k \right). \tag{4.2}$$

Proof. Note that $\log(z - z_0) = \log(z) + \log(1 - \frac{z_0}{z})$ and $|\frac{z_0}{z}| < 1$. The lemma follows from the expansion

$$\log(1 - w) = -\sum_{k=1}^{\infty} \frac{w^k}{k},$$

which is valid for any w with $|w| < 1$. □

This provides us with a method of computing the multipole expansion due to a collection of charges.

Theorem 4.1.2 (**Multipole expansion**, [13])**.** *Suppose that m charges of strengths $\{q_i,\ i = 1,\ldots,m\}$ are located at points $\{z_i,\ i = 1,\ldots,m\}$, with $|z_i| < r$. Then for any z with $|z| > r$, the potential $\phi(z)$ induced by the charges is given by*

$$\phi(z) = Q\log(z) + \sum_{k=1}^{\infty} \frac{a_k}{z^k}, \tag{4.3}$$

where

$$Q = \sum_{i=1}^{m} q_i \quad \textit{and} \quad a_k = \sum_{i=1}^{m} \frac{-q_i z_i^k}{k}. \tag{4.4}$$

Furthermore, for any $p \geq 1$,

$$\begin{aligned} \left| \phi(z) - Q\log(z) - \sum_{k=1}^{p} \frac{a_k}{z^k} \right| &\leq \frac{1}{p+1}\alpha \left|\frac{r}{z}\right|^{p+1} \\ &\leq \left(\frac{A}{p+1}\right)\left(\frac{1}{c-1}\right)\left(\frac{1}{c}\right)^p, \end{aligned} \tag{4.5}$$

where

$$c = \left|\frac{z}{r}\right|, \qquad A = \sum_{i=1}^{m} |q_i|, \quad \textit{and} \quad \alpha = \frac{A}{1 - |\frac{r}{z}|}. \tag{4.6}$$

Proof. By using the preceding Lemma 4.1.1 and the fact that $\phi(z) = \sum_{i=1}^{m} \phi_{z_i}(z)$, we can immediately get (4.3). To obtain the error bound, observe that

$$\left| \phi(z) - Q\log(z) - \sum_{k=1}^{p} \frac{a_k}{z^k} \right| \leq \left| \sum_{k=p+1}^{\infty} \frac{a_k}{z^k} \right|.$$

Substituting expression (4.4) for a_k into the above, we get

$$\left| \sum_{k=p+1}^{\infty} \frac{a_k}{z^k} \right| \leq A \sum_{k=p+1}^{\infty} \frac{r^k}{k|z|^k} \leq \frac{A}{p+1} \sum_{k=p+1}^{\infty} \left|\frac{r}{z}\right|^k = \frac{a}{p+1}\left|\frac{r}{z}\right|^{p+1} = \left(\frac{A}{p+1}\right)\left(\frac{1}{c-1}\right)\left(\frac{1}{c}\right)^p.$$

which is the required result. □

Let us consider the computational efficiency of multipole expansion. In order to compute the potential at the points $\{\tilde{z}_j\}_{j=1}^{n}$ (we will call them "targets") due to the charges at the points $\{z_i\}_{i=1}^{m}$

(we will call them "sources") as in Theorem 4.1.2, we could use

$$\sum_{i=1}^{m} \phi_{z_i}(\tilde{z}_j) \qquad \text{for all } j = 1, \cdots, n.$$

This clearly requires $O(mn)$ operations. Now suppose we compute $\{a_k\}_{k=1}^{p}$ from m "sources" with intensities $q_1, q_2, \ldots, q_m$, using (4.4) in Theorem 4.1.2. It requires $O(mp)$ operations. Then the evaluation of the potentials at the n "targets", using (4.5) in Theorem 4.1.2, requires $O(np)$ operations. Therefore, by applying the multipole expansion, the amount of computation can be reduced to $O(m) + O(n)$, which is significantly smaller than $O(mn)$ for large m and n.

The p term in Theorem 4.1.2 is very important to the approximation of potential. If $c = 2$ in (4.6), then the error bound will be

$$\left| \phi(z) - Q\log(z) - \sum_{k=1}^{p} \frac{a_k}{z^k} \right| \leq A \left(\frac{1}{2}\right)^p, \tag{4.7}$$

and if we want to obtain the relative precision ε, p must be of the order $-\log_2(\varepsilon)$.

4.1.2 The $O(N \log N)$ Algorithm

To simplify the explanation of this algorithm, let us assume for the moment that the particles are almost uniformly distributed in a square domain. We want to compute the potential for each particle due to the rest of the particles with charge intensities. If we have N particles in total, then the naive computational cost will be $O(N^2)$. But using multipole expansion, we can reduce this cost.

In order to use multipole expansion systematically, we apply the well-known tree data structure **quad tree**. In our case, the quad tree will partition the square by recursively subdividing it into four squares, as shown in Fig. 4.1. At refinement level 0, we have the entire domain. Refinement level $\ell+1$ is obtained recursively from level ℓ by subdivision of each square (we call them **parents**) into four equal squares (we call them **children**). We will call each square of level ℓ a "node" of level ℓ.

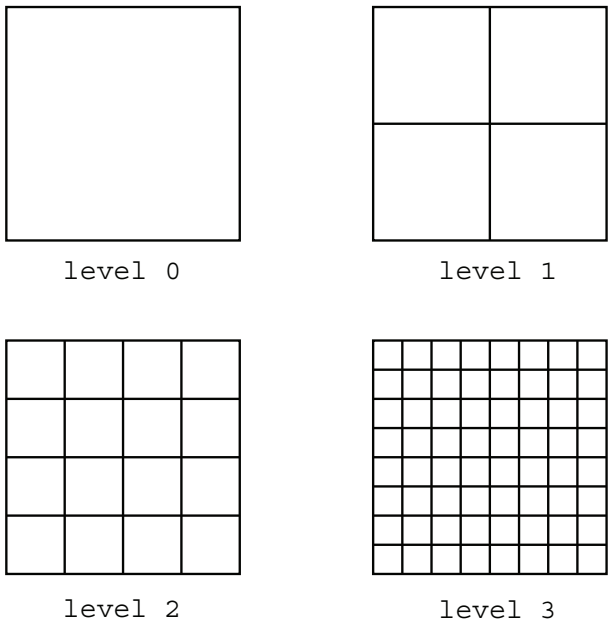


Figure 4.1: Quad tree: successive refinement of a square domain.

Definition 4.1.3. Two nodes are said to be **near neighbors** if they are at the same refinement level and share a boundary point. A node is a near neighbor of itself.

Definition 4.1.4. Two nodes are said to be **well separated** if they are at the same refinement level and are not near neighbors.

Definition 4.1.5. Each node i has its own **interaction list**, consisting of the children of the near neighbors of i's parent which are well separated from i. See details in Fig. 4.2.

The basic idea is to consider the clusters of particles at each level of quad tree and compute the interaction between well separated clusters by means of multipole expansion. Notice that in Theorem 4.1.2, the value of c control the decaying of the error bound in (4.5). Followed by Definition 4.1.4, if "targets" node and "source" node are well separated, we will have $c \geq 2$, which allows us to apply multipole expansion.

At level 0 and 1, there are no pairs of well separated nodes. At level 2, there are total sixteen nodes. Each node i of level 2 has its own interaction list (see Fig. 4.2). Then the multipole expansion can be used to compute the interactions between the well separated nodes with error

bound ε, i.e., the number of expansion terms will be $p = \log_2(\frac{1}{\varepsilon})$.

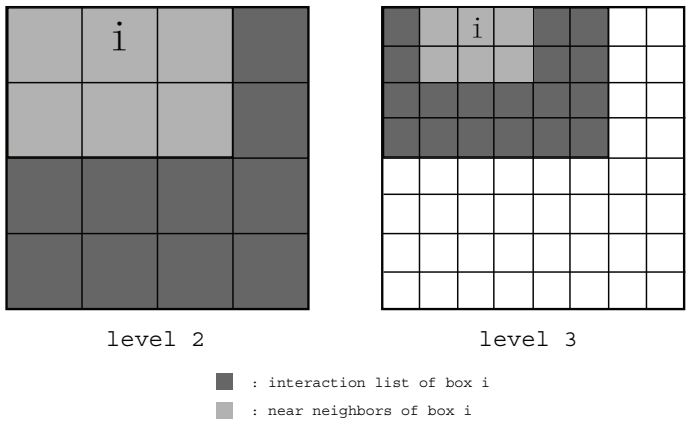


Figure 4.2: Steps of the algorithm.

It remains to compute the interactions between particles contained in each node i with those contained in i's near neighbors, and this is where recursion enters the picture. As in Fig. 4.2, at level 3, the nodes (white boxes) have been already considered (or finished) at level 2, shown in the left part. For each node i of level 3, we seek its interaction list and apply multipole expansions to compute the interactions. The near neighbors cannot be computed by means of multipole expansion. Then we should move on to next finer level.

The recursion will be stopped after roughly $\log N$ levels of refinement. The amount of work at each level is of order $O(N)$, since each particle contributes to p expansion coefficients. At the finest level, we will have $O(1)$ particles per node and compute the interactions directly. Therefore, it will cost $O(N)$ operations. In summary, the computational cost of this algorithm will roughly be of order $O(N \log N)$.

4.1.3 FMM: The $O(N)$ Method

In order to develop an $O(N)$ algorithm, we need several analytic results concerning multipole expansion and local expansion. The theorems are drawn from [13] and [14]. Here we provide the detailed proof and some important comments.

Theorem 4.1.6 (**Translation of a multipole expansion**, [13])**.** *Suppose that*

$$\phi(z) = a_0 \log(z - z_0) + \sum_{k=1}^{\infty} \frac{a_k}{(z - z_0)^k} \tag{4.8}$$

is a multipole expansion of the potential due to a set of m charges of strength $q_1, q_2, \ldots, q_m$, all of which are located inside the disk D of radius R with center at z_0. Then for z outside the disk D_1 of radius $(R + |z_0|)$ and center at the origin,

$$\phi(z) = a_0 \log(z) + \sum_{l=1}^{\infty} \frac{b_l}{z^l}, \tag{4.9}$$

where

$$b_l = -\frac{a_0 z_0^l}{l} + \sum_{k=1}^{l} a_k z_0^{l-k} \binom{l-1}{k-1}, \tag{4.10}$$

with $\binom{l}{k}$ the binomial coefficients. Furthermore, for any $p \geq 1$,

$$\left| \phi(z) - a_0 \log(z) - \sum_{l=1}^{p} \frac{b_l}{z^l} \right| \leq \left(\frac{A}{1 - \left| \frac{|z_0| + R}{z} \right|} \right) \left| \frac{|z_0| + R}{z} \right|^{p+1} \tag{4.11}$$

with A defined in (4.6)*.*

In [13], there is a hint for the proof. A detailed one is provided as follows.

Proof. First of all, by Lemma 4.1.1

$$\log(z - z_0) = \log(z) - \sum_{k=1}^{\infty} \frac{1}{k} \left(\frac{z_0}{z} \right)^k \tag{4.12}$$

Second, by Newton's Binomial series $\dfrac{1}{(1-z)^s} = \displaystyle\sum_{n=0}^{\infty} \binom{s-1+n}{s-1} z^n$, we have

$$\frac{1}{(z - z_0)^k} = \frac{1}{z^k} \cdot \frac{1}{1 - \left(\frac{z_0}{z} \right)^k} = \frac{1}{z^k} \sum_{n=0}^{\infty} \binom{k-1+n}{k-1} \frac{z_0^n}{z^n},$$

By letting $l = n + k$, we get

$$\frac{1}{(z - z_0)^k} = \sum_{l=k}^{\infty} \binom{l-1}{k-1} \frac{z_0^{l-k}}{z^l}. \tag{4.13}$$

Now substitute (4.12) and (4.13) into (4.8), we get

$$\begin{aligned}
\phi(z) &= a_0 \log(z) - a_0 \sum_{k=1}^{\infty} \left(\frac{z_0}{z}\right)^k + \sum_{k=1}^{\infty} a_k \sum_{l=k}^{\infty} \binom{l-1}{k-1} \frac{z_0^{l-k}}{z^l} \\
&= a_0 \log(z) - \sum_{l=1}^{\infty} \frac{a_0 z_0^l}{l} \cdot \frac{1}{z^l} + \sum_{k=1}^{\infty} \sum_{l=k}^{\infty} a_k z_0^{l-k} \binom{l-1}{k-1} \frac{1}{z^l} \\
&= a_0 \log(z) + \sum_{l=1}^{\infty} \left(-\frac{a_0 z_0^l}{l} + \sum_{k=1}^{l} a_k z_0^{l-k} \binom{l-1}{k-1} \right) \frac{1}{z^l} \\
&= a_0 \log(z) - \sum_{l=1}^{\infty} \frac{b_l}{z^l}
\end{aligned}$$

where b_l is defined in (4.10).

Similarly as Theorem 4.1.2, let $c = \left|\frac{z}{r}\right|$, where $r = |z_0| + R$ in this case. Therefore, by applying Theorem 4.1.2, we immediately prove (4.11). □

Observation 4.1.1. *As shown in Fig. 4.3, the multipole expansion contributed by the particles in a parent node of level $\ell - 1$ can be constructed by merging the multipole expansion contributed by the particles in each child node of level ℓ by using the translation of the multipole expansion. In other words, once we compute the multipole expansion based on each child node, we first translate it to the center of their parent and then add the translated expansions together.*

Theorem 4.1.7 (Conversion of a multipole expansion into a local expansion, [13]). *Suppose that m charges of strengths $q_1, q_2, \ldots, q_m$ are located inside the disk D_1 with radius R and center at z_0, and that $|z_0| > (c+1)R$ with $c > 1$. Then the corresponding multipole expansion* (4.8) *converges inside the disk D_2 of radius R center at origin. Inside D_2, the potential due to the charges is described by a power series:*

$$\phi(z) = \sum_{l=0}^{\infty} b_l \cdot z^l, \tag{4.14}$$

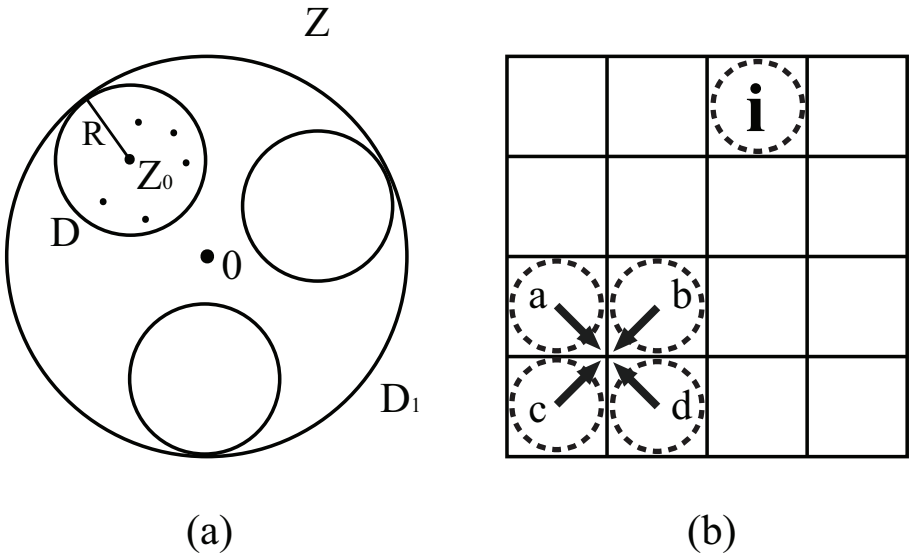


Figure 4.3: Translation of multipole expansion. (a) The multipole expansion centered at z_0, the center of the disk D can be translated to the multipole expansion centered at origin, which is the center of the parent disk D_1; (b) The merging procedure. For node i, the multipole expansion contributed by the particles inside each of the nodes a, b, c, d can be translated to the expansion centered at the center of their parent, then add the four translated expansions together to construct the multipole expansion for their parent.

where

$$b_0 = a_0 \log(-z_0) + \sum_{k=1}^{\infty} \frac{a_k}{z_0^k}(-1)^k, \tag{4.15}$$

and

$$b_l = -\frac{a_0}{l \cdot z_0^l} + \frac{1}{z_0^l} \sum_{k=1}^{\infty} \frac{a_k}{z_0^k} \binom{l+k-1}{k-1} (-1)^k, \qquad \textit{for } l \leq 1. \tag{4.16}$$

Furthermore, an error bound for the truncated series is given by

$$\left| \phi(z) - \sum_{l=0}^{p} b_l \cdot z^l \right| < \frac{A(4\mathrm{e}(p+c)(c+1)+c^2)}{c(c-1)} \left(\frac{1}{c}\right)^{p+1}, \tag{4.17}$$

where A is defined in (4.6) *and* e *is the base of natural logarithms.*

The proof of this theorem was provided in [14]. See the Fig. 4.4 for graphic explanation.

Theorem 4.1.8 (Translation of a local expansion, [13]). *For any complex z_0, z, and $\{a_k\}$, $k =$*

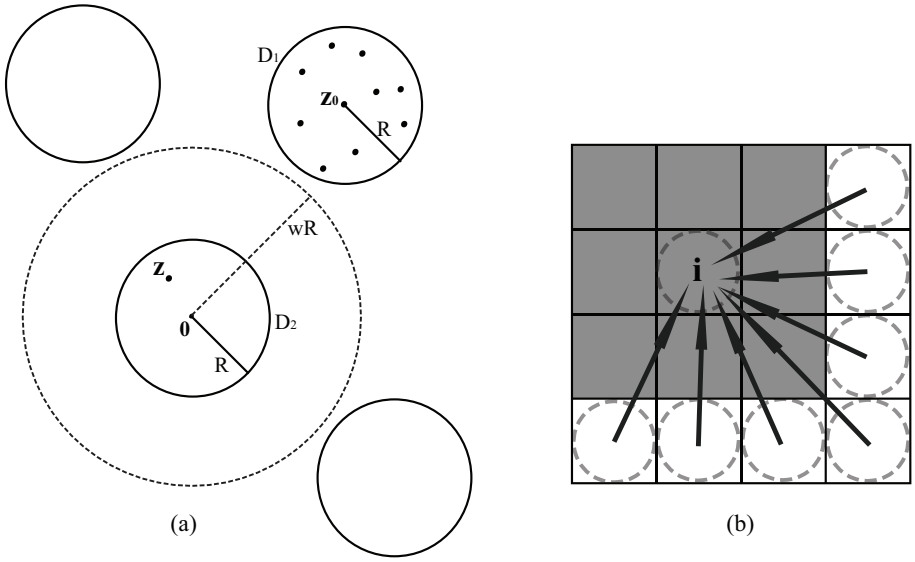


Figure 4.4: Conversion of multipole expansion into a local expansion. (a) the multipole expansion centered about z_0, the center of the circle D_1 can be converted to an expansion centered at origin, the center of the circle D_2, which is well separated from D_1; (b) For node i, the multipole expansion based on the particles inside each of the nodes (white boxes) can be converted to a local expansion about i. Notice that the set of the nodes (white boxes) is the interaction list of node i.

$0, 1, 2, \ldots, n$,

$$\sum_{k=0}^{n} a_k (z - z_0)^k = \sum_{l=0}^{n} \left(\sum_{k=l}^{n} a_k \binom{k}{l} (-z_0)^{k-l} \right) z^l. \tag{4.18}$$

Proof. The equation (4.18) is an immediate consequence of Maclaurin's theorem.

$$\begin{aligned} \sum_{k=0}^{n} a_k (z - z_0)^k &= \sum_{k=0}^{n} a_k \sum_{l=0}^{k} \binom{k}{l} (-z_0)^{k-l} z^l \\ &= \sum_{l=0}^{n} \sum_{k=l}^{n} a_k \binom{k}{l} (-z_0)^{k-l} z^l \end{aligned}$$

which finish the proof. □

Observation 4.1.2. *Assume we have a local expansion on node i of level 2, which is constructed by converting the multipole expansion based on each nodes of i's the interaction to the local expansion on i and then adding these local expansions together. Therefore, in the level 3, for any child of i, we translate the local expansion on parent i to this child, which is contributed by all the*

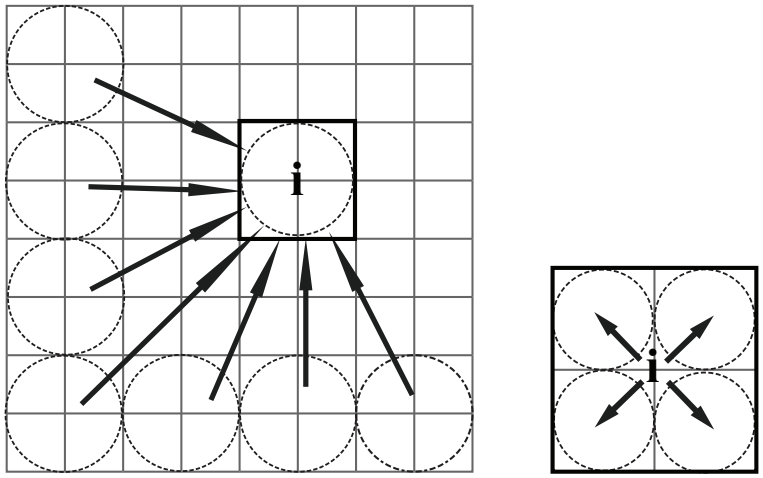


Figure 4.5: Conversion of multipole expansion into a local expansion and translation of a local expansion. (a) For node i, the multipole expansion based on the particles inside each of the nodes (white boxes) can be converted to a local expansion about node i. Notice that the set of the white boxes is the interaction list of node i. (b) The local expansion centered about the node i, which is contributed by all the particles outside of i and i's near neighbors, can be translated to the local expansion for its children.

particles outside i's near neighbors. By doing this, for the nodes of level 3, we only need to take care of the particles inside their parents and the parents' near neighbors.

Now it is time to improve the previous $O(N \log N)$ algorithm in Sec. 4.1.2 by using the analysis in this section. In the $O(N \log N)$ algorithm, all of the particles will be accessed in every level of refinement. The FMM, i.e., $O(N)$ algorithm, will avoid such computation load. Here, we simply indicate the modifications on $O(N \log N)$ which lead to the FMM implementation (see [2]).

Initialization

Given N particles distributed in a square domain. Construct a quad tree with $L + 1$ levels, which contains the refinement information of this domain. The indices of levels will be $0, 1, 2, \ldots, L-1, L$ (see Sec. 4.1.2 for the details of quad tree). Assume that, on average, s

particles per box in the finest level, we will have $4^L \cdot s = N$, or equivalently, $L = \log_4(N/s)$.

Upward Pass

In $O(N \log N)$ algorithm, we proceed from coarsest level to finest level by forming multipole expansions for every node. In the FMM, we will start with the finest level, construct multipole expansions for each node of level L. Then the multipole expansion for all nodes in the coarser levels will be constructed by the merging procedure described in Observation 4.1.1.

Downward Pass

In $O(N \log N)$ algorithm, for each node i in level $\ell \in \{2, 3, \ldots, L\}$ under consideration, we directly add the multipole expansions from the nodes of i's interaction list to the potential due to the particles outside of i's near neighbors. But in FMM, we convert each of these multipole expansions into a local expansion about i's center, using Theorem 4.1.7, then add them together.

After these calculations are completed, we have a local expansion for each node i in every level. Then beginning at the coarsest level, which in fact is level 2, we translate these local expansions to the children in level 3, as explained in Observation 4.1.2, and add to the children's local expansion. After this recursion process reaches the finest level, a local expansion of each box i will be the potential due to all of the particles outside i's near neighbors. The near neighbor interactions can be computed directly.

Let us count the approximate amount of operations needed in the above procedure.

In **Upward Pass**, to form the multipole expansions for each node in the finest level, we need about Np operations, where p is the number of terms in the multipole expansion (see Theorem 4.1.2). Then for the translations for the higher levels, we need about $(\frac{N}{s})p^2$ operations.

In **Downward Pass**, to convert the multipole expansions about all nodes of the interaction list of each node in an arbitrary level, we need about $27(\frac{N}{s})p^2$ operations. Then for the translations from the parent to its children, we need about $(\frac{N}{s})p^2$ operations. For the evaluation of a local

expansion in the finest level and computing potential directly from the near neighbor, we need about Np and $9Ns$ respectively.

Therefore, the total count will be approximately

$$Np + 29\left(\frac{N}{s}\right)p^2 + Np + 9Ns$$

where if $s = p$, this yields $40Np$.

4.2 Matrix Representations of FMM

The 2D kernel introduced in Chapter 3 is

$$K(\boldsymbol{x}, \boldsymbol{y}) = -\frac{1}{2\pi}\log\|\boldsymbol{x} - \boldsymbol{y}\|, \tag{4.19}$$

where $-\log\|\boldsymbol{x} - \boldsymbol{y}\|$ denotes the potential at point $\boldsymbol{x}$ due to a charge of unit strength located at point $\boldsymbol{y}$. Let us consider this kernel in complex domain, as explained in Sec. 4.1.1. Assume $\boldsymbol{x} = (x, y)$, $\boldsymbol{y} = (x_0, y_0)$. Let $z = x + \mathrm{i}y$, $z_0 = x_0 + \mathrm{i}y_0$. Then the kernel in (4.19) will be

$$K(\boldsymbol{x}, \boldsymbol{y}) = -\frac{1}{2\pi}\mathrm{Re}(\log(z - z_0)),$$

which can be simply written as $K(\boldsymbol{x}, \boldsymbol{y}) = -\frac{1}{2\pi}\log|z - z_0|$ for convenience.

We will investigate the following numerical computation for the eigenvalue problem in a complex domain, which is equivalent to $\mathbb{R}^2$.

Given a set of N particles located at N distinct points, i.e., $\mathrm{S} = \{z_1, z_2, \ldots, z_N\} \subset \mathbb{C}$. *For any set of reals $\{q_1, q_2, \ldots, q_N\}$, where q_i is the charge strength of the particle located at z_i, we want to compute the potential for each particle at z_i due to the rest of particles located at $\{z_j\}_{j=1, j\neq i}^{N}$.*

In terms of matrix-vector multiplication, we can rephrase this problem as follows:

$$\boldsymbol{\Phi} = \mathrm{P}\boldsymbol{q}, \tag{4.20}$$

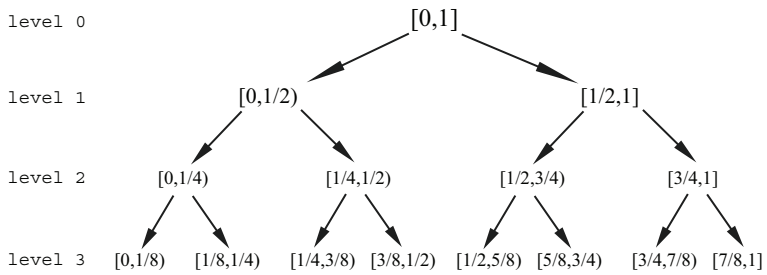


Figure 4.6: Binary tree structure induced by a uniform subdivision of the interval $[0, 1]$.

where P with size of $N \times N$, whose diagonal entries are 0's; and $\mathrm{P}_{ij} = \log|z_i - z_j|$, for $i \neq j$; $\boldsymbol{q} \in \mathbb{R}^{N\times 1}$. Therefore, by computing the matrix-vector multiplication, we will get the potential vector $\boldsymbol{\Phi}$, which contains the potentials at each point due to the others.

Now the problem is to compute the matrix-vector multiplication rapidly. The basic idea of fast matrix-vector multiplication was introduced in [4], where the 1D case is completely explained. In this thesis, we further investigate the structure of a 2D kernel matrix P and the algorithm of matrix-vector algorithm.

4.2.1 Hierarchical Data Structure

One of the important properties of d-dimensional Euclidean space $\mathbb{R}^d$ is that it can be subdivided into rectangular domains: intervals for $d = 1$, squares for $d = 2$, and cubes for $d = 3$. In practice, we often deal with finite domains. For example, a finite domain in 2D can be enclosed by a bounding square. Assign the bounding square to level 0 in the hierarchical refinement scheme. Then the square can be evenly divided into 4 squares, which are tagged as to level 1. Repeating this procedure, we generate a sequence of squares in level 2, level 3, and so on. Even though this subdivision can be repeated forever, in practice we would stop at a refinement level L, which is determined by some criterion, e.g., the number of particles in each square of level L must be less than a finite number. Such a data structure is called quad tree. Sometimes people call binary tree for $d = 1$ and 2^d tree for general d. See the binary tree as shown in Fig. 4.6 and quad tree in Fig. 4.1.

Notice that the 2^d tree structures of our interest do not depend on the distribution of the parti-

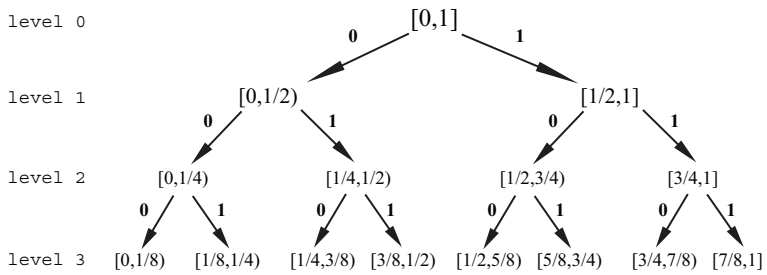


Figure 4.7: Ordinary Indexing Scheme on a Binary Tree Structure.

cles (or the sampling scheme of dataset). For example, in 2D, given the bounding square of level 0, we can do the homogeneous subdivision and stop at a particular level L, where L could be controlled by the size of whole dataset (e.g., if we want to have at most s particles in each square of level L, and if there are N particles in total, then $L \geq \log_4 \frac{N}{s}$).

To avoid the confusion, we will call each component of 2^d tree as a **node**.

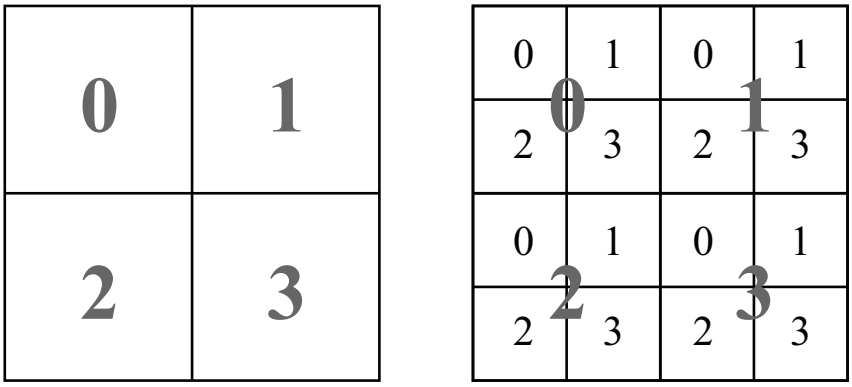
4.2.2 Indexing of a Quad Tree

To implement the matrix vector product $P\boldsymbol{q}$ in (4.20) efficiently, we want to apply the ideas of the multipole expansion and the local expansion discussed in Sec. 4.1.2 and Sec. 4.1.3. Recall that the efficiency of FMM comes from the approximations based on the well separated (see Definition. 4.1.4) subregions of a domain in terms of the Euclidean distance. And we also know that the 2^d tree structure is a subdivision of a domain into d-dimensional cubes (nodes), which is convenient for grouping particles based on the Euclidean distance.

In a 2^d tree, to index a node in an arbitrary level ℓ, we must know its index I_ℓ as one of the children of its parent, its parent's index $I_{\ell-1}$ as one of the children of its grandparent, and so on. Therefore, we can index any node in level ℓ by a vector of length ℓ, denoted by

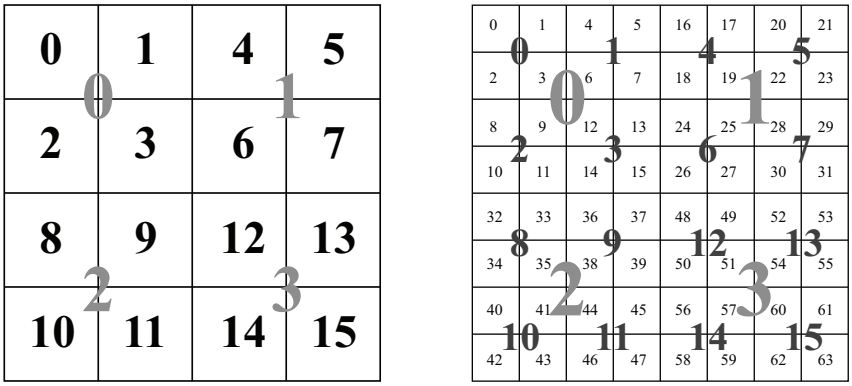
$$\boldsymbol{I} = (I_1, I_2, \cdots, I_\ell), \quad \text{with } I_j \in \{0, 1, 2, \cdots, 2^d - 1\},\ j = 1, 2, \cdots, \ell. \tag{4.21}$$

For example, in Figure 4.7 and Figure 4.8, we have binary tree indexing and quad tree indexing, respectively. It is still not convenient to trace one particular node by their indices in the vector



(a) $L = 1$ (b) $L = 2$

Figure 4.8: An ordinary indexing scheme of the quad tree.



(a) $L = 2$ (b) $L = 3$

Figure 4.9: New indexing scheme of the quad tree.

form $\boldsymbol{I}$ because every entry of $\boldsymbol{I}$ has to be accessed.

Therefore, let us apply a new indexing scheme: Given any index $\boldsymbol{I} = (I_1, \cdots, I_\ell)$, define a new index D as

$$\boldsymbol{D} \triangleq (D_{global} = \ell,\ D_{local} = \sum_{j=1}^{\ell} 4^{\ell-j} \cdot I_j). \tag{4.22}$$

where D_{global} records the level of the node and D_{local} can be considered as an identity number of the node of level D_{global}. With this indexing scheme, we can distinguish any pair of nodes of 2^d tree only by accessing two values: D_{global} and D_{local}. For example, in Figure 4.9(a), the square 12 of level 2 has index $\boldsymbol{D} = (2, 12)$ and the square 3 of level 1 has index $\boldsymbol{D} = (1, 3)$.

As shown in Figure 4.9, there are some interesting properties to be observed. In a quad tree, given a child node (c) with index $D^{(c)} = (D^{(c)}_{global}, D^{(c)}_{local})$, and its parent node (p) with index $D^{(p)} = (D^{(p)}_{global}, D^{(p)}_{local})$, we have the following relationships:

$$\begin{aligned} D^{(p)}_{global} &= D^{(c)}_{global} - 1; \\ D^{(p)}_{local} &= \left\lfloor \frac{D^{(c)}_{local}}{4} \right\rfloor; \\ D^{(c)}_{local} &= 4 \cdot D^{(p)}_{local} + i, \quad \text{where } i \in \{0, 1, 2, 3\}. \end{aligned} \tag{4.23}$$

We can generalize (4.23) to formulas in a 2^d tree as follows:

$$\begin{aligned} D^{(p)}_{global} &= D^{(c)}_{global} - 1; \\ D^{(p)}_{local} &= \left\lfloor \frac{D^{(c)}_{local}}{2^d} \right\rfloor; \\ D^{(c)}_{local} &= 2^d \cdot D^{(p)}_{local} + i, \quad \text{where } i \in \{0, 1, \cdots, 2^d - 1\}. \end{aligned} \tag{4.24}$$

Remark 4.2.1. This indexing scheme can help us construct the original kernel matrix and split it into submatrices. We will discuss about our matrix-splitting scheme in next section.

4.2.3 Matrix-splitting Scheme

Construction of the original kernel matrix

Using the indexing method in the previous section, we obtain a sequence of the nodes (from the finest level, say level L), where the indices are in the ascending order. We will use this sequence to construct the original kernel matrix. For example, Fig. 4.9(a) displays a square domain equipped with a quad tree with $L = 2$. Then the sequence used for construct the kernel matrix is $\{0, 1, 2, \ldots, 15\}$ and the original kernel matrix will have an abstract form shown in Fig. 4.10.

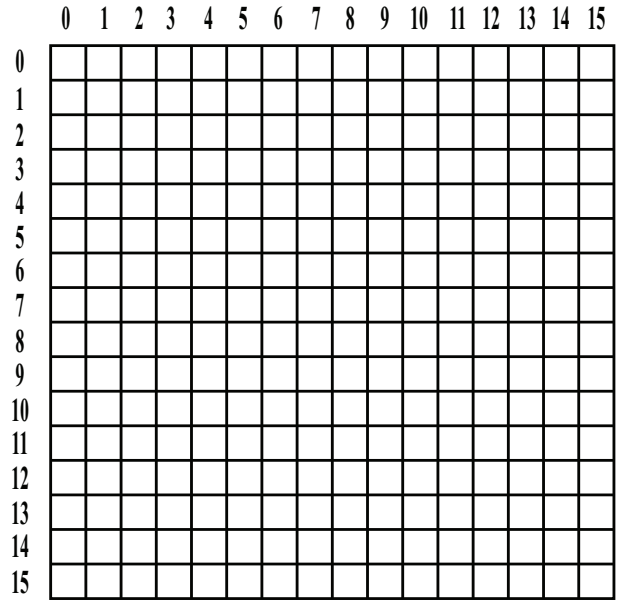


Figure 4.10: The construction of the original kernel matrix.

Splitting of the original kernel matrix

In Fig. 4.10, we already have a splitting form of the original kernel matrix. But we need to refine such splitting form according to our indexing scheme in a quad tree. Recall the definitions of the near neighbors and the interaction list from Sec. 4.1, the rules of our matrix-splitting scheme are

- Every submatrix generated by the splitting scheme should correspond to a pair of nodes from the same level of the quad tree;
- The two nodes in such a pair must either be near neighbors or belong to each other's interaction list.

Let us explain how to apply these splitting rules by an example. A square domain equipped with a quad tree with $L = 3$ is displayed in Fig. 4.9(b). By applying the splitting rules, we obtain the splitted kernel matrix shown in Fig. 4.11.

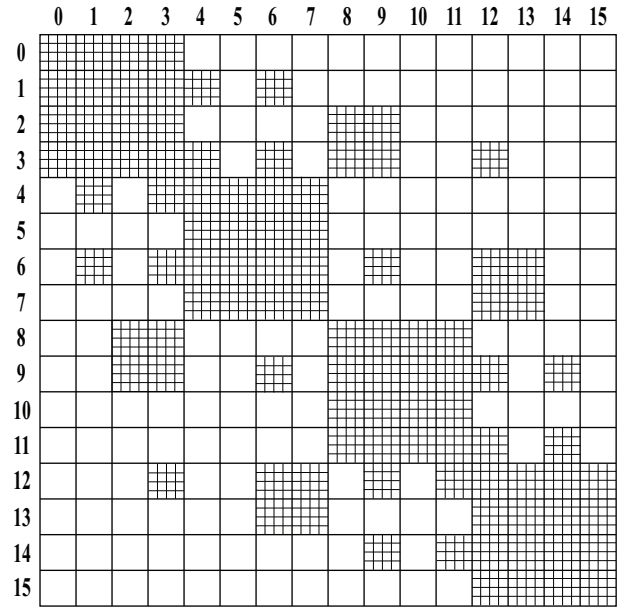


Figure 4.11: The splitted original kernel matrix generated by our matrix-splitting scheme.

4.2.4 The Low Rank Submatrices

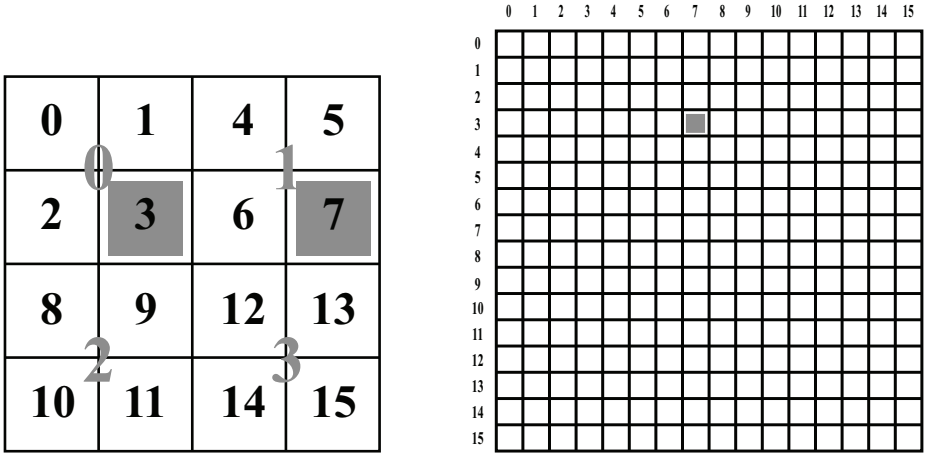
So far, we have discussed the idea of the original FMM, the quad tree structure, the matrix form of FMM, the indexing of a quad tree, and the matrix-splitting scheme. In this section, we will investigate some particular submatrices of the matrix P in (4.20).

Interaction List and Multipole Expansion

Given a square shape domain Ω with a quad tree structure as shown in Figure 4.12 (a), where the finest level is level 2. For the simplicity, assume that a S collection of point charges uniformly distributed in Ω. Then by applying the indexing scheme in the previous section, we get the corresponding division of matrix P as shown in Figure 4.12 (b).

Denote the submatrices by $B_{i,j}$, $i, j = 0, \ldots, 15$. Let us just pick one of them, say $B_{3,7}$, corresponding to two dark squares in Figure 4.12 (a), where the two corresponding subsets of S are S_3 and S_7.

Furthermore, assume $S_3 = \{z_{3,1}, z_{3,2}, \ldots, z_{3,m_3}\}$, $S_7 = \{z_{7,1}, z_{7,2}, \ldots, z_{7,m_7}\}$. Then the



(a) A square domain with a quad tree (b) The matrix P

Figure 4.12: A simple example.

submatrix of P will be written as:

$$B_{3,7} = \begin{pmatrix} \log|z_{3,1} - z_{7,1}| & \log|z_{3,1} - z_{7,2}| & \cdots & \log|z_{3,1} - z_{7,m_7}| \\ \log|z_{3,2} - z_{7,1}| & \log|z_{3,2} - z_{7,2}| & \cdots & \log|z_{3,2} - z_{7,m_7}| \\ \vdots & \vdots & \vdots & \vdots \\ \log|z_{3,m_3} - z_{7,1}| & \log|z_{3,m_3} - z_{7,2}| & \cdots & \log|z_{3,m_3} - z_{7,m_7}| \end{pmatrix}.$$

Recall Theorem 4.1.2 about the multipole expansion. Since the two subsets S_3 and S_7 are well separated (in fact, S_7 is one of the interaction list of S_3), we can write each element of $B_{3,7}$ into the form of multipole expansion. Assume the center of S_3 is $\overline{z_3}$, then by Lemma 4.1.1 and Theorem 4.1.2, we get

$$\log|z_{3,i} - z_{7,j}| \approx \log|z_{7,j} - \overline{z_3}| + \sum_{k=1}^{p} \frac{-|z_{3,i} - \overline{z_3}|^k/k}{|z_{7,j} - \overline{z_3}|^k}, \tag{4.25}$$

where $p \in \mathbb{N}$, which controls the accuracy of the approximation. Note that p depends on the separateness of S_3 and S_7, but not on the size of $B_{3,7}$. Generally, $p \ll m_3,\ m_7$. See the detail discussion about p in Sec. 4.1.1.

Therefore, we get the approximation of $B_{3,7}$ as

$$B_{3,7} \approx \underbrace{\begin{pmatrix} -|z_{3,1}-\overline{z_3}| & \cdots & -|z_{3,1}-\overline{z_3}|^p/p & 1 \\ -|z_{3,2}-\overline{z_3}| & \cdots & -|z_{3,2}-\overline{z_3}|^p/p & 1 \\ \vdots & \vdots & \vdots & \vdots \\ -|z_{3,m_3}-\overline{z_3}| & \cdots & -|z_{3,m_3}-\overline{z_3}|^p/p & 1 \end{pmatrix}}_{M_{2,3}:\, m_3 \times p+1} \cdot \underbrace{\begin{pmatrix} \frac{1}{|z_{7,1}-\overline{z_3}|} & \frac{1}{|z_{7,2}-\overline{z_3}|} & \cdots & \frac{1}{|z_{7,m_7}-\overline{z_3}|} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{1}{|z_{7,1}-\overline{z_3}|^p} & \frac{1}{|z_{7,2}-\overline{z_3}|^p} & \cdots & \frac{1}{|z_{7,m_7}-\overline{z_3}|^p} \\ \log|z_{7,1}-\overline{z_3}| & \log|z_{7,2}-\overline{z_3}| & \cdots & \log|z_{7,m_7}-\overline{z_3}| \end{pmatrix}}_{p+1 \times m_7}, \quad (4.26)$$

where "$\approx$" is derived in the sense of of Forbenius norm. Notice that this form is derived exactly from Theorem 4.1.2 about multipole expansion, where the first matrix on the right hand side of (4.26), denoted by $M_{2,3}$, consists of the multipole expansion coefficients based on the leaf node 3.

From the equation (4.26), we can conclude that the submatrix $B_{3,7}$ can be approximated by a linear combination of $p+1$ column vectors, which means the rank of $B_{3,7}$ could be as small as $p+1$. Here let us specify the "rank" of a matrix as follows: Given a matrix $A \in \mathbb{R}^{m\times n}$, let its singular values be $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_m$. If there exist an integer $1 \leq k \ll m$, such that σ_k is very small, say $\sigma_k < 10^{-8}$, then this matrix has low rank property.

Remark 4.2.2. Actually, we can intuitively get the low rank property by considering the smoothness of the kernel function $\log|z-z_0|$ when z and z_0 are far away enough from each other.

Remark 4.2.3. Notice that at the level 2 of a quad tree, node 7 is one of the interaction list of node 3, see Def. 4.1.5. Therefore, given a node, say node i, all of the submatrices generated from that node and its interaction list have low rank property. Furthermore, from (4.26), we can see that these submatrices share the same column bases $M_{2,3}$, which does not depend on the sampling locations

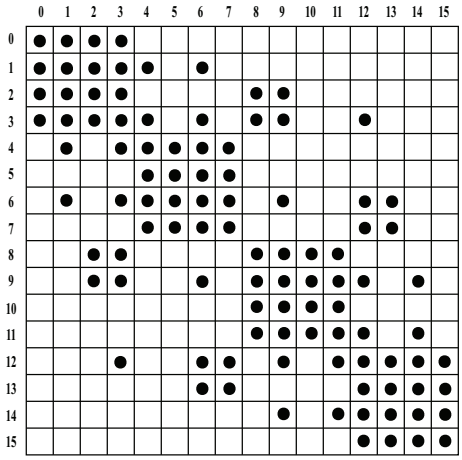


Figure 4.13: The kernel matrix P with low rank submatrices (the blank boxes) for $L = 2$.

in the nodes in the interaction list. This observation in fact is guaranteed by Theorem 4.1.2, which will be used later as an important key to our eigenvalue computation.

Remark 4.2.4. For any pair of near neighbors in the finest level of a quad tree, the corresponding submatrix does not have low rank property. In our experiments, we have to store them for the eigenvalue computation. Therefore, to save storage space for the huge dataset, it is recommended to apply a quad tree with deep levels where the number of the points in each node at the finest level is small.

Therefore, based on these considerations, we have the kernel matrix P shown in Figure 4.13, where the blank submatrices are the ones with low rank; the dotted submatrices are the ones with full rank.

Hierarchical Structure and Translation of Multipole Expansion

Assume that we have the same domain as the previous simple example, but the quad tree has one more level, i.e., the finest level is level 3 now. Figure 4.9(b) shows the indexing scheme applied on this quad tree.

We already know that the submatrices corresponding to two well-separated nodes have the low rank property. Therefore, if we assign a dot to the submatrices with full rank and blank boxes to

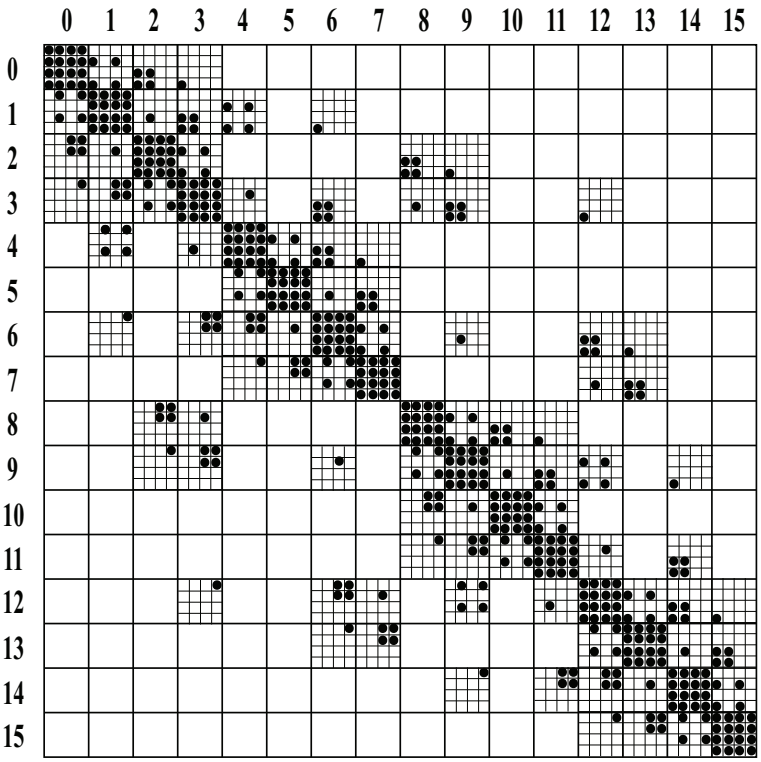


Figure 4.14: The kernel matrix P with low rank submatrices (the blank boxes) for $L = 3$.

the submatrices with low rank, we can get the kernel matrix P shown in Figure 4.14.

In Fig. 4.14, all of the blank boxes correspond to the low rank submatrices constructed by pairs of nodes who belong to each other's interaction list; all of the dotted boxes correspond to the full rank submatrices constructed by pairs of near neighbor nodes of the finest level. We can generalize such structure to the case of a quad tree with n levels. Therefore, thanks to the indexing scheme and the low rank properties, we obtain a matrix-splitting method.

There are important relationships among the low rank submatrices. For example, in Fig. 4.14, there are 12 blank boxes in P corresponding to the nodes in the interaction list of node 0 of level 3; there are also 12 blank boxes in P corresponding to its parent node, node 0 of level 2. To investigate the relationships of these low rank submatrices between child node and parent node, we will apply some ideas from FMM.

First, let us describe how to apply the translation of multipole expansion. Recall Theorem 4.1.6

as follows:

Suppose that

$$\phi(z) = a_0 \log(z - z_0) + \sum_{k=1}^{\infty} \frac{a_k}{(z - z_0)^k} \tag{4.27}$$

is a multipole expansion of the potential due to a set of m charges of strength $q_1, q_2, \ldots, q_m$, all of which are located inside the disk D of radius R with center at z_0. Then for z outside the disk D_1 of radius $(R + |z_0|)$ and center at the origin,

$$\phi(z) = a_0 \log(z) + \sum_{l=1}^{\infty} \frac{b_l}{z^l}, \tag{4.28}$$

where

$$b_l = -\frac{a_0 z_0^l}{l} + \sum_{k=1}^{l} a_k z_0^{l-k} \binom{l-1}{k-1}, \tag{4.29}$$

with $\binom{l}{k}$ the binomial coefficients. Furthermore, for any $p \geq 1$,

$$\left| \phi(z) - a_0 \log(z) - \sum_{l=1}^{p} \frac{b_l}{z^l} \right| \leq \left(\frac{A}{1 - \left| \frac{|z_0| + R}{z} \right|} \right) \left| \frac{|z_0| + R}{z} \right|^{p+1} \tag{4.30}$$

where A is some constant.

As explained in Figure 4.3 and Observation 4.1.1, the multipole expansion coefficients for a child can be transformed to the multipole expansion coefficients for its parent by applying formula (4.29). In our numerical computation, we will call the column basis of the low rank submatrices the matrix of the multiple expansion coefficients. Let us choose a particular pair of parent and its children to show this.

For a parent node in level 2 with index $D = (2, 0)$, the matrix $M_{2,0}$ is called the matrix of its multipole expansion coefficients. Then for its four children, $M_{3,i}$, $i = 0, 1, 2, 3$ are called the matrices of their multipole expansion coefficients. Thus, after we get the matrices $M_{3,i}$, $i = 0, 1, 2, 3$ in the similar manner as (4.26), which are in fact the matrices of the multipole expansion coefficients for the children, we can compute a transform matrix R_i for each $M_{3,i}$ by using (4.29).

Then the matrix $M_{2,0}$ can be computed by

$$M_{2,0} = \begin{pmatrix} M_{3,0}R_0 \\ M_{3,1}R_1 \\ M_{3,2}R_2 \\ M_{3,3}R_3 \end{pmatrix} = \begin{pmatrix} M_{3,0} & & & \\ & M_{3,1} & & \\ & & M_{3,2} & \\ & & & M_{3,3} \end{pmatrix} R, \quad \text{with } R = \begin{pmatrix} R_0 \\ R_1 \\ R_2 \\ R_3 \end{pmatrix}. \tag{4.31}$$

Notice that if we simply use p terms for all the approximation of low rank submatrices, the size of R will be $4p \times p$.

Remark 4.2.5. By knowing the translation of multipole expansions from children to parent, we find the transformation from the column bases of children's low rank matrices to the column bases of parent's low rank matrices, which is a basic hierarchical property for the original kernel matrix equipped with a quad tree structure. We will come back to this property combined with the symmetric property when we introduce our matrix decomposition algorithm.

Remark 4.2.6. In the numerical computation, we do not use exact column bases, which consist of the multipole expansion coefficients, since the approximation accuracy (which depends on the term p) cannot be precisely controlled (see the error term in Theorem 4.1.2 and Theorem 4.1.6.

To deal with these low rank submatrices, we should necessarily review some important issues about low rank matrix approximation problem, which has received much attention in the past decade and important to our matrix decomposition algorithm.

4.2.5 Low Rank Approximation Problem

Given a matrix $A \in \mathbb{R}^{m\times n}$, $m > n$, find a matrix $B \in \mathbb{R}^{m\times n}$ of rank at most k that minimizes

$$\|A - B\|_F^2 \triangleq \sum_{i,j} (A_{ij} - B_{ij})^2 \tag{4.32}$$

To solve the optimal B, one can use the Singular Value Decomposition (SVD) of a matrix. We start with a well known lemma.

Lemma 4.2.7. *For any matrix $C \in \mathbb{R}^{m\times n}$, given $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are unitary matrices, then $\|C\|_F^2 = \|U^T CV\|_F^2$.*

The proof is base on the fact that $\|C\|_F^2 = \text{tr}(C^T C) = \text{tr}(CC^T)$.

Assume the SVD of matrix A is $A = U\Sigma V^T$, where $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$ are both unitary, $\Sigma \in \mathbb{R}^{m\times n}$ with nonnegative reals $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ on the diagonal and zeros off the diagonal. Using Lemma 4.2.7, given the unitary matrices U and V from the SVD of A, we have

$$\begin{aligned} \|A - B\|_F^2 &= \|U^T AV - U^T BV\|_F^2 && (4.33)\\ &= \|\Sigma - U^T BV\|_F^2, && (4.34) \end{aligned}$$

which implies that $U^T BV$ must be a diagonal matrix in order to minimize the Frobenius norm of $\Sigma - U^T BV$, since Σ only has values on its diagonal.

So, we set $B = USV^T$, where $S \in \mathbb{R}^{m\times n}$ with zeros off the diagonal. Because we are looking for a matrix B whose rank is at most $k < n$, the diagonal entries of S should be set to $(s_{11}, \ldots, s_{kk}, 0, \ldots, 0)$. Therefore, we only need to find what values of s_{ii}, $i = 1, \ldots, k$ can minimize (4.32). We can keep working on (4.33) as follows:

$$\begin{aligned} \|A - B\|_F^2 &= \|\Sigma - U^T BV\|_F^2 \\ &= \|\Sigma - S\|_F^2 = \sum_{i=1}^{n} (\sigma_i - s_{ii})^2 \\ &= \sum_{i=1}^{k} (\sigma_i - s_{ii})^2 + \sum_{i=k+1}^{n} \sigma_i^2. \end{aligned}$$

Therefore, to minimize the Frobenius norm of $A - B$, is equivalent to minimize

$$\sum_{i=1}^{k} (\sigma_i - s_{ii})^2,$$

which means that $s_{ii} = \sigma_i$. Therefore, the solution to the low rank matrix approximation problem is matrix $B = USV^T$, where S has zeros everywhere except $s_{ii} = \sigma_i$ for $i = 1, \ldots, k$.

Remark 4.2.8. The low rank approximation problem is related to the truncated form of SVD. The

minimizer of $\|A-B\|_F$ is $B = USV^T$. Because there are only k non-zero entries on the diagonal of S, we can write B as $B = \tilde{U}\tilde{S}\tilde{V}^T$, where $\tilde{S} \in \mathbb{R}^{k\times k}$ is a diagonal matrix with $s_{ii} = \sigma_i$ for $i = 1, \ldots, k$; $\tilde{U} \in \mathbb{R}^{m\times k}$ contains the first k columns of U and $\tilde{V} \in \mathbb{R}^{n\times k}$ contains the first k columns of V.

Therefore, we can approximate the matrix-vector multiplication by truncating the SVD of the matrix based on the following inequality:

$$\|A\boldsymbol{x} - B\boldsymbol{x}\|_2^2 \leq \|A - B\|_2^2\|\boldsymbol{x}\|_2^2 \leq \|A - B\|_F^2\|\boldsymbol{x}\|_2^2.$$

Let B be the truncated form of the SVD of A, i.e., $B = \tilde{U}\tilde{S}\tilde{V}^T$. From the Lemma 4.2.7, we can get

$$\|A\boldsymbol{x} - B\boldsymbol{x}\|_2^2 \leq \sum_{i=k+1}^{n} \sigma_i^2 \cdot \|\boldsymbol{x}\|_2^2 \leq (n-k)\sigma_{k+1}^2 \cdot \|\boldsymbol{x}\|_2^2.$$

So, given the full SVD of low rank matrix A, in order to get B, we can set up a threshold $\varepsilon > 0$ and throw away the singular values smaller than ε, say $\sigma_k < \varepsilon < \sigma_{k+1}$. By doing so, we can speed up the computation of $A\boldsymbol{x}$ from the cost of direct computation $O(mn)$ to $O(k(m+n))$, the cost of $B\boldsymbol{x}$. Meanwhile, the approximation error can be controlled by ε.

4.3 Fast Matrix-Vector Multiplication

From the last section, we know that for a low rank submatrix, the truncated SVD (TSVD) will supply a good approximation and guarantee the fast matrix-vector multiplication. But if we only use TSVD for every low rank submatrix in the original kernel matrix P (see e.g., Fig. 4.14), the cost will be at least $O(N \log N)$; see the similar situation happened in Sec. 4.1.2. To achieve $O(N)$ complexity, we need several important ideas that we will describe below (see [4] for 1D case).

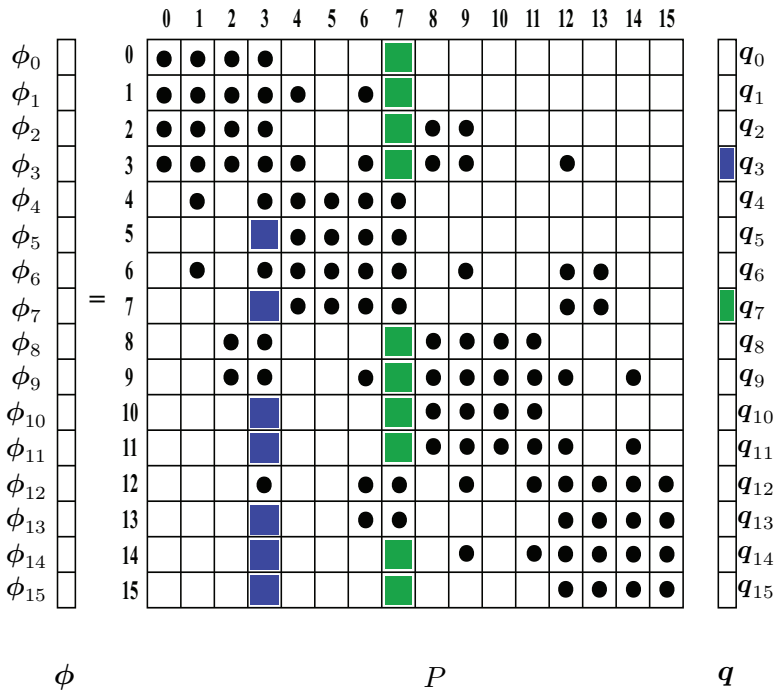


Figure 4.15: The matrix-vector multiplication.

4.3.1 Column Bases and Row Bases

Let us start with the simple example introduced in Sec. 4.2.4. In Fig. 4.15, we have the form of matrix-vector multiplication. Notice that we chop the vector ϕ and $\boldsymbol{q}$ into segments corresponding to the size of nodes of level 2. We will follow the notations used in Sec. 4.2.4. $B_{i,j}$ denotes the submatrix corresponding to the two nodes i and j, where $i, j = 0, 1, \ldots, 15$. For example, the green submatrices will multiply the green segment of $\boldsymbol{q}$, the blue submatrices will multiply the blue segment of $\boldsymbol{q}$, and $\phi_1 = \sum_{j=0}^{15} B_{1,j}\boldsymbol{q}_j$.

In Fig. 4.15, the green submatrices are $B_{i,7}$, where $i = 0, 1, 2, 3, 8, 9, 10, 11, 14, 15$, which in fact correspond to the interaction list of node 7. So the cost of computation on $B_{i,7}\boldsymbol{q}_7$ could be

reduced if they share the same row bases, which is true based on Sec. 4.2.4. Also, since the kernel matrix is symmetric, we have $B_{i,j} = B_{j,i}^T$. Therefore, the submatrix $B_{i,j}$ can be approximated by

$$B_{i,j} \approx U_i W_{i,j} U_j^T, \tag{4.35}$$

where $W_{i,j} \in \mathbb{R}^{p\times p}$; $U_i \in \mathbb{R}^{m_i \times p}$ is the column bases of $B_{i,j}$; $U_j \in \mathbb{R}^{m_j \times p}$ is the column bases of $B_{j,i}$, and the approximate rank of $B_{i,j}$ is p. Here we assume the size of $B_{i,j}$ is $m_i \times m_j$.

Therefore, for the computation in Fig. 4.15, assume we have computed all U_i, $i = 0, 1, \ldots, 15$, compute

$$G_j \triangleq U_j^T \boldsymbol{q}_j, \qquad j = 0, 1, \ldots, 15.$$

We store G_j's and use them later for computing the approximation

$$B_{i,j}\boldsymbol{q}_j \approx U_i W_{i,j} G_j.$$

Thus, we can save a lot by introducing the computation of G_j (see Sec. 4.1.2 for the similar situation). But to improve the speed from $O(N \log N)$ to $O(N)$, we must use some techniques related with the hierarchical levels. Notice that $\log N$ is constantly proportional to the number of levels of the quad tree. Therefore, if the computational cost for each level is about $O(N)$, the total algorithm will cost $O(N \log N)$.

4.3.2 Hierarchical Transformation of the Column Bases

Thanks to Sec. 4.2.4, we know that the column bases of the children can be transformed into the column bases of their parent. In Fig. 4.16, we use a particular part of original kernel matrix from Fig. 4.14 to show how to do the transformation.

Let us set $B_{i,j}^{(\ell)}$ to be the submatrix corresponding to the two nodes i and j of level ℓ. Therefore, in Fig. 4.16, the green block will be denoted by $B_{1,8}^{(2)}$, the four red blocks are the four submatrices in level 3: $B_{4,27}^{(3)}$, $B_{5,27}^{(3)}$, $B_{6,27}^{(3)}$, $B_{7,27}^{(3)}$. The four blue blocks are: $B_{12,32}^{(3)}$, $B_{12,33}^{(3)}$, $B_{12,34}^{(3)}$, $B_{12,35}^{(3)}$. The index information can be found in Fig. 4.9(b).

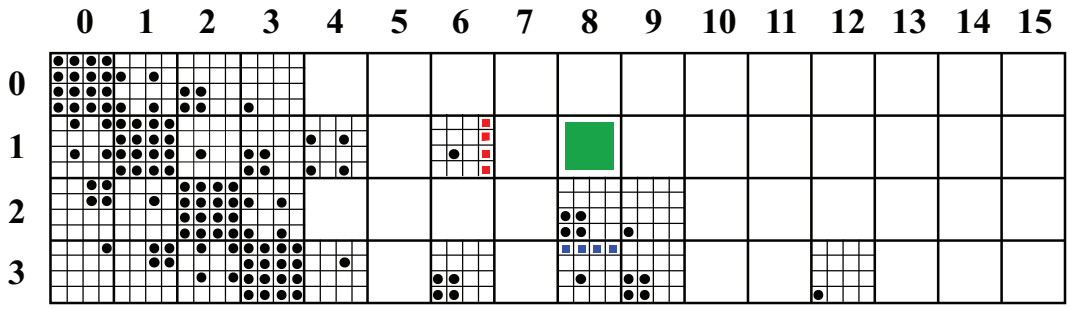


Figure 4.16: A part of kernel matrix P as in Figure 4.14.

Let us apply the formula (4.35) on these submatrices:

$$B^{(2)}_{1,8} \approx U^{(2)}_1 W^{(2)}_{1,8} U^{(2)^T}_8;$$

$$B^{(3)}_{j,27} \approx U^{(3)}_j W^{(3)}_{j,27} U^{(3)^T}_{27}, \quad j = 4, 5, 6, 7;$$

$$B^{(3)}_{12,k} \approx U^{(3)}_{12} W^{(3)}_{12,k} U^{(3)^T}_k, \quad k = 32, 33, 34, 35.$$

From Sec. 4.2.4, we know that we can compute transformation matrices $R^{(2)}_1$, and $R^{(2)}_8$ corresponding to $U^{(2)}_1$, and $U^{(2)}_8$, respectively, so that the transformations can be written as

$$U^{(2)}_1 = \begin{pmatrix} U^{(3)}_4 & & & \\ & U^{(3)}_5 & & \\ & & U^{(3)}_6 & \\ & & & U^{(3)}_7 \end{pmatrix} R^{(2)}_1 = \begin{pmatrix} U^{(3)}_4 R^{(2)}_{1,4} \\ U^{(3)}_5 R^{(2)}_{1,5} \\ U^{(3)}_6 R^{(2)}_{1,6} \\ U^{(3)}_7 R^{(2)}_{1,7} \end{pmatrix}; \tag{4.36}$$

We also have

$$U^{(2)^T}_8 = \left(R^{(2)^T}_{8,32} U^{(3)^T}_{32}, R^{(2)^T}_{8,33} U^{(3)^T}_{33}, R^{(2)^T}_{8,34} U^{(3)^T}_{34}, R^{(2)^T}_{8,35} U^{(3)^T}_{8,35} \right), \tag{4.37}$$

via transposition.

Therefore, the submatrix $B_{1,8}^{(2)}$ can be written as

$$B_{1,8}^{(2)} \approx \begin{pmatrix} U_4^{(3)} R_{1,4}^{(2)} \\ U_5^{(3)} R_{1,5}^{(2)} \\ U_6^{(3)} R_{1,6}^{(2)} \\ U_7^{(3)} R_{1,7}^{(2)} \end{pmatrix} W_{1,8}^{(2)} \left(R_{8,32}^{(2)^T} U_{32}^{(3)^T},\, R_{8,33}^{(2)^T} U_{33}^{(3)^T},\, R_{8,34}^{(2)^T} U_{34}^{(3)^T},\, R_{8,35}^{(2)^T} U_{8,35}^{(3)^T} \right).$$

In the matrix-vector multiplication, we need to compute $B_{1,8}^{(2)} \boldsymbol{q}_8^{(2)}$, where $\boldsymbol{q}_8^{(2)}$ is the segment of $\boldsymbol{q}$ corresponding to the node 8 of level 2, which can be written as

$$\begin{aligned} B_{1,8}^{(2)} \boldsymbol{q}_8^{(2)} \approx & \begin{pmatrix} U_4^{(3)} R_{1,4}^{(2)} \\ U_5^{(3)} R_{1,5}^{(2)} \\ U_6^{(3)} R_{1,6}^{(2)} \\ U_7^{(3)} R_{1,7}^{(2)} \end{pmatrix} W_{1,8}^{(2)} \\ & \cdot \left(R_{8,32}^{(2)^T} U_{32}^{(3)^T},\, R_{8,33}^{(2)^T} U_{33}^{(3)^T},\, R_{8,34}^{(2)^T} U_{34}^{(3)^T},\, R_{8,35}^{(2)^T} U_{8,35}^{(3)^T} \right) \begin{pmatrix} \boldsymbol{q}_{32}^{(3)} \\ \boldsymbol{q}_{33}^{(3)} \\ \boldsymbol{q}_{34}^{(3)} \\ \boldsymbol{q}_{35}^{(3)} \end{pmatrix}, \end{aligned} \tag{4.38}$$

where the segment $\boldsymbol{q}_8^{(2)}$ is further divided into four parts corresponding to the four children in level 3. Let us define

$$G_j^{(\ell)} \triangleq U_j^{(\ell)^T} \boldsymbol{q}_j^{(\ell)}. \tag{4.39}$$

So, (4.38) can be simplified as

$$B_{1,8}^{(2)}\boldsymbol{q}_8^{(2)} \approx \begin{pmatrix} U_4^{(3)}R_{1,4}^{(2)} \\ U_5^{(3)}R_{1,5}^{(2)} \\ U_6^{(3)}R_{1,6}^{(2)} \\ U_7^{(3)}R_{1,7}^{(2)} \end{pmatrix} W_{1,8}^{(2)}G_8^{(2)},$$

where $G_8^{(2)} = \sum_{j=32,33,34,35} R_{8,j}^{(2)^T} G_j^{(3)}$. Generally, we have the formula for $G_k^{(\ell)}$ as follows:

$$G_k^{(\ell)} = \sum_{j\in k\text{'s Children}} R_{k,j}^{(\ell)^T} G_j^{(\ell+1)}. \tag{4.40}$$

Fast Computation Technique: Begin with the finest level L, compute all possible $G_j^{(L)}$, then go on with the coarser levels, use formula (4.40) to compute all possible $G_k^{(\ell)}$, until finish up to the level 2, the coarsest level with low rank submatrices. Notice that this technique applies the similar idea as the **Upward Pass** of FMM; see Sec. 4.1.3 for details.

Also, we can immediately compute

$$F_{i,j}^{(\ell)} \triangleq W_{i,j}^{(\ell)}G_j^{(\ell)}. \tag{4.41}$$

Then let us take a look at $B_{1,8}^{(2)}\boldsymbol{q}_8^{(2)}$ again:

$$B_{1,8}^{(2)}\boldsymbol{q}_8^{(2)} \approx \begin{pmatrix} U_4^{(3)}R_{1,4}^{(2)}F_{1,8}^{(2)} \\ U_5^{(3)}R_{1,5}^{(2)}F_{1,8}^{(2)} \\ U_6^{(3)}R_{1,6}^{(2)}F_{1,8}^{(2)} \\ U_7^{(3)}R_{1,7}^{(2)}F_{1,8}^{(2)} \end{pmatrix},$$

which suggests that we can separate $B_{1,8}^{(2)}$ into four blocks according to the four children. Fig. 4.17 shows a sliced form of all submatrices $B_{1,j}^{(2)}$ for $j = 0, 1, \ldots, 15$ according to the four children of node 1 of level 2. Notice that $\boldsymbol{\phi}$ can also be separated according to all of the leaves in the finest

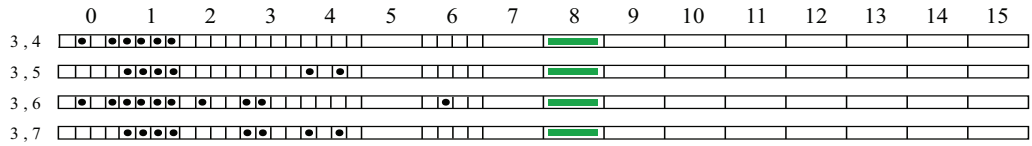


Figure 4.17: The separated form of the part of kernel matrix P as in Figure 4.16.

level, and our computation in fact amounts to computing $\boldsymbol{\phi}_j$ for $j = 0, 1, \ldots, 4^L - 1$ one after another.

Now let us consider how to compute $\boldsymbol{\phi}_j^{(3)}$ for $j = 4, 5, 6, 7$. Notice that for the full rank submatrices corresponding to j's near neighbors, we directly compute the products, which does not effect our fast algorithm on those low rank submatrices. Therefore, for the simplicity, we will ignore the results from near neighbors. Thus, we have

$$\boldsymbol{\phi}_j^{(3)} \approx U_j^{(3)} \left(\sum_{i \in j\text{'s I.L.}} F_{j,i}^{(3)} + R_{1,j}^{(2)} \sum_{k \in 1\text{'s I.L.}} F_{1,k}^{(2)} \right), \quad j = 4, 5, 6, 7, \tag{4.42}$$

where "I.L." stands for "interaction list".

It is interesting to generalize (4.42) for a quad tree with L levels. Let us focus on the node j of level L, whose parent's index, grandparent's index, and so on, are denoted by $p_{L-1}, p_{L-2}, \ldots, p_2$ respectively. We get the following important formula for $\boldsymbol{\phi}_j^{(L)}$:

$$\begin{aligned} \boldsymbol{\phi}_j^{(L)} \approx U_j^{(L)} \Bigg\{ & \sum_{i \in j\text{'s I.L.}} F_{j,i}^{(L)} + R_{p_{L-1},j}^{(L-1)} \Bigg[\sum_{k \in p_{L-1}\text{'s I.L.}} F_{p_{L-1},k}^{(L-1)} + R_{p_{L-2},p_{L-1}}^{(L-2)} \\ & \cdot \left(\sum_{m \in p_{L-2}\text{'s I.L.}} F_{p_{L-2},m}^{(L-2)} + \cdots + R_{p_3,p_4}^{(3)} \Bigg[\sum_{s \in p_3\text{'s I.L.}} F_{p_3,s}^{(3)} + R_{p_2,p_3}^{(2)} \sum_{t \in p_2\text{'s I.L.}} F_{p_2,t}^{(2)} \Bigg] \cdots \right) \Bigg] \Bigg\}. \end{aligned} \tag{4.43}$$

Let us define

$$\boldsymbol{\Psi}_j^{(2)} = \sum_{k \in j\text{'s I.L.}} F_{j,k}^{(2)}, \quad j = 0, 1, \ldots, 15; \tag{4.44}$$

$$\boldsymbol{\Psi}_j^{(\ell)} = \sum_{k \in j\text{'s I.L.}} F_{j,k}^{(\ell)} + R_{\lfloor j/4 \rfloor, j}^{(\ell-1)} \boldsymbol{\Psi}_{\lfloor j/4 \rfloor}^{(\ell-1)}, \quad j = 0, 1, \ldots, 4^\ell - 1, \ \ell = 3, 4, \ldots, L, \tag{4.45}$$

where $\lfloor j/4 \rfloor$ is in fact the index of j's parent in level $\ell - 1$; see Sec. 4.2.2.

Therefore, we can rewrite (4.43) as

$$\begin{cases} \boldsymbol{\phi}_j^{(L)} \approx U_j^{(L)} \boldsymbol{\Psi}_j^{(L)}, \quad j = 0, 1, \ldots, 4^L - 1; \\ \boldsymbol{\Psi}_j^{(\ell)} = \displaystyle\sum_{k \in j\text{'s I.L.}} F_{j,k}^{(\ell)} + R_{\lfloor j/4 \rfloor, j}^{(\ell-1)} \boldsymbol{\Psi}_{\lfloor j/4 \rfloor}^{(\ell-1)}, \quad j = 0, 1, \ldots, 4^\ell - 1, \ \ell = 3, 4, \ldots, L; \\ \boldsymbol{\Psi}_j^{(2)} = \displaystyle\sum_{k \in j\text{'s I.L.}} F_{j,k}^{(2)}, \quad j = 0, 1, \ldots, 15. \end{cases} \tag{4.46}$$

Now, we are ready to give a formal description of the algorithm. Notice that we use U, R and W as already computed, we will discuss about computing them after this algorithm.

4.3.3 Fast Matrix-Vector Multiplication Algorithm

Initialization

Given all of the information about U, R, and W. The quad tree structure is already contained in these matrices. Assume that the finest level is L. To compute $\boldsymbol{\phi} = P\boldsymbol{q}$, we will work on the each segment $\boldsymbol{\phi}_j^{(L)}$ for $j = 0, 1, \ldots, 4^L - 1$. In the algorithm, we do not need hierarchical segments of $\boldsymbol{\phi}$ and $\boldsymbol{q}$, which means that we only work on their finest segments, i.e., $\boldsymbol{\phi}_j$ and $\boldsymbol{q}_j$, for $j = 0, 1, \ldots, 4^L$.

Upward Pass

Step 1

Comment [Compute $G_j^{(L)}$ for all the nodes in the finest level L.]

do $j = 0, 1, \ldots, 4^L - 1$

Compute $G_j^{(L)} = U_j^{(L)^T} \boldsymbol{q}_j$.

enddo

Step 2

Comment [For the coarser levels, compute $G_k^{(\ell)}$.]

do $\ell = L-1, \ldots, 2$

do $k = 0, 1, \ldots, 4^\ell - 1$

Compute $G_k^{(\ell)} = \sum_{j \in k\text{'s Children}} R_{k,j}^{(\ell)^T} G_j^{(\ell+1)}$.

enddo

enddo

Downward Pass

Step 3

Comment[Compute $\boldsymbol{\Psi}_k^{(\ell)}$ for $\ell = 2, 3, \ldots, L$.]

do $\ell = 2, 3, \ldots, L$

do $k = 0, 1, \ldots, 4^\ell - 1$

First compute the needed $F_{k,j}^{(\ell)}$ from k's interaction list, using (4.41).

Then compute $\boldsymbol{\Psi}_k^{(\ell)}$ via (4.44) for $\ell = 2$ and (4.45) for $\ell > 2$.

enddo

enddo

Step 4

Comment [Compute $\boldsymbol{\phi}_j$ at the finest level L.]

do $j = 0, 1, \ldots, 4^L - 1$

Compute $\boldsymbol{\phi}_j = U_j^{(L)} \boldsymbol{\Psi}_j^{(L)}$.

enddo

Step 5

Comment [Update ϕ_j at the finest level L by adding the result from j's near neighbors (include j itself).]

do $j = 0, 1, \ldots, 4^L - 1$

For every j, there are several full rank submatrices generated by j and its near neighbors. Directly compute the multiplication and add the result to ϕ_j.

enddo

Let us investigate the complexity of this algorithm. The complexity of the algorithm does not depend on the distribution of the data points on a given domain. Hence, for the simplicity, we assume that there are N points uniformly distributed in a square domain and we are given a quad tree with the finest level L. Therefore, on the finest level, each node contains s points, where $N = 4^L \cdot s$. We also set a constant low rank p for the low rank submatrices. Therefore, the column size of $U_j^{(\ell)}$ for any ℓ will be p. The size of $R_{i,j}^{(\ell)}$ for any pair of parent i and child j will be $p \times p$. Of course, the size of any $W_{i,j}^{(\ell)}$ will be $p \times p$. In practice, the size of these matrices could be very different, but here we want it as simple as possible to analyze the complexity of the algorithm. Table. 4.3.3 gives the analysis of the complexity of the algorithm. Since we know $4^L = \dfrac{N}{s}$, the total computational cost is in order of

$$pN + p^2 \cdot \left(\frac{N}{s}\right) + 28p^2 \left(\frac{N}{s}\right) + pN + 9N$$

which will be $40pN$ when we set $p = s$. In practice, p could be much smaller than s. From this analysis of complexity, we can conclude that this algorithm is an $O(N)$ algorithm, which in fact is a matrix-vector-multiplication version of original FMM.

Table 4.1: A brief analysis of the complexity of the algorithm for matrix-vector multiplication.

Step	Cost	Explanation
1	$O(pN)$	Each $U_j^{(L)}$ has size of $p \times s$, $\boldsymbol{q}_j$ has length s
2	$O(p^2 \cdot 4^L)$	For level ℓ, the cost of computing $G_j^{(\ell)}$ for $j = 0, 1, \ldots, 4^\ell - 1$ is $O(p^2 \cdot 4^{\ell+1})$.
3	$O(28p^2 \cdot 4^L)$	In a particular level ℓ, for every node, there are at most 27 nodes in its interaction list, so the cost for all $F_{i,j}^{(\ell)}$ is $O(27p^2 \cdot 4^\ell)$; And additionaly the cost for the second part of (4.45) is $O(p^2 \cdot 4^\ell)$.
4	O(pN)	Each $\boldsymbol{\Psi}_j^{(L)}$ has length p.
5	$O(9N)$	At the finest level L, there are at most 9 near neighbors for each node including itself, so direct computation will cost $O(9s)$ for each node.

4.4 Eigenvalue/Eigenvector Computation

To compute selected eigenvectors and the corresponding eigenvalues of original kernel matrix P, the best software to use is ARPACK [19], the collection of Fortran77 subroutines designed to solve large scale eigenvalue problems. It is most appropriate for a large structured matrix A where structured means that a matrix-vector product $\boldsymbol{w} \leftarrow A\boldsymbol{v}$ requires $O(N)$ floating point operations rather than the usual $O(N^2)$ operations. Therefore, in our case, the ARPACK is the best choice since we can supply a fast routine to compute matrix-vector products. One can also use a MATLAB routine "eigs", which calls the ARPACK subroutines in the background.

Recall that our fast matrix-vector multiplication algorithm is derived from a hierarchical matrix decomposition method. It is also important, however, to develop a fast algorithm to produce the decomposition of the original kernel matrix.

4.4.1 Preprocessing of the Eigenvalue Computation

The task of the preprocessing is to compute the matrices U, R, and W using the low rank submatrices of P. As discussed before, for any pair of well separated nodes i and j on level ℓ, the

corresponding submatrix can be approximated by

$$B_{i,j}^{(\ell)} \approx U_i^{(\ell)} W_{i,j}^{(\ell)} U_j^{(\ell)^T}.$$

For example, let us consider the node 4 of level 3. We are interested in computing $U_4^{(3)}$, which is the column basis of both the red blocks and green blocks in Fig. 4.18. So the naive way of computing $U_4^{(3)}$ is to use all of these blocks.

Remark 4.4.1. We do not directly apply the formula in (4.26) for $U_4^{(3)}$ for several reasons. One reason is that the high order polynomials should be avoided in numerical computation. Another reason is that the value p is hard to predict based on the error analysis in original FMM (see Sec. 4.1), which is also the reason why we investigate the matrix representation of the FMM.

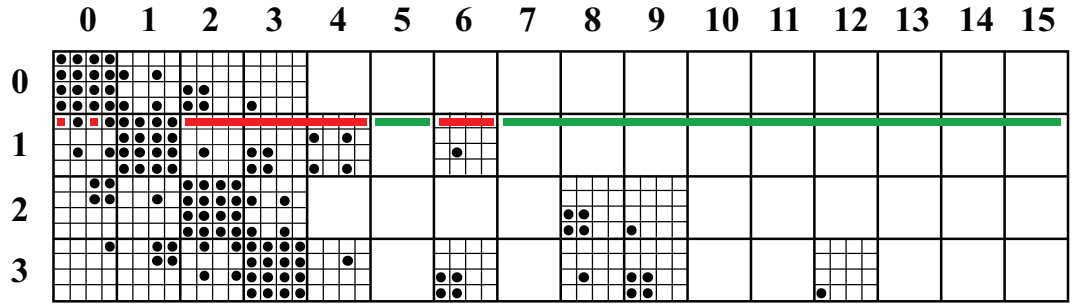


Figure 4.18: The separated form of the part of kernel matrix P as in Figure 4.16.

Thanks to the hierarchical property of the column basis discussed in Sec. 4.2.4, we know that only using the red blocks is already enough to compute $U_4^{(3)}$. Therefore, we can apply QR factorization on the matrix consist of the red blocks. Let us denote this particular matrix by C, with size $m \times n$, where $m < n$. We know that the computational cost of ordinary QR of C is $O(m^2 n)$. There are several advanced algorithms for computing $U_4^{(3)}$ from C in a fast manner, such as pivoted Gram-Schmidt with reorthogonalization (PGSR) [11, 3], and randomized approximation method (RAM) [21]). The computational cost of the PGSR is $O(pmn)$, and of the RAM is about $\log(p)mn$, where p is the desired approximated rank (see Sec. 4.2.5). For both of them, when n is large, the computation is still expensive.

In our case, we do not have to use the matrix C to compute $U_4^{(3)}$. We can use a smaller matrix, say, $T \in \mathbb{R}^{m\times h}$ with $p \leq h \ll n$. It is straightforward that there exists a submatrix of C that can be used to compute $U_4^{(3)}$. RAM [21] provide a delicate approach to find such submatrix from a general low rank matrix, which will not be applied in our task. In fact, because of our special kernel function, we can simply obtain T that is not a submatrix of C, but meanwhile it can be used to compute the column basis of C. Our method is to use the data points in the node 4 of level 3 and a particular set of data points. We will explain how to obtain this particular set of data points as follows.

The matrix C is constructed by m data points in node 4 of level 3 and n data points from the nodes of the node 4's interaction list. Therefore, the information of the column basis of C is carried by the the nodes of the node 4's interaction list. Therefore, the particular set of data points for constructing matrix T must be chosen from the nodes of the node 4's interaction list.

Thanks to Theorem 4.1.2, which we rephrase as follows: given a set of points $S = \{z_1, z_2, \ldots, z_m\}$ located within a disk D, with radius r. The potential at a point z outside of disk due to S can be written as

$$\phi(z) = Q\log(z) + \sum_{k=1}^{\infty} \frac{a_k}{z^k},$$

where the multipole expansion coefficients $a_k = \sum_{i=1}^{m} \frac{-q_i z_i^k}{k}$ only depend on the set S. The accuracy of the truncated p terms of multipole expansion is

$$\left| \phi(z) - Q\log(z) - \sum_{k=1}^{p} \frac{a_k}{z^k} \right| \leq \left(\frac{A}{p+1}\right)\left(\frac{1}{c-1}\right)\left(\frac{1}{c}\right)^p, \tag{4.47}$$

where $c = \left|\frac{z}{r}\right|$, and A is a constant. It is obvious that p and c control the accuracy of the multipole expansion. For the same p, the approximated potentials on the far away points, i.e., those for large c's, will be more accurate than the ones on the points with smaller c's.

Therefore, when choosing the data points for T from the nodes of node 4's interaction list, the points not far away, i.e., the points with smaller c's, are more important than the far away ones. For example, in Fig. 4.19, we have the red squares denote for the interaction list of node 4

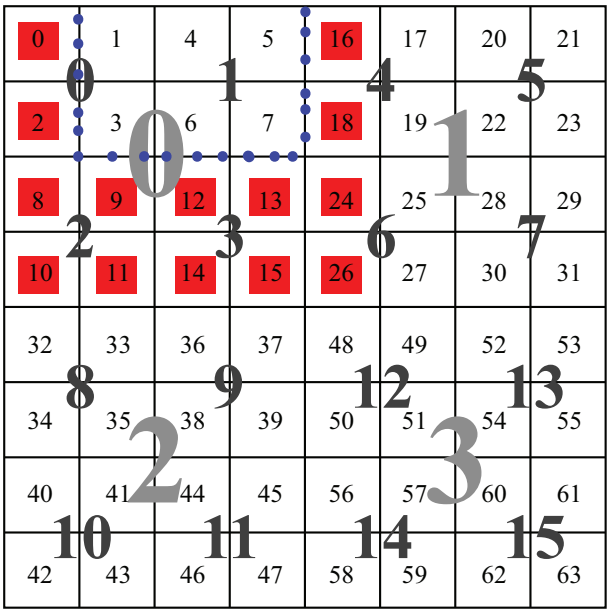


Figure 4.19: The index matrix for the quad tree with 4 levels. The red squares are the interaction list of node 4 of level 3. The blue dots are called the reference points for node 4 of level 3, which are sampled on the inner boundary of the interaction list of node 4 of level 3.

of level 3. In the matrix form of Fig. 4.18, these red squares correspond to the red submatrices. We can see that these red squares have different distances from node 4. The squares of $j = 0, 2, 8, 9, 12, 13, 16, 18, 24$ are nearer than the squares of $j = 10, 11, 14, 15, 26$. In particular, those blue dots on the boundaries of some of the red squares are the points as near as possible, which still belong to the node 4's interaction list.

We will refer to the boundary segments of the nodes in the interaction list that are closest to the node of the interest as the *inner boundary of interaction list*. We will choose the sampling points on the inner boundary of interaction list as the particular data points for constructing matrix T. We will call these sampling points as the *reference points*.

These reference points can be simply chosen as those on a regular grid along the inner boundary of the interaction list. Assume $F = \{r_1, r_2, \ldots, r_h\}$ is the set of reference points for node 4 of level 3, and $S = \{z_1, z_2, \ldots, z_m\}$ is the set of data points in node 4 of level 3. Therefore, we

will have a particular matrix of size $m \times h$:

$$T_4^{(3)} = \begin{pmatrix} \log|z_1 - r_1| & \log|z_1 - r_2| & \cdots & \log|z_1 - r_h| \\ \log|z_2 - r_1| & \log|z_2 - r_2| & \cdots & \log|z_2 - r_h| \\ \vdots & \vdots & \vdots & \vdots \\ \log|z_m - r_1| & \log|z_m - r_2| & \cdots & \log|z_m - r_h| \end{pmatrix}. \tag{4.48}$$

To decide the value of h, we must predict p at first. In (4.47), if the user specifies the upper bound of the error to be $\varepsilon > 0$, we can approximately have

$$\left(\frac{1}{c}\right)^p < \varepsilon,$$

where $c \approx 2$ for the separated nodes in the interaction list. Therefore, for $\varepsilon \geq 10^{-12}$, we can roughly have $p < 40$. Empirically, we fix the value of h slightly greater than 40. One important fact is that the constant value of h is not related to the level of the quad tree.

Now we are ready to describe the preprocessing algorithm for eigenvalue computation. Assume that the finest level is level L.

Step 1. Compute $U_j^{(L)}$, $j = 0, 1, 2, \ldots, 4^L - 1$.

For each node j, construct the particular matrix $T_j^{(3)}$ as in (4.48) and obtain $U_j^{(L)}$ by applying truncated SVD on $T_j^{(3)}$. The column size of $U_j^{(L)}$ is p.

Step 2. Compute $W_{i,j}^{(L)}$, $i, j = 0, 1, \ldots, 4^L - 1$, i belongs to j's interaction list.

Since we have

$$B_{i,j}^{(L)} = U_i^{(L)} W_{i,j}^{(L)} U_j^{(L)^T},$$

where $U_\cdot^{(L)}$ have orthonormal columns. So in order to obtain $W_{i,j}^{(L)}$, we do the multiplications as follows

$$W_{i,j}^{(L)} = U_i^{(L)^T} B_{i,j}^{(L)} U_j^{(L)}.$$

Step 3. Compute $R_j^{(\ell)}$, $\ell = L-1, L-2, \ldots, 2$, $j = 0, 1, \ldots, 4^\ell - 1$.

Recall the hierarchical property of $U_j^{(\ell)}$. Consider the example used in Sec. 4.3.2, we have

$$B_{1,8}^{(2)} \approx \begin{pmatrix} U_4^{(3)} & & & \\ & U_5^{(3)} & & \\ & & U_6^{(3)} & \\ & & & U_7^{(3)} \end{pmatrix} \cdot R_1^{(2)} W_{1,8}^{(2)} R_8^{(2)^T} \begin{pmatrix} U_{32}^{(3)^T} & & & \\ & U_{33}^{(3)^T} & & \\ & & U_{34}^{(3)^T} & \\ & & & U_{35}^{(3)^T} \end{pmatrix}.$$

So, for example, to compute $R_1^{(2)}$, we first construct the particular matrix $T_1^{(2)}$ by using h reference points for node 1 of level 2. We know that $T_1^{(2)}$ can be written as

$$T_1^{(2)} \approx \begin{pmatrix} U_4^{(3)} & & & \\ & U_5^{(3)} & & \\ & & U_6^{(3)} & \\ & & & U_7^{(3)} \end{pmatrix} R_1^{(2)} M,$$

where M is some matrix, which is not important. So, we can multiply out those $U_j^{(3)}$ by using their orthonormal property, and then apply the TSVD on the result to get $R_1^{(2)}$, which has orthonormal columns.

Step 4. Compute $W_{i,j}^{(\ell)}$, $i, j = 0, 1, \ldots, 4^\ell - 1$, i and j are well separated (i.e., i belongs to j's interaction list).

After computing U's and R's, we can easily get $W_{i,j}^{(\ell)}$ by matrix multiplication in a similar way as Step 2.

We supply a brief analysis of the complexity of these steps below. As before, we assume that the data is uniformly distributed and the finest level is level L. The total number of data points is

N. Each node of level L has s data points. So, we have $N = 4^L \cdot s$.

Step	Cost	Explanation
1	$O(N)$	Each $U_j^{(L)}$ needs $O(h^2 \cdot s)$.
$2+4$	$O(pN^2)$	The matrix multiplication is the price of this algorithm. The constant hidden by the big-O is very small.
3	$O(N)$	At a particular level k, the main cost to compute $R_j^{(k)}$ for $j = 0, 1, \ldots, 4^k - 1$ is the multiplication, which costs $O(phN)$ for each level.

4.5 Numerical Experiments

4.5.1 Laplacian Eigenvalues on the Unit Disk in $\mathbb{R}^2$

Recall the 2D example discussed in Sec. 3.3.2. Consider the unit disk $\Omega \subset \mathbb{R}^2$, where the kernel function is of the form $K(\boldsymbol{x}, \boldsymbol{y}) = -\frac{1}{2\pi} \log \|\boldsymbol{x} - \boldsymbol{y}\|$ for $\boldsymbol{x}, \boldsymbol{y} \in \overline{\Omega}$. In this experiment, we used a regular grid on the square domain covering the disk domain Ω, as shown in Fig. 4.20.

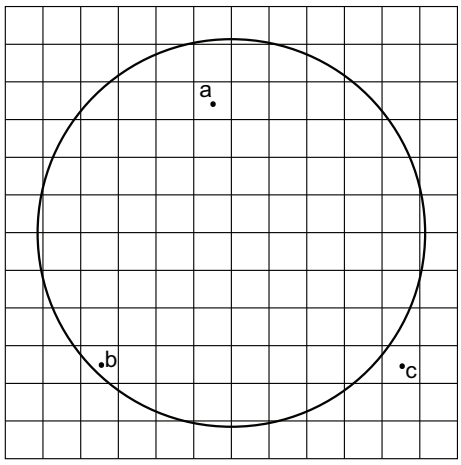


Figure 4.20: A regular grid is applied on the square domain covering the unit disk. The points a, b, and c are the centers of the three discretization cells.

As explained in Sec. 3.2.2, our sampling data points on the unit disk consist of a subset of the centers of all of the cells in the regular grid. This subset is required to contain the centers

that are located inside of the unit disk. For example, in Fig. 4.20, the centers a and b belong to our sampling data points, but not the center c. In the experiment, we fixed the location and size of the square domain, and let the density of the regular grid change. By doing so, we obtained the sampling data points on the unit disk equipped with different cell size, i.e., $(\Delta h)^2$, where Δh is the side length of each cell box in the regular grid. Assume that our sampling data points are $\{\boldsymbol{x}_i\}$, $i = 1, \ldots, N$. Then we approximated the integral eigenvalue problem $\mathcal{K}\phi = \mu\phi$, where $\mathcal{K}$ is defined as (3.1), with a simple quadrature rule as follows:

$$(\Delta h)^2 \sum_{i=1}^{N} K(\boldsymbol{x}_i, \boldsymbol{x}_j)\phi(\boldsymbol{x}_j) = \mu\phi(\boldsymbol{x}_j), \quad j = 1, \ldots, N. \tag{4.49}$$

Let $K_{i,j} \stackrel{\Delta}{=} K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Our first task was to compute the largest 5 eigenvalues of K using our fast algorithm. Our second task was to compare the computed Laplacian eigenvalues with their true values. In fact, the computed λ's have the form of

$$\tilde{\lambda} = \frac{1}{(\Delta h)^2 \cdot \mu};$$

see the proof of Corollary 3.3.2 for the details. Also, by the proof of Corollary 3.3.2, the true values of λ's are given by

$$\lambda_{m,n} = \begin{cases} \beta_{m-1,n}^2 & \text{if } m = 1, \ldots, \ n = 1, 2, \ldots, \\ \beta_{0,n}^2 & \text{if } m = 0, \ n = 1, 2, \ldots. \end{cases}, \tag{4.50}$$

where $\beta_{k,\ell}$ is the ℓth zero of the Bessel function of the first kind of order k, whose values are tabulated in many places, e.g., [1, Table 9.5]. The values of the lowest five frequencies, i.e., the smallest five λ's, are listed in below:

$$\lambda_0 = 5.7831, \ \ \lambda_1 = 5.7831, \ \ \lambda_2 = 5.7831, \ \ \lambda_3 = 14.6819, \ \ \lambda_4 = 14.6819.$$

We did four experiments base on four different values of Δh. In each experiment, we com-

puted the relative error of the eigenvalues, i.e., $e_i = \frac{|\tilde{\lambda}-\lambda|}{\lambda}$ for $i = 1, \ldots, 5$, and recorded the computational times. Our algorithm contained two steps: the first step is the preprocessing of the eigenvalue computation; the second step is the computation of the eigenvalues using "eigs"(MATLAB) by supplying our fast matrix-vector multiplication routine. The total computational time T_{total} (in seconds) and the time spent by "eigs" T_{eigs} (in seconds) were recorded. The difference $T_{\text{total}} - T_{\text{eigs}}$ is in fact the time spent by the preprocessing.

We also examined the accuracy of our algorithm for the matrix-vector multiplication during each experiment. We chose an arbitrary unit vector $\boldsymbol{v} \in \mathbb{R}^N$ and computed the approximation of $K\boldsymbol{v}$, denoted by $\tilde{K}\boldsymbol{v}$. The L^2 error $\text{MVE} = \|K\boldsymbol{v} - \tilde{K}\boldsymbol{v}\|_2$ and the computational time T_{MVP} (in seconds) of $\tilde{K}\boldsymbol{v}$ were recorded. In these experiments, we fixed a threshold $\varepsilon = 1.0\text{e} - 12$ for the truncated SVD part of our algorithm, where $O(\varepsilon)$ is exactly the upper bound of MVE. We display the results discussed above in Table. 4.5.1 and Table. 4.5.1 .

Table 4.2: The experiments of computing the smallest five eigenvalues of the Laplacian defined on a unit disk.

Δh	N	e_1	e_2	e_3	e_4	e_5
0.1	316	$2.77\text{e}-02$	$2.77\text{e}-02$	$3.44\text{e}-02$	$8.22\text{e}-02$	$8.56\text{e}-02$
0.05	1264	$4.01\text{e}-03$	$4.01\text{e}-03$	$1.00\text{e}-02$	$1.70\text{e}-02$	$2.20\text{e}-02$
0.01	31428	$1.81\text{e}-04$	$1.81\text{e}-04$	$5.66\text{e}-04$	$9.21\text{e}-04$	$1.08\text{e}-03$
0.005	125676	$7.52\text{e}-05$	$7.52\text{e}-05$	$1.73\text{e}-04$	$1.59\text{e}-04$	$4.22\text{e}-04$

Table 4.3: The accuracy and computational cost of our algorithm of matrix-vector multiplication.

Δh	N	MVE	T_{MVP}	T_{eigs}	T_{total}
0.1	316	$1.0561\text{e}-14$	$9.0\text{e}-02$	1.5	1.9
0.05	1264	$8.4889\text{e}-13$	$1.2\text{e}-01$	2.4	2.9
0.01	31428	$1.3333\text{e}-11$	$3.7\text{e}-00$	112.2	210.2
0.005	125676	$6.5331\text{e}-11$	$6.3\text{e}-00$	199.2	1999.7

From these tables, we can observe that our algorithm of matrix-vector multiplication is fast and accurate. But the relative error for the computed eigenvalues (compared with their analytic values) is not small. The reason is that our experiments were not designed to compute these

analytic eigenvalues, since at least the boundary conditions, e.g., (3.17), are not explicitly imposed in our computation. The effect of the boundary conditions on the eigenvalue computation will be investigated during our near future research.

4.5.2 An Example of Complicated and Large Domain: Japanese Islands

In previous sections, we prescribed our algorithm based on an ideal assumption that a square domain and uniformly distributed data points are given. But in practice, we often face to either a domain with complicate shape or arbitrarily distributed data points. In fact, our algorithm can be easily modified to handle these situations.

Recently we obtained a digital map called "Japan Engineering Geomorphologic Classification Map" (JEGM) [31]. The number of sampling points in this map is 387,924 over the Japanese Islands. Each point is associated with a vector of length 11 representing a type of geological layer, an elevation, a slope, etc. In other words, this is a vector-valued dataset. The coordinate of each point is specified by four values because each point here approximately represents a square region of 1km $\times$ 1km. These four values are the longitudes and the latitudes of South West corner and North East corner of each square. The corresponding kernel matrix therefore would be $387,924 \times 387,924$, which is just too huge to handle with usual eigenvalue solvers.

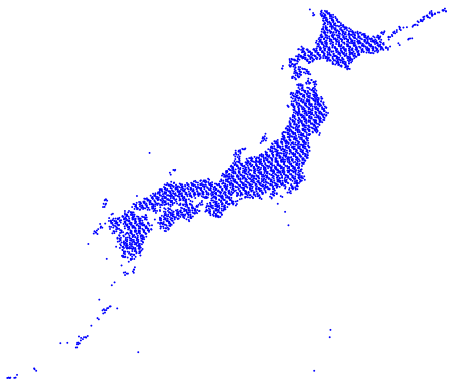


Figure 4.21: Real Challenge: The number of sampling points in this map is 387,394 over the Japanese Islands.

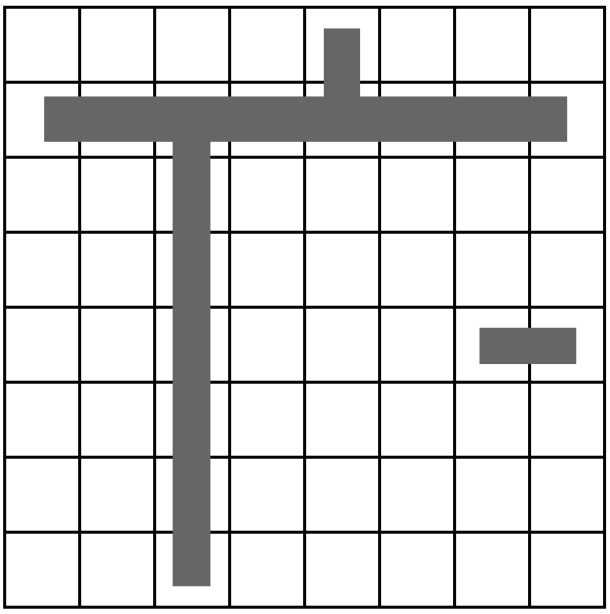


Figure 4.22: A simple domain (the dark region) with a quad tree with $L = 3$.

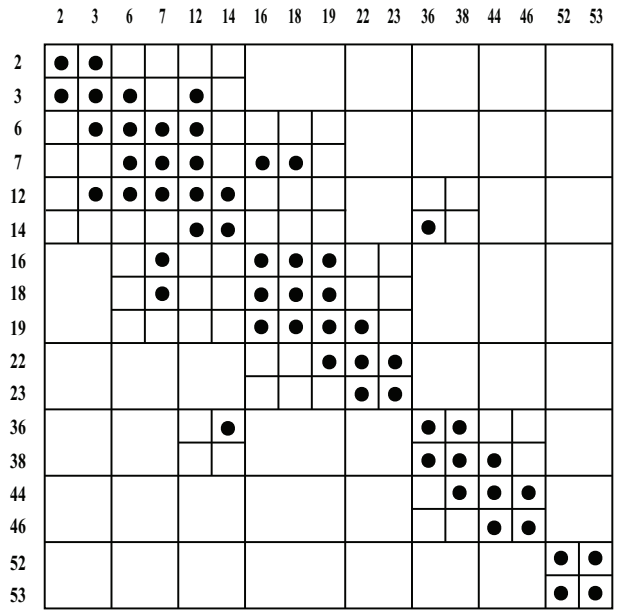


Figure 4.23: Split the kernel matrix by assigning dots to the submatrices corresponding to the near neighbors and blank boxes to the submatrices corresponding to the interaction list.

We can easily get the center of each square by knowing the South West corner and North East corner. We plot these 387,924 centers in Fig. 4.21. Obviously, the shape of the domain is very

complicated. As usual, we apply a quad tree to the domain, then throw away the nodes which contains no data points, and construct the kernel matrix using the rest of the nodes. The matrix splitting scheme is not as straightforward as the example used before in Sec. 4.2.4, since we do not have a complete tree now.

Let us use a simple example to explain the matrix splitting scheme for the incomplete quad tree structure. In Fig. 4.22, we apply a quad tree ($L = 3$) on a domain (the dark region). To construct the kernel matrix, only the nodes that contain the data points are used. By assigning the dots to the near neighbors and blank boxes for the low rank submatrices, we obtain the splitted kernel matrix shown in Fig. 4.23. The **basic rule** of our splitting scheme is that any of the submatrices generated by this scheme should correspond to either two near neighbor nodes or two nodes that belong to each other's interaction list. For example, for the submatrix $B_{0,4}^{(2)}$ (see Sec. 4.3 for the details about the notations) that corresponds to the nodes $\{2, 3\}$ and the nodes $\{16, 18, 19\}$ in level 3, we assign a blank box to it because the node 4 of level 2, the parent node of $\{16, 18, 19\}$, belongs to the interaction list of node 0 of level 2, the parent node of $\{2, 3\}$.

The fast algorithm we developed in the previous sections still work for this splitted kernel matrix, and the hierarchical property of the column basis explained in Sec. 4.3.2 still exists. Let us denote the column basis of $B_{0,4}^{(2)}$ as $U_0^{(2)}$. Then the formula (4.36) implies a particular form

$$U_0^{(2)} = \begin{pmatrix} U_2^{(3)} & \\ & U_3^{(3)} \end{pmatrix} R_0^{(2)} = \begin{pmatrix} U_2^{(3)} R_{0,2}^{(2)} \\ U_3^{(3)} R_{0,3}^{(2)} \end{pmatrix}, \tag{4.51}$$

Another interesting aspect of this simple domain is the non-connected component corresponding to the nodes $\{52, 53\}$ of level 3, whose interaction lists do not contain any sampling data points. Therefore, there is no need to compute the column basis $U_{52}^{(3)}$ or $U_{53}^{(3)}$. For the parent level, we compute the column basis $U_{13}^{(2)}$ using the directly constructed matrix $T_{13}^{(2)}$ (see Sec. 4.4), not the hierarchical transformation as in 4.51.

From the above analysis, we designed a robust matrix-splitting scheme to handle a non-complete quad tree. Then, we computed the eigenvectors corresponding to the lowest nine frequencies (i.e., the largest nine eigenvalues of the kernel matrix) and displayed them in Fig. 4.24.

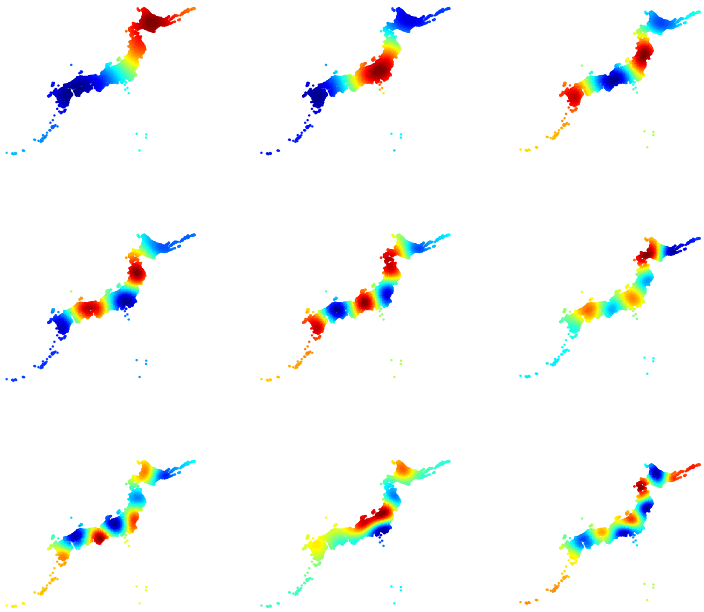


Figure 4.24: The first 9 Laplacian eigenfunctions over the islands of Japan.

We sub-sampled the data points for display purpose in this figure because of the large number of the original data points.

Chapter 5

Conclusion

5.1 Summary

In this thesis, we have designed a fast and accurate algorithm for computing the eigenvalues and eigenvectors of a special kernel matrix $K \in \mathbb{R}^{N\times N}$ associated with the kernel function

$$K(\boldsymbol{x},\boldsymbol{y}) = -\frac{1}{2\pi}\log\|\boldsymbol{x}-\boldsymbol{y}\|, \qquad \boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^2, \tag{5.1}$$

which was proposed by Saito as the kernel of a particular integral operator that commutes and shares the eigenfunctions with the Laplace operator [25], [26].

The core of our algorithm is to apply the famous eigenvalue computation algorithm "implicitly re-started Lanczos method" by providing a fast algorithm of the matrix-vector multiplication, $K\boldsymbol{v}$ for arbitrary $v \in \mathbb{R}^N$. The function $-\log\|\boldsymbol{x}-\boldsymbol{y}\|$ in (5.1) can be considered as the potential at point $\boldsymbol{x}$ due to a charge of unit strength located at point $\boldsymbol{y}$, which allows us to apply the celebrated fast multipole method (FMM, [24], [14], [13]). We have described a locally singular value decomposition after exploring the low rank properties of the submatrices of K. Thanks to the hierarchical semi-separable representation (HSS, [4]), we could further investigate the hierarchical structure of K and design a hierarchical matrix decomposition that can speed the matrix-vector multiplication considerably.

5.2 Future Studies

5.2.1 Improvement the Computational Cost

In Chapter 4, we discussed the computational cost of our algorithm, which is in the order of N^2, where N is the size of the dataset. The natural next step is to speed up this algorithm further to the order of N^α, where $1 \leq \alpha < 2$. Our fast matrix-vector multiplication is based on the hierarchical matrix decomposition and the major cost of the decomposition comes from the fact that we need to access every elements in the original kernel matrix of size $N \times N$, which implies the order of N^2 operations. Therefore, a possible way of improving the algorithm is to avoid accessing all the elements in the kernel matrix by introducing a further approximation when doing the matrix decomposition. The recent work of Martinsson, Rokhlin, and Tygert on a randomized algorithm to approximate matrices [21] could be a helpful source for our further investigation.

5.2.2 Investigate the Boundary Conditions

Our algorithm is in fact tailored for the highly structured kernel matrix constructed by the sampling points on the underlying domain. The sampling points are not required to be on the regular grids because in practice we are often given arbitrarily distributed points or we need to choose the sampling points from the domain. Also, the boundary condition (see Sec. 3.2.1) are not explicitly imposed in our computation, compared with the conventional Laplacian eigenfunctions satisfying either the Dirichlet or the Neumann boundary condition. Therefore, to better understand the behavior of the eigenvectors computed by our algorithm is important for their further application. Since these conventional Laplacian eigenfunctions are used in numerous application and yet difficult to be computed on a complicated domain, it is also interesting to investigate whether we can synthesize the conventional Laplacian eigenfunctions using Saito's eigenfunctions and to compare the mathematical and numerical properties of these two kinds of eigenfunctions.

5.2.3 Implement the Eigenvalue Problem in 3D

In this thesis, our algorithm is designed for the particular 2D kernel function (5.1). For the eigenvalue problem in 3D with the kernel function defined as

$$K(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{4\pi\|\boldsymbol{x} - \boldsymbol{y}\|}, \qquad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^3.$$

To implement the computation, 3D FMM [28] is a natural candidate. 3D FMM, however, is not easy to implement due to its complicated data structure involving an octree. Therefore, we should first look at simpler approaches such as the "best matrix decomposition" of Bremer [16] or the randomized algorithm to compute low rank approximation of the kernel matrix [21], [33].

Bibliography

[1] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions*. Dover Publications, Inc., New York, 1972. 9th printing.

[2] R. K. Beatson and L. Greengard. A short course on fast multipole methods (lectures), 1997.

[3] P. A. Businger and G. H. Golub. Linear least squares solutions by Householder transformations. *Numer. Math.*, 7:269–276, 1965.

[4] S. Chandrasekaran, M. Gu, and T. Pals. A fast ULV decomposition solver for hierarchically semiseparable representations. *SIAM J. Matrix Anal. Appl.*, 28(3):603–622, 2006.

[5] I. Daubechies. *Ten Lectures on Wavelets*, volume 61 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, PA, 1992.

[6] E. B. Davies. *Spectral Theory and Differential Operators*, volume 42 of *Cambridge Studies in Advanced Mathematics*. Cambridge Univ. Press, 1995.

[7] L. C. Evans. *Partial Differential Equations*. AMS, 1998.

[8] G. B. Folland. *Fourier Analysis and Its Applications*. Brooks/Cole Publishing Co., 1992.

[9] G. B. Folland. *Introduction to Partial Differential Equations*. Princeton Univ. Press, 2nd edition, 1995.

[10] B. Friedman. *Principles and Techniques of Applied Mathematics*. John Wiley & Sons, Inc., New York, 1990. Republished by Dover Publications, Inc. in 1990.

[11] G. H. Golub. Numerical methods for solving linear least squares problems. *Numer. Math.*, 7:206–216, 1965.

[12] I. S. Gradshteyn and I. M. Ryzhik. *Table of Integrals, Series, and Products*. Academic Press, 6th edition, 2000.

[13] L. Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems*. MIT Press, Cambridge, MA, 1988.

[14] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73:325–348, 1987.

[15] F. John. *Partial Differential Equations*. Number 1 in Applied Mathematical Sciences. Springer-Verlag, New York, 4th edition, 1982.

[16] J. C. Bremer Jr. *Adaptive Multiscale Analysis of Graphs and Applications*. PhD thesis, Dept. Math., Yale University, May, 2007.

[17] M. A. Khabou, L. Hermi, and M. B. H. Rhouma. Shape recognition using eigenvalues of the dirichlet laplacian. *Pattern Recognition*, 40:141–153, 2007.

[18] E. Kreyszig. *Introductory Functional Analysis with Applications*. Wiley Classics Library. John Wiley & Sons, Inc., New York, 1989.

[19] R. Lehoucq, D. Sorensen, and C. Yang. *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM, Philadelphia, PA, 1997.

[20] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM Publications, Philadelphia, 1998.

[21] P. G. Martinsson, V. Rokhlin, and M. Tygert. A randomized algorithm for the approximation of matrices. Technical Report 1361, Yale University, 2006.

[22] D. Porter and D. S. G. Stirling. *Integral Equations: A Practical Treatment from Spectral Theory to Applications*. Cambridge Univ. Press, New York, 1990.

[23] M. Reuter, F. E. Wolter, and N. Peinecke. Laplace-beltrami spectra as shape-dna of surfaces and solids. *Computer-Aided Design*, 38:342–366, 2006.

[24] V. Rokhlin. Rapid solution of integral equations of classical potential theory. *J. Comput. Phys.*, 60:187–207, 1985.

[25] N. Saito. Geometric harmonics as a statistical image processing tool for images defined on irregularly-shaped domains. In *Proc. 13th IEEE Workshop on Statistical Signal Processing*, pages 425–430. IEEE, 2005.

[26] N. Saito. Data analysis and representation on a general domain via eigenfunctions of Laplacian. *Applied and Computational Harmonic Analysis*, 2007. To appear.

[27] N. Saito, K. Yamatani, and J. Zhao. Generalized polyharmonic trigonometric transform: A tool for object-oriented image analysis and synthesis. Technical report, Dept. Math., Univ. California, Davis, 2007. In preparation.

[28] K. E. Schmidt and Michael A. Lee. Implementing the fast multipole method in three dimensions. *J. Stat. Phys.*, 63:1223–1235, 1991.

[29] W. A. Strauss. *Partial Differential Equations: An Introduction*. Brooks/Cole Publishing Co., 1992.

[30] L. N. Trefethen and D. Bau, III. *Numerical Linear Algebra*. SIAM, Philadelphia, 1997.

[31] K. Wakamatsu, S. Kubo, M. Matsuoka, K. Hasegawa, and M. Sugiura. *Japan Engineering Geomorphologic Classification Map*. University of Tokyo Press, 2005.

[32] H. Wendland. *Scattered Data Approximation*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge Univ. Press, 2005.

[33] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert. A fast randomized algorithm for the approximation of matrices. Technical Report 1386, Yale University, 2007.

[34] R. Young. *An Introduction to Hilbert Space*. Cambridge Univ. Press, 1988.

[35] J. Zhao. *Efficient Approximations: Overcoming Boundary Effects*. PhD thesis, Dept. Math., Univ. California, Davis, 2006.