

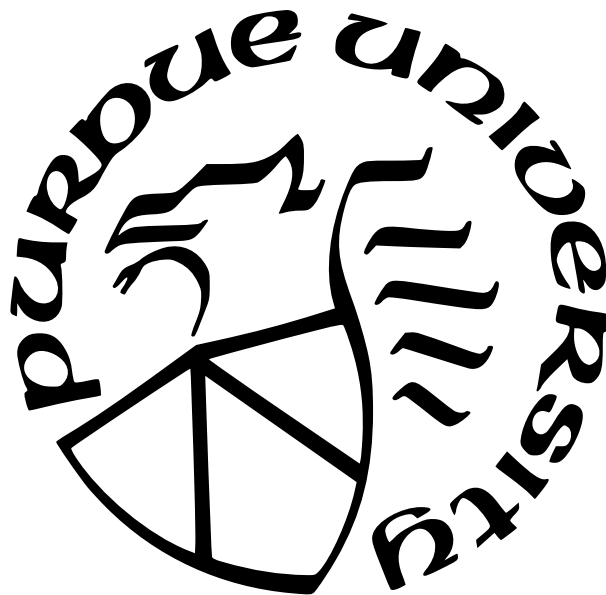
**STABLE AND EFFICIENT METHODS FOR STRUCTURED
MATRIX COMPUTATIONS**

by
Xiaofeng Ou

A Dissertation

*Submitted to the Faculty of Purdue University
In Partial Fulfillment of the Requirements for the degree of*

Doctor of Philosophy



Department of Mathematics

West Lafayette, Indiana

May 2022

**THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL**

Dr. Jianlin Xia, Chair
Department of Mathematics

Dr. Jie Shen
Department of Mathematics

Dr. Peijun Li
Department of Mathematics

Dr. Dan Jiao
School of Electrical and Computer Engineering

Approved by:
Dr. Plamen Stefanov

To my family

ACKNOWLEDGMENTS

My deepest gratitude goes to my academic supervisor Professor Jianlin Xia. He is the most supportive and considerate advisor I have ever known. I receive lots of help and cares from him, not only on researches but also on all the small things in life and graduate school. He shows lots of trusts and supports to give me the chances to attend many conferences, especially in my first year when I only had some very elementary research progresses. There were many times when my researches were stagnant, or I did not have the motivations, but he never push me on that and always shows lots of patience and encouragements. I learn so much from him, and I will never forget that.

I would also like to show my gratitude to the committee members for their valuable advises and guidance on my researches. They give me the motivations to explore more possible research area. I would also like to thank Professor Balakrishnan, Professor de Hoop and all the professors I met for their guidance and supports.

My appreciation goes to our former and current research group members, Zixing Xin, Xin Ye, Xiao Liu, Jimmy Vogel, Chenyang Cao, Mikhail Lepilov, Tong Ding. The process of sharing and learning with them has always been enjoyable and meaningful.

My gratitude goes to my friends at Purdue, Fukeng Huang, Xiaodong Huang, Heng Du, Zhuoran Qiu, Hengrong Du, Yubo Wang, Zhiguo Yang, Xiaokai Yuan, Jianliang Li, Can Zhang etc. I received lots of help from them during my stay at Purdue. It is my fortune to have them accompanied through this journey.

My appreciation goes to my friends in China, Qihui Feng, Xuejun Zhong, Yipei Chen, Yang Chen, Haoli Sun, Yu Sun, Ziyue Gao, Quanjun Lang, Jiawei Deng, Yaocheng Lu, Zihua Ye, Junhua Luo, Guangjun Luo. It is my fortune to have them behind my back.

My gratitude goes to my training coach Chao Xu, and my training partners Kaiqiang Liu, Yuxi Chen, as well as all the people I met in Corec. They motivate me to train harder and smarter.

My deepest love goes to my family. I am blessed to be born in a big family with love and cares. They are always so supportive and considerate. They give me the power and courage to overcome any difficulties.

Time flies. It has been five years since the moment I stepped out the plane landing at Chicago. This Ph.D career is definitely not an easy journey, but it must be one of the most memorable time of my life.

TABLE OF CONTENTS

LIST OF TABLES	9
LIST OF FIGURES	10
ABSTRACT	11
1 INTRODUCTION	12
2 STABLE MATRIX VERSION OF THE FAST MULTIPOLE METHOD IN 2D	16
2.1 Background	16
2.2 Mathematics apparatus	18
2.2.1 Binomial series and the truncation error	19
2.2.2 Some basic properties of Bessel functions	19
2.2.3 Graf's addition formula and the truncation error	23
2.3 Stable kernel expansion and low-rank approximation	25
2.3.1 Generalized Cauchy kernel	26
2.3.2 Poisson kernel	31
2.3.3 Helmholtz kernel	35
2.4 Stable translation relation	44
2.4.1 Stable translation relation for Cauchy and Poisson kernel	45
2.4.2 Stable translation for Helmholtz kernel	48
2.5 Matrix version of the fast multipole method	50
2.5.1 Hierarchical partitioning and interaction list	51
2.5.2 Level-wise low rank approximation	53
2.5.3 Translation relation and nested basis	53
2.6 Backward stability of FMM	55
3 SUPERDC: SUPERFAST DIVIDE-AND-CONQUER RANK-STRUCTURED MA- TRIX EIGENVALUE DECOMPOSITION WITH IMPROVED STABILITY	66
3.1 Introduction	67

3.2	Review of the basic superfast divide-and-conquer eigensolver	70
3.2.1	Dividing stage	72
3.2.2	Conquering stage	73
3.3	Improved structured dividing strategy	75
3.4	Improved structured conquering stage	81
3.4.1	Deflation	81
3.4.2	Fast secular equation solution	82
3.4.3	Local shifting in triangular FMM for shifted secular equation solution	88
3.4.4	Structured eigenvectors via FMM with local shifting	93
3.4.5	Overall eigendecomposition and structure of the eigenmatrix	95
3.5	Numerical experiments	98
3.6	Pseudocodes and algorithms	110
3.7	Some improvements on implementations of SuperDC	111
3.7.1	Rank-revealing factorization in dividing stage	111
3.7.2	Precomputations in triangular FMM	116
3.7.3	Improved convergence criteria	118
3.8	Generalization to SVD solver	119
3.9	Conclusions	127
4	SUPERFAST FACTORIZATION UPDATE FOR DIAGONALLY-SHIFTED SPARSE DISCRETIZED MATRICES	129
4.1	Introduction	129
4.2	Fast eigenvalue decomposition for the interior problem	131
4.3	Fast diagonally-shifted factorization update	134
4.3.1	Structured approximation of Schur complements with multiple shifts	134
4.3.2	Factorization update	137
4.4	Complexity and discussions	138
4.5	Numerical experiments	141
4.6	Conclusions	145
	REFERENCES	147

VITA 155

LIST OF TABLES

3.1	<i>Example 1.</i> Accuracy of SuperDC, where some errors (δ) are not reported since <code>eig</code> runs out of memory, and the cases $n \geq 262,144$ are not shown since it takes too long to compute γ and θ	101
3.2	<i>Example 2.</i> Accuracy of SuperDC, where some errors (δ) are not reported since <code>eig</code> runs out of memory, and the cases $n \geq 262,144$ are not shown since it takes too long to compute γ and θ	104
3.3	<i>Example 3.</i> Accuracy of SuperDC, where the error (δ) for $n = 65,536$ is not reported since <code>eig</code> runs out of memory. Note that for $n = 4,096$, A has a zero eigenvalue.	107
3.4	<i>Example 4.</i> Accuracy of SuperDC, where the error (δ) for $n = 65,536$ is not reported since <code>eig</code> runs out of memory.	107
3.5	<i>Example 4.</i> Maximum percentage (μ) of eigenvalues not converged within 5 iterations for solving the r secular equations associated with <code>root(T)</code>	109
3.6	<i>Example 4.</i> Norms of the D, B generators after the dividing stage, where ∞ means overflow.	110
4.1	Costs of some basic operations.	139
4.2	Costs of the main operations in the precomputation and factorization update.	140
4.3	Precomputation flops count for <code>NEW</code> , <code>tol=10⁻³</code>	142
4.4	Factorization update flops count and storage, <code>NEW</code> v.s. <code>SMF</code>	144
4.5	Solution stage, <code>SMF</code> v.s. <code>NEW</code>	145

LIST OF FIGURES

2.1	Well separated sets	26
2.2	A two-level partition example and the corresponding post-ordered quadtree . . .	52
2.3	Interaction list and far-field approximations	52
3.1	A 4-level symmetric HSS matrix and its associated HSS binary tree	71
3.2	Roots of secular equation $1 + \frac{v_1^2}{d_1-x} + \frac{v_2^2}{d_2-x} + \frac{v_3^2}{d_3-x} + \frac{v_4^2}{d_4-x} = 0$	74
3.3	Nodes involved in the dividing process.	78
3.4	Structure eigenmatrix Q , where $l_{\max} = 4$	97
3.5	<i>Example 1.</i> Timing and storage of SuperDC and eig and flops of SuperDC. . . .	100
3.6	<i>Example 2.</i> Timing and storage of SuperDC and eig and flops of SuperDC. . . .	103
3.7	<i>Example 3.</i> Timing and storage of SuperDC and eig and flops of SuperDC. . . .	106
3.8	<i>Example 4.</i> Timing and storage of SuperDC and eig and flops of SuperDC. . . .	108
3.9	Multilevel block broken-arrowhead form	123
4.1	Precomputation flops count of NEW, $\text{tol}=10^{-3}$	142
4.2	Storage of Q , $\text{tol}=10^{-3}$	143
4.3	Storage of W , $\text{tol}=10^{-3}$	143
4.4	Factorization update flops	144
4.5	Factorization update storage	144
4.6	Solution flops, SMF v.s. NEW	145

ABSTRACT

In this dissertation, we study stable and efficient methods for structured matrix computations. In particular, we present stability analysis, stabilization strategies, and efficiency improvements for some major structured matrix algorithms. Structured matrix computations now play an important role in many applications. Compared to traditional algorithms, it can reduce significantly the algorithm complexity, by capturing and taking advantage of the intrinsic structures of the underlying problems. This dissertation overcomes some major challenges and presents some novel results in the design and analysis of structured matrix algorithms. These results have been previously overlooked due to the complex nature of the structured methods. We show how to integrate many stabilization techniques into efficient structured methods. The stability and efficiency of the resulting algorithms are justified both theoretically and numerically. In Chapter 2, we propose a stable matrix version of the fast multipole method (FMM) in the complex plane. The advantage of our stable FMM is that all the intermediate low-rank matrices will have bounded entries and bounded norms, so that they can be computed stably and efficiently. Based on this, we give a formal proof of the backward stability of our stable FMM. In Chapter 3, we propose a robust and superfast divide-and-conquer eigensolver (SuperDC) for symmetric structured matrices, which significantly improves some earlier basic algorithms. The complexity of SuperDC for getting the full eigenvalue decomposition is quasilinear, as compared to the cubic cost of traditional divide and conquer. Such acceleration is achieved by integrating our stable FMM in a novel but subtle way with the modified Newton-Raphson method for the solution of some intermediate rank-one modification eigenvalue problems. Numerical tests demonstrate its superior performance in terms of speed and memory. To accommodate more general problems, we also extend SuperDC to compute the singular value decomposition of nonsymmetric structured matrices. In Chapter 4, we consider the efficient factorization update for some shifted discretized matrices. After an $O(n)$ precomputation stage, the factorization update for each new shift needs only $O(\sqrt{n} \log n)$, which is significantly more efficient than doing refactorizations. The various contributions of this dissertation significantly advance the reliability, accuracy, and efficiency of structured matrix computations.

1. INTRODUCTION

In this dissertation, we study stable and efficient methods for structured matrix computations. As the increasing problem size renders classical numerical linear algorithms less competitive, structured matrix computations now plays an important role in modern scientific computations. While most classical methods will treat the underlying matrix as a general one, structured matrix methods attempt to capture the intrinsic properties or structures of the underlying specific problem (e.g., low-rank property, divide-and-conquer, multilevel). Based on that, efficient structured algorithms can be designed. Compared to their classical counterparts, the complexity and memory of these structured algorithms are significantly reduced, making them much more competitive and appropriate for large-scaled problems.

Among various structured matrix methods, there are the famous fast multipole methods (FMM) [1]–[4], $\mathcal{H}/\mathcal{H}^2$ matrix [5]–[8], quasiseparable/semiseparable matrix [9], [10], hierarchical semiseparable matrix (HSS) [11]–[14], block low-rank matrix (BLR) [15]. There are also methods dedicated to sparse matrix, like the divide-and-conquer methods for bidiagonal/tridiagonal matrix [16]–[21], the multifrontal method for sparse factorization [22]–[24], and the structured multifrontal methods [25]–[28]. Some randomization techniques can also be integrated with structured matrix methods to speed up the algorithms [11]–[13], [29], [30]. In this dissertation, we focus on the fast multipole methods, and HSS matrix. We also consider the factorization updates for some sparse discretized matrices.

In Chapter 2, we shall describe a *stable matrix version* of the fast multipole method (FMM) in the complex plane. This work can also serve as an elementary introduction to FMM for people who are more familiar with matrix. It is well-known that the essential components of FMM are some *degenerate/separable* expansions of the underlying kernel for *far-field* point clusters. In matrix language, it is equivalent to the analytic constructions of some separable low-rank approximations to the off-diagonal blocks of the kernel matrix. However, some commonly used expansions may lead to fast-growing coefficients, which renders the algorithm prone to numerical instability. In some circumstances where the points are highly clustered and full precision are needed (e.g., the secular equation solution in Chapter 3), these fast-growing coefficients may destroy the accuracy. We attempt to overcome

this issue via some novel derivations of the expansions. Our examples include generalized Cauchy kernel, Poisson kernel and the Helmholtz kernel. In our derivations, the intermediate low-rank matrices of the FMM shall have bounded entries as well as bounded norms, so that their entries can be computed stably and efficiently via some recurrence formulas. We say such low-rank approximations are stable. We also study the backward stability of the FMM, which, to the best of my knowledge, has been lacking for years due to the complexity of the algorithm. In particular, we can show that with our stable low-rank approximations, the FMM is backward stable, and its entry-wise backward error only depends logarithmically on the matrix size. This is significantly better than standard matrix-vector product routine, for which the backward error depends linearly on the matrix size. This also confirms the advantage of structured matrix methods over classical matrix methods in terms of numerical stability.

In Chapter 3, we shall describe a superfast divide-and-conquer eigenvalue decomposition (SuperDC) for dense symmetric matrices with small off-diagonal ranks and in HSS form. SuperDC significantly improves an earlier basic algorithm in [Vogel, Xia, et al., SIAM J. Sci. Comput., 38 (2016)]. The overall complexity of SuperDC is $O(r^2 n \log^2 n)$, where r is the maximal off-diagonal rank of the HSS matrix. The eigenmatrix is given in a structured form, consisting a sequence of Cauchy-like matrices and orthogonal transformations. The matrix-vector product routines of the eigenmatrix and its transpose have only $O(rn \log n)$ complexity. We incorporate a sequence of key stability techniques and provide many improvements in the algorithm design. Various stability risks in the original basic algorithm are analyzed, including potential exponential norm growth, cancellations, loss of accuracy with clustered eigenvalues or intermediate eigenvalues, etc. In the dividing stage, we give a new structured low-rank updating strategy with balancing that eliminates the exponential norm growth and also minimizes the ranks of low-rank updates. In the conquering stage with low-rank updated eigenvalue solution, the original algorithm directly uses the standard FMM to accelerate secular function evaluations, which has the risks of cancellation, division by zero, and slow convergence. Here, we design a triangular FMM to avoid cancellation and to accelerate the rate of convergence. Furthermore, when there are clustered intermediate eigenvalues or when updates to existing eigenvalues are very small, we design a novel

local shifting strategy to integrate FMM accelerations into the solution of shifted secular equations. This significantly enhances the efficiency and reliability. We also provide several improvements or clarifications on some structures and techniques that are missing or unclear in the previous work. While keeping the nearly linear complexity for finding the entire eigenvalue decomposition, the resulting SuperDC eigensolver has significantly better stability. In a set of comprehensive tests, SuperDC shows significantly lower runtime and storage than some other fast structured eigensolvers as well as the highly optimized MATLAB `eig` function. The stability benefits are also confirmed by both analysis and numerical comparisons. We also discuss some other techniques and improvements (e.g., rank-revealing factorization in the dividing stage, precomputations for FMM in the conquering stage, refined convergence criteria for roots of secular equations) for more efficient and reliable implementations.

We also extend the SuperDC to compute the full singular value decomposition (SVD) of unsymmetric HSS matrices. For this purpose, we will need to exploit the HSS structure in a different way from the symmetric case. The idea is to reduce the HSS matrix to a *multilevel block broken-arrowhead form* via a sequence of orthogonal transformations (similar orthogonal transformations are also used in the ULV-type factorization of HSS matrix). After the reduction, the full SVD shall be computed by recursively solving the SVDs of a series of broken-arrowhead matrix. Then we can combine the well-studied theory on the SVD of broken-arrowhead matrix, with the FMM acceleration techniques we develop for SuperDC. Analogous to the case of symmetric eigensolver, the complexity of the entire SVD solver is $O(r^2n \log^2 n)$, and the left and right singular matrices are given in structured form consisting a sequence of Cauchy-like matrices and orthogonal transformations. The matrix-vector product routine of left and right singular matrices also only have $O(rn \log n)$ complexity.

The SuperDC eigensolver/SVD solver makes it feasible to use full eigendecompositions to solve various challenging numerical problems. These include fast computations of Gauss quadratures via Golub-Welsch algorithm, fast spectral transform, fast solution of roots of Bessel functions, fast matrix function evaluation, separable PDE solver, etc. We expect to explore these applications in details in future work. In addition, we expect that the novel local shifting strategy and triangular FMM accelerations are also useful for other FMM-

related matrix computations when stability and accuracy are crucial. In our future work, we plan to prove the backward stability of SuperDC, as well as implement a high-performance parallel version, which will extend the applicability of the algorithm to more large-scaled numerical computations.

In Chapter 4, we consider the superfast factorization update for some important discretized matrices. It is commonly known that the LU factorization of a matrix A cannot be reused for the LU factorization of a diagonally shifted matrix $A - s_j I$, where s_j is a scalar. That is, $A - s_j I$ needs to be refactorized. In this chapter, we consider some important discretized matrices and develop a series of techniques that enable us to quickly obtain a factorization of A and then perform factorization update for multiple shifts s_j . We first compute a structured partial factorization of A , which can be reused to obtain new factorizations for free for multiple s_j . This idea is feasible because of several innovative ideas. One is to compute a fast structured eigenvalue decomposition for large a pivot submatrix A_{11} . This eigenvalue decomposition can be updated for free for different shifts. Then we use structured HSS form to approximate the Schur complements. This is done via randomization and matrix-vector multiplication. A key idea is to assemble all the matrix-vector multiplications for all the shifts together in a highly structured matrix-matrix multiplication. We fully utilize all the sparsity and structures in this process. By carefully designing all the steps, most of the operations can be done in a precomputation step. The update is essentially only limited to a small subproblem (such as that corresponding to boundary mesh points). For two dimensional elliptic problems and Helmholtz problems with certain coefficients and discretizations, the algorithm requires a precomputation cost of roughly $O(n)$ flops. The factorization update for each new shift needs only $O(\sqrt{n} \log n)$. The factorization update is said to be superfast and is significantly more efficient than refactorizations.

2. STABLE MATRIX VERSION OF THE FAST MULTIPOLE METHOD IN 2D

In this chapter, we shall describe a matrix version of the fast multipole method (FMM) in the complex plane. In our derivations, the intermediate low-rank matrices of the FMM have bounded entries and norms, so that they can be computed stably and efficiently. We say such low-rank approximations are *stable*. We also study the backward stability of the FMM. In particular, we can show that with our stable low-rank approximations, then the relative backward error of the FMM only depends logarithmically on the matrix size. Our examples include generalized Cauchy kernel, Poisson kernel and the Helmholtz kernel.

Throughout this chapter, the following notations are used.

- Bold lower-case letters like \mathbf{x} are used to denote sets of points, or nodes of a tree.
- $(K_{ij})_{n \times m}$ means an $n \times m$ matrix with the (i, j) -entry K_{ij} .
- $\text{diag}(\dots)$ denotes a (block) diagonal matrix.
- $\text{fl}(x)$ denotes the floating point result of x .
- ϵ_{mach} represents the machine precision.
- $i = \sqrt{-1}$ is the imaginary unit.
- For a complex number z , let $\theta_z \in (-\pi, \pi]$ denote its phase, $\text{Re}(z)$ denote its real part, and $\text{Im}(z)$ denote its imaginary part.
- Let $\alpha \in \mathbb{C}$, then $\binom{\alpha}{n} = \frac{\alpha(\alpha-1)\dots(\alpha-n+1)}{n!}$ denotes the generalized binomial coefficients.

2.1 Background

Given a kernel function $\kappa(x, y)$ and two sets of points $\mathbf{x} = \{x_i\}_{i=1}^n$ and $\mathbf{y} = \{y_j\}_{j=1}^m$ on the complex plane \mathbb{C} , the *interactions* between points in \mathbf{x} and \mathbf{y} are

$$\phi(x_i) = \sum_{j=1}^m \kappa(x_i, y_j) q_j, \quad 1 \leq i \leq n. \quad (2.1)$$

where $q_j \in \mathbb{R}$. Some frequently used kernels in 2D include the Poisson kernel $\log|x-y|$, the generalized Cauchy kernel $\frac{1}{(x-y)^{1+d}}$, as well as the Helmholtz kernel $H_0(\tilde{k}|x-y|)$, where H_0 is the first kind Hankel function of order 0. In this chapter, we assume without loss of generality the wavenumber $\tilde{k} = 1$ since $H_0(\tilde{k}|x-y|) = H_0(|\tilde{k}x - \tilde{k}y|)$. Let $\phi = (\phi(x_1), \dots, \phi(x_n))^T$, $q = (q_1, \dots, q_m)^T$, then (2.1) can be reformed as a matrix-vector product

$$\phi = Kq, \tag{2.2}$$

where the matrix $K = (\kappa(x_i, y_j))_{x_i \in \mathbf{x}, y_j \in \mathbf{y}} \in \mathbb{C}^{n \times m}$ is referred as the *interaction matrix* or *kernel matrix* of \mathbf{x} and \mathbf{y} .

The fast multipole method (FMM) is a fast algorithm that computes (2.1) or (2.2) to any given accuracy with $O(n+m)$ complexity [2], [3], [31]. Early literature of FMM [2], [3] focus on the summation form (2.1). Later, it is shown that the FMM essentially constructs a matrix approximation to the kernel matrix K (see, e.g., [1], [32]). This approximation matrix is also referred as the *FMM matrix*, and belongs to the class of hierarchically structured \mathcal{H}^2 -matrix [5], [6]. The FMM algorithm has lots of success in accelerating numerical computations, including N body simulations [3], integral equation solutions [31], fast Gauss transform [33], symmetric eigenvalue solutions [16], [34]. In this chapter, we will focus on the matrix form (2.2).

The construction of the FMM matrix relies on the *degenerate* or *separable expansion* of the underlying kernels κ . Some commonly used separable expansions include multipole expansion [3], Taylor expansion [32], spherical harmonic expansion [1], [2], Chebyshev interpolation [35], proxy point method [36]. Such expansions can be used to construct effectively the separable low-rank approximation of the off-diagonal blocks of the kernel matrix K .

There exists a lots of implementations of the FMM algorithm [4], [31], [32], [35], [37], [38] and they achieve satisfactory accuracy and efficiency. Nevertheless, as pointed out in [4], [31], [32], some stability risk may arise when we apply those aforementioned separable expansions to construct the low-rank factorization. Examples include fast-growing coefficients in the expansions, products of extremely large numbers and extremely small numbers, extremely large entries in the low-rank approximations. Such stability concerns may cause the

algorithm to break down in some special circumstances when full double-precision accuracy are desired [16], [34], [39]. In this work, we attempt to resolve such stability dangers for some kernels in 2D, including the generalized Cauchy kernel, the Poisson kernel and the Helmholtz kernel. In particular, via meticulous selections of forms of the separable expansions, we construct *stable* low-rank approximations to the off-diagonal block of the kernel matrix, such that the low-rank factors shall have bounded entries and norms. We also provide recurrence formulas to compute the entries stably and efficiently.

Furthermore, due to the complicated nature of the FMM, its backward stability has not been fully understood. In [32], an elementary study in the backward stability of the matrix-vector product with the off-diagonal block is provided. In [40], some relevant stability studies on HSS matrix are provided, built on the assumptions of orthogonal off-diagonal basis matrices. In this work, we attempt to provide a more comprehensive and specific study of the backward stability of the FMM algorithm. In particular, we shall prove that via our stable low-rank approximations, the FMM algorithm is backward stable. The backward error depends logarithmically on the size of the kernel matrix K . As comparisons, the backward error in the standard matrix-vector product routine depends linearly on the matrix size.

The rest of this chapter is organized as follows. In Section 2.2, we provide some mathematics apparatus for the derivation and analysis of the FMM. In Section 2.3, we derive the stable low-rank approximation for different kernels. Then in Section 2.4, we derive the stable translation relation in the FMM. The overall framework of the matrix version of the FMM, as well as the backward stability are presented in Section 2.5.

2.2 Mathematics apparatus

In this section, we present some mathematics apparatus for the derivation and analysis of the separable expansions for various kernels including the generalized Cauchy kernel, Poisson kernel and the Helmholtz kernel.

2.2.1 Binomial series and the truncation error

It is well-known that the binomial series is the Taylor series of the function $F(z) = \frac{1}{(1-z)^{1+d}}$ where $d \in \mathbb{C}$

$$\frac{1}{(1-z)^{1+d}} = \sum_{p=0}^{\infty} \binom{p+d}{p} z^p, \quad |z| < 1. \quad (2.3)$$

The following lemma studies the truncation error of (2.3).

Lemma 2.2.1. *Let E_r be the remainder term in the Taylor expansion of F ,*

$$E_r(z) \equiv F(z) - \sum_{p=0}^{r-1} \binom{p+d}{p} z^p = \sum_{p=r}^{\infty} \binom{p+d}{p} z^p, \quad |z| < 1.$$

For any given $v > 0$ such that $\mu \equiv (1+v)|z| < 1$, there exists a nonnegative integer $N_0 = \lceil \frac{|d|}{v} \rceil$ such that for $r \geq N_0$,

$$|E_r(z)| \leq \frac{\left| \binom{N_0+d}{N_0} \right|}{(1+v)^{N_0}} \cdot \frac{\mu^r}{1-\mu} = O(\mu^r).$$

Proof. Let $a_p(z) = \binom{p+d}{p} z^p$. If $p \geq N_0 = \lceil \frac{|d|}{v} \rceil$, then

$$\left| \frac{a_{p+1}(z)}{a_p(z)} \right| = \left| \left(1 + \frac{d}{p+1} \right) z \right| \leq (1+v)|z| = \mu,$$

hence,

$$|a_p(z)| \leq |a_{N_0}(z) \mu^{p-N_0}| = \left| \binom{N_0+d}{N_0} \right| \frac{|z|^{N_0}}{\mu^{N_0}} \mu^p = \frac{\left| \binom{N_0+d}{N_0} \right|}{(1+v)^{N_0}} \mu^p.$$

As a result, for $r \geq N_0$,

$$|E_r(z)| \leq \sum_{p=r}^{\infty} |a_p(z)| \leq \frac{\left| \binom{N_0+d}{N_0} \right|}{(1+v)^{N_0}} \sum_{p=r}^{\infty} \mu^p = \frac{\left| \binom{N_0+d}{N_0} \right|}{(1+v)^{N_0}} \frac{\mu^r}{1-\mu}.$$

□

2.2.2 Some basic properties of Bessel functions

In this subsection, we present some basic properties of Bessels functions. This will be needed in our derivation and analysis of the FMM for the Helmholtz kernel.

We respect the convention to let $J_p(z)$ denote the order p Bessel function of the first kind, and $Y_p(z)$ the order p Bessel function of second kind. Both $J_p(z)$ and $Y_p(z)$ are real values if $z > 0$ and $p \in \mathbb{R}$. Let $H_p(z) = J_p(z) + iY_p(z)$ denote the Hankel function of the first kind. The following proposition can be found in standard references like [41], [42].

Proposition 2.2.1 ([41], [42]). *Suppose $p \in \mathbb{N}$ and $z > 0$, the following relations hold for the Bessel functions.*

$$|J_p(z)| \leq 1, \quad |J_p(z)| \leq \frac{1}{p!} \left(\frac{z}{2}\right)^p \leq \frac{1}{\sqrt{2\pi p}} \left(\frac{ez}{2p}\right)^p,$$

$$J_{-p}(z) = (-1)^p J_p(z), \quad Y_{-p}(z) = (-1)^p Y_p(z), \quad H_{-p}(z) = (-1)^p H_p(z).$$

They satisfy the following recurrence relation where T can be J, Y, H ,

$$T_{p-1}(z) + T_{p+1}(z) = \frac{2p}{z} T_p(z), \tag{2.4}$$

$$T_{p-1}(z) - T_{p+1}(z) = 2T'_p(z). \tag{2.5}$$

If in addition $0 < z \leq p$, then $J_p(z) > 0$ and $J'_p(z) > 0$, and

$$|J_p(z)| \leq |Y_p(z)|, \quad |H_p(z)| \leq \sqrt{2}|Y_p(z)|.$$

The following proposition describes the asymptotic behavior of $Y_p(z)$ (see [43]).

Proposition 2.2.2 ([43]). *Suppose $p \in \mathbb{N}_+$ and $z > 0$, and let $C_p(z)$ be defined by*

$$Y_p(z) = -C_p(z) \sqrt{\frac{2}{\pi p}} \left(\frac{2p}{ez}\right)^p.$$

Then $\lim_{p \rightarrow \infty} C_p(z) = 1$. If in addition $p \geq z$, we have

$$C_p(z) > 0, \quad \text{and} \quad \frac{Y_{p+1}(z)}{Y_p(z)} > \frac{p}{z}.$$

If in addition $p \geq z$ and $p \geq 2$, we have

$$C_p(z) > C_{p+1}(z) > \cdots > 1, \quad \text{and} \quad \frac{Y_{p+1}(z)}{Y_p(z)} < \frac{2p}{z}.$$

Based on Proposition 2.2.2, we can prove the following proposition for $C_p(z)$.

Proposition 2.2.3. *The function $C_p(z)$ has the following properties.*

(i) If $p \geq 1$ and $0 < z \leq p$, then $C_p(z) < \frac{4\sqrt{2}}{e} C_{p+1}(z)$.

(ii) If $p \geq 2$ and $0 < z \leq p$, $C_p(z)$ is a strictly increasing function of z .

(iii) If $p \geq 2$ and $0 < z \leq p$, $C_p(z) \leq |H_p(p)| \sqrt{\frac{\pi p}{2}} \left(\frac{e}{2}\right)^p$.

Proof. We first prove (i). By definition, $C_p(z) = -Y_p(z) \sqrt{\frac{\pi p}{2}} \left(\frac{ez}{2p}\right)^p$. Then

$$\frac{C_p(z)}{C_{p+1}(z)} = \frac{Y_p(z) \sqrt{\frac{\pi p}{2}} \left(\frac{ez}{2p}\right)^p}{Y_{p+1}(z) \sqrt{\frac{\pi(p+1)}{2}} \left(\frac{ez}{2(p+1)}\right)^{p+1}}.$$

According to Proposition 2.2.2, we have $\frac{Y_p(z)}{Y_{p+1}(z)} < \frac{z}{p}$. Therefore,

$$\frac{C_p(z)}{C_{p+1}(z)} < \frac{2}{e} \left(1 + \frac{1}{p}\right)^{p+\frac{1}{2}} \leq \frac{4\sqrt{2}}{e}.$$

This proves property (i). For property (ii), the derivative of $C_p(z)$ is

$$C'_p(z) = -\sqrt{\frac{\pi p}{2}} \left(\frac{ez}{2p}\right)^p \left(Y'_p(z) + \frac{p}{z} Y_p(z)\right).$$

For Bessel functions, we have the following recurrence formulas (see (2.4) and (2.5))

$$Y_{p-1}(z) + Y_{p+1}(z) = \frac{2p}{z} Y_p(z), \tag{2.6}$$

$$Y_{p-1}(z) - Y_{p+1}(z) = 2Y'_p(z). \tag{2.7}$$

Adding (2.6) and (2.7) to get

$$C'_p(z) = -\sqrt{\frac{\pi p}{2}} \left(\frac{ez}{2p}\right)^p Y_{p-1}(z).$$

According to [42, Section 9.5] and [43, Lemma 1], $Y_{p-1}(z) < 0$ if $p \geq 2$ and $0 < z \leq p$. Therefore, $C_p(z)$ is a strictly increasing function of z if $p \geq 2$ and $0 < z \leq p$. This proves property (ii). As a result,

$$C_p(z) \leq C_p(p) = |Y_p(p)| \sqrt{\frac{\pi p}{2}} \left(\frac{e}{2}\right)^p \leq |H_p(p)| \sqrt{\frac{\pi p}{2}} \left(\frac{e}{2}\right)^p,$$

which proves property (iii). □

The following proposition studies the monotonicity of $|H_p(z)|$.

Proposition 2.2.4. *The Hankel function $H_p(z)$ satisfies the following properties*

- (i) For fixed $z > 0$, $|H_p(z)|$ is a strictly increasing function of $p \geq 0$.
- (ii) For fixed $p \geq 0$, $|H_p(z)|$ is a strictly decreasing function of $z > 0$.
- (iii) If $z \geq \frac{1}{2}$ and $0 \leq p \leq z$, then $|H_p(z)| \leq \sqrt{\frac{4}{\pi}}$.
- (iv) If $z > 0$, $p \geq 0$, and $0 < \lambda < 1$, then $|H_p(\lambda z)| \leq \frac{1}{\sqrt{\lambda}} |H_{\frac{p}{\lambda}}(z)|$.

Proof. The proofs of properties (i) and (ii) are given in [43, Lemma 2]. We prove properties (iii) and (iv) below.

To prove (iii), suppose $z \geq \frac{1}{2}$ and $0 \leq p \leq z$. By property (i), $|H_p(z)| \leq |H_z(z)|$. Therefore, it suffices to show $|H_z(z)| \leq \sqrt{\frac{4}{\pi}}$ for $z \geq \frac{1}{2}$. By Nicholson's formula (see, e.g. [41], [43]),

$$|H_p(z)|^2 = \frac{8}{\pi^2} \int_0^\infty \int_0^\infty e^{-2z \sinh t \cosh v} \cosh(2pt) dt dv.$$

Take the derivative of $|H_z(z)|^2$ with respect to z to get

$$\frac{d}{dz} |H_z(z)|^2 = \frac{16}{\pi^2} \int_0^\infty \int_0^\infty e^{-2z \sinh t \cosh v} \left(t \sinh(2zt) - \sinh t \cosh(2zt) \cosh v \right) dt dv.$$

For $z, v, t > 0$, $t < \sinh t$, $\sinh(2zt) < \cosh(2zt)$, and $1 < \cosh v$. Then

$$t \sinh(2zt) < \sinh t \cosh(2zt) \cosh v.$$

Therefore, $\frac{d}{dz}|H_z(z)|^2 < 0$. As a result, for $z \geq \frac{1}{2}$,

$$|H_z(z)|^2 \leq \left|H_{\frac{1}{2}}\left(\frac{1}{2}\right)\right|^2 = \frac{8}{\pi^2} \int_0^\infty \int_0^\infty e^{-\sinh t \cosh v} \cosh t dt dv = \frac{4}{\pi}.$$

To prove (iv), by Nicholson's formula, we have

$$|H_p(\lambda z)|^2 = \frac{8}{\pi^2} \int_0^\infty \int_0^\infty e^{-2\lambda z \sinh t \cosh v} \cosh(2pt) dt dv.$$

Since $\sinh t = \frac{e^t - e^{-t}}{2}$ is convex on $[0, \infty)$ and $\sinh 0 = 0$, we have for any $\lambda \in (0, 1)$,

$$\lambda \sinh t \geq \sinh(\lambda t), \quad 0 \leq t < \infty.$$

Therefore,

$$\begin{aligned} |H_p(\lambda z)|^2 &\leq \frac{8}{\pi^2} \int_0^\infty \int_0^\infty e^{-2z \sinh(\lambda t) \cosh v} \cosh(2pt) dt dv \\ &= \frac{8}{\pi^2} \int_0^\infty \int_0^\infty \frac{1}{\lambda} e^{-2z \sinh t \cosh v} \cosh\left(2\frac{p}{\lambda}t\right) dt dv \\ &= \frac{1}{\lambda} |H_{\frac{p}{\lambda}}(z)|^2. \end{aligned}$$

□

2.2.3 Graf's addition formula and the truncation error

The following *Graf's addition formula* is essential to the theory behind the fast multipole methods for the Helmholtz kernel (see, e.g., [42, (9.1.79)], [31, Theorem 3.1]).

Theorem 2.2.2 ([31], [42]). *Suppose $z_1, z_2 \in \mathbb{C}$, then*

$$J_p(|z_1 - z_2|)e^{\pm ip\theta_{z_1 - z_2}} = \sum_{l=-\infty}^{\infty} J_{p+l}(|z_1|)e^{\pm i(p+l)\theta_{z_1}} J_l(|z_2|)e^{\mp il\theta_{z_2}}. \quad (2.8)$$

Suppose $w, t \in \mathbb{C}$ such that $|w| > |t|$, then

$$H_p(|w - t|)e^{\pm ip\theta_{w-t}} = \sum_{l=-\infty}^{\infty} H_{p+l}(|w|)e^{\pm i(p+l)\theta_w} J_l(|t|)e^{\mp il\theta_t}. \quad (2.9)$$

Given $r \in \mathbb{N}$, we can truncate the summands of (2.8) and (2.9) when $|l| \geq r$ or $|l+p| \geq r$. The following two lemmas study the truncation errors. For relevant analysis, see [2], [43]–[45].

Lemma 2.2.3. *Let $E_r^{J_p}$ be the remainder term in Graf's addition formula for J_p ,*

$$E_r^{J_p}(z_1, z_2) = \sum_{|l| \geq r \text{ or } |p+l| \geq r} J_{p+l}(|z_1|)e^{\pm i(p+l)\theta_{z_1}} J_l(|z_2|)e^{\mp il\theta_{z_2}}.$$

Suppose $z_{\max} \geq \max(|z_1|, |z_2|)$. If $r \geq z_{\max}$, then

$$|E_r^{J_p}(z_1, z_2)| \leq 4 \sum_{l=r}^{\infty} |J_l(z_{\max})| \leq \frac{8}{r!} \left(\frac{z_{\max}}{2} \right)^r.$$

Proof. By Proposition 2.2.1, $|J_k(z)|$ is an increasing function of z if $0 < z \leq |k|$.

$$\begin{aligned} |E_r^{J_p}(z_1, z_2)| &\leq \sum_{|l| \geq r \text{ or } |p+l| \geq r} |J_{p+l}(z_{\max}) J_l(z_{\max})| \\ &\leq \sum_{|p+l| \geq r} |J_{p+l}(z_{\max})| + \sum_{|l| \geq r} |J_l(z_{\max})| = 4 \sum_{l=r}^{\infty} |J_l(z_{\max})|. \end{aligned}$$

This proves the first inequality. To prove the second one, since $r \geq z_{\max}$,

$$\sum_{l=r}^{\infty} |J_l(z_{\max})| \leq \sum_{l=r}^{\infty} \frac{1}{l!} \left(\frac{z_{\max}}{2} \right)^l \leq \sum_{l=r}^{\infty} \frac{1}{r!} \left(\frac{z_{\max}}{2} \right)^r \frac{1}{2^{l-r}} = \frac{2}{r!} \left(\frac{z_{\max}}{2} \right)^r.$$

□

Lemma 2.2.4. *Let $E_r^{H_0}$ be the remainder term in Graf's addition formula for H_0 ,*

$$E_r^{H_0}(w, t) = \sum_{|l| \geq r} H_l(|w|)e^{\pm il\theta_w} J_l(|t|)e^{\mp il\theta_t}.$$

Suppose $|t| \leq t_{\max} \leq \tau|w|$ for some $\frac{1}{2} \leq \tau < 1$. If $r \geq \max(\frac{t_{\max}}{\tau}, 2)$, then

$$|E_r^{H_0}(w, t)| \leq \frac{2\sqrt{2}C_r\left(\frac{t_{\max}}{\tau}\right)}{\pi r} \frac{\tau^r}{1-\tau}.$$

Proof. According to Propositions 2.2.1, 2.2.2, and 2.2.4(ii),

$$\begin{aligned} |E_r^{H_0}(w, t)| &\leq 2 \sum_{l=r}^{\infty} \left| H_l\left(\frac{t_{\max}}{\tau}\right) J_l(t_{\max}) \right| \leq 2\sqrt{2} \sum_{l=r}^{\infty} \left| Y_l\left(\frac{t_{\max}}{\tau}\right) J_l(t_{\max}) \right| \\ &\leq 2\sqrt{2} \sum_{l=r}^{\infty} C_l\left(\frac{t_{\max}}{\tau}\right) \sqrt{\frac{2}{\pi l}} \left(\frac{2l\tau}{et_{\max}}\right)^l \frac{1}{\sqrt{2\pi l}} \left(\frac{et_{\max}}{2l}\right)^l \\ &\leq \frac{2\sqrt{2}}{\pi r} \sum_{l=r}^{\infty} C_l\left(\frac{t_{\max}}{\tau}\right) \tau^l \leq \frac{2\sqrt{2}C_r\left(\frac{t_{\max}}{\tau}\right)}{\pi r} \frac{\tau^r}{1-\tau}. \end{aligned}$$

□

2.3 Stable kernel expansion and low-rank approximation

If the kernel function κ can be approximated by a *degenerate* or *separable* expansion such that for $x_i \in \mathbf{x}, y_j \in \mathbf{y}$,

$$\kappa(x_i, y_j) \approx \sum_{p,l=0}^{r-1} b_{pl} u_p(x_i) v_l(y_j), \quad (2.10)$$

then the kernel matrix has a low-rank approximation

$$K \approx UBV^T, \quad \text{where} \quad (2.11)$$

$$U = (u_p(x_i))_{n \times r}, \quad B = (b_{pl})_{r \times r}, \quad V = (v_l(y_j))_{m \times r}.$$

In this section, we provide the details on the derivations of such separable expansion for kernels including the Poisson kernel $\log \frac{1}{|x-y|}$, generalized Cauchy kernel $\frac{1}{(x-y)^{1+d}}$, and the Helmholtz kernel $H_0(|x-y|)$. For a set of points $\mathbf{x} \subset \mathbb{C}$, we use $\delta_{\mathbf{x}}$ and $o_{\mathbf{x}}$ to denote the radius and center of a circle that encloses \mathbf{x} . Note that such circle may not be unique, but

this will not concern the discussions in this paper. We generalize the definition in [1], [32] to describe *well-separated* sets.

Definition 2.3.1 ([1]). Let $\tau \in [\frac{1}{2}, 1)$, $\alpha \in (0, 1]$. Two sets of points \mathbf{x} and \mathbf{y} are said to be (τ, α) -*well-separated* if

$$\frac{\delta_{\mathbf{x}} + \delta_{\mathbf{y}}}{|o_{\mathbf{x}} - o_{\mathbf{y}}|} \leq \tau, \quad (2.12)$$

$$\alpha \delta_{\mathbf{y}} \leq \delta_{\mathbf{x}} \leq \frac{1}{\alpha} \delta_{\mathbf{y}}. \quad (2.13)$$

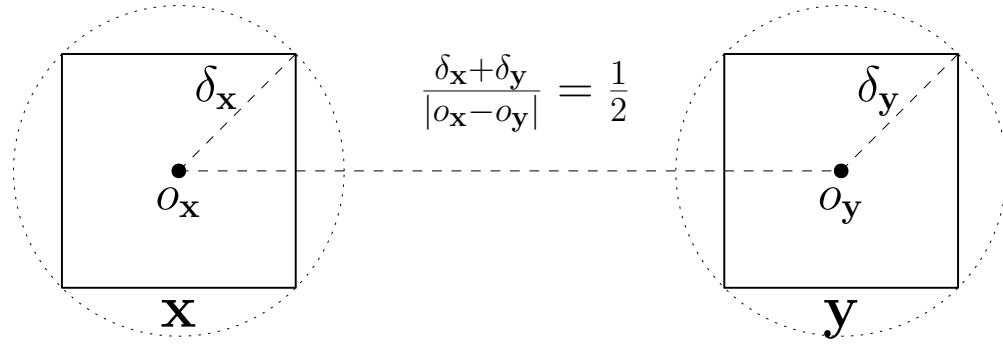


Figure 2.1. Well separated sets

Note that by triangular inequality, (2.12) implies that for $x \in \mathbf{x}, y \in \mathbf{y}$,

$$\begin{aligned} |(x - o_{\mathbf{x}}) - (y - o_{\mathbf{y}})| &\leq \tau |o_{\mathbf{x}} - o_{\mathbf{y}}|, \\ (1 - \tau)|x - y| &\leq \frac{1}{1 + \tau}|x - y| \leq |o_{\mathbf{x}} - o_{\mathbf{y}}| \leq \frac{1}{1 - \tau}|x - y|. \end{aligned}$$

(2.13) implies that

$$\delta_{\mathbf{x}} + \delta_{\mathbf{y}} \geq (1 + \alpha)\delta_{\mathbf{x}}, \quad \delta_{\mathbf{x}} + \delta_{\mathbf{y}} \geq (1 + \alpha)\delta_{\mathbf{y}}.$$

2.3.1 Generalized Cauchy kernel

The standard Cauchy kernel (i.e., $\kappa(x, y) = \frac{1}{x-y}$) is treated in [32], such that for well-separated \mathbf{x} and \mathbf{y} , a low-rank approximation like $K \approx UBV^T$ is derived via Taylor expansion. The authors of [32] also apply a sequence of scaling factors to the matrices U, B, V , so

that the scaled matrices have bounded entries. These scaling factors are designed based on Stirling's formula, and it is difficult to compute the entries of the scaled B efficiently.

In this subsection, we extend the studies of [32] to the generalized Cauchy kernel $\frac{1}{(x-y)^{1+d}}$, $d \in \mathbb{Z}$. Without resorting to the Stirling's formula, we only need Taylor expansion to establish (2.10) and (2.11) with a stronger bound on the entries and norms of the low-rank factors U , B and V . We also give a recurrence formula to compute the entries of B efficiently and stably, which is an advantage over [32]. In some special cases (e.g., \mathbf{x} and \mathbf{y} are on a straight line), we can even relax the assumption $d \in \mathbb{Z}$ to $d \in \mathbb{C}$ (see Corollary 2.3.2). To be more specific, we have the following theorem.

Theorem 2.3.1. *Suppose $\kappa(x, y) = \frac{1}{(x-y)^{1+d}}$, $d \in \mathbb{Z}$. Suppose $\mathbf{x} = \{x_i\}_{i=1}^n$ and $\mathbf{y} = \{y_j\}_{j=1}^m$ are well-separated with separation ratio τ . Let $\epsilon > 0$ be any small positive number, then the kernel matrix $K = (\kappa(x_i, y_j))_{x_i \in \mathbf{x}, y_j \in \mathbf{y}}$ has a rank $r = O\left(\frac{\log \epsilon + |1+d| \cdot \log(1-\tau)}{\log \tau}\right)$ approximation*

$$K = UBVT^T + E, \quad \text{where } |E| \leq \epsilon|K|. \quad (2.14)$$

These matrices have the following forms

$$U = \left(\left(\frac{x_i - o_{\mathbf{x}}}{\delta_{\mathbf{x}}} \right)^p \right)_{n \times r}, \quad V = \left(\left(\frac{y_j - o_{\mathbf{y}}}{\delta_{\mathbf{y}}} \right)^l \right)_{m \times r}, \quad (2.15)$$

$$B = \begin{pmatrix} b_{00} & b_{01} & \cdots & b_{0,r-1} \\ b_{10} & \cdots & b_{1,r-2} & \\ \vdots & \ddots & & \\ b_{r-1,0} & & & \end{pmatrix}_{r \times r}, \quad (2.16)$$

$$b_{pl} = \frac{(-1)^p}{(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}} \binom{p+l+d}{p+l} \binom{p+l}{p} \left(\frac{\delta_{\mathbf{x}}}{o_{\mathbf{x}} - o_{\mathbf{y}}} \right)^p \left(\frac{\delta_{\mathbf{y}}}{o_{\mathbf{x}} - o_{\mathbf{y}}} \right)^l. \quad (2.17)$$

Moreover, the entries b_{pl} can be computed efficiently and stably via the recurrence relation

$$\begin{cases} b_{p,-1} = b_{-1,l} = 0, & b_{00} = \frac{(-1)^p}{(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}}, \\ b_{pl} = \frac{p+l+d}{p+l} \left(\frac{\delta_{\mathbf{y}}}{o_{\mathbf{x}} - o_{\mathbf{y}}} b_{p,l-1} - \frac{\delta_{\mathbf{x}}}{o_{\mathbf{x}} - o_{\mathbf{y}}} b_{p-1,l} \right), & 1 \leq p+l \leq r-1 \end{cases} \quad (2.18)$$

Furthermore, their entries satisfy the following bounds

$$\|U\|_{\max} \leq 1, \quad \|V\|_{\max} \leq 1, \quad \|B\|_{1,1} = \sum_{p,l} |b_{pl}| \leq \frac{\min_{i,j} |K_{ij}|}{(1-\tau)^{2+2|d|}}. \quad (2.19)$$

As corollaries, their norms satisfy the following bounds

$$\|U\|_2 \leq \sqrt{rn}, \quad \|V\|_2 \leq \sqrt{rm}, \quad \|B\|_2 \leq \frac{\|K\|_2}{(1-\tau)^{2+2|d|}}. \quad (2.20)$$

Proof. Let $x \in \mathbf{x}$, $y \in \mathbf{y}$, $t = \frac{(x-o_{\mathbf{x}})-(y-o_{\mathbf{y}})}{o_{\mathbf{y}}-o_{\mathbf{x}}}$, then $x - y = (o_{\mathbf{x}} - o_{\mathbf{y}})(1 - t)$. By the assumption $d \in \mathbb{Z}$, we have

$$(x - y)^{1+d} = (o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}(1 - t)^{1+d}. \quad (2.21)$$

Note that this identity (2.21) does not generally hold if $d \notin \mathbb{Z}$. Since \mathbf{x} and \mathbf{y} are well separated, we have $|t| \leq \tau < 1$. By Taylor expansion,

$$\begin{aligned} \frac{1}{(x - y)^{1+d}} &= \frac{1}{(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}} \cdot \frac{1}{(1 - t)^{1+d}} \\ &= \frac{1}{(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}} \sum_{k=0}^{r-1} \binom{k+d}{k} t^k + \frac{E_r(t)}{(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}} \\ &= \sum_{k=0}^{r-1} \binom{k+d}{k} \sum_{p=0}^k \binom{k}{p} (-1)^p \frac{(x - o_{\mathbf{x}})^p (y - o_{\mathbf{y}})^{k-p}}{(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d+k}} + \frac{E_r(t)}{(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}} \\ &= \sum_{k=0}^{r-1} \sum_{p=0}^k \binom{k+d}{k} \binom{k}{p} (-1)^p \frac{(x - o_{\mathbf{x}})^p (y - o_{\mathbf{y}})^{k-p}}{(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d+k}} + \frac{E_r(t)}{(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}} \\ &= \sum_{k=0}^{r-1} \sum_{p=0}^k b_{p,k-p} \left(\frac{x - o_{\mathbf{x}}}{\delta_{\mathbf{x}}} \right)^p \left(\frac{y - o_{\mathbf{y}}}{\delta_{\mathbf{y}}} \right)^{k-p} + \frac{E_r(t)}{(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}}. \end{aligned}$$

This gives the forms in (2.14)-(2.17). Since $(1 - \tau)|x - y| \leq |o_{\mathbf{x}} - o_{\mathbf{y}}| \leq \frac{1}{1-\tau}|x - y|$ and $d \in \mathbb{Z}$, we have

$$\left| (o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d} \right| \geq (1 - \tau)^{|1+d|} \left| (x - y)^{1+d} \right|. \quad (2.22)$$

Therefore, if $r = O\left(\frac{\log \epsilon + |1+d| \cdot \log(1-\tau)}{\log \tau}\right)$, the remainder can be bounded according to (2.22) and Lemma 2.2.1

$$\frac{|E_r(t)|}{|(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}|} \leq \frac{|E_r(t)|}{(1-\tau)^{|1+d|}} |\kappa(x, y)| \leq \epsilon |\kappa(x, y)|.$$

Next, we prove the recurrence relation (2.18). Let $k = p + l$. Note that we have the identities for binomial coefficients $\binom{k+d}{k} = \frac{k+d}{k} \binom{k-1+d}{k-1}$ and $\binom{k}{p} = \binom{k-1}{p-1} + \binom{k-1}{p}$. Combining these two identities, we have

$$\binom{k+d}{k} \binom{k}{p} = \frac{k+d}{k} \binom{k-1+d}{k-1} \binom{k-1}{p-1} + \frac{k+d}{k} \binom{k-1+d}{k-1} \binom{k-1}{p}. \quad (2.23)$$

Substituting (2.23) into the formula (2.17) of b_{pl} to get

$$\begin{aligned} b_{pl} &= \frac{(-1)^p}{(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}} \binom{k+d}{k} \binom{k}{p} \left(\frac{\delta_{\mathbf{x}}}{o_{\mathbf{x}} - o_{\mathbf{y}}}\right)^p \left(\frac{\delta_{\mathbf{y}}}{o_{\mathbf{x}} - o_{\mathbf{y}}}\right)^l \\ &= \frac{(-1)^p}{(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}} \frac{k+d}{k} \binom{k-1+d}{k-1} \binom{k-1}{p-1} \left(\frac{\delta_{\mathbf{x}}}{o_{\mathbf{x}} - o_{\mathbf{y}}}\right)^p \left(\frac{\delta_{\mathbf{y}}}{o_{\mathbf{x}} - o_{\mathbf{y}}}\right)^l \\ &\quad + \frac{(-1)^p}{(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}} \frac{k+d}{k} \binom{k-1+d}{k-1} \binom{k-1}{p} \left(\frac{\delta_{\mathbf{x}}}{o_{\mathbf{x}} - o_{\mathbf{y}}}\right)^p \left(\frac{\delta_{\mathbf{y}}}{o_{\mathbf{x}} - o_{\mathbf{y}}}\right)^l \\ &= -\frac{k+d}{k} \frac{\delta_{\mathbf{x}}}{o_{\mathbf{x}} - o_{\mathbf{y}}} b_{p-1,l} + \frac{k+d}{k} \frac{\delta_{\mathbf{y}}}{o_{\mathbf{x}} - o_{\mathbf{y}}} b_{p,l-1} \\ &= \frac{p+l+d}{p+l} \left(\frac{\delta_{\mathbf{y}}}{o_{\mathbf{x}} - o_{\mathbf{y}}} b_{p,l-1} - \frac{\delta_{\mathbf{x}}}{o_{\mathbf{x}} - o_{\mathbf{y}}} b_{p-1,l} \right), \end{aligned}$$

which establish the recurrence relation (2.18).

Next, we prove the entry-wise bounds (2.19). Since $|x - o_{\mathbf{x}}| \leq \delta_{\mathbf{x}}$ and $|y - o_{\mathbf{y}}| \leq \delta_{\mathbf{y}}$, we have $\|U\|_{\max} \leq 1$ and $\|V\|_{\max} \leq 1$. To show the bound on $\|B\|_{1,1}$, note that $\left|\binom{k+d}{k}\right| = \frac{(k+d)(k-1+d)\cdots(1+d)}{k!} \leq \frac{(k+|d|)(k-1+|d|)\cdots(1+|d|)}{k!} = \binom{k+|d|}{k}$. Therefore,

$$\begin{aligned} \|B\|_{1,1} &= \sum_{k=0}^{r-1} \sum_{p=0}^k |b_{p,k-p}| \\ &\leq \frac{1}{|(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}|} \sum_{k=0}^{r-1} \sum_{p=0}^k \binom{k+|d|}{k} \binom{k}{p} \left| \frac{\delta_{\mathbf{x}}}{o_{\mathbf{x}} - o_{\mathbf{y}}} \right|^p \left| \frac{\delta_{\mathbf{y}}}{o_{\mathbf{x}} - o_{\mathbf{y}}} \right|^{k-p} \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{|(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}|} \sum_{k=0}^{r-1} \binom{k+|d|}{k} \sum_{p=0}^k \binom{k}{p} \left| \frac{\delta_{\mathbf{x}}}{o_{\mathbf{x}} - o_{\mathbf{y}}} \right|^p \left| \frac{\delta_{\mathbf{y}}}{o_{\mathbf{x}} - o_{\mathbf{y}}} \right|^{k-p} \\
&\leq \frac{1}{|(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}|} \sum_{k=0}^{r-1} \binom{k+|d|}{k} \sum_{p=0}^k \binom{k}{p} \left| \frac{\tau \delta_{\mathbf{x}}}{\delta_{\mathbf{x}} + \delta_{\mathbf{y}}} \right|^p \left| \frac{\tau \delta_{\mathbf{y}}}{\delta_{\mathbf{x}} + \delta_{\mathbf{y}}} \right|^{k-p} \\
&= \frac{1}{|(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}|} \sum_{k=0}^{r-1} \binom{k+|d|}{k} \tau^k \left(\sum_{p=0}^k \binom{k}{p} \left(\frac{\delta_{\mathbf{x}}}{\delta_{\mathbf{x}} + \delta_{\mathbf{y}}} \right)^p \left(\frac{\delta_{\mathbf{y}}}{\delta_{\mathbf{x}} + \delta_{\mathbf{y}}} \right)^{k-p} \right) \\
&= \frac{1}{|(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}|} \sum_{k=0}^{r-1} \binom{k+|d|}{k} \tau^k \left(\frac{\delta_{\mathbf{x}}}{\delta_{\mathbf{x}} + \delta_{\mathbf{y}}} + \frac{\delta_{\mathbf{y}}}{\delta_{\mathbf{x}} + \delta_{\mathbf{y}}} \right)^k \\
&= \frac{1}{|(o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}|} \sum_{k=0}^{r-1} \binom{k+|d|}{k} \tau^k \\
&\leq \frac{|\kappa(x, y)|}{(1-\tau)^{1+d}} \sum_{k=0}^{\infty} \binom{k+|d|}{k} \tau^k = \frac{|\kappa(x, y)|}{(1-\tau)^{1+d}} \frac{1}{(1-\tau)^{1+d}} \leq \frac{|\kappa(x, y)|}{(1-\tau)^{2+2|d|}},
\end{aligned}$$

where we use $\delta_{\mathbf{x}} + \delta_{\mathbf{y}} \leq \tau |o_{\mathbf{x}} - o_{\mathbf{y}}|$ in the second inequality, and (2.22) in the third inequality. Choosing x and y such that $|\kappa(x, y)| = \min_{i,j} |K_{ij}|$ yields the bound of $\|B\|_{1,1}$ in (2.19). The norm bounds in (2.20) can be also verified via

$$\begin{aligned}
\|U\|_2 &\leq \|U\|_{\mathbb{F}} \leq \sqrt{rn}, \quad \|V\|_2 \leq \|V\|_{\mathbb{F}} \leq \sqrt{rm}, \\
\|B\|_2 &\leq \sqrt{\|B\|_1 \|B\|_{\infty}} \leq \|B\|_{1,1} \leq \frac{\min_{i,j} |K_{ij}|}{(1-\tau)^{2+2|d|}} \leq \frac{\|K\|_2}{(1-\tau)^{2+2|d|}}.
\end{aligned}$$

The proof is completed. \square

Note that Theorem 2.3.1 has the assumption $d \in \mathbb{Z}$. If \mathbf{x} and \mathbf{y} are subsets of a straight line in \mathbb{C} , Theorem 2.3.1 can be generalized to $\kappa(x, y) = \frac{1}{(x-y)^{1+d}}$, $d \in \mathbb{C}$. This will be particularly useful when points in \mathbf{x} and \mathbf{y} are real.

Corollary 2.3.2. *If \mathbf{x} and \mathbf{y} are subsets of a straight line in \mathbb{C} , then in Theorem 2.3.1, the assumption $d \in \mathbb{Z}$ can be relaxed to $d \in \mathbb{C}$.*

Proof. Note that in Theorem 2.3.1, the assumption $d \in \mathbb{Z}$ is only needed for the identity (2.21) and the inequality (2.22). If \mathbf{x} and \mathbf{y} are subsets of a straight line, then the centers

$o_{\mathbf{x}}$ and $o_{\mathbf{y}}$ are also on that line. Hence, $t = \frac{(x-o_{\mathbf{x}})-(y-o_{\mathbf{y}})}{o_{\mathbf{y}}-o_{\mathbf{x}}} \in \mathbb{R}$. Since $|t| \leq \tau < 1$, we must have $1 - t > 0$. Therefore, the identity (2.21)

$$(x - y)^{1+d} = (o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d}(1 - t)^{1+d}$$

still holds for $d \in \mathbb{C}$. We can use the inequality

$$|z|^{\operatorname{Re} \mu} e^{-\pi |\operatorname{Im} \mu|} \leq |z^\mu| \leq |z|^{\operatorname{Re} \mu} e^{\pi |\operatorname{Im} \mu|}, \quad z \in \mathbb{C}, \quad \mu \in \mathbb{C},$$

to extend the inequality (2.22) to the more general one

$$\left| (o_{\mathbf{x}} - o_{\mathbf{y}})^{1+d} \right| \geq e^{-2\pi |\operatorname{Im}(1+d)|} (1 - \tau)^{|\operatorname{Re}(1+d)|} \left| (x - y)^{1+d} \right|. \quad (2.24)$$

Then the rest of the proof of Theorem 2.3.1 can go through with an extra factor $e^{2\pi |\operatorname{Im}(1+d)|}$.

□

2.3.2 Poisson kernel

For the Poisson kernel $\log \frac{1}{|x-y|}$, we can also use Taylor expansion to establish (2.10) and (2.11) for well-separated \mathbf{x} and \mathbf{y} . To be more specific, we have the following theorem.

Theorem 2.3.3. *Suppose $\kappa(x, y) = \log \frac{1}{|x-y|}$. Suppose $\mathbf{x} = \{x_i\}_{i=1}^n$ and $\mathbf{y} = \{y_j\}_{j=1}^m$ are well-separated with separation ratio τ . Let $\epsilon > 0$ be any small positive number, then the kernel matrix $K = (\kappa(x_i, y_j))_{x_i \in \mathbf{x}, y_j \in \mathbf{y}}$ has a rank $r = O\left(\frac{\log \epsilon}{\log \tau}\right)$ approximation*

$$K = \operatorname{Re}(UBV^T) + E, \quad \text{where } |E| \leq \epsilon. \quad (2.25)$$

These matrices have the following forms

$$U = \left(\left(\frac{x_i - o_{\mathbf{x}}}{\delta_{\mathbf{x}}} \right)^p \right)_{n \times r}, \quad V = \left(\left(\frac{y_j - o_{\mathbf{y}}}{\delta_{\mathbf{y}}} \right)^l \right)_{m \times r}, \quad (2.26)$$

$$B = \begin{pmatrix} b_{00} & b_{01} & \cdots & b_{0,r-1} \\ b_{10} & \cdots & b_{1,r-2} & \\ \vdots & \ddots & & \\ b_{r-1,0} & & & \end{pmatrix}_{r \times r}, \quad (2.27)$$

$$b_{pl} = \begin{cases} \log \frac{1}{|o_{\mathbf{x}} - o_{\mathbf{y}}|}, & p = l = 0 \\ \frac{(-1)^p}{p+l} \binom{p+l}{p} \left(\frac{\delta_{\mathbf{x}}}{o_{\mathbf{x}} - o_{\mathbf{y}}} \right)^p \left(\frac{\delta_{\mathbf{y}}}{o_{\mathbf{x}} - o_{\mathbf{y}}} \right)^l, & 0 < p + l \leq r - 1 \end{cases}. \quad (2.28)$$

Moreover, the entries b_{pl} can be computed efficiently and stably via the recurrence relation

$$\begin{cases} b_{p,-1} = b_{-1,l} = 0, \\ b_{00} = \log \frac{1}{|o_{\mathbf{x}} - o_{\mathbf{y}}|}, \quad b_{10} = \frac{-\delta_{\mathbf{x}}}{o_{\mathbf{x}} - o_{\mathbf{y}}}, \quad b_{01} = \frac{\delta_{\mathbf{y}}}{o_{\mathbf{x}} - o_{\mathbf{y}}}, \\ b_{pl} = \frac{p+l-1}{p+l} \left(\frac{\delta_{\mathbf{y}}}{o_{\mathbf{x}} - o_{\mathbf{y}}} b_{p,l-1} - \frac{\delta_{\mathbf{x}}}{o_{\mathbf{x}} - o_{\mathbf{y}}} b_{p-1,l} \right), \quad 2 \leq p + l \leq r - 1. \end{cases} \quad (2.29)$$

Furthermore, their entries satisfy the following bounds

$$\|U\|_{\max} \leq 1, \quad \|V\|_{\max} \leq 1, \quad \|B\|_{1,1} \leq \min_{i,j} |K_{ij}| + 2 \log \frac{1}{1 - \tau}. \quad (2.30)$$

As corollaries, their norms satisfy the following bounds

$$\|U\|_2 \leq \sqrt{rn}, \quad \|V\|_2 \leq \sqrt{rm}, \quad \|B\|_2 \leq \|K\|_2 + 2 \log \frac{1}{1 - \tau}. \quad (2.31)$$

Proof. Let $x \in \mathbf{x}$, $y \in \mathbf{y}$, $t = \frac{(x - o_{\mathbf{x}}) - (y - o_{\mathbf{y}})}{o_{\mathbf{y}} - o_{\mathbf{x}}}$, then $x - y = (o_{\mathbf{x}} - o_{\mathbf{y}})(1 - t)$. Since \mathbf{x} and \mathbf{y} are well separated, we have $|t| \leq \tau < 1$. By Taylor expansion,

$$\begin{aligned} \log \frac{1}{1 - t} &= \sum_{k=1}^{r-1} \frac{t^k}{k} + \tilde{E}_r(t) \\ &= \sum_{k=1}^{r-1} \sum_{p=0}^k \frac{(-1)^p}{k} \binom{k}{p} \frac{(x - o_{\mathbf{x}})^p (y - o_{\mathbf{y}})^{k-p}}{(o_{\mathbf{x}} - o_{\mathbf{y}})^k} + \tilde{E}_r(t) \\ &= \sum_{k=1}^{r-1} \sum_{p=0}^k b_{p,k-p} \left(\frac{x - o_{\mathbf{x}}}{\delta_{\mathbf{x}}} \right)^p \left(\frac{y - o_{\mathbf{y}}}{\delta_{\mathbf{y}}} \right)^{k-p} + \tilde{E}_r(t), \end{aligned}$$

where $|\tilde{E}_r(t)| \leq \sum_{p=r}^{\infty} \frac{\tau^p}{p} \leq \frac{\tau^r}{r(1-\tau)}$. Then

$$\begin{aligned}
\log \frac{1}{|x-y|} &= \log \frac{1}{|o_x - o_y|} + \log \frac{1}{|1-t|} \\
&= \log \frac{1}{|o_x - o_y|} + \operatorname{Re} \left(\log \frac{1}{1-t} \right) \\
&= \operatorname{Re} \left(b_{00} + \log \frac{1}{1-t} \right) \\
&= \operatorname{Re} \left(b_{00} + \sum_{k=1}^{r-1} \sum_{p=0}^k b_{p,k-p} \left(\frac{x-o_x}{\delta_x} \right)^p \left(\frac{y-o_y}{\delta_y} \right)^{k-p} + \tilde{E}_r(t) \right) \\
&= \operatorname{Re} \left(\sum_{k=0}^{r-1} \sum_{p=0}^k b_{p,k-p} \left(\frac{x-o_x}{\delta_x} \right)^p \left(\frac{y-o_y}{\delta_y} \right)^{k-p} \right) + \operatorname{Re} \left(\tilde{E}_r(t) \right).
\end{aligned}$$

If the expansion order $r = O\left(\frac{\log \epsilon}{\log \tau}\right)$, we will have $|\tilde{E}_r(t)| \leq \epsilon$. This establishes (2.25)-(2.28).

Next, we show the recurrence relation (2.29). By definition, $b_{00} = \log \frac{1}{|o_x - o_y|}$, $b_{10} = \frac{-\delta_x}{o_x - o_y}$, $b_{01} = \frac{\delta_y}{o_x - o_y}$. For $p+l \geq 2$, we can use the identity $\binom{p+l}{p} = \binom{p+l-1}{p-1} + \binom{p+l-1}{p}$ to get

$$\begin{aligned}
\frac{p+l}{p+l-1} b_{pl} &= \frac{(-1)^p}{p+l-1} \binom{p+l}{p} \left(\frac{\delta_x}{o_x - o_y} \right)^p \left(\frac{\delta_y}{o_x - o_y} \right)^l \\
&= \frac{(-1)^p}{p+l-1} \binom{p+l-1}{p-1} \left(\frac{\delta_x}{o_x - o_y} \right)^p \left(\frac{\delta_y}{o_x - o_y} \right)^l \\
&\quad + \frac{(-1)^p}{p+l-1} \binom{p+l-1}{p} \left(\frac{\delta_x}{o_x - o_y} \right)^p \left(\frac{\delta_y}{o_x - o_y} \right)^l \\
&= -b_{p-1,l} \frac{\delta_x}{o_x - o_y} + b_{p,l-1} \frac{\delta_y}{o_x - o_y}.
\end{aligned}$$

which establish the recurrence relation (2.29).

Next, we prove the entry-wise bounds (2.30). Since $|x - o_x| \leq \delta_x$ and $|y - o_y| \leq \delta_y$, we have $\|U\|_{\max} \leq 1$ and $\|V\|_{\max} \leq 1$. To show the bound on $\|B\|_{1,1}$,

$$\begin{aligned}
\|B\|_{1,1} &= |b_{00}| + \sum_{k=1}^{r-1} \sum_{p=0}^k |b_{p,k-p}| \\
&= |b_{00}| + \sum_{k=1}^{r-1} \frac{1}{k} \sum_{p=0}^k \binom{k}{p} \left| \frac{\delta_x}{o_x - o_y} \right|^p \left| \frac{\delta_y}{o_x - o_y} \right|^{k-p}
\end{aligned}$$

$$\begin{aligned}
&\leq |b_{00}| + \sum_{k=1}^{r-1} \frac{1}{k} \sum_{p=0}^k \binom{k}{p} \left| \frac{\tau \delta_{\mathbf{x}}}{\delta_{\mathbf{x}} + \delta_{\mathbf{y}}} \right|^p \left| \frac{\tau \delta_{\mathbf{y}}}{\delta_{\mathbf{x}} + \delta_{\mathbf{y}}} \right|^{k-p} \\
&= |b_{00}| + \sum_{k=1}^{r-1} \frac{\tau^k}{k} \left(\sum_{p=0}^k \binom{k}{p} \left(\frac{\delta_{\mathbf{x}}}{\delta_{\mathbf{x}} + \delta_{\mathbf{y}}} \right)^p \left(\frac{\delta_{\mathbf{y}}}{\delta_{\mathbf{x}} + \delta_{\mathbf{y}}} \right)^{k-p} \right) \\
&= |b_{00}| + \sum_{k=1}^{r-1} \frac{\tau^k}{k} \left(\frac{\delta_{\mathbf{x}}}{\delta_{\mathbf{x}} + \delta_{\mathbf{y}}} + \frac{\delta_{\mathbf{y}}}{\delta_{\mathbf{x}} + \delta_{\mathbf{y}}} \right)^k \\
&\leq |b_{00}| + \sum_{k=1}^{\infty} \frac{\tau^k}{k} \\
&= \left| \log \frac{1}{|o_{\mathbf{x}} - o_{\mathbf{y}}|} \right| + \log \frac{1}{1 - \tau} \leq \left| \log \frac{1}{|x - y|} \right| + 2 \log \frac{1}{1 - \tau},
\end{aligned}$$

where we use $\delta_{\mathbf{x}} + \delta_{\mathbf{y}} \leq \tau |o_{\mathbf{x}} - o_{\mathbf{y}}|$ in the first inequality, and $(1 - \tau)|x - y| \leq |o_{\mathbf{x}} - o_{\mathbf{y}}| \leq \frac{1}{1 - \tau}|x - y|$ in the last inequality. Choosing x and y such that $|\kappa(x, y)| = \min_{i,j} |K_{ij}|$ gives the bound of $\|B\|_{1,1}$ in (2.30). The norm bounds in (2.31) can also be verified via

$$\begin{aligned}
\|U\|_2 &\leq \|U\|_{\mathbb{F}} \leq \sqrt{rn}, \quad \|V\|_2 \leq \|V\|_{\mathbb{F}} \leq \sqrt{rm}, \\
\|B\|_2 &\leq \sqrt{\|B\|_1 \|B\|_{\infty}} \leq \|B\|_{1,1} \leq \min_{i,j} |K_{ij}| + 2 \log \frac{1}{1 - \tau} \leq \|K\|_2 + 2 \log \frac{1}{1 - \tau}.
\end{aligned}$$

The proof is completed. \square

Note that in Theorem 2.3.3, $\log \frac{1}{|x-y|} = \operatorname{Re} \left(\log \frac{1}{x-y} \right)$, and we need to take the real part of the low-rank approximation $\operatorname{Re} \left(UBV^T \right)$. If \mathbf{x} and \mathbf{y} are subsets of a straight line in \mathbb{C} , Theorem 2.3.3 can be generalized to $\kappa(x, y) = \log \frac{1}{x-y}$, where \log is the principle branch of natural logarithm. This will be particularly useful when points in \mathbf{x} and \mathbf{y} are real.

Corollary 2.3.4. *If \mathbf{x} and \mathbf{y} are subsets of a straight line in \mathbb{C} , then Theorem 2.3.3 can be generalized to $\kappa(x, y) = \log \frac{1}{x-y}$, such that*

$$K = UBV^T + E, \quad \text{where } |E| \leq \epsilon,$$

where U , B and V are the same as in Theorem 2.3.3 except that $b_{00} = \log \frac{1}{o_{\mathbf{x}} - o_{\mathbf{y}}}$.

Proof. If \mathbf{x} and \mathbf{y} are subsets of a straight line, then the centers $o_{\mathbf{x}}$ and $o_{\mathbf{y}}$ are also on that line. Hence, $t = \frac{(x - o_{\mathbf{x}}) - (y - o_{\mathbf{y}})}{o_{\mathbf{y}} - o_{\mathbf{x}}} \in \mathbb{R}$. Since $|t| \leq \tau < 1$, we must have $1 - t > 0$. Therefore,

the identity $\log \frac{1}{x-y} = \log \frac{1}{o_{\mathbf{x}}-o_{\mathbf{y}}} + \log \frac{1}{1-t}$ holds, and the rest of the proof of Theorem 2.3.3 can go through. \square

2.3.3 Helmholtz kernel

For the Helmholtz kernel $H_0(|x-y|)$, we can use Graf's addition formula to establish (2.10) and (2.11) for well-separated \mathbf{x} and \mathbf{y} . To be more specific, we have the following theorem, which is implicitly derived in [2], [31]. We assume that the size of \mathbf{x} and \mathbf{y} are comparable such that $\alpha\delta_{\mathbf{y}} \leq \delta_{\mathbf{x}} \leq \frac{1}{\alpha}\delta_{\mathbf{y}}$, $0 < \alpha \leq 1$.

Theorem 2.3.5. *Suppose $\kappa(x, y) = H_0(|x-y|)$. Suppose $\mathbf{x} = \{x_i\}_{i=1}^n$ and $\mathbf{y} = \{y_j\}_{j=1}^m$ are well-separated with separation ratio τ . Let $\epsilon > 0$ be any small positive number, and $r = O\left(\frac{\delta_{\mathbf{x}}+\delta_{\mathbf{y}}}{\tau} + \frac{\log \epsilon}{\log \tau}\right)$, then the kernel matrix $K = (\kappa(x_i, y_j))_{x_i \in \mathbf{x}, y_j \in \mathbf{y}}$ has a $(2r+1)$ -rank approximation*

$$K = UBV^T + E, \quad \text{where } |E| \leq \epsilon. \quad (2.32)$$

These matrices have the following forms

$$U = \begin{pmatrix} g_{-r}(x_1 - o_{\mathbf{x}}) & \cdots & g_r(x_1 - o_{\mathbf{x}}) \\ \vdots & \ddots & \vdots \\ g_{-r}(x_n - o_{\mathbf{x}}) & \cdots & g_r(x_n - o_{\mathbf{x}}) \end{pmatrix}_{n \times (2r+1)}, \quad (2.33)$$

$$V = \begin{pmatrix} g_{-r}(y_1 - o_{\mathbf{y}}) & \cdots & g_r(y_1 - o_{\mathbf{y}}) \\ \vdots & \ddots & \vdots \\ g_{-r}(y_m - o_{\mathbf{y}}) & \cdots & g_r(y_m - o_{\mathbf{y}}) \end{pmatrix}_{m \times (2r+1)}, \quad (2.34)$$

$$B = \begin{pmatrix} & b_{-r,0} & \cdots & b_{-r,r} \\ & \ddots & & \vdots \\ b_{0,-r} & & \ddots & b_{0,r} \\ \vdots & \ddots & & \ddots \\ b_{r,-r} & \cdots & b_{r,0} & \end{pmatrix}_{(2r+1) \times (2r+1)}, \quad (2.35)$$

where for $-r \leq p, l \leq r$, we write

$$g_p(z) = J_p(|z|)e^{ip\theta_z}, \quad b_{pl} = (-1)^l H_{p+l}(|o_{\mathbf{x}} - o_{\mathbf{y}}|)e^{-i(p+l)\theta_{o_{\mathbf{y}} - o_{\mathbf{x}}}}.$$

And the entries of U, V, B satisfy

$$\|U\|_{\max} \leq 1, \quad \|V\|_{\max} \leq 1, \quad \|B\|_{\max} \geq \sqrt{\frac{2}{\pi r}} \left(\frac{2r}{e|o_{\mathbf{x}} - o_{\mathbf{y}}|} \right)^r C_r(|o_{\mathbf{x}} - o_{\mathbf{y}}|). \quad (2.36)$$

Proof. Let $x \in \mathbf{x}, y \in \mathbf{y}, t = (x - o_{\mathbf{x}}) - (y - o_{\mathbf{y}}), w = o_{\mathbf{y}} - o_{\mathbf{x}}$, then $x - y = -(w - t)$. Since \mathbf{x} and \mathbf{y} are well separated, we have $|t| \leq \tau|w| < |w|$. Apply Graf's addition formula (2.9) to $H_0(|x - y|) = H_0(|w - t|)$, then we have

$$H_0(|x - y|) = \sum_{|k| \leq r} H_k(|w|)e^{-ik\theta_w} J_k(|t|)e^{ik\theta_t} + E_{r+1}^{H_0}(w, t). \quad (2.37)$$

Apply Graf's addition formula (2.8) to $J_k(|t|)e^{ik\theta_t}$, then we have

$$\begin{aligned} J_k(|t|)e^{ik\theta_t} &= \sum_{l=-\infty}^{\infty} J_{k-l}(|x - o_{\mathbf{x}}|)e^{i(k-l)\theta_{x-o_{\mathbf{x}}}} J_{-l}(|y - o_{\mathbf{y}}|)e^{il\theta_{y-o_{\mathbf{y}}}} \\ &= \sum_{|l| \leq r, |k-l| \leq r} (-1)^l g_{k-l}(x - o_{\mathbf{x}}) g_l(y - o_{\mathbf{y}}) + E_{r+1}^{J_k}(x - o_{\mathbf{x}}, y - o_{\mathbf{y}}). \end{aligned} \quad (2.38)$$

Therefore, substituting (2.38) into (2.37) we have

$$\begin{aligned} &H_0(|x - y|) \\ &= \sum_{|k| \leq r} H_k(|w|)e^{-ik\theta_w} \left(\sum_{|l| \leq r, |k-l| \leq r} (-1)^l g_{k-l}(x - o_{\mathbf{x}}) g_l(y - o_{\mathbf{y}}) \right) + E(x, y) \\ &= \sum_{|k| \leq r} \left(\sum_{|l| \leq r, |k-l| \leq r} H_k(|w|)e^{-ik\theta_w} (-1)^l g_{k-l}(x - o_{\mathbf{x}}) g_l(y - o_{\mathbf{y}}) \right) + E(x, y) \\ &= \sum_{|l| \leq r, |p| \leq r} \left(\sum_{|p+l| \leq r} b_{pl} g_p(x - o_{\mathbf{x}}) g_l(y - o_{\mathbf{y}}) \right) + E(x, y), \end{aligned}$$

which establishes (2.32)-(2.35) with the remainder

$$\begin{aligned} |E(x, y)| &= \left| E_{r+1}^{H_0}(w, t) + \sum_{|k| \leq r} H_k(|w|) e^{-ik\theta w} \cdot E_{r+1}^{J_k}(x - o_{\mathbf{x}}, y - o_{\mathbf{y}}) \right| \\ &\leq \left| E_{r+1}^{H_0}(w, t) \right| + \sum_{|k| \leq r} \left| H_k(|w|) \cdot E_{r+1}^{J_k}(x - o_{\mathbf{x}}, y - o_{\mathbf{y}}) \right|. \end{aligned}$$

Since $|t| \leq \delta_{\mathbf{x}} + \delta_{\mathbf{y}} \leq \tau|w|$, we can set $t_{\max} = \delta_{\mathbf{x}} + \delta_{\mathbf{y}}$ in Lemma 2.2.4, and bound

$$\left| E_{r+1}^{H_0}(w, t) \right| \leq \frac{2\sqrt{2}C_{r+1} \left(\frac{t_{\max}}{\tau} \right) \tau^{r+1}}{\pi(r+1) 1 - \tau} = O \left(\frac{C_{r+1} \left(\frac{t_{\max}}{\tau} \right) \tau^{r+1}}{r+1} \right), \quad r \geq \frac{t_{\max}}{\tau}.$$

Since $\frac{\delta_{\mathbf{x}} + \delta_{\mathbf{y}}}{1 + \alpha} \geq \max(\delta_{\mathbf{x}}, \delta_{\mathbf{y}})$, we can set $z_{\max} = \frac{t_{\max}}{1 + \alpha}$ in Lemma 2.2.3 and bound

$$\left| E_{r+1}^{J_k}(x - o_{\mathbf{x}}, y - o_{\mathbf{y}}) \right| \leq 4 \sum_{l=r+1}^{\infty} \left| J_l \left(\frac{t_{\max}}{1 + \alpha} \right) \right|.$$

Since $|H_k(z)|$ is increasing in $|k|$ and decreasing in z , we can bound

$$\begin{aligned} &\left| H_k(|w|) \cdot E_{r+1}^{J_k}(x - o_{\mathbf{x}}, y - o_{\mathbf{y}}) \right| \\ &\leq \left| H_k \left(\frac{t_{\max}}{\tau} \right) \cdot E_{r+1}^{J_k}(x - o_{\mathbf{x}}, y - o_{\mathbf{y}}) \right| \leq 4 \left| H_k \left(\frac{t_{\max}}{\tau} \right) \cdot \sum_{l=r+1}^{\infty} \left| J_l \left(\frac{t_{\max}}{1 + \alpha} \right) \right| \right| \\ &\leq 4 \sum_{l=r+1}^{\infty} \left| H_l \left(\frac{t_{\max}}{\tau} \right) J_l \left(\frac{t_{\max}}{1 + \alpha} \right) \right| \leq 4\sqrt{2} \sum_{l=r+1}^{\infty} \left| Y_l \left(\frac{t_{\max}}{\tau} \right) J_l \left(\frac{t_{\max}}{1 + \alpha} \right) \right| \\ &\leq 4\sqrt{2} \sum_{l=r+1}^{\infty} C_l \left(\frac{t_{\max}}{\tau} \right) \sqrt{\frac{2}{\pi l}} \left(\frac{2l\tau}{et_{\max}} \right)^l \frac{1}{\sqrt{2\pi l}} \left(\frac{et_{\max}}{2l(1 + \alpha)} \right)^l \\ &\leq \frac{4\sqrt{2}}{\pi(r+1)} \sum_{l=r+1}^{\infty} C_l \left(\frac{t_{\max}}{\tau} \right) \left(\frac{\tau}{1 + \alpha} \right)^l = O \left(\frac{C_{r+1} \left(\frac{t_{\max}}{\tau} \right)}{r+1} \left(\frac{\tau}{1 + \alpha} \right)^{r+1} \right). \end{aligned}$$

Therefore,

$$\begin{aligned} |E(x, y)| &\leq \left| E_{r+1}^{H_0}(w, t) \right| + \sum_{|k| \leq r} \left| H_k(|w|) \cdot E_{r+1}^{J_k}(x - o_{\mathbf{x}}, y - o_{\mathbf{y}}) \right| \\ &= O \left(\frac{C_{r+1} \left(\frac{t_{\max}}{\tau} \right) \tau^{r+1}}{r+1} \right) + O \left(C_{r+1} \left(\frac{t_{\max}}{\tau} \right) \left(\frac{\tau}{1 + \alpha} \right)^{r+1} \right). \end{aligned}$$

By Proposition 2.2.2, $C_{r+1} \left(\frac{t_{\max}}{\tau} \right)$ is monotonically decreasing to 1 as $r \rightarrow \infty$. As a result, we have the bound $|E(x, y)| \leq \epsilon$ if $r = O \left(\frac{\delta_{\mathbf{x}} + \delta_{\mathbf{y}}}{\tau} + \frac{\log \epsilon}{\log \tau} \right)$.

Next we show the entries bounds (2.36). According to Proposition 2.2.1, $|J_p(z)| \leq 1$ for all $p \in \mathbb{Z}$ and $z > 0$. Hence, we have $\|U\|_{\max} \leq 1$ and $\|V\|_{\max} \leq 1$. By Proposition 2.2.4(i), $|H_k(|o_{\mathbf{x}} - o_{\mathbf{y}}|)|$ is strictly increasing in $|k|$, we have

$$\|B\|_{\max} = |H_r(|o_{\mathbf{x}} - o_{\mathbf{y}}|)| \geq |Y_r(|o_{\mathbf{x}} - o_{\mathbf{y}}|)| = \sqrt{\frac{2}{\pi r}} \left(\frac{2r}{e|o_{\mathbf{x}} - o_{\mathbf{y}}|} \right)^r C_r(|o_{\mathbf{x}} - o_{\mathbf{y}}|).$$

The proof is completed. \square

In the cases of generalized Cauchy kernel and Poisson kernel, the entries and norms of the B matrices are bounded (see Theorems 2.3.1 and 2.3.3). In contrast, in the low-rank factorization (2.32) for Helmholtz kernel, many entries of B are extremely large if $r > \frac{e|o_{\mathbf{x}} - o_{\mathbf{y}}|}{2}$. In particular, when $o_{\mathbf{x}} - o_{\mathbf{y}}$ is very small, or when a large truncation order r is used for high accuracy, we will have $\frac{2r}{e|o_{\mathbf{x}} - o_{\mathbf{y}}|} \gg 1$, and hence $\|B\|_{\max}$ grows very fast. This may have stability risk. It is often preferred to have bounded entries in B . For this purpose, we follow the diagonal-scaling strategy in [32] to scale the low-rank factors U , B and V . The diagonal-scaling in [32] is based on Stirling's formula, while we use the asymptotic behaviors of J_p and H_p (see Propositions 2.2.1, 2.2.2, 2.2.3, 2.2.4) to design the scaling factors.

For $0 \leq p \leq r$, define the *scaling factors* for \mathbf{x} and \mathbf{y} respectively,

$$\lambda_p = \lambda_{-p} = \max \left(1, p! \left(\frac{2}{\delta_{\mathbf{x}}} \right)^p \right), \quad \omega_p = \omega_{-p} = \max \left(1, p! \left(\frac{2}{\delta_{\mathbf{y}}} \right)^p \right).$$

Lemma 2.3.6. *The sequences $\{\lambda_p\}$ and $\{\omega_p\}$ are increasing in $|p|$.*

Proof. Assume $p \geq 0$, and let $a_p = p! \left(\frac{2}{\delta_{\mathbf{x}}} \right)^p$, then $\frac{a_{p+1}}{a_p} = \frac{2(p+1)}{\delta_{\mathbf{x}}}$. The sequence $\{a_p\}$ is either monotonically increasing, or first monotonically decreasing then monotonically increasing. In either case, we have

$$a_p \leq \max(a_0, a_{p+1}) = \max \left(1, (p+1)! \left(\frac{2}{\delta_{\mathbf{x}}} \right)^{p+1} \right) = \lambda_{p+1}.$$

Therefore, $\lambda_p = \max(1, a_p) \leq \lambda_{p+1}$. The proof for ω_p is analogous. \square

With the scaling factors, we scale the low-rank approximation (2.32) as

$$UBV^T = (U\Lambda) \cdot (\Lambda^{-1}B\Omega^{-1}) \cdot (V\Omega)^T = \hat{U}\hat{B}\hat{V}^T, \quad \text{where}$$

$$\Lambda = \text{diag}(\lambda_{-r} \quad \cdots \quad \lambda_r), \quad \Omega = \text{diag}(\omega_{-r} \quad \cdots \quad \omega_r).$$

Then we can show that the scaled matrices have bounded entries.

Theorem 2.3.7. *With the same assumption in Theorem 2.3.5 and assume $\tau \leq \frac{2}{e}$. We have the scaled low-rank approximation*

$$K = \hat{U}\hat{B}\hat{V}^T + E, \quad \text{where } |E| \leq \epsilon. \quad (2.39)$$

such that \hat{U} , \hat{B} and \hat{V} have the following forms,

$$\hat{U} = \begin{pmatrix} g_{-r}(x_1 - o_{\mathbf{x}})\lambda_{-r} & \cdots & g_r(x_1 - o_{\mathbf{x}})\lambda_r \\ \vdots & \ddots & \vdots \\ g_{-r}(x_n - o_{\mathbf{x}})\lambda_{-r} & \cdots & g_r(x_n - o_{\mathbf{x}})\lambda_r \end{pmatrix}_{n \times (2r+1)}, \quad (2.40)$$

$$\hat{V} = \begin{pmatrix} g_{-r}(y_1 - o_{\mathbf{y}})\omega_{-r} & \cdots & g_r(y_1 - o_{\mathbf{y}})\omega_r \\ \vdots & \ddots & \vdots \\ g_{-r}(y_m - o_{\mathbf{y}})\omega_{-r} & \cdots & g_r(y_m - o_{\mathbf{y}})\omega_r \end{pmatrix}_{m \times (2r+1)}, \quad (2.41)$$

$$\hat{B} = \begin{pmatrix} & \hat{b}_{-r,0} & \cdots & \hat{b}_{-r,r} \\ & \ddots & & \vdots \\ \hat{b}_{0,-r} & & \ddots & \hat{b}_{0,r} \\ \vdots & \ddots & & \ddots \\ \hat{b}_{r,-r} & \cdots & \hat{b}_{r,0} & \end{pmatrix}_{(2r+1) \times (2r+1)}, \quad (2.42)$$

where for $-r \leq p, l \leq r$

$$\hat{b}_{pl} = (-1)^l \lambda_p^{-1} \omega_l^{-1} H_{p+l}(|o_{\mathbf{x}} - o_{\mathbf{y}}|) e^{-i(p+l)\theta_{o_{\mathbf{y}} - o_{\mathbf{x}}}}. \quad (2.43)$$

And their entries satisfy the following bounds.

$$\|\hat{U}\|_{\max} \leq 1, \quad \|\hat{V}\|_{\max} \leq 1, \quad \|\hat{B}\|_{\max} \leq \max\left(\frac{8}{\pi}, \sqrt{1+\tau} \min_{i,j} |K_{ij}|\right). \quad (2.44)$$

Proof. For $1 \leq i \leq n$ and $-r \leq p \leq r$, by Proposition 2.2.1, we have

$$\begin{aligned} |\hat{U}_{ip}| &= |g_p(x_i - o_{\mathbf{x}})\lambda_p| \\ &= \max\left(|J_p(|x_i - o_{\mathbf{x}}|)|, |J_p(|x_i - o_{\mathbf{x}}|)| \cdot |p|! \left(\frac{2}{\delta_{\mathbf{x}}}\right)^{|p|}\right) \\ &\leq \max\left(1, \frac{1}{|p|!} \left(\frac{|x_i - o_{\mathbf{x}}|}{2}\right)^{|p|} |p|! \left(\frac{2}{\delta_{\mathbf{x}}}\right)^{|p|}\right) \\ &= \max\left(1, \left(\frac{|x_i - o_{\mathbf{x}}|}{\delta_{\mathbf{x}}}\right)^{|p|}\right) \leq 1. \end{aligned}$$

Therefore, $\|\hat{U}\|_{\max} \leq 1$. Similarly, we can show $\|\hat{V}\|_{\max} \leq 1$.

Next, we show the upper bound for entries of \hat{B} . Note that for $-r \leq p, l \leq r$,

$$|\hat{b}_{pl}| = \left|\lambda_p^{-1}\omega_l^{-1}H_{p+l}(|o_{\mathbf{x}} - o_{\mathbf{y}}|)\right|. \quad (2.45)$$

By Proposition 2.2.4(i), we have $|H_{p+l}(|o_{\mathbf{x}} - o_{\mathbf{y}}|)| \leq |H_{|p+l|}(|o_{\mathbf{x}} - o_{\mathbf{y}}|)|$. Therefore, without loss of generality, we can assume $0 \leq p, l \leq r$ in (2.45). Note that by definition of the scaling factors,

$$\lambda_p^{-1} = \min\left(1, \frac{1}{p!} \left(\frac{\delta_{\mathbf{x}}}{2}\right)^p\right), \quad \omega_l^{-1} = \min\left(1, \frac{1}{l!} \left(\frac{\delta_{\mathbf{y}}}{2}\right)^l\right).$$

There are three cases to discuss.

(i) $p = l = 0$. If $|o_{\mathbf{x}} - o_{\mathbf{y}}| \geq |x - y|$, then by Proposition 2.2.4(ii),

$$|\hat{b}_{00}| = |H_0(|o_{\mathbf{x}} - o_{\mathbf{y}}|)| \leq |H_0(|x - y|)| \leq \sqrt{1+\tau} |H_0(|x - y|)|.$$

If $|o_{\mathbf{x}} - o_{\mathbf{y}}| < |x - y|$, then by Proposition 2.2.4(iv)

$$\begin{aligned} |\hat{b}_{00}| &= \left| H_0 \left(\frac{|o_{\mathbf{x}} - o_{\mathbf{y}}|}{|x - y|} \cdot |x - y| \right) \right| \leq \sqrt{\frac{|x - y|}{|o_{\mathbf{x}} - o_{\mathbf{y}}|}} \cdot |H_0(|x - y|)| \\ &\leq \sqrt{1 + \tau} |H_0(|x - y|)|. \end{aligned}$$

(ii) $1 \leq p + l \leq |o_{\mathbf{x}} - o_{\mathbf{y}}|$. In this case, according to Proposition 2.2.4(iii)

$$|\hat{b}_{pl}| \leq \left| \lambda_p^{-1} \omega_l^{-1} H_{p+l}(|o_{\mathbf{x}} - o_{\mathbf{y}}|) \right| \leq |H_{p+l}(|o_{\mathbf{x}} - o_{\mathbf{y}}|)| \leq \sqrt{\frac{4}{\pi}}.$$

(iii) $p + l > |o_{\mathbf{x}} - o_{\mathbf{y}}|$. Note that by Propositions 2.2.1 and 2.2.2, we have

$$|H_{p+l}(|o_{\mathbf{x}} - o_{\mathbf{y}}|)| \leq \sqrt{2} |Y_{p+l}(|o_{\mathbf{x}} - o_{\mathbf{y}}|)|, \quad \text{and} \quad C_{p+l}(|o_{\mathbf{x}} - o_{\mathbf{y}}|) > 0.$$

Hence, we can use Stirling's approximation $\left(\frac{p+l}{e}\right)^{p+l} < \frac{(p+l)!}{\sqrt{2\pi(p+l)}}$ to bound

$$\begin{aligned} |\hat{b}_{pl}| &\leq \sqrt{2} \left| \lambda_p^{-1} \omega_l^{-1} Y_{p+l}(|o_{\mathbf{x}} - o_{\mathbf{y}}|) \right| \\ &\leq \frac{\sqrt{2}}{p!!} \left(\frac{\delta_{\mathbf{x}}}{2}\right)^p \left(\frac{\delta_{\mathbf{y}}}{2}\right)^l C_{p+l}(|o_{\mathbf{x}} - o_{\mathbf{y}}|) \sqrt{\frac{2}{\pi(p+l)}} \left(\frac{2(p+l)}{e|o_{\mathbf{x}} - o_{\mathbf{y}}|}\right)^{p+l} \\ &= \frac{2C_{p+l}(|o_{\mathbf{x}} - o_{\mathbf{y}}|)}{p!! \sqrt{\pi(p+l)}} \left(\frac{\delta_{\mathbf{x}}}{|o_{\mathbf{x}} - o_{\mathbf{y}}|}\right)^p \left(\frac{\delta_{\mathbf{y}}}{|o_{\mathbf{x}} - o_{\mathbf{y}}|}\right)^l \left(\frac{p+l}{e}\right)^{p+l} \\ &\leq \frac{\sqrt{2}C_{p+l}(|o_{\mathbf{x}} - o_{\mathbf{y}}|)}{\pi(p+l)} \frac{(p+l)!}{p!!} \left(\frac{\delta_{\mathbf{x}}}{|o_{\mathbf{x}} - o_{\mathbf{y}}|}\right)^p \left(\frac{\delta_{\mathbf{y}}}{|o_{\mathbf{x}} - o_{\mathbf{y}}|}\right)^l \\ &\leq \frac{\sqrt{2}C_{p+l}(|o_{\mathbf{x}} - o_{\mathbf{y}}|)}{\pi(p+l)} \tau^{p+l} \frac{(p+l)!}{p!!} \left(\frac{\delta_{\mathbf{x}}}{\delta_{\mathbf{x}} + \delta_{\mathbf{y}}}\right)^p \left(\frac{\delta_{\mathbf{y}}}{\delta_{\mathbf{x}} + \delta_{\mathbf{y}}}\right)^l \\ &\leq \frac{\sqrt{2}C_{p+l}(|o_{\mathbf{x}} - o_{\mathbf{y}}|)}{\pi(p+l)} \tau^{p+l}. \end{aligned}$$

By Propositions 2.2.3(i), 2.2.3(iii) and 2.2.4(iii), we can bound $C_{p+l}(|o_{\mathbf{x}} - o_{\mathbf{y}}|)$ by

$$\begin{aligned} C_{p+l}(|o_{\mathbf{x}} - o_{\mathbf{y}}|) &\leq \frac{4\sqrt{2}}{e} C_{p+l+1}(|o_{\mathbf{x}} - o_{\mathbf{y}}|) \\ &\leq \frac{4\sqrt{2}}{e} |H_{p+l+1}(p+l+1)| \sqrt{\frac{\pi(p+l+1)}{2}} \left(\frac{e}{2}\right)^{p+l+1} \end{aligned}$$

$$\begin{aligned}
&\leq \frac{4\sqrt{2}}{e} \sqrt{\frac{4}{\pi}} \sqrt{\frac{\pi(p+l+1)}{2}} \left(\frac{e}{2}\right)^{p+l+1} \\
&= 4\sqrt{p+l+1} \left(\frac{e}{2}\right)^{p+l}.
\end{aligned}$$

As a result, if $\tau \leq \frac{2}{e}$,

$$|\hat{b}_{pl}| \leq \frac{4\sqrt{2}\sqrt{p+l+1}}{\pi(p+l)} \left(\frac{e\tau}{2}\right)^{p+l} \leq \frac{4\sqrt{2}\sqrt{2(p+l)}}{\pi(p+l)} \left(\frac{e\tau}{2}\right)^{p+l} \leq \frac{8}{\pi}.$$

Combining these three cases, we have $\|B\|_{\max} \leq \max\left(\frac{8}{\pi}, \sqrt{1+\tau} \min_{i,j} |K_{ij}|\right)$. And the proof is completed. \square

Theorem 2.3.7 states that the entries in \hat{U} , \hat{V} and \hat{B} are bounded independent of the expansion order r . However, we avoid to compute them directly via definitions (2.40)-(2.43), since some of them could be the products of extremely large numbers and extremely small numbers. Analogous to the generalized Cauchy kernel and Poisson kernel, we shall compute those entries via recurrence formulas. Note that for Bessel functions, we do not have explicit formulas, but we can use their recurrence relations (2.4).

Theorem 2.3.8. *The entries of \hat{B} can be computed stably and efficiently via the following three-step recurrence relations*

$$\begin{cases} \hat{b}_{0,0} = H_0(|o_{\mathbf{x}} - o_{\mathbf{y}}|), \\ \hat{b}_{1,0} = \lambda_1^{-1} H_1(|o_{\mathbf{x}} - o_{\mathbf{y}}|) e^{-i\theta_{o_{\mathbf{y}} - o_{\mathbf{x}}}}, \\ \hat{b}_{-1,0} = -\lambda_1^{-1} H_1(|o_{\mathbf{x}} - o_{\mathbf{y}}|) e^{i\theta_{o_{\mathbf{y}} - o_{\mathbf{x}}}}, \end{cases} \quad (2.46)$$

$$\hat{b}_{p,0} = \begin{cases} e^{-i\theta_{o_{\mathbf{y}} - o_{\mathbf{x}}}} \frac{2(p-1)}{|o_{\mathbf{x}} - o_{\mathbf{y}}|} \frac{\lambda_{p-1}}{\lambda_p} \hat{b}_{p-1,0} - e^{-2i\theta_{o_{\mathbf{y}} - o_{\mathbf{x}}}} \frac{\lambda_{p-2}}{\lambda_{p-1}} \frac{\lambda_{p-1}}{\lambda_p} \hat{b}_{p-2,0}, & p \geq 2, \\ e^{i\theta_{o_{\mathbf{y}} - o_{\mathbf{x}}}} \frac{2(p+1)}{|o_{\mathbf{x}} - o_{\mathbf{y}}|} \frac{\lambda_{p+1}}{\lambda_p} \hat{b}_{p+1,0} - e^{2i\theta_{o_{\mathbf{y}} - o_{\mathbf{x}}}} \frac{\lambda_{p+2}}{\lambda_{p+1}} \frac{\lambda_{p+1}}{\lambda_p} \hat{b}_{p+2,0}, & p \leq -2, \end{cases} \quad (2.47)$$

$$\hat{b}_{p,l} = \begin{cases} -\frac{\lambda_{p+1}}{\lambda_p} \frac{\omega_{l-1}}{\omega_l} \hat{b}_{p+1,l-1}, & l \geq 1, \\ -\frac{\lambda_{p-1}}{\lambda_p} \frac{\omega_{l+1}}{\omega_l} \hat{b}_{p-1,l+1}, & l \leq -1. \end{cases} \quad (2.48)$$

The entries of \hat{U} can be computed stably and efficiently via the following recurrence relations, for $1 \leq i \leq n$,

$$\hat{U}_{i,p} = \begin{cases} \hat{U}_{i,0} = J_0(|x_i - o_{\mathbf{x}}|), \\ \hat{U}_{i,1} = \lambda_1 J_1(|x_i - o_{\mathbf{x}}|) e^{i\theta_{x_i - o_{\mathbf{x}}}}, \\ \hat{U}_{i,-1} = -\lambda_1 J_1(|x_i - o_{\mathbf{x}}|) e^{-i\theta_{x_i - o_{\mathbf{x}}}}, \\ e^{i\theta_{x_i - o_{\mathbf{x}}}} \frac{2(p-1)}{|x_i - o_{\mathbf{x}}|} \frac{\lambda_p}{\lambda_{p-1}} \hat{U}_{i,p-1} - e^{2i\theta_{x_i - o_{\mathbf{x}}}} \frac{\lambda_{p-1}}{\lambda_{p-2}} \frac{\lambda_p}{\lambda_{p-1}} \hat{U}_{i,p-2}, & p \geq 2, \\ e^{-i\theta_{x_i - o_{\mathbf{x}}}} \frac{2(p+1)}{|x_i - o_{\mathbf{x}}|} \frac{\lambda_p}{\lambda_{p+1}} \hat{U}_{i,p+1} - e^{-2i\theta_{x_i - o_{\mathbf{x}}}} \frac{\lambda_{p+1}}{\lambda_{p+2}} \frac{\lambda_p}{\lambda_{p+1}} \hat{U}_{i,p+2}, & p \leq -2. \end{cases} \quad (2.49)$$

The entries of \hat{V} can be computed stably and efficiently via the following recurrence relations, for $1 \leq j \leq m$,

$$\hat{V}_{j,l} = \begin{cases} \hat{V}_{j,0} = J_0(|y_j - o_{\mathbf{y}}|), \\ \hat{V}_{j,1} = \omega_1 J_1(|y_j - o_{\mathbf{y}}|) e^{i\theta_{y_j - o_{\mathbf{y}}}}, \\ \hat{V}_{j,-1} = -\omega_1 J_1(|y_j - o_{\mathbf{y}}|) e^{-i\theta_{y_j - o_{\mathbf{y}}}}, \\ e^{i\theta_{y_j - o_{\mathbf{y}}}} \frac{2(l-1)}{|y_j - o_{\mathbf{y}}|} \frac{\omega_l}{\omega_{l-1}} \hat{V}_{j,l-1} - e^{2i\theta_{y_j - o_{\mathbf{y}}}} \frac{\omega_{l-1}}{\omega_{l-2}} \frac{\omega_l}{\omega_{l-1}} \hat{V}_{j,l-2}, & l \geq 2, \\ e^{-i\theta_{y_j - o_{\mathbf{y}}}} \frac{2(l+1)}{|y_j - o_{\mathbf{y}}|} \frac{\omega_l}{\omega_{l+1}} \hat{V}_{j,l+1} - e^{-2i\theta_{y_j - o_{\mathbf{y}}}} \frac{\omega_{l+1}}{\omega_{l+2}} \frac{\omega_l}{\omega_{l+1}} \hat{V}_{j,l+2}, & l \leq -2. \end{cases} \quad (2.50)$$

Proof. We show the recurrence relations (2.47) and (2.48) for \hat{B} . The recurrence relations (2.49) and (2.50) for \hat{U} and \hat{V} can be derived in a similar way.

For convenience, we write $c = |o_{\mathbf{x}} - o_{\mathbf{y}}|$ and $\eta = e^{-i\theta_{o_{\mathbf{y}} - o_{\mathbf{x}}}}$. Since the Hankel function satisfies the recurrence relation $H_{p-2}(c) + H_p(c) = \frac{2(p-1)}{c} H_{p-1}(c)$, we have

$$\begin{aligned} \hat{b}_{p,0} &= \lambda_p^{-1} H_p(c) \eta^p = \lambda_p^{-1} \left(\frac{2(p-1)}{c} H_{p-1}(c) - H_{p-2}(c) \right) \eta^p \\ &= \frac{2(p-1)\eta}{c} \frac{\lambda_{p-1}}{\lambda_p} \hat{b}_{p-1,0} - \eta^2 \frac{\lambda_{p-2}}{\lambda_{p-1}} \frac{\lambda_{p-1}}{\lambda_p} \hat{b}_{p-2,0}. \end{aligned}$$

Similarly, the recurrence relation $H_p(c) + H_{p+2}(c) = \frac{2(p+1)}{c} H_{p+1}(c)$ implies

$$\hat{b}_{p,0} = \lambda_p^{-1} H_p(c) \eta^p = \lambda_p^{-1} \left(\frac{2(p+1)}{c} H_{p+1}(c) - H_{p+2}(c) \right) \eta^p$$

$$= \frac{2(p+1)}{c\eta} \frac{\lambda_{p+1}}{\lambda_p} \hat{b}_{p+1,0} - \frac{1}{\eta^2} \frac{\lambda_{p+2}}{\lambda_{p+1}} \frac{\lambda_{p+1}}{\lambda_p} \hat{b}_{p+2,0}.$$

This proves (2.47). The other one (2.48) follows directly from the formula (2.43) of \hat{b}_{pl} . \square

2.4 Stable translation relation

In this section, we exploit the *translation relation* in the U , V matrices in the low-rank approximation,

$$K_{\mathbf{x},\mathbf{y}} \approx U_{\mathbf{x}} B_{\mathbf{x},\mathbf{y}} V_{\mathbf{y}}^T,$$

where we write the subscript to emphasize their dependencies on the underlying point sets. The matrix $U_{\mathbf{x}}$ is called the *local expansion* matrix associated with \mathbf{x} . It is the (approximate) column basis matrix of $K_{\mathbf{x},\mathbf{y}}$ and represents the contributions of \mathbf{x} to $K_{\mathbf{x},\mathbf{y}}$. Similarly, $V_{\mathbf{y}}$ is the *multipole expansion* matrix for \mathbf{y} . It is the (approximate) row basis matrix of $K_{\mathbf{x},\mathbf{y}}$ and represents the contributions of \mathbf{y} . The translation relation connects the contributions of \mathbf{x} and \mathbf{y} with those of their subsets, respectively. It is an essential idea for the FMM to reach linear complexity.

In [32], the authors derive the translation matrix for standard Cauchy kernel and scale it using Stirling's formula such that its entries are bounded by 1. In this work, we extend the study to generalized Cauchy kernel and Poisson kernel. Our derivation is much simpler (without Stirling's formula), yet yields a stronger guarantee that the L_1 norm of such translation matrix equals 1. We also derive the translation matrix for the Helmholtz kernel.

It is shown that in Section 2.3 that for generalized Cauchy kernel and Poisson kernel, the local and multipole expansion matrices have the following forms

$$U_{\mathbf{x}} = \left(\left(\frac{x_i - o_{\mathbf{x}}}{\delta_{\mathbf{x}}} \right)^p \right)_{n \times r}, \quad V_{\mathbf{y}} = \left(\left(\frac{y_j - o_{\mathbf{y}}}{\delta_{\mathbf{y}}} \right)^l \right)_{m \times r},$$

and for Helmholtz kernel, they are

$$\hat{U}_{\mathbf{x}} = \left(\lambda_{\mathbf{x},p} J_p(|x_i - o_{\mathbf{x}}|) e^{ip\theta_{x_i - o_{\mathbf{x}}}} \right)_{n \times (2r+1)}, \quad \lambda_{\mathbf{x},p} = \max \left(1, |p|! \left(\frac{2}{\delta_{\mathbf{x}}} \right)^{|p|} \right),$$

$$\hat{V}_{\mathbf{y}} = \left(\omega_{\mathbf{y},l} J_p(|y_j - o_{\mathbf{y}}|) e^{il\theta_{y_j - o_{\mathbf{y}}}} \right)_{m \times (2r+1)}, \quad \omega_{\mathbf{y},l} = \max \left(1, |l|! \left(\frac{2}{\delta_{\mathbf{y}}} \right)^{|l|} \right).$$

In this section, we derive the translation relation between the local expansion matrices of \mathbf{x} and its subset. The translation relation for \mathbf{y} and its subset can be derived analogously.

Suppose a subset \mathbf{x}' of \mathbf{x} has center $o_{\mathbf{x}'}$ and radius $\delta_{\mathbf{x}'}$. We assume that the disk $\mathbb{D}_{\mathbf{x}'} = \{z : |z - o_{\mathbf{x}'}| \leq \delta_{\mathbf{x}'}\}$ is inside the disk $\mathbb{D}_{\mathbf{x}} = \{z : |z - o_{\mathbf{x}}| \leq \delta_{\mathbf{x}}\}$ so that

$$|o_{\mathbf{x}'} - o_{\mathbf{x}}| + \delta_{\mathbf{x}'} \leq \delta_{\mathbf{x}}. \quad (2.51)$$

To accommodate general cases, we also assume that there exists $0 < \beta \leq 1$ such that

$$\beta |o_{\mathbf{x}'} - o_{\mathbf{x}}| \leq \delta_{\mathbf{x}'} \leq \frac{1}{\beta} |o_{\mathbf{x}'} - o_{\mathbf{x}}|, \quad (2.52)$$

hence $|o_{\mathbf{x}'} - o_{\mathbf{x}}| \leq \frac{\delta_{\mathbf{x}}}{1+\beta}$, $\delta_{\mathbf{x}'} \leq \frac{\delta_{\mathbf{x}}}{1+\beta}$.

2.4.1 Stable translation relation for Cauchy and Poisson kernel

Suppose $x \in \mathbf{x}'$, then a row of $U_{\mathbf{x}'}$ is

$$u' = \left(1 \quad \frac{x - o_{\mathbf{x}'}}{\delta_{\mathbf{x}'}} \quad \dots \quad \left(\frac{x - o_{\mathbf{x}'}}{\delta_{\mathbf{x}'}} \right)^{r-1} \right)_{1 \times r},$$

and a row of $U_{\mathbf{x}}$ is

$$u = \left(1 \quad \frac{x - o_{\mathbf{x}}}{\delta_{\mathbf{x}}} \quad \dots \quad \left(\frac{x - o_{\mathbf{x}}}{\delta_{\mathbf{x}}} \right)^{r-1} \right)_{1 \times r}.$$

The translation from u' to u can be derived as follows. For $0 \leq k \leq r - 1$,

$$\begin{aligned} \left(\frac{x - o_{\mathbf{x}}}{\delta_{\mathbf{x}}} \right)^k &= \sum_{p=0}^k \binom{k}{p} \left(\frac{x - o_{\mathbf{x}'}}{\delta_{\mathbf{x}}} \right)^p \left(\frac{o_{\mathbf{x}'} - o_{\mathbf{x}}}{\delta_{\mathbf{x}}} \right)^{k-p} \\ &= \sum_{p=0}^k \binom{k}{p} \left(\frac{x - o_{\mathbf{x}'}}{\delta_{\mathbf{x}'}} \right)^p \left(\frac{\delta_{\mathbf{x}'}}{\delta_{\mathbf{x}}} \right)^p \left(\frac{o_{\mathbf{x}'} - o_{\mathbf{x}}}{\delta_{\mathbf{x}}} \right)^{k-p} \\ &\equiv \sum_{p=0}^k \left(\frac{x - o_{\mathbf{x}'}}{\delta_{\mathbf{x}'}} \right)^p t_{pk}, \quad \text{where } t_{pk} = \binom{k}{p} \left(\frac{\delta_{\mathbf{x}'}}{\delta_{\mathbf{x}}} \right)^p \left(\frac{o_{\mathbf{x}'} - o_{\mathbf{x}}}{\delta_{\mathbf{x}}} \right)^{k-p}. \end{aligned}$$

Theorem 2.4.1. *There exists an $r \times r$ matrix $T_{\mathbf{x}',\mathbf{x}}$ such that $u = u' \cdot T_{\mathbf{x}',\mathbf{x}}$. This matrix $T_{\mathbf{x}',\mathbf{x}}$ is called the translation matrix between \mathbf{x}' and \mathbf{x} , and it has the following form*

$$T_{\mathbf{x}',\mathbf{x}} = \begin{pmatrix} t_{00} & t_{01} & \cdots & t_{0,r-1} \\ & t_{11} & \cdots & t_{1,r-1} \\ & & \ddots & \vdots \\ & & & t_{r-1,r-1} \end{pmatrix}, \quad t_{pk} = \binom{k}{p} \left(\frac{\delta_{\mathbf{x}'}}{\delta_{\mathbf{x}}} \right)^p \left(\frac{o_{\mathbf{x}'} - o_{\mathbf{x}}}{\delta_{\mathbf{x}}} \right)^{k-p}. \quad (2.53)$$

and its entries t_{pk} can be computed efficiently and stably via the recurrence formula

$$\begin{cases} t_{00} = 1, & t_{-1,k-1} = t_{k,k-1} = 0, \\ t_{pk} = \frac{\delta_{\mathbf{x}'}}{\delta_{\mathbf{x}}} t_{p-1,k-1} + \frac{o_{\mathbf{x}'} - o_{\mathbf{x}}}{\delta_{\mathbf{x}}} t_{p,k-1}, & 0 \leq p \leq k, \end{cases} \quad (2.54)$$

It also satisfies

$$\|T_{\mathbf{x}',\mathbf{x}}\|_1 = 1, \quad \|T_{\mathbf{x}',\mathbf{x}}\|_\infty \leq \min \left(r, \frac{1 + \beta}{\beta} \right).$$

Furthermore, if \mathbf{x}'' is a subset of \mathbf{x}' , then

$$T_{\mathbf{x}'',\mathbf{x}} = T_{\mathbf{x}'',\mathbf{x}'} T_{\mathbf{x}',\mathbf{x}}. \quad (2.55)$$

Proof. Using the identity $\binom{k}{p} = \binom{k-1}{p-1} + \binom{k-1}{p}$, we have the recurrence relation

$$\begin{aligned} t_{pk} &= \binom{k-1}{p-1} \left(\frac{\delta_{\mathbf{x}'}}{\delta_{\mathbf{x}}} \right)^{p-1} \left(\frac{o_{\mathbf{x}'} - o_{\mathbf{x}}}{\delta_{\mathbf{x}}} \right)^{k-p} + \binom{k-1}{p} \left(\frac{\delta_{\mathbf{x}'}}{\delta_{\mathbf{x}}} \right)^p \left(\frac{o_{\mathbf{x}'} - o_{\mathbf{x}}}{\delta_{\mathbf{x}}} \right)^{k-p} \\ &= \frac{\delta_{\mathbf{x}'}}{\delta_{\mathbf{x}}} t_{p-1,k-1} + \frac{o_{\mathbf{x}'} - o_{\mathbf{x}}}{\delta_{\mathbf{x}}} t_{p,k-1}. \end{aligned}$$

Next, we show $\|T_{\mathbf{x}',\mathbf{x}}\|_1 = 1$. Since $t_{00} = 1$, we have $\|T_{\mathbf{x}',\mathbf{x}}\|_1 \geq 1$. On the other hand, for $0 \leq k \leq r-1$, the L_1 -norm of the k^{th} column satisfies,

$$\begin{aligned} \sum_{p=0}^k |t_{pk}| &= \sum_{p=0}^k \binom{k}{p} \left(\frac{\delta_{\mathbf{x}'}}{\delta_{\mathbf{x}}} \right)^p \left(\frac{|o_{\mathbf{x}'} - o_{\mathbf{x}}|}{\delta_{\mathbf{x}}} \right)^{k-p} \\ &\leq \sum_{p=0}^k \binom{k}{p} \left(\frac{\delta_{\mathbf{x}'}}{\delta_{\mathbf{x}}} \right)^p \left(\frac{\delta_{\mathbf{x}} - \delta_{\mathbf{x}'}}{\delta_{\mathbf{x}}} \right)^{k-p} = 1, \end{aligned}$$

therefore $\|T_{\mathbf{x}',\mathbf{x}}\|_1 = 1$.

Next, we show $\|T_{\mathbf{x}',\mathbf{x}}\|_\infty \leq \min\left(r, \frac{1+\beta}{\beta}\right)$. Let $s_p = \sum_{k=p}^{r-1} |t_{pk}|$ be the L_1 -norm of the p^{th} row. Since

$$s_0 = \sum_{k=0}^{r-1} |t_{0k}| = \sum_{k=0}^{r-1} \left(\frac{|o_{\mathbf{x}'} - o_{\mathbf{x}}|}{\delta_{\mathbf{x}}}\right)^k \leq \sum_{k=0}^{r-1} \left(\frac{1}{1+\beta}\right)^k \leq \min\left(r, \frac{1+\beta}{\beta}\right),$$

it suffices to show s_p is decreasing in p . Indeed, by recurrence relation (2.54),

$$\begin{aligned} s_p &= \sum_{k=p}^{r-1} |t_{pk}| \leq \frac{\delta_{\mathbf{x}'}}{\delta_{\mathbf{x}}} \sum_{k=p}^{r-1} |t_{p-1,k-1}| + \frac{|o_{\mathbf{x}'} - o_{\mathbf{x}}|}{\delta_{\mathbf{x}}} \sum_{k=p}^{r-1} |t_{p,k-1}| \\ &\leq \frac{\delta_{\mathbf{x}'}}{\delta_{\mathbf{x}}} s_{p-1} + \frac{|o_{\mathbf{x}'} - o_{\mathbf{x}}|}{\delta_{\mathbf{x}}} s_p \\ &\leq \frac{\delta_{\mathbf{x}'}}{\delta_{\mathbf{x}}} s_{p-1} + \frac{\delta_{\mathbf{x}} - \delta_{\mathbf{x}'}}{\delta_{\mathbf{x}}} s_p, \end{aligned}$$

which simplifies to $s_p \leq s_{p-1}$. Thus, $\|T_{\mathbf{x}',\mathbf{x}}\|_\infty = s_0 \leq \min\left(r, \frac{1+\beta}{\beta}\right)$.

Next, we show (2.55). For any $x \in \mathbf{x}''$, let u'' be the row for x in $U_{\mathbf{x}''}$, then we have $u'' \cdot T_{\mathbf{x}'',\mathbf{x}} = u = u' \cdot T_{\mathbf{x}',\mathbf{x}} = u'' \cdot T_{\mathbf{x}'',\mathbf{x}'} \cdot T_{\mathbf{x}',\mathbf{x}}$. Choosing r different $x \in \mathbf{x}''$, we have an invertible Vandermonde matrix U'' such that $U'' \cdot T_{\mathbf{x}'',\mathbf{x}} = U'' \cdot T_{\mathbf{x}'',\mathbf{x}'} \cdot T_{\mathbf{x}',\mathbf{x}}$, which proves (2.55). \square

Remark 2.4.2. *The translation matrix $T_{\mathbf{x}',\mathbf{x}}$ connects the local expansion matrix of \mathbf{x} to those of subsets of \mathbf{x} . For example, suppose we have the partition $\mathbf{x} = \mathbf{x}_1 \cup \mathbf{x}_2 \cup \mathbf{x}_3 \cup \mathbf{x}_4$, then by Theorem 2.4.1,*

$$U_{\mathbf{x}} = \begin{pmatrix} U_{\mathbf{x}_1} T_{\mathbf{x}_1,\mathbf{x}} \\ U_{\mathbf{x}_2} T_{\mathbf{x}_2,\mathbf{x}} \\ U_{\mathbf{x}_3} T_{\mathbf{x}_3,\mathbf{x}} \\ U_{\mathbf{x}_4} T_{\mathbf{x}_4,\mathbf{x}} \end{pmatrix} = \begin{pmatrix} U_{\mathbf{x}_1} & & & \\ & U_{\mathbf{x}_2} & & \\ & & U_{\mathbf{x}_3} & \\ & & & U_{\mathbf{x}_4} \end{pmatrix} \underbrace{\begin{pmatrix} T_{\mathbf{x}_1,\mathbf{x}} \\ T_{\mathbf{x}_2,\mathbf{x}} \\ T_{\mathbf{x}_3,\mathbf{x}} \\ T_{\mathbf{x}_4,\mathbf{x}} \end{pmatrix}}_{T_{\mathbf{x}} \in \mathbb{C}^{4r \times r}},$$

where we say $T_{\mathbf{x}}$ is the translation matrix between \mathbf{x} and $\{\mathbf{x}_k\}_{k=1}^4$, and it satisfies $\|T_{\mathbf{x}}\|_1 = 4$, $\|T_{\mathbf{x}}\|_\infty \leq \frac{1+\beta}{\beta}$. Obviously, this can be generalized to any partition of $\mathbf{x} = \cup_{k=1}^s \mathbf{x}_k$ and the translation matrix

$$T_{\mathbf{x}} = \left(T_{\mathbf{x}_1,\mathbf{x}}^T \quad \cdots \quad T_{\mathbf{x}_s,\mathbf{x}}^T \right)^T \in \mathbb{C}^{sr \times r}$$

satisfies $\|T_{\mathbf{x}}\|_1 = s$, $\|T_{\mathbf{x}}\|_\infty \leq \min(r, \frac{1+\beta}{\beta})$.

2.4.2 Stable translation for Helmholtz kernel

In this subsection, we derive the translation relation for the Helmholtz kernel. Suppose $x \in \mathbf{x}' \subset \mathbf{x}$, then a row of $\hat{U}_{\mathbf{x}'}$ is

$$u' = \left(g_p(x - o_{\mathbf{x}'}) \lambda_{\mathbf{x}', p} \right)_{-r \leq p \leq r},$$

and a row of $\hat{U}_{\mathbf{x}}$ is

$$u = \left(g_p(x - o_{\mathbf{x}}) \lambda_{\mathbf{x}, p} \right)_{-r \leq p \leq r},$$

where $g_p(z) = J_p(|z|) e^{ip\theta z}$ and $r \geq \frac{\delta_{\mathbf{x}} + \delta_{\mathbf{y}}}{\tau} \geq \frac{(1+\alpha)\delta_{\mathbf{x}}}{\tau}$. By Graf's addition formula,

$$g_p(x - o_{\mathbf{x}}) = \sum_{|l| \leq r, |p-l| \leq r} g_l(x - o_{\mathbf{x}'}) g_{p-l}(o_{\mathbf{x}} - o_{\mathbf{x}'}) (-1)^{p-l} + E_{r+1}^{J_p}(x - o_{\mathbf{x}'}, o_{\mathbf{x}} - o_{\mathbf{x}'}).$$

From this we can obtain the translation relation from u' to u

$$u_p = \sum_{l=-r}^r u'_l t_{lp} + E_p, \quad \text{where } E_p = \lambda_{\mathbf{x}, p} E_{r+1}^{J_p}(x - o_{\mathbf{x}'}, o_{\mathbf{x}} - o_{\mathbf{x}'}),$$

$$t_{lp} = \begin{cases} (-1)^{p-l} \lambda_{\mathbf{x}, p} g_{p-l}(o_{\mathbf{x}} - o_{\mathbf{x}'}) \lambda_{\mathbf{x}', l}^{-1}, & |p-l| \leq r \\ 0, & |p-l| > r \end{cases}.$$

Theorem 2.4.3. *There exists a $(2r+1) \times (2r+1)$ translation matrix $T_{\mathbf{x}', \mathbf{x}}$ such that $u = u' \cdot T_{\mathbf{x}', \mathbf{x}} + E$, where*

$$T_{\mathbf{x}', \mathbf{x}} = \begin{pmatrix} t_{-r, -r} & \cdots & t_{-r, 0} & & \\ \vdots & \ddots & \vdots & \ddots & \\ t_{0, -r} & \cdots & t_{0, 0} & \cdots & t_{0, r} \\ & \ddots & \vdots & \ddots & \vdots \\ & & t_{r, 0} & \cdots & t_{r, r} \end{pmatrix}, \quad \|T_{\mathbf{x}', \mathbf{x}}\|_{\max} \leq 1,$$

$$|E| \leq O\left(\left(\frac{\gamma}{1+\beta}\right)^{r+1}\right), \quad \gamma = \max\left(1, \frac{e\tau}{2(1+\alpha)}\right).$$

The entries t_{lp} can be computed stably and efficiently via a recurrence relation analogous to (2.47) and (2.48).

Proof. We first show the bound on $\|T_{\mathbf{x}',\mathbf{x}}\|_{\max}$. By Lemma 2.3.6, $\lambda_{\mathbf{x},p}$ is increasing in $|p|$, hence $\lambda_{\mathbf{x},p} \leq \lambda_{\mathbf{x},\tilde{p}}$, where $\tilde{p} = |l| + |p - l|$. Note that we have

$$\begin{aligned} \lambda_{\mathbf{x},\tilde{p}} &= \max\left(1, \tilde{p}! \left(\frac{2}{\delta_{\mathbf{x}}}\right)^{\tilde{p}}\right), \quad \lambda_{\mathbf{x}',l}^{-1} = \min\left(1, \frac{1}{|l|!} \left(\frac{\delta_{\mathbf{x}'}}{2}\right)^{|l|}\right), \\ |J_{p-l}(|o_{\mathbf{x}} - o_{\mathbf{x}'}|)| &\leq 1, \quad |J_{p-l}(|o_{\mathbf{x}} - o_{\mathbf{x}'}|)| \leq \frac{1}{|p-l|!} \left(\frac{|o_{\mathbf{x}} - o_{\mathbf{x}'}|}{2}\right)^{|p-l|}. \end{aligned}$$

Therefore, we can bound

$$\begin{aligned} |t_{lp}| &\leq \lambda_{\mathbf{x},\tilde{p}} \cdot |J_{p-l}(|o_{\mathbf{x}} - o_{\mathbf{x}'}|)| \cdot \lambda_{\mathbf{x}',l}^{-1} \\ &= \max\left(1, \tilde{p}! \left(\frac{2}{\delta_{\mathbf{x}}}\right)^{\tilde{p}}\right) \cdot |J_{p-l}(|o_{\mathbf{x}} - o_{\mathbf{x}'}|)| \cdot \lambda_{\mathbf{x}',l}^{-1} \\ &= \max\left(|J_{p-l}(|o_{\mathbf{x}} - o_{\mathbf{x}'}|)| \cdot \lambda_{\mathbf{x}',l}^{-1}, \tilde{p}! \left(\frac{2}{\delta_{\mathbf{x}}}\right)^{\tilde{p}} |J_{p-l}(|o_{\mathbf{x}} - o_{\mathbf{x}'}|)| \cdot \lambda_{\mathbf{x}',l}^{-1}\right) \\ &\leq \max\left(1, \tilde{p}! \left(\frac{2}{\delta_{\mathbf{x}}}\right)^{\tilde{p}} \frac{1}{|p-l|!} \left(\frac{|o_{\mathbf{x}} - o_{\mathbf{x}'}|}{2}\right)^{|p-l|} \frac{1}{|l|!} \left(\frac{\delta_{\mathbf{x}'}}{2}\right)^{|l|}\right) \\ &= \max\left(1, \binom{|p-l|+|l|}{|l|} \left(\frac{|o_{\mathbf{x}} - o_{\mathbf{x}'}|}{\delta_{\mathbf{x}}}\right)^{|p-l|} \left(\frac{\delta_{\mathbf{x}'}}{\delta_{\mathbf{x}}}\right)^{|l|}\right) \\ &\leq \max\left(1, \binom{|p-l|+|l|}{|l|} \left(\frac{\delta_{\mathbf{x}} - \delta_{\mathbf{x}'}}{\delta_{\mathbf{x}}}\right)^{|p-l|} \left(\frac{\delta_{\mathbf{x}'}}{\delta_{\mathbf{x}}}\right)^{|l|}\right) = 1. \end{aligned}$$

To show the bound for the remainder term, since $\frac{\delta_{\mathbf{x}}}{1+\beta} \geq \max(\delta_{\mathbf{x}'}, |o_{\mathbf{x}} - o_{\mathbf{x}'}|)$ and $r \geq \frac{(1+\alpha)\delta_{\mathbf{x}}}{\tau}$, we can set $z_{\max} = \frac{\delta_{\mathbf{x}}}{1+\beta}$ in Lemma 2.2.3 to bound

$$\begin{aligned} |E_{r+1}^{J_p}(x - o_{\mathbf{x}'}, o_{\mathbf{x}} - o_{\mathbf{x}'})| &\leq \frac{8}{(r+1)!} \left(\frac{\delta_{\mathbf{x}}}{2(1+\beta)}\right)^{r+1} \\ &\leq \frac{8}{\sqrt{2\pi}(r+1)} \left(\frac{e\delta_{\mathbf{x}}}{2(1+\beta)(r+1)}\right)^{r+1} \end{aligned}$$

$$\leq \frac{8}{\sqrt{2\pi(r+1)}} \left(\frac{e\tau}{2(1+\beta)(1+\alpha)} \right)^{r+1},$$

where we use Stirling's approximation in the second inequality. Therefore, we can bound

$$\begin{aligned} |E_p| &= \lambda_{\mathbf{x},p} \cdot \left| E_{r+1}^{J_p}(x - o_{\mathbf{x}'}, o_{\mathbf{x}} - o_{\mathbf{x}'}) \right| \leq \lambda_{\mathbf{x},r} \cdot \left| E_{r+1}^{J_p}(x - o_{\mathbf{x}'}, o_{\mathbf{x}} - o_{\mathbf{x}'}) \right| \\ &\leq \max \left(\left| E_{r+1}^{J_p}(x - o_{\mathbf{x}'}, o_{\mathbf{x}} - o_{\mathbf{x}'}) \right|, \frac{8}{(r+1)!} \left(\frac{\delta_{\mathbf{x}}}{2(1+\beta)} \right)^{r+1} r! \left(\frac{2}{\delta_{\mathbf{x}}} \right)^r \right) \\ &\leq \max \left(\frac{8}{\sqrt{2\pi(r+1)}} \left(\frac{e\tau}{2(1+\beta)(1+\alpha)} \right)^{r+1}, \frac{4\delta_{\mathbf{x}}}{r+1} \left(\frac{1}{1+\beta} \right)^{r+1} \right) \\ &= O \left(\left(\frac{\gamma}{1+\beta} \right)^r \right), \quad \text{where } \gamma = \max \left(1, \frac{e\tau}{2(1+\alpha)} \right). \end{aligned}$$

□

2.5 Matrix version of the fast multipole method

In this section, we present the matrix version of the fast multipole method in 2D. In particular, given the approximation accuracy $\epsilon > 0$, the fast multipole method will construct an *FMM approximation matrix* \tilde{K} to the kernel matrix $K = (\kappa(x_i, y_j))_{x_i \in \mathbf{x}, y_j \in \mathbf{y}} \in \mathbb{C}^{n \times m}$,

$$K = \tilde{K} + E, \quad |E| \leq \epsilon |K|.$$

The matrix \tilde{K} is hierarchically structured and admits $O(n+m)$ matrix-vector multiplication.

We will illustrate the matrix version FMM with a square domain in 2D. The cases in 3D and 1D are considered in earlier work [1] and [32], respectively. Our frameworks are parallel to them. Without loss of generality, we assume $n = m$ hereafter. Throughout this section, the symbols Σ and Π will be interpreted as

$$\begin{aligned} \sum_{k=l_1}^{l_2} M_k &= \begin{cases} M_{l_1} + M_{l_1+1} \cdots + M_{l_2}, & l_1 \leq l_2 \\ 0, & l_1 > l_2 \end{cases}, \\ \prod_{k=l_2}^{l_1} M_k &= \begin{cases} M_{l_2} M_{l_2-1} \cdots M_{l_1+1} M_{l_1}, & l_1 \leq l_2 \\ I, & l_1 > l_2 \end{cases}. \end{aligned}$$

2.5.1 Hierarchical partitioning and interaction list

Suppose the point sets \mathbf{x} and \mathbf{y} are located in a square domain \mathbf{S} in \mathbb{C} . The FMM begins with a hierarchical partitioning of \mathbf{S} . The domain \mathbf{S} is quadrisected recursively until the number of points is $O(n_0)$, where n_0 is pre-specified constant. A two-level partitioning example is given in Figure 2.2a. We also use a post-ordered quadtree \mathcal{T} to organize the partition such that the root of \mathcal{T} corresponds to \mathbf{S} , and each descendant node of \mathcal{T} corresponds to a descendant square of \mathbf{S} , see Figure 2.2b. We say the root node is at level 0, and the children of a level l node are at level $l + 1$. The total number of levels of \mathcal{T} shall be $L = O\left(\log \frac{n}{n_0}\right)$. We will use letters like \mathbf{i}, \mathbf{j} to denote nodes of the quadtree \mathcal{T} , and $\text{lvl}(\mathbf{i})$ to denote the level of node \mathbf{i} . We do not distinguish a node and its corresponding square.

We use \mathbf{x}_i to denote the subset of \mathbf{x} that are located in the square \mathbf{i} . The center $o_{\mathbf{x}_i}$ and radius $\delta_{\mathbf{x}_i}$ are chosen to be the center and radius of \mathbf{i} , that is, we choose $o_{\mathbf{x}_i} = o_i$ and $c_{\mathbf{x}_i} = c_i$. For convenience, we will also write $K_{\mathbf{i},\mathbf{j}}$ as the block of K defined by subsets \mathbf{x}_i and \mathbf{y}_j , that is, $K_{\mathbf{i},\mathbf{j}} = K_{\mathbf{x}_i,\mathbf{y}_j} = (\kappa(x, y))_{x \in \mathbf{x}_i, y \in \mathbf{y}_j}$. When \mathbf{i} and \mathbf{j} are well-separated, the block $K_{\mathbf{i},\mathbf{j}}$ is referred as *far-field interaction* or *far-field block*. According to Section 2.3, each far-field block has a low-rank approximation

$$K_{\mathbf{i},\mathbf{j}} = U_i B_{\mathbf{i},\mathbf{j}} V_j^T + E_{\mathbf{i},\mathbf{j}}, \quad |E_{\mathbf{i},\mathbf{j}}| \leq \epsilon |K_{\mathbf{i},\mathbf{j}}|,$$

where we write $U_i = U_{\mathbf{x}_i}$, $B_{\mathbf{i},\mathbf{j}} = B_{\mathbf{x}_i,\mathbf{y}_j}$, $V_j = V_{\mathbf{y}_j}$ for convenience. When they are not well-separated, they are said to be *neighbors*, and the block $K_{\mathbf{i},\mathbf{j}}$ is referred as *near-field interaction* or *near-field block*.

The *interaction list* \mathcal{L}_i of a node $\mathbf{i} \in \mathcal{T}$ is defined to be the collection of nodes \mathbf{j} that satisfies the conditions (i) $\text{lvl}(\mathbf{j}) = \text{lvl}(\mathbf{i})$; (ii) \mathbf{i} and \mathbf{j} are well-separated; (iii) their parents are neighbors [1], [3], [32]. For example, the interaction list of node 4 in Figure 2.2a is $\mathcal{L}_4 = \{7, 9, 13, 14, 17, 18, 19\}$. An three-level example is also given in Figure 2.3a, where $\text{lvl}(\mathbf{i}) = 3$, $\text{lvl}(\mathbf{p}) = 2$. It is easy to see for any node $\mathbf{i} \in \mathcal{T}$, its interaction list \mathcal{L}_i has at most 27 nodes. The interaction list decides which far-field interactions to be considered at each

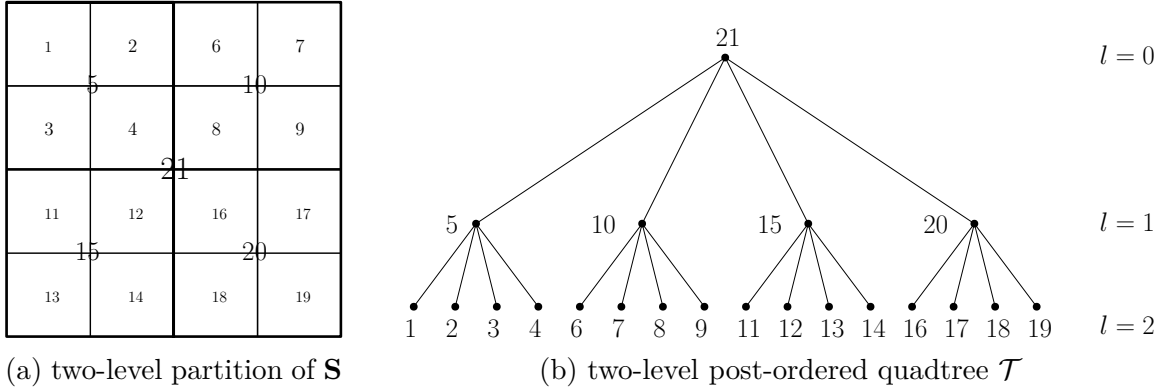


Figure 2.2. A two-level partition example and the corresponding post-ordered quadtree

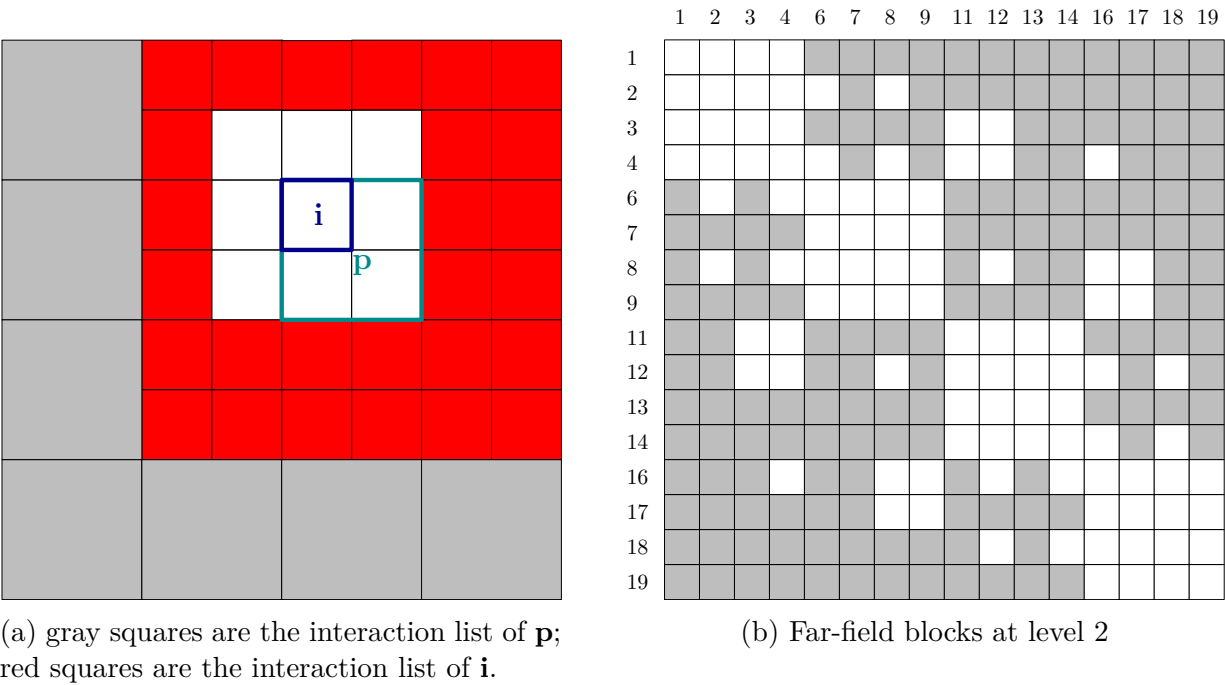


Figure 2.3. Interaction list and far-field approximations

level. Figure 2.3b gives an example of the far-field interactions at level 2, in which each gray block is a far-field block that has a low-rank approximation like $U_i B_{i,j} V_j^T$.

2.5.2 Level-wise low rank approximation

For $l \geq 2$, we can define the level- l far-field matrix $K^{(l)}$ by retaining only the far-field blocks $K_{\mathbf{i},\mathbf{j}}$, where $\text{lvl}(\mathbf{i}) = \text{lvl}(\mathbf{j}) = l$ and $\mathbf{j} \in \mathcal{L}_{\mathbf{i}}$, and setting other blocks of K to zero. Then $K^{(l)}$ have the decomposition

$$K^{(l)} = U^{(l)} B^{(l)} \left(V^{(l)} \right)^T + E^{(l)}, \quad \text{where} \quad |E^{(l)}| \leq \epsilon |K^{(l)}|, \quad (2.56)$$

$$U^{(l)} = \text{diag}(U_{\mathbf{i}_1}, \dots, U_{\mathbf{i}_k}), \quad V^{(l)} = \text{diag}(V_{\mathbf{i}_1}, \dots, V_{\mathbf{i}_k}), \quad \{\mathbf{i}_\alpha\}_{\alpha=1}^k \text{ are nodes at level } l,$$

and $B^{(l)}$ and $E^{(l)}$ have the same block structure as $K^{(l)}$ with $K_{\mathbf{i},\mathbf{j}}$ replaced by $B_{\mathbf{i},\mathbf{j}}$ and $E_{\mathbf{i},\mathbf{j}}$ respectively. We also define the near-field matrix K_n by retaining only the near-field blocks $K_{\mathbf{i},\mathbf{j}}$, where \mathbf{i} and \mathbf{j} are leaf nodes and neighbors, and setting other blocks to zero. Then the kernel matrix K can be decomposed as the sum

$$K = K_n + \sum_{l=2}^L K^{(l)} = K_n + \sum_{l=2}^L U^{(l)} B^{(l)} \left(V^{(l)} \right)^T + E, \quad (2.57)$$

where the error term $E = \sum_{l=2}^L E^{(l)}$. Because the nonzero pattern of $E^{(l)}$ does not overlap for different l , we have the error bound $|E| \leq \epsilon |K|$.

Then, we can define the FMM approximation matrix

$$\tilde{K} = K_n + \sum_{l=2}^L U^{(l)} B^{(l)} \left(V^{(l)} \right)^T. \quad (2.58)$$

Computing the matrix-vector product with \tilde{K} in form (2.58) has $O(Ln) = O(n \log n)$ complexity, and the resulting algorithm is called the *TreeCode* algorithm.

2.5.3 Translation relation and nested basis

In this subsection, we exploit the nested relations in $U^{(l)}$ and $V^{(l)}$, so that the matrix-vector product with the FMM matrix \tilde{K} can be accelerated to $O(n)$ complexity. For this purpose, we need to use the translation relation in Section 2.4.

Suppose a non-leaf node $\mathbf{i} \in \mathbf{T}$ has four children $\{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4\}$ in the hierarchical partition of \mathbf{S} . Then we have the partition for \mathbf{x}_i

$$\mathbf{x}_i = \mathbf{x}_{\mathbf{c}_1} \cup \mathbf{x}_{\mathbf{c}_2} \cup \mathbf{x}_{\mathbf{c}_3} \cup \mathbf{x}_{\mathbf{c}_4}.$$

For $1 \leq j \leq 4$, let $T_{\mathbf{c}_j, \mathbf{i}}$ be the translation matrix defined in Theorem 2.4.1 or 2.4.3, then

$$U_{\mathbf{i}} = \begin{pmatrix} U_{\mathbf{c}_1} T_{\mathbf{c}_1, \mathbf{i}} \\ U_{\mathbf{c}_2} T_{\mathbf{c}_2, \mathbf{i}} \\ U_{\mathbf{c}_3} T_{\mathbf{c}_3, \mathbf{i}} \\ U_{\mathbf{c}_4} T_{\mathbf{c}_4, \mathbf{i}} \end{pmatrix} = \begin{pmatrix} U_{\mathbf{c}_1} & & & \\ & U_{\mathbf{c}_2} & & \\ & & U_{\mathbf{c}_3} & \\ & & & U_{\mathbf{c}_4} \end{pmatrix} \underbrace{\begin{pmatrix} T_{\mathbf{c}_1, \mathbf{i}} \\ T_{\mathbf{c}_2, \mathbf{i}} \\ T_{\mathbf{c}_3, \mathbf{i}} \\ T_{\mathbf{c}_4, \mathbf{i}} \end{pmatrix}}_{T_{\mathbf{i}}}.$$

A similar relation is also true for $V_{\mathbf{i}}$. As a result, we obtain the nested relation

$$U^{(l)} = U^{(l+1)} R^{(l)}, \quad V^{(l)} = V^{(l+1)} R^{(l)} \quad \text{where} \quad (2.59)$$

$$R^{(l)} = \text{diag}(T_{\mathbf{i}_1}, \dots, T_{\mathbf{i}_k}), \quad \{\mathbf{i}_\alpha\}_{\alpha=1}^k \text{ are nodes at level } l. \quad (2.60)$$

Apply the nested relation (2.59) recursively, we obtain the factorization

$$U^{(l)} B^{(l)} (V^{(l)})^T = U^{(L)} \left(\prod_{l'=L-1}^l R^{(l')} \right) B^{(l)} \left(\prod_{l'=L-1}^l R^{(l')} \right)^T (V^{(L)})^T.$$

Thus, we have a *telescoping expansion* of the FMM matrix \tilde{K}

$$\begin{aligned} \tilde{K} &= \sum_{l=2}^L U^{(L)} \left(\prod_{l'=L-1}^l R^{(l')} \right) B^{(l)} \left(\prod_{l'=L-1}^l R^{(l')} \right)^T (V^{(L)})^T + K_n \\ &= U^{(L)} \left(R^{(L-1)} \left(\dots \left(R^{(2)} B^{(2)} (R^{(2)})^T + B^{(3)} \right) \right. \right. \\ &\quad \left. \left. \dots \right) (R^{(L-1)})^T + B^{(L)} \right) (V^{(L)})^T + K_n. \end{aligned} \quad (2.61)$$

In actual computation, we do not have to assemble the matrices $U^{(l)}$, $V^{(l)}$, $B^{(l)}$, $R^{(l)}$ and K_n . Instead, we only need to assemble the small matrices U_i and V_i for leaf nodes, T_i for non-leaf nodes, $B_{i,j}$ for $j \in \mathcal{L}_i$, and $K_{i,j}$ for neighbor leaf nodes i, j . Then the matrix-vector product with \tilde{K} can be computed via a bottom-up and a top-down traversal of the tree \mathcal{T} , see Algorithm 1. If we choose $n_0 = O(r)$, where r is the expansion order in the far-field low-rank approximation $K_{i,j} = U_i B_{i,j} V_j^T + E_{i,j}$, then the complexity of the FMM matrix-vector product is $O(r^2 n)$.

Algorithm 1 2D FMM matrix vector product $\phi = \tilde{K}q$

```

1:  $v^{(L)} \leftarrow \left(V^{(L)}\right)^T q$ 
2:  $t^{(L)} \leftarrow B^{(L)}v^{(L)}$ 
3: for level  $l = L - 1, \dots, 2$  do  $\triangleright$  bottom-up traversal
4:    $v^{(l)} \leftarrow \left(R^{(l)}\right)^T v^{(l+1)}$ 
5:    $t^{(l)} \leftarrow B^{(l)}v^{(l)}$ 
6: end for
7:  $u^{(2)} \leftarrow t^{(2)}$ 
8: for level  $l = 3, \dots, L$  do  $\triangleright$  top-down traversal
9:    $u^{(l)} \leftarrow R^{(l-1)}u^{(l-1)} + t^{(l)}$ 
10: end for
11:  $\phi \leftarrow U^{(L)}u^{(L)} + K_n q$   $\triangleright$  evaluation

```

2.6 Backward stability of FMM

In this section, we will study the backward stability of FMM. In particular, we will prove that the FMM algorithm is *backward stable*. For simplicity, we assume the underlying kernel $\kappa(x, y) = \frac{1}{(x-y)^{1+d}}$. We also assume \mathcal{T} is a full quadtree, so that for $0 \leq l \leq L$, there are 4^l nodes at level l . We also assume that for each leaf node $i \in \mathcal{T}$, $|\mathbf{x}_i| = |\mathbf{y}_i| = n_0 = O(r)$, so that U_i and V_i have size $n_0 \times r$ and the total number of level $L = \frac{1}{2} \log_2 \frac{n}{n_0} = O(\log_2 n)$.

We first review the backward stability of the standard matrix-vector multiplication, see [46].

Lemma 2.6.1 ([46]). *Let $M \in \mathbb{C}^{p \times r}$, $q \in \mathbb{C}^r$, $\gamma_r = \frac{r\epsilon_{mach}}{1-r\epsilon_{mach}}$, then the numerical matrix-vector product satisfies*

$$\text{fl}(Mq) = (M + \Delta M)q, \quad |\Delta M| \leq \gamma_r |M|.$$

As corollary, $\|\Delta M\|_1 \leq \gamma_r \|M\|_1$, $\|\Delta M\|_\infty \leq \gamma_r \|M\|_\infty$, $\|\Delta M\|_{1,1} \leq \gamma_r \|M\|_{1,1}$.

We will use the following norm inequalities repeatedly.

Lemma 2.6.2. *Let M , A , N be matrices with appropriate sizes, then*

$$\begin{aligned}\|MAN\|_{\max} &\leq \|M\|_\infty \|A\|_{\max} \|N\|_1, \\ \|MAN\|_{\max} &\leq \|M\|_{\max} \|A\|_{1,1} \|N\|_{\max},\end{aligned}$$

where $\|A\|_{1,1} = \sum_{p,l} |A_{pl}|$.

Proof. For the first inequality, since

$$|(MA)_{ij}| = \left| \sum_p M_{ip} A_{pj} \right| \leq \|A\|_{\max} \sum_p |M_{ip}| \leq \|M\|_\infty \|A\|_{\max},$$

using this inequality twice we get

$$\begin{aligned}\|MAN\|_{\max} &\leq \|M\|_\infty \|AN\|_{\max} = \|M\|_\infty \|N^T A^T\|_{\max} \\ &\leq \|M\|_\infty \|N^T\|_\infty \|A^T\|_{\max} = \|M\|_\infty \|A\|_{\max} \|N\|_1.\end{aligned}$$

For the second inequality,

$$|(MAN)_{ij}| = \left| \sum_{p,l} M_{ip} A_{pl} N_{lj} \right| \leq \|M\|_{\max} \|N\|_{\max} \sum_{p,l} |A_{pl}| = \|M\|_{\max} \|A\|_{1,1} \|N\|_{\max}.$$

The proof is completed. □

Lemma 2.6.3. *For $2 \leq l \leq L$, we have*

$$\begin{aligned}\|U^{(l)}\|_{\max} &= 1, \quad \|U^{(l)}\|_\infty \leq r, \quad \|U^{(l)}\|_1 \leq 4^{L-l} n_0, \\ \|V^{(l)}\|_{\max} &= 1, \quad \|V^{(l)}\|_\infty \leq r, \quad \|V^{(l)}\|_1 \leq 4^{L-l} n_0, \\ \|B^{(l)}\|_\infty &\leq \frac{27\|K\|_{\max}}{(1-\tau)^{2+2|d|}} \quad \|B^{(l)}\|_1 \leq \frac{27\|K\|_{\max}}{(1-\tau)^{2+2|d|}}, \quad \|B^{(l)}\|_2 \leq \frac{27\|K\|_{\max}}{(1-\tau)^{2+2|d|}}.\end{aligned}$$

Proof. Suppose $\{\mathbf{i}_\alpha\}_{\alpha=1}^k$ are nodes at level l , where $k = 4^l$. By definition,

$$U^{(l)} = \text{diag}(U_{\mathbf{i}_1}, \dots, U_{\mathbf{i}_k}), \quad U_{\mathbf{i}_\alpha} \in \mathbb{C}^{\frac{n}{4^l} \times r}.$$

According to Theorem 2.3.1, $\|U_{\mathbf{i}_\alpha}\|_{\max} = 1$. Therefore, $\|U^{(l)}\|_{\max} = 1$, and

$$\begin{aligned} \|U^{(l)}\|_{\infty} &= \max_{\alpha} \|U_{\mathbf{i}_\alpha}\|_{\infty} \leq r, \\ \|U^{(l)}\|_1 &= \max_{\alpha} \|U_{\mathbf{i}_\alpha}\|_1 \leq \frac{n}{4^l} = 4^{L-l} n_0. \end{aligned}$$

The bound for $V^{(l)}$ is analogous. Note that $B^{(l)}$ is a block $4^l \times 4^l$ matrix, and the nonzero blocks on the α^{th} block row $B_{\alpha,:}^{(l)}$ are $\{B_{\mathbf{i}_\alpha, \mathbf{i}_\beta}\}_{\mathbf{i}_\beta \in \mathcal{L}_{\mathbf{i}_\alpha}}$, where $\|B_{\mathbf{i}_\alpha, \mathbf{i}_\beta}\|_{1,1} \leq \frac{\|K\|_{\max}}{(1-\tau)^{1+|\mathbf{d}|}}$. Since each interaction list $\mathcal{L}_{\mathbf{i}_\alpha}$ contains at most 27 nodes, we have

$$\begin{aligned} \|B^{(l)}\|_{\infty} &= \max_{\alpha} \|B_{\alpha,:}^{(l)}\|_{\infty} \leq \max_{\alpha} \sum_{\beta \in \mathcal{L}_{\mathbf{i}_\alpha}} \|B_{\mathbf{i}_\alpha, \mathbf{i}_\beta}\|_{1,1} \\ &\leq 27 \max_{\alpha} \max_{\beta \in \mathcal{L}_{\mathbf{i}_\alpha}} \|B_{\mathbf{i}_\alpha, \mathbf{i}_\beta}\|_{1,1} \leq \frac{27\|K\|_{\max}}{(1-\tau)^{2+2|\mathbf{d}|}}. \end{aligned}$$

Similarly, we can consider block column $B_{:, \alpha}^{(l)}$ to see $\|B^{(l)}\|_1 \leq \frac{27\|K\|_{\max}}{(1-\tau)^{2+2|\mathbf{d}|}}$. \square

Lemma 2.6.4. *Suppose $2 \leq l_1 \leq l_2 \leq L-1$. The matrix $R^{(l_2, l_1)} \equiv \prod_{l=l_2}^{l_1} R^{(l)}$ is block $4^{l_1} \times 4^{l_1}$ diagonal. Each diagonal block $T_{\mathbf{i}}$ corresponds to a node \mathbf{i} at level l_1 and has size $4^{l_2+1-l_1} r \times r$. Furthermore, $T_{\mathbf{i}}$ is the vertical concatenation of $4^{l_2+1-l_1}$ matrices of size $r \times r$, and each of these $r \times r$ matrices corresponds to a level $l_2 + 1$ descendant of \mathbf{i} . Suppose \mathbf{d} is a level $l_2 + 1$ descendant of \mathbf{i} , and the path from \mathbf{d} to \mathbf{i} is*

$$\mathbf{d} = \mathbf{d}_{l_2+1} \rightarrow \mathbf{d}_{l_2} \rightarrow \dots \rightarrow \mathbf{d}_{l_1+1} \rightarrow \mathbf{d}_{l_1} = \mathbf{i},$$

then the $r \times r$ submatrix of $T_{\mathbf{i}}$ corresponding to \mathbf{d} is

$$T_{\mathbf{d}_{l_2+1}, \mathbf{d}_{l_2}} T_{\mathbf{d}_{l_2}, \mathbf{d}_{l_2-1}} \cdots T_{\mathbf{d}_{l_1+1}, \mathbf{d}_{l_1}} = \prod_{l=l_2}^{l_1} T_{\mathbf{d}_{l+1}, \mathbf{d}_l} = T_{\mathbf{d}, \mathbf{i}}, \quad (2.62)$$

where $T_{\mathbf{d}_{l+1}, \mathbf{d}_l}$ is the translation matrix defined in Theorem 2.4.1. As corollaries,

$$\|R^{(l_2, l_1)}\|_{\max} = 1, \quad \|R^{(l_2, l_1)}\|_{\infty} \leq \min\left(r, \frac{1+\beta}{\beta}\right), \quad \|R^{(l_2, l_1)}\|_1 = 4^{l_2-l_1+1}. \quad (2.63)$$

Proof. We prove this by induction. If $l_1 = l_2$, by the definition (2.60), $R^{(l_2, l_1)} = R^{(l_1)}$ is block $4^{l_1} \times 4^{l_1}$ diagonal, and each diagonal block $T_{\mathbf{i}}$ is just

$$T_{\mathbf{i}} = \left(T_{\mathbf{c}_1, \mathbf{i}}^T \quad T_{\mathbf{c}_2, \mathbf{i}}^T \quad T_{\mathbf{c}_3, \mathbf{i}}^T \quad T_{\mathbf{c}_4, \mathbf{i}}^T \right)^T, \quad \{\mathbf{c}_j\}_{j=1}^4 \text{ are children of } \mathbf{i},$$

so that (2.62) is true.

Assume $l_1 < l_2$, and $R^{(l_2, l_1+1)}$ is block $4^{l_1+1} \times 4^{l_1+1}$ diagonal, and each diagonal block corresponds to a node at level $l_1 + 1$ and has size $4^{l_2-l_1}r \times r$. Then we can merge 4 diagonal blocks into a larger diagonal block, so that $R^{(l_2, l_1+1)}$ is block $4^{l_1} \times 4^{l_1}$ diagonal and each larger diagonal block now corresponds to a node at level l_1 and has size $4^{l_2-l_1+1}r \times 4r$. By definition, $R^{(l_1)}$ is also block $4^{l_1} \times 4^{l_1}$ diagonal, and each diagonal block corresponds to a node at level l_1 and has size $4r \times r$. As a result, the product

$$R^{(l_2, l_1)} = \prod_{l=l_2}^{l_1} R^{(l)} = \underbrace{R^{(l_2, l_1+1)}}_{\text{block } 4^{l_1} \times 4^{l_1} \text{ diagonal}} \cdot \underbrace{R^{(l_1)}}_{\text{block } 4^{l_1} \times 4^{l_1} \text{ diagonal}} \quad (2.64)$$

is also block $4^{l_1} \times 4^{l_1}$ diagonal, and each diagonal block has size $4^{l_2-l_1+1}r \times r$ and corresponds to a node at level l_1 .

In (2.64), the submatrix of $R^{(l_2, l_1+1)}$ corresponding to \mathbf{d} is multiplied with the submatrix $T_{\mathbf{d}_{l+1}, \mathbf{d}_l}$ of $R^{(l_1)}$ to get the submatrix of $R^{(l_2, l_1)}$ corresponding to \mathbf{d} . By induction hypothesis, it is

$$\left(\prod_{l=l_2}^{l_1+1} T_{\mathbf{d}_{l+1}, \mathbf{d}_l} \right) T_{\mathbf{d}_{l+1}, \mathbf{d}_l} = \prod_{l=l_2}^{l_1} T_{\mathbf{d}_{l+1}, \mathbf{d}_l} = T_{\mathbf{d}, \mathbf{i}},$$

where the last equality uses (2.55) of Theorem 2.4.1. This completes the induction.

Since $\|T_{\mathbf{d}, \mathbf{i}}\|_{\max} = \|T_{\mathbf{d}, \mathbf{i}}\|_1 = 1$, $\|T_{\mathbf{d}, \mathbf{i}}\|_{\infty} \leq \min\left(r, \frac{1+\beta}{\beta}\right)$, and the diagonal block $T_{\mathbf{i}}$ is the vertical concatenation of $4^{l_2+1-l_1}$ such $T_{\mathbf{d}, \mathbf{i}}$, we have the norm bounds (2.63). This completes the proof. \square

Remark 2.6.5. If we replace $R^{(k)}$ by $\hat{R}^{(k)}$ in the product $\prod_{l=l_2}^{l_1} R^{(l)}$,

$$R^{(l_2)} \dots R^{(k+1)} \hat{R}^{(k)} R^{(k-1)} \dots R^{(l_1)},$$

where $\hat{R}^{(k)}$ could be any matrix with the same structure as $R^{(k)}$, then the statement in Lemma 2.6.4 still holds for this new product, except that we just need to replace $T_{\mathbf{d}_{k+1}, \mathbf{d}_k}$ in (2.62) with the appropriate block in $\hat{R}^{(k)}$.

Now, we study the backward error in the fast multipole method. In particular, we show that its backward error depends logarithmically on the size of K . This is an advantage over standard matrix-vector product, where the backward error depends linearly on the size of the matrix (see Lemma 2.6.1).

Theorem 2.6.6. *The fast multipole method multiplication is backward stable:*

$$\begin{aligned} \text{fl}(\tilde{K}q) &= (K + \Delta K)q, \quad \text{where} \\ |\Delta K| &\leq \left(O(r \log n) \epsilon_{mach} + \epsilon \right) |K|, \end{aligned}$$

where ϵ_{mach} is the machine precision, and ϵ is the approximation accuracy.

Proof. Throughout this proof, let $\tilde{\gamma}_r$ denote the generic constant $c\gamma_r$ for some small $c > 0$. The FMM matrix-vector product routine consists of a bottom-up traversal, a top-down traversal, and a final evaluation step, see Algorithm 1.

- *Bottom-up traversal.* In this stage, we compute

$$\begin{aligned} \text{fl}(v^{(L)}) &= \left(V^{(L)} + \Delta V^{(L)} \right)^T q, & |\Delta V^{(L)}| &\leq \tilde{\gamma}_r |V^{(L)}|, \\ \text{fl}(v^{(l)}) &= \left(R^{(l)} + \Delta R^{(l)} \right)^T \text{fl}(v^{(l+1)}), & |\Delta R^{(l)}| &\leq \tilde{\gamma}_r |R^{(l)}|, \end{aligned} \quad (2.65)$$

$$\text{fl}(t^{(l)}) = \left(B^{(l)} + \Delta B^{(l)} \right) \text{fl}(v^{(l)}), \quad |\Delta B^{(l)}| \leq \tilde{\gamma}_r |B^{(l)}|, \quad (2.66)$$

where $\Delta V^{(L)}$ has the same block structure as $V^{(L)}$, and $\Delta R^{(l)}$ has the same block structure as $R^{(l)}$, and $\Delta B^{(l)}$ has the same block structure as $B^{(l)}$. Expand the recursion (2.65) and use the nested relation (2.59), we get

$$\text{fl}(v^{(l)}) = \left(V^{(l)} + \Delta V^{(l)} + O(\epsilon_{\text{mach}}^2) \right)^T q, \quad \text{where} \quad (2.67)$$

$$\Delta V^{(l)} = \Delta V^{(L)} \cdot R^{(L-1,l)} + \sum_{k=l}^{L-1} V^{(k+1)} \cdot \Delta R^{(k)} \cdot R^{(k-1,l)}, \quad 2 \leq l \leq L. \quad (2.68)$$

Then, we can expand (2.66) to get

$$\begin{aligned} \text{fl}(t^{(l)}) &= \left(B^{(l)} + \Delta B^{(l)} \right) \left(V^{(l)} + \Delta V^{(l)} + O(\epsilon_{\text{mach}}^2) \right)^T q \\ &= \left(B^{(l)} \left(V^{(l)} \right)^T + \Delta B_l + O(\epsilon_{\text{mach}}^2) \right) q, \end{aligned} \quad (2.69)$$

where the error matrix ΔB_l is

$$\Delta B_l = \Delta B^{(l)} \left(V^{(l)} \right)^T + B^{(l)} \left(\Delta V^{(l)} \right)^T. \quad (2.70)$$

- *Top-down traversal.* In this stage, we compute

$$\begin{aligned} \text{fl}(u^{(2)}) &= \text{fl}(t^{(2)}), \\ \text{fl}(u^{(l)}) &= \left(I + \Delta Z^{(l)} \right) \left(\left(R^{(l-1)} + \Delta R^{(l-1)} \right) \text{fl}(u^{(l-1)}) + \text{fl}(t^{(l)}) \right), \quad l \geq 3, \end{aligned}$$

where $|\Delta Z^{(l)}| \leq \epsilon_{\text{mach}} |I|$ for $l \geq 3$, and we also set $\Delta Z^{(2)} = 0$. Substituting (2.69) into the above expression, we obtain the following recursion for $l \geq 3$,

$$\begin{aligned} \text{fl}(u^{(l)}) &= \left(I + \Delta Z^{(l)} \right) \left(R^{(l-1)} + \Delta R^{(l-1)} \right) \text{fl}(u^{(l-1)}) \\ &\quad + \left(I + \Delta Z^{(l)} \right) \left(B^{(l)} \left(V^{(l)} \right)^T + \Delta B_l + O(\epsilon_{\text{mach}}^2) \right) q. \end{aligned} \quad (2.71)$$

In order to expand the above recursion, we assume $\text{fl}(u^{(l-1)})$ has the following form

$$\text{fl}(u^{(l-1)}) = \left(\sum_{k=2}^{l-1} R^{(l-2,k)} B^{(k)} \left(V^{(k)} \right)^T + \Delta F_{l-1} + O(\epsilon_{\text{mach}}^2) \right) q. \quad (2.72)$$

And we need to find the recurrence formula for the error matrix ΔF_{l-1} . To do this, substitute (2.72) into (2.71), then we get

$$\text{fl}(u^{(l)}) = \left(\sum_{k=2}^l R^{(l-1,k)} B^{(k)} (V^{(k)})^T + \Delta F_l + O(\epsilon_{\text{mach}}^2) \right) q, \quad (2.73)$$

and we obtain the following recurrence formula for F_l

$$\begin{aligned} \Delta F_2 &= \Delta B_2, \\ \Delta F_l &= R^{(l-1)} \cdot \Delta F_{l-1} + \Delta G_l, \quad \text{where} \\ \Delta G_l &= \Delta B_l + \Delta Z^{(l)} \cdot B^{(l)} (V^{(l)})^T \\ &\quad + \sum_{k=2}^{l-1} \left(\Delta Z^{(l)} \cdot R^{(l-1,k)} \right) B^{(k)} (V^{(k)})^T \\ &\quad + \sum_{k=2}^{l-1} \left(\Delta R^{(l-1)} \cdot R^{(l-2,k)} \right) B^{(k)} (V^{(k)})^T \end{aligned} \quad (2.74)$$

Expand the recurrence formula (2.74), we can get the error matrix ΔF_L

$$\begin{aligned} \Delta F_L &= R^{(L-1,2)} \cdot \Delta B_2 + \sum_{l=3}^L R^{(L-1,l)} \cdot \Delta G_l, \\ &= \sum_{l=2}^L R^{(L-1,l)} \cdot \Delta B_l + \sum_{l=2}^L \left(R^{(L-1,l)} \cdot \Delta Z^{(l)} \right) B^{(l)} (V^{(l)})^T \\ &\quad + \sum_{l=3}^L \sum_{k=2}^{l-1} \left(R^{(L-1,l)} \cdot \Delta Z^{(l)} \cdot R^{(l-1,k)} \right) B^{(k)} (V^{(k)})^T \\ &\quad + \sum_{l=3}^L \sum_{k=2}^{l-1} \left(R^{(L-1,l)} \cdot \Delta R^{(l-1)} \cdot R^{(l-2,k)} \right) B^{(k)} (V^{(k)})^T \end{aligned}$$

Substitute the expression (2.70) for ΔB_l and change the order in the double sums,

$$\begin{aligned} \Delta F_L &= \sum_{l=2}^L R^{(L-1,l)} \cdot \Delta B^{(l)} (V^{(l)})^T + \sum_{l=2}^L R^{(L-1,l)} B^{(l)} (\Delta V^{(l)})^T \\ &\quad + \sum_{l=2}^L \left(R^{(L-1,l)} \cdot \Delta Z^{(l)} \right) B^{(l)} (V^{(l)})^T \\ &\quad + \sum_{l=2}^{L-1} \sum_{k=l}^{L-1} \left(R^{(L-1,k+1)} \cdot \Delta Z^{(k+1)} \cdot R^{(k,l)} \right) B^{(l)} (V^{(l)})^T \end{aligned}$$

$$+ \sum_{l=2}^{L-1} \sum_{k=l}^{L-1} \left(R^{(L-1,k+1)} \cdot \Delta R^{(k)} \cdot R^{(k-1,l)} \right) B^{(l)} \left(V^{(l)} \right)^T.$$

• *Evaluation.* In this stage, we compute

$$\text{fl}(\phi) = (I + \Delta Z) \left((U^{(L)} + \Delta U^{(L)}) \text{fl}(u^L) + (K_n + \Delta K_n)q \right), \quad (2.75)$$

where $|\Delta U^{(L)}| \leq \tilde{\gamma}_r |U^{(L)}|$, $|\Delta K_n| \leq \tilde{\gamma}_r |K_n|$, $|\Delta Z| \leq \epsilon_{\text{mach}} |I|$, and

$$\text{fl}(u^{(L)}) = \left(\sum_{l=2}^L R^{(L-1,l)} B^{(l)} \left(V^{(l)} \right)^T + \Delta F_L + O(\epsilon_{\text{mach}}^2) \right) q, \quad (2.76)$$

Substituting (2.76) into (2.75), and using (2.59) and (2.57), we have

$$\text{fl}(\phi) = (K + \Delta K)q,$$

where the overall error matrix

$$\begin{aligned} \Delta K &= \Delta F + \Delta K_n + \Delta Z \cdot K - \Delta Z \cdot E - E + O(\epsilon_{\text{mach}}^2), \quad \text{where} \\ \Delta F &= U^{(L)} \cdot \Delta F_L + \sum_{l=2}^L \left(\Delta U^{(L)} \cdot R^{(L-1,l)} \right) B^{(l)} \left(V^{(l)} \right)^T \\ &= \sum_{l=2}^L U^{(l)} \cdot \Delta B^{(l)} \left(V^{(l)} \right)^T + \sum_{l=2}^L U^{(l)} B^{(l)} \left(\Delta V^{(l)} \right)^T \\ &\quad + \sum_{l=2}^L \left(U^{(l)} \cdot \Delta Z^{(l)} \right) B^{(l)} \left(V^{(l)} \right)^T \\ &\quad + \sum_{l=2}^{L-1} \sum_{k=l}^{L-1} \left(U^{(k+1)} \cdot \Delta Z^{(k+1)} \cdot R^{(k,l)} \right) B^{(l)} \left(V^{(l)} \right)^T \\ &\quad + \sum_{l=2}^{L-1} \sum_{k=l}^{L-1} \left(U^{(k+1)} \cdot \Delta R^{(k)} \cdot R^{(k-1,l)} \right) B^{(l)} \left(V^{(l)} \right)^T \\ &\quad + \sum_{l=2}^L \left(\Delta U^{(L)} \cdot R^{(L-1,l)} \right) B^{(l)} \left(V^{(l)} \right)^T, \end{aligned}$$

where $\Delta V^{(l)}$ has the form (2.68)

$$\Delta V^{(l)} = \Delta V^{(L)} \cdot R^{(L-1,l)} + \sum_{k=l}^{L-1} V^{(k+1)} \cdot \Delta R^{(k)} \cdot R^{(k-1,l)}, \quad 2 \leq l \leq L.$$

To simplify the notations in ΔF , let

$$\begin{aligned} \Delta U^{(l)} &= \Delta U^{(L)} \cdot R^{(L-1,l)} + \sum_{k=l}^{L-1} U^{(k+1)} \cdot \Delta R^{(k)} \cdot R^{(k-1,l)}, \quad 2 \leq l \leq L, \\ \Delta \tilde{U}^{(l)} &= U^{(l)} \cdot \Delta Z^{(l)} + \sum_{k=l}^{L-1} U^{(k+1)} \cdot \Delta Z^{(k+1)} \cdot R^{(k,l)}, \quad 2 \leq l \leq L, \end{aligned}$$

then we have

$$\begin{aligned} \Delta F &= \sum_{l=2}^L U^{(l)} \cdot \Delta B^{(l)} (V^{(l)})^T + \sum_{l=2}^L U^{(l)} B^{(l)} (\Delta V^{(l)})^T \\ &\quad + \sum_{l=2}^L \Delta U^{(l)} \cdot B^{(l)} (V^{(l)})^T + \sum_{l=2}^L \Delta \tilde{U}^{(l)} \cdot B^{(l)} (V^{(l)})^T. \end{aligned}$$

We show how to bound the entries of $\sum_{l=2}^L \Delta U^{(l)} \cdot B^{(l)} (V^{(l)})^T$. In light of Lemma 2.6.4 and Remark 2.6.5, $\Delta U^{(l)}$ has the same structure as $U^{(l)}$, so $\Delta U^{(l)} \cdot B^{(l)} (V^{(l)})^T$ has the same structure as $K^{(l)} = U^{(l)} B^{(l)} (V^{(l)})^T + E^{(l)}$. Therefore, the non-zero pattern of $\Delta U^{(l)} \cdot B^{(l)} (V^{(l)})^T$ for different l does not overlap. Suppose $\text{lvl}(\mathbf{i}) = \text{lvl}(\mathbf{j}) = l$ and $\mathbf{j} \in \mathcal{L}_{\mathbf{i}}$, then the corresponding block in $\Delta U^{(l)} \cdot B^{(l)} (V^{(l)})^T$ has the form $\Delta U_{\mathbf{i}} \cdot B_{\mathbf{i},\mathbf{j}} (V_{\mathbf{j}})^T$, where $\Delta U_{\mathbf{i}}$ is a diagonal block in $\Delta U^{(l)}$. According to Lemma 2.6.2 and Theorem 2.3.1,

$$\begin{aligned} \left\| \Delta U_{\mathbf{i}} \cdot B_{\mathbf{i},\mathbf{j}} (V_{\mathbf{j}})^T \right\|_{\max} &\leq \|\Delta U_{\mathbf{i}}\|_{\max} \|B_{\mathbf{i},\mathbf{j}}\|_{1,1} \|V_{\mathbf{j}}\|_{\max} \\ &\leq \|\Delta U_{\mathbf{i}}\|_{\max} \frac{\|K_{\mathbf{i},\mathbf{j}}\|_{\min}}{(1-\tau)^{2+2|d|}} \\ &\leq \left\| \Delta U^{(l)} \right\|_{\max} \frac{\|K_{\mathbf{i},\mathbf{j}}\|_{\min}}{(1-\tau)^{2+2|d|}}, \end{aligned}$$

where the $\|\cdot\|_{\max}$ norm of $\Delta U^{(l)}$ can be bounded via Lemma 2.6.4 and Remark 2.6.5 as follows. For its first term $\Delta U^{(L)} \cdot R^{(L-1,l)}$, the block corresponding a node \mathbf{d}_L at level L has the form

$$\Delta U_{\mathbf{d}_L} \cdot \left(T_{\mathbf{d}_L, \mathbf{d}_{L-1}} \cdots T_{\mathbf{d}_{l+1}, \mathbf{d}_l} \right) = \Delta U_{\mathbf{d}_L} \cdot T_{\mathbf{d}_L, \mathbf{d}_l},$$

and its $\|\cdot\|_{\max}$ norm can be bounded by

$$\|\Delta U_{\mathbf{d}_L}\|_{\max} \cdot \|T_{\mathbf{d}_L, \mathbf{d}_l}\|_1 \leq \tilde{\gamma}_r \|U_{\mathbf{d}_L}\|_{\max} \cdot \|T_{\mathbf{d}_L, \mathbf{d}_l}\|_1 = \tilde{\gamma}_r.$$

For each summand $U^{(k+1)} \cdot \Delta R^{(k)} \cdot R^{(k-1,l)}$, the block corresponding a node \mathbf{d}_L at level L has the form

$$\underbrace{U_{\mathbf{d}_L} \cdot \left(T_{\mathbf{d}_L, \mathbf{d}_{L-1}} \cdots T_{\mathbf{d}_{k+2}, \mathbf{d}_{k+1}} \right)}_{U_{\mathbf{d}_{k+1}}} \cdot \Delta T_{\mathbf{d}_{k+1}, \mathbf{d}_k} \cdot \underbrace{\left(T_{\mathbf{d}_k, \mathbf{d}_{k-1}} \cdots T_{\mathbf{d}_{l+1}, \mathbf{d}_l} \right)}_{T_{\mathbf{d}_k, \mathbf{d}_l}},$$

and its $\|\cdot\|_{\max}$ norm can be bounded by

$$\|U_{\mathbf{d}_{k+1}}\|_{\max} \cdot \|\Delta T_{\mathbf{d}_{k+1}, \mathbf{d}_k}\|_1 \cdot \|T_{\mathbf{d}_k, \mathbf{d}_l}\|_1 \leq \tilde{\gamma}_r \|U_{\mathbf{d}_{k+1}}\|_{\max} \cdot \|T_{\mathbf{d}_{k+1}, \mathbf{d}_k}\|_1 \cdot \|T_{\mathbf{d}_k, \mathbf{d}_l}\|_1 = \tilde{\gamma}_r.$$

Thus, we have the bounds

$$\begin{aligned} \|\Delta U^{(l)}\|_{\max} &\leq \tilde{\gamma}_r + \sum_{k=l}^{L-1} \tilde{\gamma}_r \leq (L-l+1)\tilde{\gamma}_r, \\ \|\Delta U_{\mathbf{i}} \cdot B_{\mathbf{i}, \mathbf{j}} (V_{\mathbf{j}})^T\|_{\max} &\leq \frac{(L-l+1)\tilde{\gamma}_r}{(1-\tau)^{2+2|d|}} \|K_{\mathbf{i}, \mathbf{j}}\|_{\min}. \end{aligned}$$

Hence, we have the entry-wise bound

$$\begin{aligned} \left| \Delta U_{\mathbf{i}} \cdot B_{\mathbf{i}, \mathbf{j}} (V_{\mathbf{j}})^T \right| &\leq \frac{(L-l+1)\tilde{\gamma}_r}{(1-\tau)^{2+2|d|}} |K_{\mathbf{i}, \mathbf{j}}|, \\ \left| \Delta U^{(l)} \cdot B^{(l)} (V^{(l)})^T \right| &\leq O(\tilde{\gamma}_r \log n) |K^{(l)}|. \end{aligned}$$

Because the non-zero blocks of $\Delta U^{(l)} \cdot B^{(l)} (V^{(l)})^T$ for different l does not overlap, we have the entry-wise bound

$$\left| \sum_{l=2}^L \Delta U^{(l)} \cdot B^{(l)} (V^{(l)})^T \right| \leq O(\tilde{\gamma}_r \log n) |K|.$$

We can analogously derive the bounds for other terms in ΔF

$$\begin{aligned} \left| \sum_{l=2}^L U^{(l)} \cdot \Delta B^{(l)} (V^{(l)})^T \right| &\leq O(\tilde{\gamma}_r \log n) |K|, \\ \left| \sum_{l=2}^L U^{(l)} B^{(l)} \cdot (\Delta V^{(l)})^T \right| &\leq O(\tilde{\gamma}_r \log n) |K|, \\ \left| \sum_{l=2}^L \Delta \tilde{U}^{(l)} \cdot B^{(l)} (V^{(l)})^T \right| &\leq O(\tilde{\gamma}_r \log n) |K|. \end{aligned}$$

Because $\Delta K = \Delta F + \Delta K_n + \Delta Z \cdot K - \Delta Z \cdot E - E + O(\epsilon_{\text{mach}}^2)$ and $|E| \leq \epsilon |K|$, we obtain the entry-wise bound for ΔK

$$|\Delta K| \leq \left(O(r \log n) \epsilon_{\text{mach}} + \epsilon \right) |K|,$$

where ϵ_{mach} is the machine precision, and ϵ is the approximation accuracy. The proof is completed. \square

3. SUPERDC: SUPERFAST DIVIDE-AND-CONQUER RANK-STRUCTURED MATRIX EIGENVALUE DECOMPOSITION WITH IMPROVED STABILITY

This chapter is based on the paper [Ou, Xiaofeng and Xia, Jianlin, "*SuperDC: superfast divide-and-conquer eigenvalue decomposition for rank-structured matrices with improved stability*", submitted to SIAM Journal on Scientific Computing].

In this chapter, we shall describe a superfast divide-and-conquer eigenvalue decomposition for dense symmetric matrices with small off-diagonal ranks and in a hierarchically semiseparable form. Compared to an earlier basic algorithm in [Vogel, Xia, et al., SIAM J. Sci. Comput., 38 (2016)], this eigensolver (SuperDC) has significantly better stability while keeping the nearly linear complexity for finding the entire eigenvalue decomposition. In a set of comprehensive tests, SuperDC shows significantly lower runtime and storage than the Matlab `eig` function. The stability benefits of our new algorithm are also confirmed with both analysis and numerical comparisons.

Throughout this chapter, the following notations are used.

- Lower-case letters in bold fonts like \mathbf{u} are used to denote vectors.
- A vector $\mathbf{u} = (u_i)_{i=1}^n$ will also be considered as a set with elements $\{u_i\}_{i=1}^n$.
- $(A_{ij})_{n \times n}$ means an $n \times n$ matrix with the (i, j) -entry A_{ij} . Sometimes, a matrix defined by the evaluation of a function $\kappa(s, t)$ at points s_i in a set \mathbf{s} and t_j in a set \mathbf{t} is written as $(\kappa(s_i, t_j))_{s_i \in \mathbf{s}, t_j \in \mathbf{t}}$.
- $\text{diag}(\dots)$ denotes a (block) diagonal matrix.
- $\text{rowsize}(A)$ and $\text{colsize}(A)$ mean the row and column sizes of A , respectively.
- $\mathbf{u} \odot \mathbf{v}$ denotes the entrywise (Hadamard) product of two vectors \mathbf{u} and \mathbf{v} .
- For a binary tree \mathcal{T} , we order its nodes in postorder, so that it has nodes $i = 1, 2, \dots, \text{root}(\mathcal{T})$, where $\text{root}(\mathcal{T})$ is the root.

- $\text{fl}(x)$ denotes the floating point result of x .
- ϵ_{mach} represents the machine precision.

3.1 Introduction

We consider the full eigenvalue decomposition of $n \times n$ symmetric matrices A with small off-diagonal ranks. Such matrices belong to the class of rank-structured matrices. Examples include banded matrices with finite bandwidth [47], Toeplitz matrices in Fourier space [30], [48], [49], some matrices arising from discretized PDEs and integral equations [25]–[27], [50], [51], some kernel matrices [32], [36], etc. The eigenvalue decompositions are very useful in computations such as matrix function evaluations [52], discretized linear system solutions [53], [54], matrix equation solutions, and quadrature approximations [55], [56]. They are also very useful in fields such as optimization, imaging, Gaussian processes, and machine learning. In addition, symmetric eigendecompositions can be used to compute SVDs of non-symmetric matrices.

There are several types of rank-structured forms such as $\mathcal{H}/\mathcal{H}^2$ matrices [5], [6], hierarchical semiseparable (HSS) matrices [14], [57], quasiseparable/semiseparable matrices [10], [58], [59], BLR matrices [15], and HODLR matrices [60]. Examples of eigensolvers for these rank-structured methods include divide-and-conquer methods [39], [47], [61]–[64], QR iterations [9], [10], [65]–[67], and bisection [30], [68]. There are also relevant works like [16], [69] dedicated to the acceleration and stabilization of relevant eigenvalue solutions.

Our work here focuses on the divide-and-conquer method for HSS matrices (that may be dense or sparse). The divide-and-conquer method has previously been well studied for tridiagonal matrices (which may be considered as special HSS forms). See, e.g., [16]–[18], [70]–[72]. In particular, a stable version is given in [16]. The algorithms can compute all the eigenvalues in $O(n^2)$ flops and can compute the eigenvectors in $O(n^3)$ flops. It is also mentioned in [16] that it is possible to accelerate the operations in the divide-and-conquer process via the fast multipole method (FMM) [3] to reach nearly linear complexity. However, this has not actually been done in [16] or later relevant work [47], [61]. Until recently in [39], a divide-and-conquer algorithm is designed for HSS matrices without the need of tridiagonal

reductions. For an HSS matrix with off-diagonal ranks bounded by r (which may be a constant or a power of $\log n$), the method in [39] computes a structured eigendecomposition in $O(r^2 n \log^2 n)$ flops with storage $O(rn \log n)$. The method is said to be *superfast*.

The work in [39] gives a proof-of-concept study of superfast eigendecompositions for HSS matrices A . Yet it does not consider some crucial stability issues in the HSS divide-and-conquer process, such as the risks of exponential norm growth and potential cancellations in some function evaluations. Moreover, it does not incorporate several key stability strategies that are used in practical tridiagonal divide-and-conquer algorithms. In fact, these limitations are due to some major challenges in combining FMM accelerations with those stability strategies. More specifically, the limitations are as follows.

1. During the dividing stage, the diagonal blocks of A (also as HSS blocks) are repeatedly updated along a top-down hierarchical tree traversal. Upper-level off-diagonal blocks are used to update lower-level HSS diagonal blocks. If some upper-level off-diagonal blocks have large norms, then the norms of updated lower-level HSS blocks will grow exponentially during the top-down traversal. This brings stability risks and may even cause overflow.
2. In the conquering stage, the eigenvalues are solved via modified Newton's method applied to some secular equations. On the one hand, in order to apply FMM accelerations, evaluations of the secular function need to be assembled into matrix-vector products. On the other hand, each evaluation of the secular function shall be split into two (say, for the positive terms and negative terms in a summation) to avoid cancellation and also to employ different interpolation methods (see, e.g., [16], [18], [70], [71], [73]). However, such splitting depends on individual eigenvalues, so that the standard FMM acceleration cannot be applied directly. (See Section 3.4.2.) In [39], the FMM is used directly without such splitting, which gives another stability risk.
3. When the eigenvalues of A or any of the intermediate eigenvalue problems are clustered or when an updated eigenvalue is close to a previous one, evaluations of the standard secular function might lose accuracy or even encounter division by zero due

to catastrophic cancellations. This would negatively affect the convergence of modified Newton’s method, and destroy the orthogonality of eigenvectors. To avoid such cancellations, in [16], [18], [70], [71], [73], the *shifted* secular equations are considered rather than the standard ones. However, such shifting is eigenvalue-dependent and there is no uniform shift that works for all the eigenvalues. This makes it difficult to apply FMM accelerations. (See Section 3.4.3 for the details.) Again, the algorithm in [39] directly applies FMM accelerations to standard secular equations without shifting. This is potentially dangerous for practical use.

The main purpose of this work is to overcome these limitations. That is, we seek to design a more reliable superfast divide-and-conquer eigensolver (called SuperDC) to find an approximate eigenvalue decomposition

$$A \approx Q\Lambda Q^T, \tag{3.1}$$

where Λ is a diagonal matrix for the eigenvalues, and Q is for the orthogonal eigenvectors. Here we assume A to be a real and symmetric HSS matrix. The ideas can be immediately extended to the Hermitian case. For convenience, we call the matrix Q an *eigenmatrix*. Compared to [39], we give a sequence of strategies that resolves the stability issues. We also provide many other improvements in terms of the reliability, efficiency, along with certain analysis. The main significance of the work includes the following.

1. We analyze why the original hierarchical dividing strategy in [39] can lead to exponential norm growth. A more stable dividing strategy is designed, with a balancing technique which guarantees the norm growth is well under control. The number of rank-one updates is also minimized to save the cost of intermediate eigenvalue solutions.
2. For accelerations of the solution of the secular equations, we design a triangular FMM to split secular function evaluations into two . When shifted secular equations are solved to avoid cancellations, a *local shifting* strategy is developed to integrate shifting

into FMM matrices without destroying the FMM structure. This improves not only the stability, but also the rate of convergence.

3. We also provide various other discussions and improvements. Examples include the precise structure of the resulting eigenmatrix, the FMM-accelerated iterative eigenvalue solution, the eigenvalue deflation criterion with a user-supplied tolerance.
4. With all the stabilization strategies, SuperDC still has $O(r^2 n \log^2 n)$ complexity with $O(rn \log n)$ storage. Numerical tests for different types of matrices are included. For matrix of moderate size, SuperDC already has significantly lower runtime and storage than the Matlab `eig` function while producing satisfactory accuracy. Benefits of our stability strategies are also demonstrated.

In the remaining sections, we begin in Section 3.2 with a quick review of the basic HSS divide-and-conquer eigensolver in [39]. Then the improved structured dividing strategy is discussed in Section 3.3, followed by the efficient structured conquering scheme in Section 3.4. Section 3.5 gives some comprehensive numerical experiments to demonstrate the efficiency and accuracy. A list of the major algorithms is given in the Section 3.6. We also describe some improvements on the implementations in Section 3.7, followed by the generalization of SuperDC to compute the SVD in Section 3.8. Section 3.9 concludes this chapter.

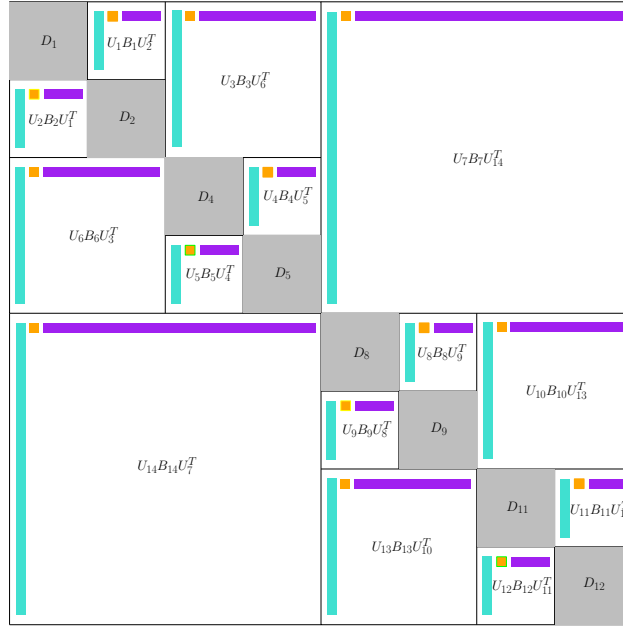
3.2 Review of the basic superfast divide-and-conquer eigensolver

We first briefly summarize the basic superfast divide-and-conquer eigensolver in [39], which generalizes the classical divide-and-conquer method for tridiagonal matrices to HSS matrices.

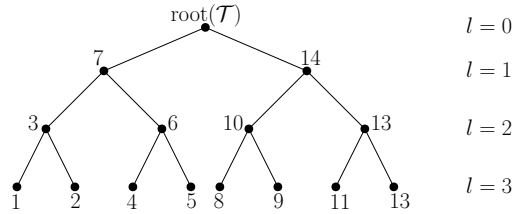
A symmetric HSS matrix A [14] defined with the aid of a postordered full binary tree \mathcal{T} called *HSS tree* has a nested structure that looks like

$$D_p = \begin{pmatrix} D_i & U_i B_i U_j^T \\ U_j B_i^T U_i^T & D_j \end{pmatrix}, \quad (3.2)$$

where $p \in \mathcal{T}$ has child nodes i and j , so that D_p with $p = \text{root}(\mathcal{T})$ is the entire HSS matrix A . Here, the U matrices are off-diagonal basis matrices and also satisfy a nested relationship $U_p = \begin{pmatrix} U_i R_i \\ U_j R_j \end{pmatrix}$. The D_i, U_i, B_i matrices are called *HSS generators* associated with node i . The maximum size of the B generators is usually referred as the *HSS rank* of A . We suppose the root of the HSS tree \mathcal{T} for A is at level 0, and the children of a node i at level l are at level $l + 1$.



(a) 4-level HSS matrix



(b) 4-level HSS tree

Figure 3.1. A 4-level symmetric HSS matrix and its associated HSS binary tree

The superfast divide-and-conquer eigensolver in [39] finds the eigendecomposition (3.1) of A through a dividing stage and a conquering stage as follows.

3.2.1 Dividing stage

In the dividing stage in [39], A and its submatrices are recursively divided into block-diagonal HSS forms plus low-rank updates. Starting with $p = \text{root}(\mathcal{T})$ and its two children i and j . $A = D_p$ in (3.2) can be written as

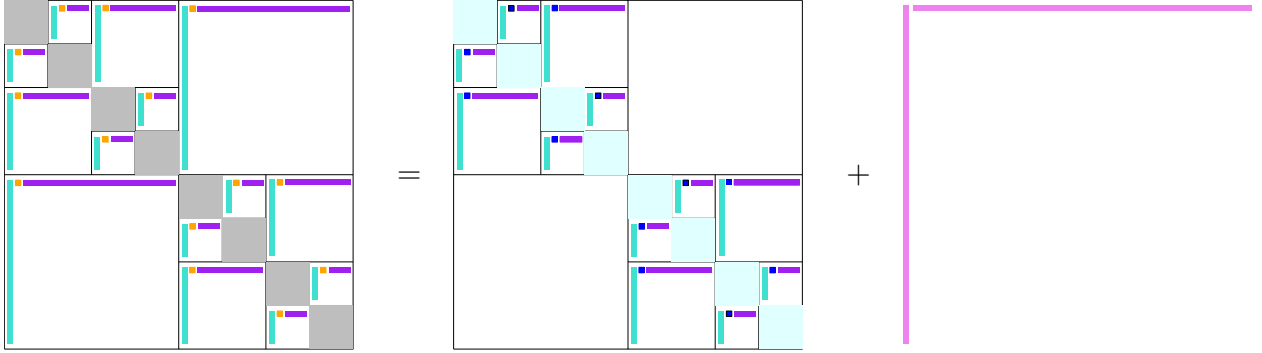
$$D_p = \begin{pmatrix} D_i - U_i B_i B_i^T U_i^T & \\ & D_j - U_j U_j^T \end{pmatrix} + \begin{pmatrix} U_i B_i \\ U_j \end{pmatrix} \begin{pmatrix} B_i^T U_i^T & U_j^T \end{pmatrix}. \quad (3.3)$$

For notational convenience, we suppose the HSS rank of A is r and each B generator has column size r . Let

$$\hat{D}_i = D_i - U_i B_i B_i^T U_i^T, \quad \hat{D}_j = D_j - U_j U_j^T, \quad Z_p = \begin{pmatrix} U_i B_i \\ U_j \end{pmatrix}, \quad (3.4)$$

then we have

$$D_p = \text{diag}(\hat{D}_i, \hat{D}_j) + Z_p Z_p^T. \quad (3.5)$$



Here, the diagonal blocks D_i and D_j are modified so that a rank- r update $Z_p Z_p^T$ can be used instead of a rank- $2r$ update. The column size of Z_p is referred as the *rank of the low-rank update* and here we have $\text{colsize}(Z_p) = \text{colsize}(B_i)$. During this process, the blocks \hat{D}_i and \hat{D}_j remain to be HSS forms. In fact, it is shown in [14], [39] that any matrix of the form $D_i - U_i H U_i^T$ can preserve the off-diagonal basis matrices of D_i . Specifically, the following lemma can be used for generator updates.

Lemma 3.2.1. [39] Let \mathcal{T}_i be the subtree of the HSS tree \mathcal{T} that has the node i as the root. Then $D_i - U_i H U_i^T$ has HSS generators $\tilde{D}_k, \tilde{U}_k, \tilde{R}_k, \tilde{B}_k$ for each node $k \in \mathcal{T}_i$ as follows:

$$\begin{aligned}\tilde{U}_k &= U_k, & \tilde{R}_k &= R_k, \\ \tilde{B}_k &= B_k - (R_k R_{k_l} \cdots R_{k_1}) H (R_{k_1}^T \cdots R_{k_l}^T R_{\tilde{k}}^T), \\ \tilde{D}_k &= D_k - U_k (R_k R_{k_l} \cdots R_{k_1}) H (R_{k_1}^T \cdots R_{k_l}^T R_k^T) U_k^T \quad \text{for a leaf } k,\end{aligned}\tag{3.6}$$

where \tilde{k} is the sibling node of k and $k \rightarrow k_l \rightarrow \cdots \rightarrow k_1 \rightarrow i$ is the path connecting k to i . Accordingly, $D_i - U_i H U_i^T$ and D_i have the same off-diagonal basis matrices.

Thus, the HSS generators of \hat{D}_i and \hat{D}_j can be conveniently obtained via the generator update procedure (3.6). Then the dividing process can continue on \hat{D}_i and \hat{D}_j like above with p in (3.3) replaced by i and j , respectively.

3.2.2 Conquering stage

Suppose eigenvalue decompositions of the subproblems \hat{D}_i and \hat{D}_j in (3.4) have been computed as

$$\hat{D}_i = Q_i \Lambda_i Q_i^T, \quad \hat{D}_j = Q_j \Lambda_j Q_j^T.\tag{3.7}$$

Then from (3.5), we have

$$D_p = \text{diag}(Q_i, Q_j) \left(\text{diag}(\Lambda_i, \Lambda_j) + \hat{Z}_p \hat{Z}_p^T \right) \text{diag}(Q_i^T, Q_j^T),\tag{3.8}$$

where

$$\hat{Z}_p = \text{diag}(Q_i^T, Q_j^T) Z_p.\tag{3.9}$$

Consequently, if we can solve the rank- r update problem

$$\text{diag}(\Lambda_i, \Lambda_j) + \hat{Z}_p \hat{Z}_p^T = \hat{Q}_p \Lambda_p \hat{Q}_p^T,\tag{3.10}$$

then the eigendecomposition of D_p can be simply retrieved as

$$D_p = Q_p \Lambda_p Q_p^T, \quad \text{with} \quad Q_p = \text{diag}(Q_i, Q_j) \hat{Q}_p. \quad (3.11)$$

Therefore, the main task is to compute the eigendecomposition of the low-rank update problem (3.10). To this end, suppose $\hat{Z}_p = (\mathbf{z}_1 \ \cdots \ \mathbf{z}_r)$, where \mathbf{z}_k 's are the columns of \hat{Z}_p . Then (3.10) can be treated as r rank-1 update problems $\text{diag}(\Lambda_i, \Lambda_j) + \sum_{k=1}^r \mathbf{z}_k \mathbf{z}_k^T$. As a result, a basic component is to quickly find the eigenvalue decomposition of a diagonal plus rank-1 update problem in the following form:

$$\tilde{\Lambda} + \mathbf{v} \mathbf{v}^T = \tilde{Q} \Lambda \tilde{Q}^T, \quad (3.12)$$

where $\tilde{\Lambda} = \text{diag}(d_1, \dots, d_n)$ with $d_1 \leq \dots \leq d_n$, $\mathbf{v} = (v_1 \ \cdots \ v_n)^T$, $\tilde{Q} = (\tilde{\mathbf{q}}_1 \ \cdots \ \tilde{\mathbf{q}}_n)$, and $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$.

As in the standard divide-and-conquer eigensolver (see, e.g., [16], [17], [70]), finding λ_k is equivalent to solving the following secular equation [74]:

$$f(x) = 1 + \sum_{k=1}^n \frac{v_k^2}{d_k - x} = 0. \quad (3.13)$$

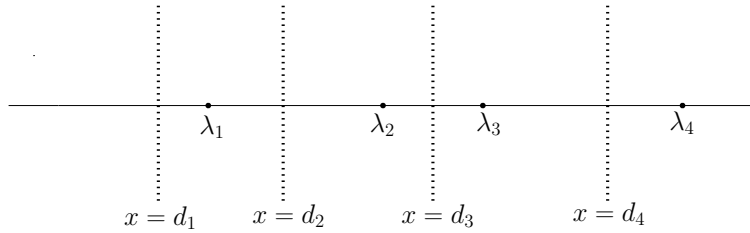


Figure 3.2. Roots of secular equation $1 + \frac{v_1^2}{d_1 - x} + \frac{v_2^2}{d_2 - x} + \frac{v_3^2}{d_3 - x} + \frac{v_4^2}{d_4 - x} = 0$

Newton iterations with rational interpolations may be used, and the cost for finding all the n roots is $O(n^2)$. Once λ_k is computed, a corresponding eigenvector looks like $\tilde{\mathbf{q}}_k = (\tilde{\Lambda} - \lambda_k I)^{-1} \mathbf{v}$. Such an analytical form is not directly used, because any loss of precision in the computed λ_k can be significantly amplified in $(\tilde{\Lambda} - \lambda_k I)^{-1} \mathbf{v}$, and this will result in loss

of eigenvectors orthogonality. Instead, a method in [16] based on Löwner’s formula can be used to obtain $\tilde{\mathbf{q}}_k$ stably.

It is also mentioned in [16] that nearly $O(n)$ complexity may be achieved by assembling multiple operations into matrix-vector multiplications that can be accelerated by the FMM. This is first verified in [39], where the complexity of the algorithm for finding the entire eigendecomposition is $O(r^2n \log^2 n)$ instead of $O(n^3)$, with the eigenmatrix Q in (3.1) given in a structured form that needs $O(rn \log n)$ storage instead of $O(n^2)$. In the following sections, we give a series of stability enhancements to get an improved superfast divide-and-conquer eigensolver.

3.3 Improved structured dividing strategy

In this section, we point out a stability risk in the original dividing method as given in (3.3)–(3.4) and propose a more stable dividing strategy. We also minimize $\text{colsize}(Z_p)$, the rank of the low-rank update.

The stability risk can be illustrated as follows. Consider \hat{D}_i in (3.3) which is the result of updating D_i in the dividing process associated with the parent p of i . Suppose i has children c_1 and c_2 such that

$$D_i = \begin{pmatrix} D_{c_1} & U_{c_1} B_{c_1} U_{c_2}^T \\ U_{c_2} B_{c_2} U_{c_1}^T & D_{c_2} \end{pmatrix}, \quad U_i = \begin{pmatrix} U_{c_1} R_{c_1} \\ U_{c_2} R_{c_2} \end{pmatrix}. \quad (3.14)$$

Then

$$\hat{D}_i = D_i - U_i B_i B_i^T U_i^T = \begin{pmatrix} \tilde{D}_{c_1} & U_{c_1} \tilde{B}_{c_1} U_{c_2}^T \\ U_{c_2} \tilde{B}_{c_1}^T U_{c_1}^T & \tilde{D}_{c_2} \end{pmatrix},$$

where

$$\begin{aligned} \tilde{D}_{c_1} &= D_{c_1} - U_{c_1} R_{c_1} B_i B_i^T R_{c_1}^T U_{c_1}^T, & \tilde{D}_{c_2} &= D_{c_2} - U_{c_2} R_{c_2} B_i B_i^T R_{c_2}^T U_{c_2}^T, \\ \tilde{B}_{c_1} &= B_{c_1} - R_{c_1} B_i B_i^T R_{c_2}^T. \end{aligned} \quad (3.15)$$

In HSS constructions [14], to ensure stability of HSS algorithms, the U basis generators often have orthonormal columns [30], [40], and the R generators also satisfy that $\begin{pmatrix} R_{c_1} \\ R_{c_2} \end{pmatrix}$ has orthonormal columns. Then each B generator has 2-norm equal to its associated off-diagonal block. For example $\|B_i\|_2 = \|U_i B_i U_j^T\|_2$. Furthermore, $\|R_{c_1}\|_2 \leq 1$, $\|R_{c_2}\|_2 \leq 1$, and (3.15) means

$$\|\tilde{B}_{c_1}\|_2 \leq \|B_{c_1}\|_2 + \|B_i\|_2^2. \quad (3.16)$$

If the off-diagonal block $U_i B_i U_j^T$ has a large norm, $\|\tilde{B}_{c_1}\|_2$ can potentially be much larger than $\|B_{c_1}\|_2$. We can similarly observe the norm growth with the updated D generators. Moreover, when the dividing process proceeds on \tilde{D}_{c_1} , the norms of the updated B, D generators at lower levels can grow exponentially.

Proposition 3.3.1. *Suppose the U_k generator of A associated with each node k of \mathcal{T} with $k \neq \text{root}(\mathcal{T})$ has orthonormal columns and all the original B_k generators satisfy $\|B_k\|_2 \leq \beta$ with $\beta \gg 1$. Also suppose the leaves of \mathcal{T} are at level $l_{\max} \leq \log_2 n$. When the original dividing process in Section 3.2.1 proceeds from $\text{root}(\mathcal{T})$ to a nonleaf node i , immediately after finishing the dividing process associated with node i ,*

- *with i at level $l \leq l_{\max} - 2$, the updated B_k generator (denoted \tilde{B}_k) associated with any descendant k of i satisfies*

$$\|\tilde{B}_k\|_2 = O(\beta^{2^l}) \leq O(\beta^{2^{l_{\max}-2}}); \quad (3.17)$$

- *with i at level $l \leq l_{\max} - 1$, the updated D_k generator (denoted \tilde{D}_k) associated with any leaf descendant k of i satisfies*

$$\|\tilde{D}_k\|_2 = \|D_k\|_2 + O(\beta^{2^l}) \leq \|D_k\|_2 + O(\beta^{2^{l_{\max}-1}}). \quad (3.18)$$

Here $O(\cdot)$ means the asymptotic upper bound taken as the highest order term in β . Both upper bounds for $\|\tilde{B}_k\|_2$ and $\|\tilde{D}_k\|_2$ are attainable.

Proof. Following the update formulas in Lemma 3.2.1, we just need to show the norm bound for $\|\tilde{B}_k\|_2$. The bound for $\|\tilde{D}_k\|_2$ can be shown similarly.

After the dividing process associated with $\text{root}(\mathcal{T})$ is finished, according to (3.6), \tilde{B}_k associated with any descendant k of a child i of $\text{root}(\mathcal{T})$ looks like

$$\tilde{B}_k = B_k - (R_k R_{k_{m-1}} \cdots R_{k_1}) H_i (R_{k_1}^T \cdots R_{k_{m-1}}^T R_k^T), \quad (3.19)$$

where $H_i = B_i B_i^T$ if i is the left child of $\text{root}(\mathcal{T})$ or $H_i = I$ otherwise, k is supposed to be at level m with sibling \tilde{k} , and $k \rightarrow k_{m-1} \rightarrow \cdots \rightarrow k_1 \rightarrow i$ is the path connecting k to i in the HSS tree \mathcal{T} . Clearly, $\|H_i\|_2 \leq \beta^2$. With the orthogonality condition of the U basis generators, $\begin{pmatrix} R_{c_1} \\ R_{c_2} \end{pmatrix}$ also has orthogonal columns. Then we get

$$\|\tilde{B}_k\|_2 \leq \|B_k\|_2 + \|H_i\|_2 \leq \beta + \beta^2 = O(\beta^2). \quad (3.20)$$

Then in the dividing process associated with node i at level 1, for a child c of i (see Figure 3.3 for an illustration), the generator \tilde{D}_c is further updated to

$$\hat{D}_c = \tilde{D}_c - U_c H_c U_c^T, \quad (3.21)$$

where $H_c = \tilde{B}_c \tilde{B}_c^T$ if c is the left child of i or $H_c = I$ otherwise. We have $\|H_c\|_2 \leq \|\tilde{B}_c\|_2^2$ for the first case and $\|H_c\|_2 = 1$ for the second case. From (3.20), we have $\|H_c\|_2 \leq (\beta^2 + \beta)^2$. For any descendant k of c with sibling \tilde{k} , (3.21) needs to update the generator B_k to

$$\begin{aligned} \tilde{B}_k &= B_k - (R_k R_{k_{m-1}} \cdots R_{k_2} R_c) H_i (R_c^T R_{k_2}^T \cdots R_{k_{m-1}}^T R_k^T) \\ &\quad - (R_k R_{k_{m-1}} \cdots R_{k_2}) H_c (R_{k_2}^T \cdots R_{k_{m-1}}^T) R_k^T, \end{aligned} \quad (3.22)$$

where the last term on the right-hand side is because of the update associated with the dividing of D_i like in (3.19). Then

$$\|\tilde{B}_k\|_2 \leq \|B_k\|_2 + \|H_i\|_2 + \|H_c\|_2 \leq \beta + \beta^2 + (\beta^2 + \beta)^2 = O(\beta^4). \quad (3.23)$$

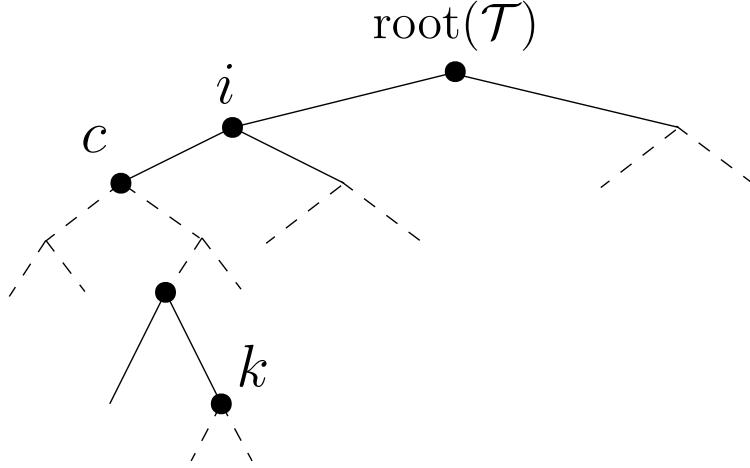


Figure 3.3. Nodes involved in the dividing process.

If the dividing process continues to c , it is similar to obtain $\|\tilde{B}_k\|_2 = O(\beta^8)$ for any descendant k of a child of c . We can then similarly reach the conclusion on the general pattern of the norm growth as in (3.17). Also, if i is at level $l_{\max} - 1$, then B_k associated with a child k of i is not updated, which is why only i at level $l \leq l_{\max} - 2$ contributes to the norm growth of lower level B generators. The upper bounds are attainable. To see this, suppose k_0 is a child of $\text{root}(\mathcal{T})$ and $\|B_{k_0}\|_2 = \beta$, then following the proof we can see that the asymptotic upper bounds (3.17) and (3.18) can be attained at some leaf level node k after the dividing process associated with the parent of k is completed. \square

This proposition indicates that, during the original hierarchical dividing process, the updated B, D generators associated with a lower-level node may potentially have exponential norm accumulation, as long as one of its ancestors is associated with a B generator with a large norm. This can cause stability issues or even overflow, as confirmed in the numerical tests later.

To resolve this, we introduce *balancing*/scaling into the updates and propose a new dividing strategy. That is, we replace the original dividing method (3.3) by

$$D_p = \begin{pmatrix} D_i - \frac{1}{\|B_i\|_2} U_i B_i B_i^T U_i^T & \\ & D_j - \|B_i\|_2 U_j U_j^T \end{pmatrix} \quad (3.24)$$

$$+ \begin{pmatrix} \frac{1}{\sqrt{\|B_i\|_2}} U_i B_i \\ \sqrt{\|B_i\|_2} U_j \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{\|B_i\|_2}} B_i^T U_i^T & \sqrt{\|B_i\|_2} U_j^T \end{pmatrix}.$$

Then we still have (3.5), but with

$$\hat{D}_i = D_i - \frac{1}{\|B_i\|_2} U_i B_i B_i^T U_i^T, \quad \hat{D}_j = D_j - \|B_i\|_2 U_j U_j^T, \quad Z_p = \begin{pmatrix} \frac{1}{\sqrt{\|B_i\|_2}} U_i B_i \\ \sqrt{\|B_i\|_2} U_j \end{pmatrix}. \quad (3.25)$$

With this strategy, we can prove that the norms of the updated B, D generators are well controlled.

Proposition 3.3.2. *Suppose the same conditions as in Proposition 3.3.1 hold, except that (3.3) is replaced by (3.24) so that (3.4) is replaced by (3.25). Then (3.17) becomes*

$$\|\tilde{B}_k\|_2 \leq 2^l \beta \leq 2^{l_{\max}-2} \beta, \quad (3.26)$$

and (3.18) becomes

$$\|\tilde{D}_k\|_2 \leq \|D_k\|_2 + 2^l \beta \leq \|D_k\|_2 + 2^{l_{\max}-1} \beta.$$

Analogously, the upper bounds are attainable.

Proof. The proof follows a procedure similar to the proof for Proposition 3.3.1. Again, we just show the result for $\|\tilde{B}_k\|_2$. After the dividing process associated with $\text{root}(\mathcal{T})$ is finished, we still have (3.19) for any descendant k of a child i of $\text{root}(\mathcal{T})$, except that $H_i = \frac{B_i B_i^T}{\|B_i\|_2}$ if i is the left child of $\text{root}(\mathcal{T})$ or $H_i = \|B_i\|_2 I$ otherwise. In either case, we have $\|H_i\|_2 \leq \beta$. Then (3.20) becomes

$$\|\tilde{B}_k\|_2 \leq 2\beta. \quad (3.27)$$

Then in the dividing process associated with node i at level 1, for a child c of i , the generator \tilde{D}_c is updated like in (3.21), except that $H_c = \frac{\tilde{B}_c \tilde{B}_c^T}{\|\tilde{B}_c\|_2}$ if c is the left child of i or $H_c = \|\tilde{B}_c\|_2 I$ otherwise. We have $\|H_c\|_2 \leq \|\tilde{B}_c\|_2$ for both cases. From (3.27), $\|H_c\|_2 \leq 2\beta$.

For any descendant k of c , (3.21) still requires the update of the generator B_k to \tilde{B}_k like in (3.22), except that (3.23) now becomes

$$\|\tilde{B}_k\|_2 \leq \|B_k\|_2 + \|H_i\|_2 + \|H_c\|_2 \leq \beta + \beta + 2\beta = 4\beta.$$

If the dividing process continues to c , it is similar to obtain $\|\tilde{B}_k\|_2 \leq 8\beta$ for any descendant k of the left child of c . It is clear to observe the norm growth as in (3.26) in general. \square

Therefore, the norm growth now becomes at most linear in n and is well controlled, in contrast to the exponential growth in Proposition 3.3.1.

Next, we can also minimize $\text{colsize}(Z_p)$ (the rank of the low-rank update). Note that in the original dividing method (3.3) in [39], the updates to the two diagonal blocks involve the B_i generator in different ways. No reason is given in [39] to tell why \hat{D}_i and \hat{D}_j should involve B_i differently. In fact, in (3.3) and also (3.24)–(3.25), the rank of the low-rank update is equal to $\text{colsize}(B_i)$. In practice, B_i may not be a square matrix. Thus, (3.25) shall be used only if $\text{colsize}(B_i) \leq \text{rowsize}(B_i)$. Otherwise, we replace (3.25) by the following:

$$\hat{D}_i = D_i - \|B_i\|_2 U_i U_i^T, \quad \hat{D}_j = D_j - \frac{1}{\|B_i\|_2} U_j B_i^T B_i U_j^T, \quad Z_p = \begin{pmatrix} \sqrt{\|B_i\|_2} U_i \\ \frac{1}{\sqrt{\|B_i\|_2}} U_j B_i^T \end{pmatrix}, \quad (3.28)$$

so that (3.5) still holds. In (3.28), the low-rank update size is now $\text{rowsize}(B_i)$. Therefore, the rank of the low-rank update is

$$\text{colsize}(Z_p) = \min(\text{rowsize}(B_i), \text{colsize}(B_i)).$$

When B_i is rectangular, we will have a smaller $\text{colsize}(Z_p)$, which can benefit the efficiency in the conquering stage. With these new ideas, we have a more stable and efficient dividing stage.

3.4 Improved structured conquering stage

In this section, we discuss the solution of the eigenvalues and eigenvectors in the conquering stage via the integration of various stability strategies into FMM accelerations. As reviewed in Section 3.2.2, the key problem in the conquering stage is to quickly find the eigendecomposition of the rank-one update problem (3.12): $\tilde{\Lambda} + \mathbf{v}\mathbf{v}^T = \tilde{Q}\Lambda\tilde{Q}^T$.

In the following, a flexible deflation strategy is first introduced. Then we show a triangular FMM idea for accelerating secular equation solution, a local shifting idea for solving shifted secular equations and constructing structured eigenvectors. We also discuss the framework of the overall eigendecomposition, and the precise structure of the overall eigenmatrix.

3.4.1 Deflation

Following the discussions in [18], [71], a standard deflation step can be first applied to simplify the problem (3.12) when v_k or the difference $|d_k - d_{k+1}|$ is small. In the implementations of the tridiagonal divide-and-conquer eigensolver (see, e.g., [70]), the deflation is performed in a two-step procedure with a tolerance related to ϵ_{mach} . Here, we follow the same steps, but replace ϵ_{mach} with a user-supplied deflation tolerance parameter τ to get a more flexible deflation procedure:

- (i) If $|v_k| < \tau$, without loss of generality we assume $k = n$, then

$$\tilde{\Lambda} + \mathbf{v}\mathbf{v}^T = \begin{pmatrix} \tilde{\Lambda}_1 & \\ & d_n \end{pmatrix} + \begin{pmatrix} \mathbf{v}_1 \\ v_n \end{pmatrix} \begin{pmatrix} \mathbf{v}_1^T & v_n \end{pmatrix} = \begin{pmatrix} \tilde{\Lambda}_1 + \mathbf{v}_1\mathbf{v}_1^T & \\ & d_n \end{pmatrix} + O(\tau).$$

Then we only need to solve the smaller problem $\tilde{\Lambda}_1 + \mathbf{v}_1\mathbf{v}_1^T = \tilde{Q}_1\Lambda_1\tilde{Q}_1^T$.

(ii) If $|(d_k - d_{k+1})v_k v_{k+1}| < (v_k^2 + v_{k+1}^2)\tau$, we can find a Givens rotation G

$$\begin{aligned} & G \left(\begin{pmatrix} d_k & & \\ & d_{k+1} & \\ & & \end{pmatrix} + \begin{pmatrix} v_k \\ v_{k+1} \end{pmatrix} \begin{pmatrix} v_k & v_{k+1} \end{pmatrix}^T \right) G^T \\ &= \begin{pmatrix} d_k & \mu \\ \mu & d_{k+1} \end{pmatrix} + \begin{pmatrix} 0 \\ \sqrt{v_k^2 + v_{k+1}^2} \end{pmatrix} \begin{pmatrix} 0 & \sqrt{v_k^2 + v_{k+1}^2} \end{pmatrix}^T \\ &\approx \begin{pmatrix} d_k & \\ & d_{k+1} \end{pmatrix} + \begin{pmatrix} 0 \\ \sqrt{v_k^2 + v_{k+1}^2} \end{pmatrix} \begin{pmatrix} 0 & \sqrt{v_k^2 + v_{k+1}^2} \end{pmatrix}^T, \end{aligned}$$

where $|\mu| = \left| \frac{(d_k - d_{k+1})v_k v_{k+1}}{v_k^2 + v_{k+1}^2} \right| \leq \tau$. This is reduced to the first case.

After the above two deflation steps, the problem size of (3.12) is reduced and the simplified problem satisfies

$$|v_k| \geq \tau, \quad \text{and} \quad |d_k - d_{k+1}| \geq \frac{(v_k^2 + v_{k+1}^2)\tau}{|v_k v_{k+1}|}. \quad (3.29)$$

The parameter τ offers the flexibility to control the accuracy of the eigenvalues. When only moderate accuracy is needed, a larger τ can be used for a larger reduction in problem size. Moreover, this can sometimes avoid the need to deal with situations where $|\lambda_k - d_k|$ or $|\lambda_k - d_{k+1}|$ is too small, see Section 3.4.3.

3.4.2 Fast secular equation solution

Assume (3.29) holds for (3.12) so that no more deflation is needed. We consider the solution of the secular equation (3.13) for its eigenvalues $\lambda_k, k = 1, 2, \dots, n$. Without loss of generality, suppose the diagonal entries d_k of $\tilde{\Lambda}$ are ordered from the smallest to the largest.

Standard FMM for fast evaluations of the secular function

When modified Newton's method is used to solve for λ_k , it needs to evaluate the secular function f and its derivative f' at certain $x_k \in (d_k, d_{k+1})$. The idea in [16], [39], [61] is to

assemble the function evaluations for all k together as matrix-vector products, so that fast evaluations can be done via the standard FMM matrix multiplication. That is, let

$$\begin{aligned} \mathbf{f} &= \left(f(x_1) \ \cdots \ f(x_n) \right)^T, \quad \mathbf{f}' = \left(f'(x_1) \ \cdots \ f'(x_n) \right)^T, \\ \mathbf{v} &= \left(v_1 \ \cdots \ v_n \right)^T, \quad \mathbf{w} = \mathbf{v} \odot \mathbf{v}, \quad \mathbf{e} = \left(1 \ \cdots \ 1 \right)^T, \end{aligned} \quad (3.30)$$

$$C = \left(\frac{1}{d_j - x_i} \right)_{n \times n}, \quad S = \left(\frac{1}{(d_j - x_i)^2} \right)_{n \times n}. \quad (3.31)$$

$$\mathbf{f} = \mathbf{e} + C\mathbf{w}, \quad \mathbf{f}' = S\mathbf{w}. \quad (3.32)$$

The vectors \mathbf{f} and \mathbf{f}' can be quickly evaluated by the 1D FMM with the kernel functions $\kappa(s, t) = \frac{1}{s-t}$ and $\kappa(s, t) = \frac{1}{(s-t)^2}$, respectively. We will briefly introduce the 1D FMM for completeness, since its essential idea is analogous to the 2D FMM in Chapter 2. The basic idea of the 1D FMM for computing, say, $C\mathbf{w}$ is as follows. Note that C is the evaluation of the kernel $\kappa(s, t) = \frac{1}{s-t}$ at real points $s \in \{d_j\}_{1 \leq j \leq n}$ and $t \in \{x_i\}_{1 \leq i \leq n}$ that are interlaced:

$$d_i < x_i < d_{i+1} < x_{i+1}, \quad 1 \leq i \leq n-1. \quad (3.33)$$

The sets $\{x_i\}_{1 \leq i \leq n}$ and $\{d_j\}_{1 \leq j \leq n}$ are first partitioned into subsets. This is done via a hierarchical bisection of the interval that covers all x_i 's and d_j 's. Consider two subsets produced in this partitioning:

$$\mathbf{s}_x \subset \{x_i\}_{1 \leq i \leq n}, \quad \mathbf{s}_d \subset \{d_j\}_{1 \leq j \leq n}. \quad (3.34)$$

Use $C_{\mathbf{s}_x, \mathbf{s}_d} = (\kappa(d_j, x_i))_{x_i \in \mathbf{s}_x, d_j \in \mathbf{s}_d}$ to denote the block of C defined by \mathbf{s}_x and \mathbf{s}_d , which is often referred as the *interaction* between \mathbf{s}_x and \mathbf{s}_d .

- (i) If \mathbf{s}_x and \mathbf{s}_d are well separated (see (2.12)), then $C_{\mathbf{s}_x, \mathbf{s}_d}$ can be approximated by a low-rank form

$$C_{\mathbf{s}_x, \mathbf{s}_d} \approx U_{\mathbf{s}_x} B_{\mathbf{s}_x, \mathbf{s}_d} V_{\mathbf{s}_d}^T. \quad (3.35)$$

As in Theorem 2.3.1, such low-rank approximation can be obtained via a degenerate expansion of $\kappa(s, t)$. The size of $B_{\mathbf{s}_x, \mathbf{s}_d}$ depends logarithmically on the desired approx-

imation accuracy, and can be treated as bounded even when (3.35) achieves double-precision. Subsets \mathbf{s}_x and \mathbf{s}_d are also said to be *far-field* clusters and the submatrix $C_{\mathbf{s}_x, \mathbf{s}_d}$ to be *far-field interactions* or *far-field blocks*.

- (ii) If \mathbf{s}_x and \mathbf{s}_d are not well separated, then they are said to be *near-field* clusters, and $C_{\mathbf{s}_x, \mathbf{s}_d} = (\kappa(d_j, x_i))_{x_i \in \mathbf{s}_x, d_j \in \mathbf{s}_d}$ is treated as a regular dense block (also referred as *near-field interactions* or *near-field blocks*).

Moreover, the interactions between parent cluster and child cluster during the hierarchical partition are considered, so that the U, V basis matrices in (3.35) satisfy nested relationships (see Theorem 2.4.1 and Remark 2.4.2). The 1D FMM essentially constructs an *FMM matrix* approximation to C and multiplies it with \mathbf{w} . The complexity of each FMM matrix-vector multiplication is $O(n)$.

In light of (3.31) and (3.32), a straightforward idea in [16], [39], [61] is to apply the standard FMM to C and S for fast evaluations of \mathbf{f} and \mathbf{f}' . However, in practical implementations of secular equation solution methods, it is preferred to write $f(x)$ in the following form to avoid cancellation (see, [18], [70], [71]):

$$f(x) = 1 + \psi_k(x) + \phi_k(x),$$

where the splitting depends on k (when $\lambda_k \in (d_k, d_{k+1})$ is to be found):

$$\psi_k(x) = \sum_{j=1}^k \frac{v_j^2}{d_j - x}, \quad \phi_k(x) = \sum_{j=k+1}^n \frac{v_j^2}{d_j - x}. \quad (3.36)$$

Because of the interlacing property (3.33), all the terms in the sum for ψ_k (resp. ϕ_k) have the same sign for $x \in (d_k, d_{k+1})$. Furthermore, ψ_k and ϕ_k capture the behavior of f near two poles d_k and d_{k+1} respectively. Therefore, ψ_k and ϕ_k can be used in any interpolation-based strategy for finding the roots of f . A reliable and widely used strategy to solve (3.13) is proposed in [73], and based on a modified Newton's method with a hybrid scheme for rational interpolations. The scheme mixes a middle way method and a fixed weight method and is implemented in LAPACK [70]. In the middle way method, rational functions

$\xi_{k,1}(x) = a_1 + \frac{b_1}{d_k - x}$ and $\xi_{k,2}(x) = a_2 + \frac{b_2}{d_{k+1} - x}$ are decided to interpolate ψ_k and ϕ_k respectively at $x_k \in (d_k, d_{k+1})$, so that

$$\xi_{k,1}(x_k) = \psi_k(x_k), \quad \xi'_{k,1}(x_k) = \psi'_k(x_k), \quad \xi_{k,2}(x_k) = \phi_k(x_k), \quad \xi'_{k,2}(x_k) = \phi'_k(x_k).$$

(SuperDC also follows this hybrid scheme to find the first $n - 1$ roots $\lambda_1, \lambda_2, \dots, \lambda_{n-1}$. The last root λ_n has only one pole d_n next to it, so a simple rational interpolation is used as in [70], [73]).

The modified Newton's method requires evaluations of the functions ψ_k , ϕ_k , ψ'_k , and ϕ'_k at some $x_k \in (d_k, d_{k+1})$, $1 \leq k \leq n - 1$. (Note that even though the summands in ψ'_k and ϕ'_k have the same sign, ψ'_k and ϕ'_k are used separately in the rational interpolations by $\xi_{k,1}$ and $\xi_{k,2}$, respectively [73].) Since these functions all depend on individual k , the standard FMM cannot be applied directly. The reason is that the standard FMM handles the evaluation of a kernel $\kappa(s, t)$ at a fixed set of data points, while here these k -dependent functions need to evaluate $\kappa(s, t)$ at some k -dependent subsets of the data points. This poses a challenge to applying the standard FMM accelerations to (3.36).

Triangular FMM for fast evaluations of ψ_k and ϕ_k

To resolve the challenge of applying FMM accelerations to (3.36), we let

$$\boldsymbol{\psi} = \begin{pmatrix} \psi_1(x_1) & \cdots & \psi_n(x_n) \end{pmatrix}^T, \quad \boldsymbol{\phi} = \begin{pmatrix} \phi_1(x_1) & \cdots & \phi_{n-1}(x_{n-1}) & 0 \end{pmatrix}^T, \quad (3.37)$$

$$\boldsymbol{\psi}' = \begin{pmatrix} \psi'_1(x_1) & \cdots & \psi'_n(x_n) \end{pmatrix}^T, \quad \boldsymbol{\phi}' = \begin{pmatrix} \phi'_1(x_1) & \cdots & \phi'_{n-1}(x_{n-1}) & 0 \end{pmatrix}^T. \quad (3.38)$$

The key idea is to write

$$\mathbf{f} = \mathbf{e} + \boldsymbol{\psi} + \boldsymbol{\phi} = \mathbf{e} + C_L \mathbf{w} + C_U \mathbf{w}, \quad \mathbf{f}' = \boldsymbol{\psi}' + \boldsymbol{\phi}' = S_L \mathbf{w} + S_U \mathbf{w}, \quad (3.39)$$

where \mathbf{e} is given in (3.30), C_L and S_L are the lower triangular parts of C and S , respectively, and C_U and S_U are the strictly upper triangular parts of C and S , respectively. This suggests that the FMM idea should be applied to the lower and upper triangular parts of C and S

separately. That is, we need a special *triangular FMM* that can be used to quickly evaluate the triangular matrix-vector products $C_L \mathbf{w}$, $C_U \mathbf{w}$, $S_L \mathbf{w}$, $S_U \mathbf{w}$.

In the following, we use $C_L \mathbf{w}$ as a specific example to describe the design of triangular FMM. For two subsets $\mathbf{s}_x, \mathbf{s}_d$ in (3.34), we similarly use $(C_L)_{\mathbf{s}_x, \mathbf{s}_d}$ to denote the block of C_L defined by \mathbf{s}_x and \mathbf{s}_d .

For the far-field block $(C_L)_{\mathbf{s}_x, \mathbf{s}_d}$ where \mathbf{s}_x and \mathbf{s}_d are well-separated, one and only one of the following two cases is true:

1. \mathbf{s}_x is on the left of \mathbf{s}_d , i.e., $\max \mathbf{s}_x < \min \mathbf{s}_d$. In this case, because of the interlacing property (3.33), the block $(C_L)_{\mathbf{s}_x, \mathbf{s}_d}$ is in the upper triangular part of C_L . Since C_L is a lower triangular matrix, $(C_L)_{\mathbf{s}_x, \mathbf{s}_d}$ must be the zero matrix,

$$(C_L)_{\mathbf{s}_x, \mathbf{s}_d} = 0.$$

2. \mathbf{s}_x is on the right of \mathbf{s}_d , i.e., $\min \mathbf{s}_x > \max \mathbf{s}_d$. In this case, because of the interlacing property (3.33), the block $(C_L)_{\mathbf{s}_x, \mathbf{s}_d}$ is in the lower triangular part of C_L . Since C_L is defined to be the lower triangular part of C , we have

$$(C_L)_{\mathbf{s}_x, \mathbf{s}_d} = C_{\mathbf{s}_x, \mathbf{s}_d} \approx U_{\mathbf{s}_x} B_{\mathbf{s}_x, \mathbf{s}_d} V_{\mathbf{s}_d}^T.$$

The above two cases can be unified as

$$(C_L)_{\mathbf{s}_x, \mathbf{s}_d} \approx U_{\mathbf{s}_x} \tilde{B}_{\mathbf{s}_x, \mathbf{s}_d} V_{\mathbf{s}_d}^T, \quad (3.40)$$

where

$$\tilde{B}_{\mathbf{s}_x, \mathbf{s}_d} = \begin{cases} 0 & \text{if } \max \mathbf{s}_x < \min \mathbf{s}_d \\ B_{\mathbf{s}_x, \mathbf{s}_d} & \text{if } \min \mathbf{s}_x > \max \mathbf{s}_d \end{cases}.$$

On the other hand, suppose \mathbf{s}_x and \mathbf{s}_d are neighbor clusters, then $(C_L)_{\mathbf{s}_x, \mathbf{s}_d}$ is a dense near-field block. Again, because of the interlacing property (3.33), $(C_L)_{\mathbf{s}_x, \mathbf{s}_d}$ is on the diagonal part of C_L . Since C_L is the lower triangular part of C , we can assemble $(C_L)_{\mathbf{s}_x, \mathbf{s}_d}$ by selecting appropriate nonzero elements from $C_{\mathbf{s}_x, \mathbf{s}_d}$.

Therefore, analogous to the standard FMM, we can construct a *lower triangular FMM approximation matrix* to C_L and multiplies it with \mathbf{w} in $O(n)$ operations. With the triangular FMM, fast and accurate function evaluations in modified Newton's method is now feasible. The cost of one simultaneous iteration step for all x_k 's is $O(n)$.

Iterative secular equation solution

During the modified Newton's method, let $x_k^{(j)}$ be an approximation to the eigenvalue λ_k at the iteration step j . A correction $\Delta x_k^{(j)}$ is computed to update $x_k^{(j)}$ as

$$x_k^{(j+1)} \leftarrow x_k^{(j)} + \Delta x_k^{(j)}. \quad (3.41)$$

(We sometimes write x_k instead of $x_k^{(j)}$ unless we specifically discuss the details of the iterations.)

We adopt the stopping criterion from [16]:

$$|f(x_k^{(j)})| < cn(1 + |\psi_k(x_k^{(j)})| + |\phi_k(x_k^{(j)})|)\epsilon_{\text{mach}}, \quad (3.42)$$

where c is a small constant. This stopping criterion can be conveniently checked after the FMM-accelerated function evaluations, which is an advantage over a criterion in [73]. Although (3.42) might be loose for an large n , it works well in our tests and produces satisfactory accuracy for large matrices. It is possible to refine (3.42) to a tighter convergence estimate using the results for the backward stability of hierarchical algorithms in [32], [40]. This is our ongoing work.

Typically, a very small number of iterations is needed for convergence. In our experiments, each eigenvalue converges in 2 to 5 iterations on average. We want to point out that this is slightly more than the case in the standard tridiagonal divide and conquer (2 or 3 iterations on average, as pointed out in [75]). This can be explained by the fact that FMM is an approximation algorithm to evaluate the secular functions, so it might need one or two extra iterations. Note that it is possible for few eigenvalues to converge slower than most others. We may just use the standard iteration method in [16], [73] for those eigenvalues.

With the total number of iterations bounded, the total iterative solution cost for finding all the eigenvalues from one secular equation is $O(n)$.

3.4.3 Local shifting in triangular FMM for shifted secular equation solution

When there are clustered eigenvalues or when updates to previous eigenvalues are small, typically the standard secular equation (3.13) is not directly solved. Instead, shifted secular equations are solved for the purpose of stability and accuracy, as discussed in [16], [18], [71]. However, it is nontrivial to apply FMM to accelerate shifted secular equation solution. In fact, the paper [16] mentions the possibility of FMM accelerations for the standard secular equation but does not consider the shifted ones. The FMM-accelerated algorithm in [39] does not use shifted secular equations either and thus has stability risks.

In this subsection, we discuss the necessity of shifting and its challenges to FMM accelerations. Moreover, we develop a new strategy that makes feasible applying FMM accelerations to shifted secular equations. In the following, we suppose deflation in Section 3.4.1 has already been applied.

Shifted secular equation solution and its challenge to FMM accelerations

The original secular equation (3.13) can be rewritten as the equivalent *shifted secular equation* (see, e.g., [16], [18], [70]):

$$g_k(y) \equiv f(d_k + y) = 1 + \sum_{j=1}^n \frac{v_j^2}{\delta_{jk} - y} = 0, \quad (3.43)$$

where

$$\delta_{jk} = d_j - d_k, \quad j = 1, 2, \dots, n. \quad (3.44)$$

Here we assume $f(\frac{d_k+d_{k+1}}{2}) \geq 0$ so that $d_k < \lambda_k \leq \frac{d_k+d_{k+1}}{2}$ and λ_k is closer to d_k . If $f(\frac{d_k+d_{k+1}}{2}) < 0$, then λ_k is closer to d_{k+1} and we can replace d_k in (3.43) and (3.44) with d_{k+1} . The difference (also referred as the gap) $\eta_k \equiv \lambda_k - d_k$ can be computed by solving (3.43) for $y = \eta_k$. Note that in exact arithmetic we have $\delta_{ik} - \eta_k = d_i - \lambda_k$, however it is

preferred to compute $\delta_{ik} - \eta_k$ since it does not suffer from cancellation (see, e.g., [16], [71]). We would like to provide more details on the benefits of this shifting within our context.

One benefit is to avoid catastrophic cancellation or division by zero (see, e.g., [16], [70], [71]). To be more specific, we illustrate this with the following example. Let x_k be an computed approximation to λ_k . In exact arithmetic, x_k shall lie strictly between d_k and d_{k+1} . At each modified Newton iteration, it needs to guarantee $d_k < \text{fl}(x_k) < d_{k+1}$. However, this might not be satisfied in floating point arithmetic when x_k is very close to d_k :

$$|d_k - x_k| = O(\epsilon_{\text{mach}}) \text{ or smaller,} \quad (3.45)$$

which may lead to cancellation when computing $d_k - \text{fl}(x_k)$:

$$\text{fl}(d_k - \text{fl}(x_k)) = o(\epsilon_{\text{mach}}) \quad \text{or even} \quad \text{fl}(d_k - \text{fl}(x_k)) = 0. \quad (3.46)$$

This will cause stability issues in the numerical solutions of the standard secular function: $\text{fl}\left(\frac{v_k^2}{d_k - \text{fl}(x_k)}\right)$ either is highly inaccurate or becomes ∞ .

Note that (3.45) and (3.46) are still possible even if deflation in Section 3.4.1 has been applied with a tolerance τ that is not too small. To see this, suppose $v_k = O(\tau) \geq \tau$ and the exact k^{th} root λ_k satisfies $|\lambda_k - d_j| \gg v_j^2$ for $j \neq k$. Substituting λ_k into the secular equation (3.13), we get $\frac{v_k^2}{d_k - \lambda_k} = -1 + \sum_{j \neq k}^n \frac{v_j^2}{\lambda_k - d_j} = O(1)$. In this case, λ_k shall be very close to d_k in the following sense:

$$|d_k - \lambda_k| = v_k^2 \cdot O(1) = O(\tau^2).$$

If $\tau = O(\epsilon_{\text{mach}}^{1/2})$ which is not extremely small, we can have (3.45) so that (3.46) may happen when solving the standard secular equation.

Another benefit for solving the shifted equation is faster convergence. It is observed in our tests that computing with η_k instead of λ_k can speed up the convergence of modified Newton's method. To illustrate this, suppose λ_k is solved directly from the standard secular equation (3.13), then the approximation $x_k^{(j)}$ at iteration step j is updated as in (3.41). Suppose $|\lambda_k| = O(1)$ and $|\eta_k| = |\lambda_k - d_k| = O(\epsilon_{\text{mach}})$. Since $x_k^{(j)}$ converges to λ_k as $j \rightarrow \infty$, we also have $|x_k^{(j)}| = O(1)$ and $|x_k^{(j)} - d_k| = O(\epsilon_{\text{mach}})$ after some iterations. By modified

Newton's method, the correction $\Delta x_k^{(j)}$ approaches 0 as j increase. This may lead to loss of digits in the updated $x_k^{(j+1)}$: $\text{fl}(x_k^{(j+1)}) = \text{fl}(x_k^{(j)} + \Delta x_k^{(j)}) = \text{fl}(x_k^{(j)})$. As a result, the iteration stagnates. On the other hand, if η_k is solved from the shifted secular equation (3.43), as in [18], [70], [71], the update (3.41) is replaced by

$$y_k^{(j+1)} \leftarrow y_k^{(j)} + \Delta x_k^{(j)}, \quad (3.47)$$

where $y_k^{(j)} = x_k^{(j)} - d_k$ is an approximation to η_k at step j of the iterative solution. Although (3.41) and (3.47) are equivalent in exact arithmetic, the latter preserves a lot more digits of accuracy since $|y_k^{(j)}| = O(\epsilon_{\text{mach}})$.

These discussions illustrate the importance of solving the shifted secular equation (3.43) instead of the original equation (3.13). However, in an FMM-accelerated scheme where all λ_k 's are solved simultaneously, it is not plausible to shift the secular equation simultaneously for all λ_k 's. The reason is the shift in (3.43) depends on each individual eigenvalue and there is no such a uniform shift that would work for all λ_k 's. To see this, let $y_k = x_k - d_k$ be an approximation to η_k during the iterative solution of (3.43). The evaluations of $g_k(y)$ in (3.43) at $y = y_k$ for all $k = 1, 2, \dots, n$ can be assembled into the matrix form

$$\mathbf{g} = \mathbf{e} + \hat{C}\mathbf{w} = \mathbf{e} + \hat{C}_L\mathbf{w} + \hat{C}_U\mathbf{w}, \quad \text{with} \quad (3.48)$$

$$\mathbf{g} = \begin{pmatrix} g_1(y_1) & \cdots & g_n(y_n) \end{pmatrix}^T, \quad \hat{C} = \begin{pmatrix} 1 \\ \delta_{jk} - y_k \end{pmatrix}_{1 \leq k, j \leq n}, \quad (3.49)$$

where δ_{jk} is given in (3.44) and \hat{C}_L, \hat{C}_U are similarly defined as in (3.39).

Recall that when the triangular FMM is used to accelerate the matrix-vector product $C\mathbf{w}$ in (3.39), it relies on the separability of s and t in a degenerate approximation of $\kappa(s, t) = \frac{1}{s-t}$. (Note that in $\kappa(d_j, x_k)$, x_k only involves the row index k and d_j only involves the column index j , so that the separability can be understood in terms of the row and column indices.) However, to evaluate $\hat{C}\mathbf{w}$ in (3.48), we have

$$\kappa(d_j, x_k) = \kappa(d_j - d_k, x_k - d_k) = \kappa(\delta_{jk}, y_k). \quad (3.50)$$

δ_{jk} involves both the row and column indices, so that the separability in terms of the row and column indices does not hold. Also, there is no obvious way of rewriting $\kappa(\delta_{jk}, \eta_k)$ to produce separability in j and k . If there exists such a uniform shift d_0 , then $\kappa(d_j, x_k) = \kappa(d_j - d_0, x_k - d_0)$ and the triangular FMM framework would still apply. However, the shift d_k as above for λ_k depends on the local behavior of the secular function in (d_k, d_{k+1}) so such d_0 does not exist.

One possible remedy is as follows. The FMM-accelerated iterations are applied to solve the original secular equation (3.13) via K . In the meantime, whenever the difference $|x_k - d_k|$ is too small for a certain eigenvalue λ_k , switch to solve the shifted equation (3.43) without FMM accelerations to get λ_k . However, if (3.45) happens very often when a small tolerance τ is used for high accuracy or when the problem is not very nice, then the efficiency will be reduced significantly since every such a case costs extra $O(n)$ flops. Also, when a shift like this is involved, the corresponding eigenvector needs to be represented in the usual way for the accuracy purpose (instead of using the structured form as in Section 3.4.4 later). This requires extra storage for extra (regular) eigenvectors. Thus, this remedy is not fully satisfactory.

Therefore, we need to adapt the triangular FMM for fast evaluation of the shifted matrix-vector product (3.48).

FMM accelerations with local shifting

In this subsection, we propose a strategy called *local shifting* that makes it feasible to apply triangular FMM accelerations to solve (3.43).

As mentioned in Section 3.4.2, multiple terms involving $x_k - d_j$ are assembled into matrices in order to apply FMM accelerations. See, e.g., (3.31). When $|x_k - d_j|$ is small, the shifting helps get $x_k - d_j$ accurately. However, when k is not near j or when $|k - j|$ is large, $x_k - d_j$ can actually be computed accurately *without involving any shift* d_k . To see this, recall that $d_k < x_k < d_{k+1}$ and also after deflation in Section 3.4.1, we have for all j ,

$$|d_j - d_{j+1}| \geq \frac{v_j^2 + v_{j+1}^2}{|v_j v_{j+1}|} \geq 2\tau.$$

Thus, for $j \neq k, k + 1$,

$$|x_k - d_j| \geq \min(|d_k - d_j|, |d_{k+1} - d_j|) \geq 2(|k - j| - 1)\tau. \quad (3.51)$$

Hence, $x_k - d_j$ can be computed accurately when $|k - j|$ is large.

Following this justification, we have our local shifting strategy with the following basic ideas: (i) use the gap η_k for each eigenvalue λ_k in near-field interactions of FMM, which does not interfere with the structures needed for FMM accelerations; (ii) it is safe to directly use λ_k recovered from

$$\lambda_k = d_k + \eta_k, \quad k = 1, 2, \dots, n, \quad (3.52)$$

in far-field interactions to exploit the rank structure and facilitate FMM accelerations.

The major details are as follows.

1. For $k = 1, 2, \dots, n$, the shifted secular equations (3.43) are solved together for the gaps $\eta_k = \lambda_k - d_k$. An intermediate gap during the iterative solution looks like $y_k = x_k - d_k$. The relevant function evaluations in the iterative solutions are assembled into matrix-vector products like in (3.48).
2. The FMM is used to accelerate the resulting matrix-vector products like $\hat{C}\mathbf{w}$ in (3.48) as follows. On the one hand, suppose two subsets \mathbf{s}_x and \mathbf{s}_d like in (3.34) are well-separated. As mentioned above, for $x_k \in \mathbf{s}_x$ and $d_j \in \mathbf{s}_d$, x_k and d_j are far away from each other and $|k - j|$ is large, so $x_k - d_j$ can then be computed accurately because of (3.51). Thus, we can recover x_k from $d_k + y_k$ to directly exploit the low-rank structure like in (3.35). As a result, the far-field block $\hat{C}_{\mathbf{s}_x, \mathbf{s}_d}$ of \hat{C} is now just a block of C in (3.31):

$$\hat{C}_{\mathbf{s}_x, \mathbf{s}_d} = (\kappa(\delta_{jk}, y_k))_{x_k \in \mathbf{s}_x, d_j \in \mathbf{s}_d} = (\kappa(d_j, x_k))_{x_k \in \mathbf{s}_x, d_j \in \mathbf{s}_d} = C_{\mathbf{s}_x, \mathbf{s}_d}$$

3. On the other hand, when two subsets \mathbf{s}_x and \mathbf{s}_d are not well separated, the near-field interaction $\hat{C}_{\mathbf{s}_x, \mathbf{s}_d} = (\kappa(\delta_{jk}, y_k))_{x_k \in \mathbf{s}_x, d_j \in \mathbf{s}_d}$ is kept dense and each entry $\kappa(\delta_{jk}, y_k)$ can be evaluated accurately via y_k and δ_{jk} . This has no impact on the structures needed for FMM accelerations.

4. These ideas are combined with the triangular FMM in Section 3.4.2 to stably and quickly perform function evaluations like (3.48) and solve the shifted secular equations.

This local shifting strategy successfully integrates shifting into the triangular FMM framework without sacrificing performance. As a result, we can quickly and reliably solve the shifted secular equations (3.43) via modified Newton’s method. The overall complexity to find all the n roots is still $O(n)$. In addition, since the relevant functions are now evaluated more accurately than with the method in [39], the convergence is also improved. (This can be confirmed from our tests later.) When the iterative solution of the shifted secular equations converge, we can use the resulting η_k values to recover the desired eigenvalues as in (3.52).

The local shifting strategy can also be used to stably apply FMM accelerations to other operations like finding the eigenmatrix. See the next subsection.

3.4.4 Structured eigenvectors via FMM with local shifting

With the identified eigenvalues λ_k in (3.52), the eigenvectors can be obtained stably as in [16]. An eigenvector corresponding to λ_k looks like

$$\mathbf{q}_k = \left(\frac{\hat{v}_1}{d_1 - \lambda_k} \quad \cdots \quad \frac{\hat{v}_k}{d_k - \lambda_k} \quad \cdots \quad \frac{\hat{v}_n}{d_n - \lambda_k} \right)^T, \quad (3.53)$$

where $\hat{\mathbf{v}} \equiv (\hat{v}_1 \quad \cdots \quad \hat{v}_n)^T$ is given by Löwner’s formula

$$\hat{v}_i = \sqrt{\frac{\prod_j (\lambda_j - d_i)}{\prod_{j \neq i} (d_j - d_i)}}, \quad i = 1, 2, \dots, n. \quad (3.54)$$

To quickly form $\hat{\mathbf{v}}$, the standard FMM acceleration would look like the following [16]. Rewrite (3.54) as

$$\log \hat{v}_i = \frac{1}{2} \sum_{j=1}^n \log(|d_i - \lambda_j|) - \frac{1}{2} \sum_{j=1, j \neq i}^n \log |d_i - d_j|. \quad (3.55)$$

Now let $G_1 = (\log |d_i - \lambda_j|)_{n \times n}$, $G_2 = (\log |d_i - d_j|)_{n \times n}$, where the diagonals of G_2 are set to be zero. Then

$$\log \hat{\mathbf{v}} = \frac{1}{2}(G_1 \mathbf{e} - G_2 \mathbf{e}). \quad (3.56)$$

$G_1 \mathbf{e}$ and $G_2 \mathbf{e}$ can thus be quickly evaluated by the FMM with the kernel $\log |s - t|$.

As in [16], [39], the eigenvectors are often normalized to form an orthogonal matrix

$$\hat{Q} = \left(\frac{\hat{v}_i b_j}{d_i - \lambda_j} \right)_{n \times n}, \quad (3.57)$$

where

$$\mathbf{b} \equiv (b_1 \ \dots \ b_n)^T, \quad \text{with} \quad b_j = \left(\sum_{i=1}^n \frac{\hat{v}_i^2}{(d_i - \lambda_j)^2} \right)^{-1/2}. \quad (3.58)$$

Again, the vector \mathbf{b} can be quickly obtained via the FMM with the kernel $\kappa(s, t) = \frac{1}{(s-t)^2}$. \hat{Q} is a Cauchy-like matrix which gives a structured form of the eigenvectors. The FMM with the kernel $\kappa(s, t) = \frac{1}{s-t}$ can be used to quickly multiply \hat{Q} to a vector.

Again, with the same reasons as before, it is challenging to stably apply the standard FMM to accelerate operations like the evaluations of $\log \mathbf{v}$ in (3.56) and \mathbf{b} in (3.58) and the application of \hat{Q} to a vector. On the other hand, just like the discussions in Section 3.4.3, the local shifting strategy still applies with appropriate kernels $\kappa(s, t)$. Thus, instead of directly applying the standard FMM accelerations in [39], we use FMM accelerations with local shifting. For example, with the gaps η_k solved from the shifted secular equation solution, it is preferred to use $\delta_{ik} - \eta_k$ in place of $d_i - \lambda_k$ in the computation of some entries of \mathbf{q}_k for accuracy purpose [16], [18], [70], [71] when d_i and λ_k are very close. Note that, with δ_{jk} in (3.44), (3.53) can be written as

$$\mathbf{q}_k = \left(\frac{\hat{v}_1}{\delta_{1k} - \eta_k} \quad \dots \quad \frac{\hat{v}_k}{-\eta_k} \quad \dots \quad \frac{\hat{v}_n}{\delta_{nk} - \eta_k} \right)^T. \quad (3.59)$$

When an entry of \mathbf{q}_k belongs to a near-field block of \hat{Q} , its representation in (3.59) is used. Otherwise, we use its form in (3.53). This preserves the far-field rank structure and makes the local shifting idea go through.

Thus, FMM accelerations with local shifting can be used to reliably represent and apply \hat{Q} or \hat{Q}^T . Note that

$$\hat{Q} = \text{diag}(\hat{\mathbf{v}}) \left(\frac{1}{d_i - \lambda_j} \right)_{n \times n} \text{diag}(\mathbf{b}), \quad (3.60)$$

so that \hat{Q} can be stored just via five vectors:

$$\hat{\mathbf{v}}, \mathbf{b}, \mathbf{d} \equiv (d_1 \ \dots \ d_n)^T, \ \boldsymbol{\lambda} \equiv (\lambda_1 \ \dots \ \lambda_n)^T, \ \boldsymbol{\eta} \equiv (\eta_1 \ \dots \ \eta_n)^T. \quad (3.61)$$

Here, we have the storage of one more vector $\boldsymbol{\eta}$ than that in [39]. This only slightly increase the storage, but the stability is significantly enhanced.

3.4.5 Overall eigendecomposition and structure of the eigenmatrix

The overall conquering framework is similar to [39], but with all the new stability strategies integrated. Also, the structure of the eigenmatrix Q is only briefly mentioned in [39] in a vague way. Here, we would like to give a precise description of Q resulting from the conquering process.

The conquering process is performed following the postordered traversal of the HSS tree \mathcal{T} of A , where at each node $i \in \mathcal{T}$, a local eigenproblem is solved. For a leaf node i , suppose \hat{D}_i is the (small) diagonal generator resulting from the overall dividing process. We just compute the dense eigenproblem $\hat{D}_i = Q_i \Lambda_i Q_i^T$. Then Q_i is a *local eigenmatrix* associated with i .

For a non-leaf node p with children i and j , the local eigenproblem is to find an eigendecomposition like in (3.11) based on (3.7) and (3.8). However, unlike (3.10) where a diagonal plus low-rank update eigendecomposition is computed, it is *necessary to reorder* the diagonal entries of $\text{diag}(\Lambda_i, \Lambda_j)$ in order to explore structures in the FMM accelerations that rely on the locations of the eigenvalues. Let P_p represent a sequence of permutations for deflation and for ordering the diagonal entries of $\text{diag}(\Lambda_i, \Lambda_j)$ from the smallest to the largest. The

need for P_p is not clearly mentioned in [39]. Also let the eigendecomposition of the *permuted* diagonal plus low-rank update problem be

$$P_p[\text{diag}(\Lambda_i, \Lambda_j) + \hat{Z}_p \hat{Z}_p^T] P_p^T = \hat{Q}_p \Lambda_p \hat{Q}_p^T, \quad (3.62)$$

where \hat{Z}_p is given in (3.9). Write D_p in (3.8) as \hat{D}_p since D_p is likely updated after the multilevel dividing process. Then we have the following eigendecomposition:

$$\hat{D}_p = Q_p \Lambda_p Q_p^T, \quad \text{with} \quad Q_p = \text{diag}(Q_i, Q_j) P_p^T \hat{Q}_p, \quad (3.63)$$

where Q_i and Q_j are eigenmatrices of \hat{D}_i and \hat{D}_j obtained in steps i and j , respectively. Then the conquering process proceeds similarly.

Here for convenience, we say Q_p is a *local eigenmatrix* and \hat{Q}_p is an *intermediate eigenmatrix*. The difference between the two is that a local eigenmatrix is an eigenmatrix of a local HSS block while the latter is an eigenmatrix of a diagonal plus low-rank update problem. A local eigenmatrix is formed by a sequence of intermediate eigenmatrices. Since $\hat{Q}_p \Lambda_p \hat{Q}_p^T$ in (3.62) is obtained by solving r consecutive rank-1 update eigenproblems, the intermediate eigenmatrix \hat{Q}_p is the product of r Cauchy-like matrices like in (3.57). Of course, when FMM accelerations and deflation are applied, the eigendecomposition is approximate.

Then the overall eigenmatrix Q is given in terms of all the intermediate eigenmatrices, organized with the aid of the tree \mathcal{T} . Its precise form is missing from [39]. Here, we give an accurate way to understand its structure as follows.

Lemma 3.4.1. *Assemble all the intermediate eigenmatrices and permutation matrices corresponding to the nodes at a level l of \mathcal{T} as*

$$Q^{(l)} = \text{diag}(\hat{Q}_i, i: \text{at level } l \text{ of } \mathcal{T}), \quad P^{(l)} = \text{diag}(P_i, i: \text{at level } l \text{ of } \mathcal{T}). \quad (3.64)$$

Then the final eigenmatrix Q has the form (illustrated in Figure 3.4)

$$Q = Q^{(L)} \prod_{l=l_{\max}-1}^0 (P^{(l)} Q^{(l)}), \quad (3.65)$$

where level l_{\max} is the leaf level of \mathcal{T} and $\text{root}(\mathcal{T})$ is at level 0. In addition, Q also corresponds to (3.63) with p set to be $\text{root}(\mathcal{T})$.

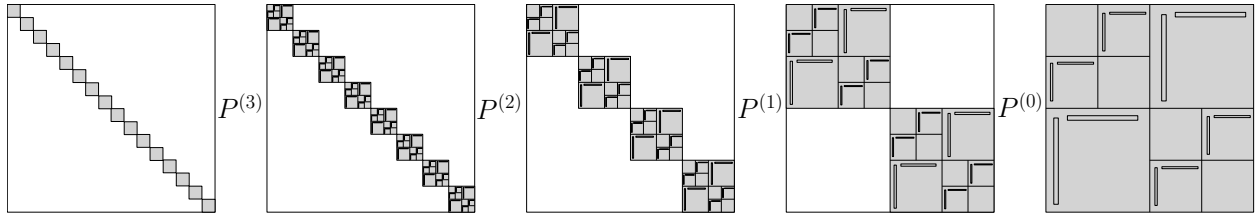
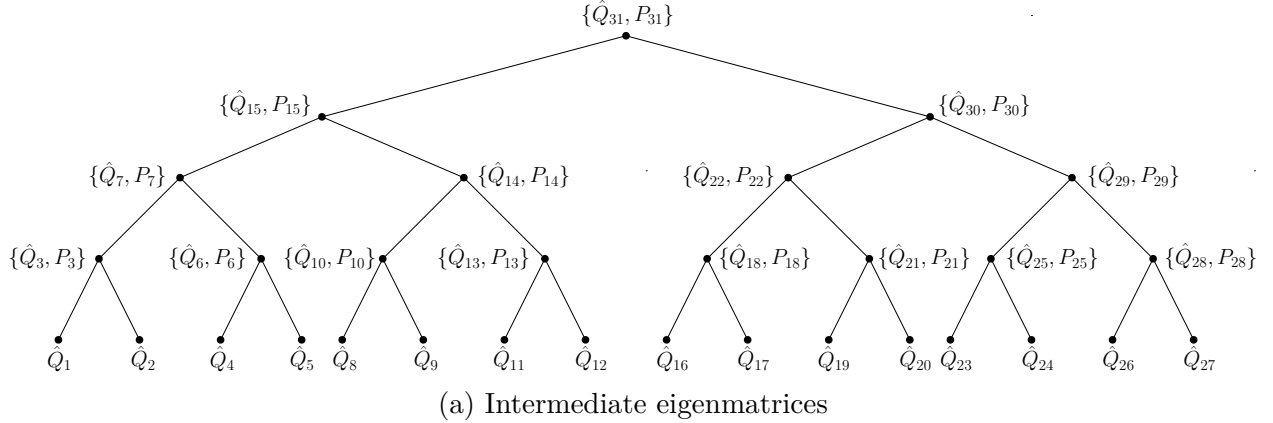


Figure 3.4. Structure eigenmatrix Q , where $l_{\max} = 4$

Thus, Q can be understood in terms of either (3.65) or the local eigenmatrices. Lemma 3.4.1 gives an efficient way to apply Q or Q^T to a vector, where the triangular FMM with local shifting is again used to multiply the intermediate eigenmatrices with vectors. Note that with a very similar procedure, a local eigenmatrix Q_i or its transpose can be conveniently applied to a vector. Such an application process is used to multiply the local eigenmatrices Q_i^T and Q_j^T to Z_p as in (3.9) to quickly form \hat{Z}_p used in (3.62).

The main algorithms used in SuperDC are shown in the supplementary materials. When A is given in terms of an HSS form with HSS rank r , the total complexity for computing the eigendecomposition (3.1) can be counted following [39, Section 3.1] and is $O(r^2 n \log^2 n)$. (There is an erratum for [39] in the flop count since r in equation (3.1) of [39, Section 3.1] should be r^2 .) Note that the use of all the new stability techniques here does not change the overall complexity. Every local eigenmatrix \hat{Q}_i is represented by a sequence of r Cauchy-like

matrices like in (3.57). Each such a Cauchy-like matrix is stored with the aid of five vectors like in (3.61). The storage for Q is then $O(rn \log n)$ and the cost to apply Q or Q^T to a vector is $O(rn \log n)$ as in [39].

3.5 Numerical experiments

In this section, we carry out a comprehensive test of the SuperDC eigensolver with different types of matrices and demonstrate its efficiency and accuracy. We compare SuperDC with the divide and conquer algorithms for band-symmetric matrices (BandDC, [16], [18], [63]), and the HSS bisection eigensolver based on structured LDL factorization (HSS-LDL, [76]), as well as the highly optimized Matlab `eig` function as a performance reference. In order for comparisons of larger sizes, we use BandDC and HSS-LDL to compute only the eigenvalues, which also gives them advantages over SuperDC. For the bisection-based HSS-LDL, we use $\tilde{\rho}(A) \equiv \sqrt{\|A\|_1 \|A\|_\infty} \geq \|A\|_2$ as an estimate of the spectral radius of A . We also show the necessity of our stability improvements in some test cases. We use the following accuracy measurements:

$$\begin{aligned} \gamma &= \max_{1 \leq k \leq n} \frac{\|A\mathbf{q}_k - \lambda_k \mathbf{q}_k\|_2}{\sqrt{n} \|A\|_2} && \text{(residual),} \\ \theta &= \max_{1 \leq k \leq n} \frac{\|Q^T \mathbf{q}_k - \mathbf{e}_k\|_2}{\sqrt{n}} && \text{(loss of orthogonality),} \\ \delta_{\text{rs}} &= \frac{\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^*\|_2}{\|\boldsymbol{\lambda}^*\|_2} && \text{(relative spectral error),} \\ \delta_{\text{rm}} &= \frac{\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^*\|_\infty}{\|\boldsymbol{\lambda}^*\|_\infty} && \text{(relative maximum error),} \\ \delta_{\text{mr}} &= \max_{1 \leq k \leq n} \frac{|\lambda_k^* - \lambda_k|}{|\lambda_k^*|} && \text{(maximum relative error),} \end{aligned}$$

where $\boldsymbol{\lambda}^* = (\lambda_1^* \ \dots \ \lambda_n^*)^T$ are eigenvalues from `eig` and are considered as the exact results. We also measure the *flops* (the total number of floating point arithmetic operations of the algorithm), the *storage* (total number of nonzeros to store the eigenmatrix), and the *timing* (total seconds elapsed when the call of the eigensolver routine is completed). The triangular

FMM routine is developed based on a code used in [32], and its accuracy during each call is set to reach full machine precision.

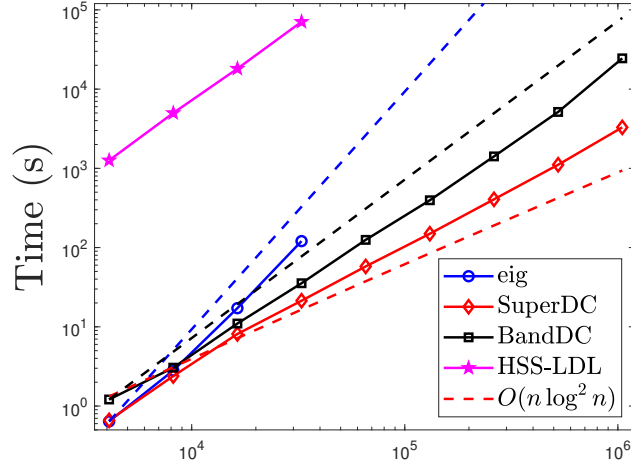
All the algorithms are implemented in pure Matlab. SuperDC is available at <https://www.math.purdue.edu/~xiaj>. The tests are performed with four 2.60GHz cores and 80GB memory on a node at a cluster of Purdue RCAC. The request of 80GB memory is just to accommodate the need of `eig` for larger matrices.

Example 1

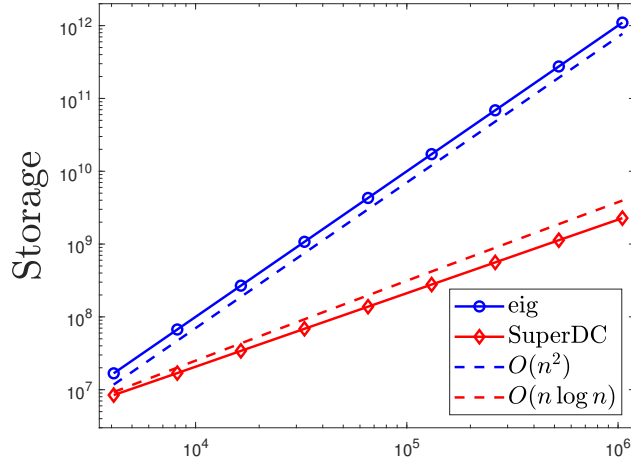
First, we consider a symmetric tridiagonal matrix A . For our SuperDC eigensolver, the HSS representation of A can be explicitly written out without any extra cost and its HSS rank is $r = 2$ [77]. (The HSS structure does not rely on the actual nonzero entries, which are 3 on the main diagonal and -1 on the first superdiagonal and subdiagonal. Other numbers such as random ones are also tested with similar performance observed.) As comparisons, we also apply BandDC, HSS-LDL and `eig` to A . The size of A in the test ranges from 4096 to 1048576. In the HSS form, the leaf-level diagonal block size is 2048. We use $\tau = 10^{-10}$ in the deflation criterion (Section 3.4.1).

The timing are reported in Figure 3.5(a). The storage for the eigenmatrix Q is given in Figure 3.5(b). The costs of SuperDC are given in Figure 3.5(c), in terms of the flops to get the eigendecomposition and the flops to apply Q to a vector. SuperDC achieves nearly linear complexity in all the aspects (timing, flops, and storage), while BandDC and HSS-LDL have a quadratic trend in timing, and `eig` has a cubic trend in timing, and an obvious quadratic storage (which is just n^2 for storing the dense Q). In fact, the flop count of SuperDC in Figure 3.5(c) shows a pattern even slightly better than $O(n \log^2 n)$. The timing trend of SuperDC is slightly deviated from the reference line $O(n \log^2 n)$, and we expect to optimize our implementations in future work.

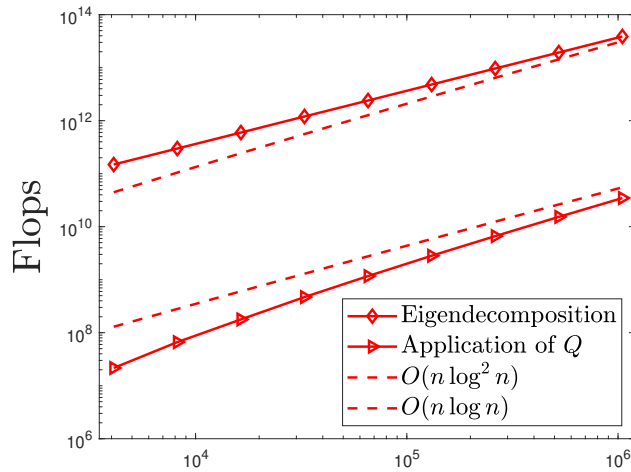
SuperDC is faster than BandDC and HSS-LDL for all the tested sizes, and its breakeven point with `eig` is around $n = 4096$. With $n = 32768$, SuperDC is already about 6 times faster than `eig` and takes only about 6% of the memory. Note that `eig` runs out of memory



(a) Eigendecomposition timing



(b) Storage



(c) Flops of SuperDC

Figure 3.5. *Example 1.* Timing and storage of SuperDC and eig and flops of SuperDC.

for larger n due to the dense eigenmatrix, while SuperDC takes much less memory and can reach much larger n .

The conquering stage is usually much more time-consuming than the dividing stage. For example, for $n = 65536$, SuperDC takes totally 57.8 seconds, where the dividing stage needs just 1.4 seconds and the conquering stage needs 56.4 seconds. This confirms that our strategy for minimizing $\text{colsize}(Z_p)$ is important, since it can reduce the number of rank-one updates to improve the efficiency of the conquering stage.

Table 3.1 shows the accuracy of SuperDC. The eigenvalues and eigendecompositions are computed accurately with numerically orthogonal eigenvectors.

Table 3.1. *Example 1.* Accuracy of SuperDC, where some errors (δ) are not reported since `eig` runs out of memory, and the cases $n \geq 262,144$ are not shown since it takes too long to compute γ and θ .

n	4,096	8,192	16,384	32,768	65,536	131,072
γ	$1.2e - 15$	$6.4e - 15$	$1.1e - 13$	$9.4e - 14$	$7.5e - 14$	$5.3e - 14$
θ	$1.8e - 14$	$2.9e - 14$	$3.8e - 14$	$5.5e - 14$	$8.6e - 14$	$1.2e - 13$
δ_{rs}	$2.6e - 16$	$4.6e - 16$	$1.3e - 13$	$9.4e - 14$		
δ_{rm}	$8.9e - 16$	$1.2e - 14$	$8.0e - 12$	$6.3e - 12$		
δ_{mr}	$9.7e - 16$	$1.2e - 14$	$8.0e - 12$	$6.3e - 12$		

Example 2

Next, we consider a symmetric matrix A which is sparse and nearly banded. That is, A has a banded form with half bandwidth 5 together with some nonzero entries away from the band. The HSS form for A can be explicitly written out with the method in [77] and has HSS rank 10. The nonzero entries away from the band are introduced by modifying some HSS generators. The main diagonal entries are set as 30 and the other entries in the band are set as -10 so that the upper bound for all $\|B_k\|$ in Proposition 3.3.1 is $\beta = 35.1 \gg 1$. As comparisons, we apply `eig` and HSS-LDL to A , and BandDC can also be conveniently adapted to A . The size n in the test ranges from 4096 to 1048576. In the HSS form, the

leaf-level diagonal block size is 2048. We use $\tau = 10^{-10}$ in the deflation criterion (Section 3.4.1).

The entries away from the band break the banded structure of A . The efficiency benefit of SuperDC becomes even more significant, as shown in Figure 3.6. The breakeven point of SuperDC and `eig` is around $n = 4096$. At $n = 32678$, SuperDC is already over 8 times faster than `eig` and takes only about 7% of the memory. At $n = 1048576$, SuperDC is over 12 times faster than BandDC (note that we do not compute the eigenmatrix with BandDC). Again, `eig` runs out of memory when n increases, but SuperDC works for much larger n and demonstrates nearly linear complexity.

We also compare our new dividing strategy (3.25) with the original one (3.3), and the results are reported in Table 3.2. The accuracies associated with the original dividing strategy deteriorate as the matrix size gets larger, while the accuracies associated with the new dividing strategy are well controlled by the tolerance. This can be explained as follows. At the leaf node, we need to use a (backward stable) dense method to compute a (numerical) eigendecomposition of the updated generator (see Lemma 3.2.1)

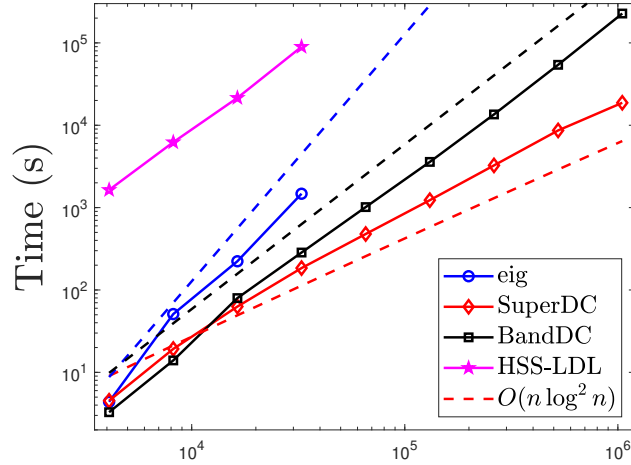
$$\tilde{D}_k = Q_k \Lambda_k Q_k^T + \Delta \tilde{D}_k, \quad \text{with} \quad \|\Delta \tilde{D}_k\|_2 = O(\|\tilde{D}_k\|_2 \cdot \epsilon_{\text{mach}}). \quad (3.66)$$

By Proposition 3.3.1 and 3.3.2, this backward error is

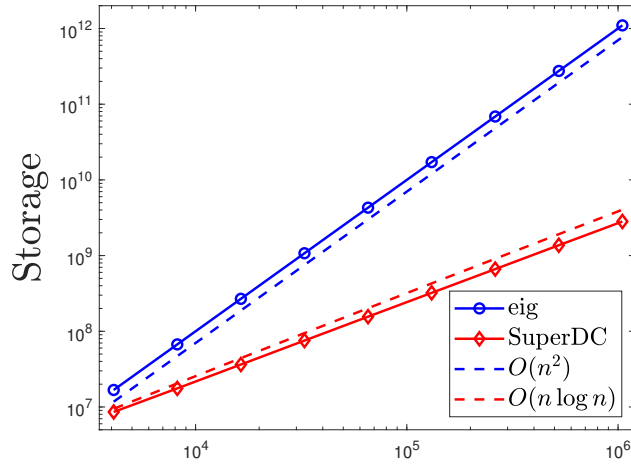
$$\|\Delta \tilde{D}_k\|_2 = \begin{cases} O(\beta^{2^{l_{\max}-1}} \cdot \epsilon_{\text{mach}}) & \text{with the original dividing (3.3)} \\ O(2^{l_{\max}-1} \beta \cdot \epsilon_{\text{mach}}) & \text{with the new dividing (3.24)} \end{cases}, \quad (3.67)$$

where l_{\max} is the total level of the HSS tree. Therefore, using the original dividing strategy will introduce large error since $\beta = 35.1$ in this case.

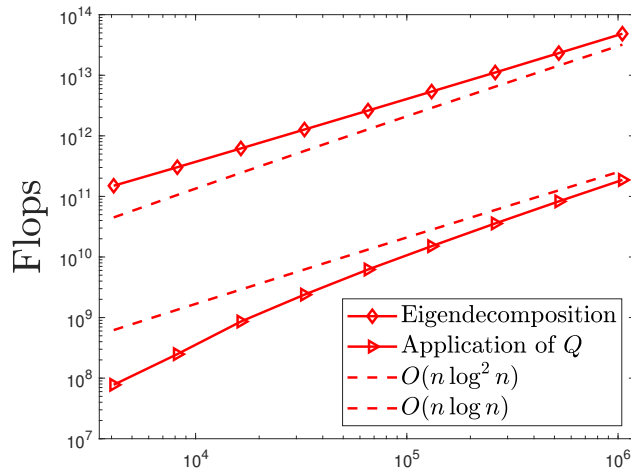
We also demonstrate the importance of our local shifting strategy by testing the eigensolver with triangular FMM accelerations applied to the standard secular equation. Due to cancellations, Matlab returns NaN (not-a-number) for all cases except $n = 4096$ and 8192. This confirms the risk of directly applying FMM accelerations to the standard secular equation like in [39].



(a) Eigendecomposition timing



(b) Storage



(c) Flops of SuperDC

Figure 3.6. *Example 2.* Timing and storage of SuperDC and eig and flops of SuperDC.

Table 3.2. *Example 2.* Accuracy of SuperDC, where some errors (δ) are not reported since `eig` runs out of memory, and the cases $n \geq 262,144$ are not shown since it takes too long to compute γ and θ .

n	4,096	8,192	16,384	32,768	65,536	131,072
new dividing strategy						
γ	$9.4e - 14$	$1.8e - 12$	$3.7e - 13$	$3.4e - 13$	$5.6e - 13$	$1.2e - 12$
θ	$1.9e - 13$	$4.4e - 13$	$5.6e - 13$	$1.1e - 12$	$1.4e - 12$	$2.0e - 12$
δ_{rs}	$1.6e - 14$	$3.2e - 12$	$5.3e - 13$	$2.6e - 13$		
δ_{rm}	$6.0e - 13$	$1.5e - 10$	$2.2e - 11$	$1.1e - 11$		
δ_{mr}	$1.6e - 12$	$2.9e - 10$	$3.2e - 11$	$2.2e - 11$		
original dividing strategy						
δ_{rs}	$7.0e - 13$	$8.3e - 13$	$5.4e - 11$	$7.4e - 10$		
δ_{rm}	$2.7e - 11$	$3.5e - 11$	$2.6e - 9$	$4.1e - 8$		
δ_{mr}	$2.8e - 11$	$5.0e - 10$	$2.5e - 8$	$3.9e - 7$		

Example 3

Next, we consider a dense symmetric Toeplitz matrix A with its first row

$$\boldsymbol{\xi} = (\xi_1 \ \dots \ \xi_n), \quad \text{given by}$$

$$\xi_1 = 2\alpha, \quad \xi_j = \frac{\sin(2\alpha(j-1)\pi)}{(j-1)\pi}, \quad j = 2, 3, \dots, n,$$

where $0 < \alpha < 1/2$. This is the so-called Prolate matrix that appears frequently in signal processing. It is known to be extremely ill-conditioned and has special spectral properties (see, e.g., [78]). In fact, the Prolate matrix has many small eigenvalues of magnitude $O(\epsilon_{\text{mach}})$. In this example, we set $\alpha = \frac{1}{4}$. It is known that any Toeplitz matrix can be converted into a Cauchy-like matrix \mathcal{C} which has small off-diagonal numerical ranks [39], [48], [49]. That is, $\mathcal{C} = \mathcal{F}A\mathcal{F}^*$, where \mathcal{F} is the normalized inverse DFT matrix. Then the eigendecomposition of A can be done via that of \mathcal{C} . An HSS approximation to \mathcal{C} may be quickly constructed based on randomized methods in [12], [13], [26], [30] and fast Toeplitz matrix-vector multiplications.

The cost is nearly linear in n . Here, we use a tolerance 10^{-10} in relevant compression steps, which is the same as the deflation tolerance τ . In the HSS form, the leaf-level diagonal block size is 2048. SuperDC and HSS-LDL are applied to the resulting HSS form and compared with `eig` applied to A . The size n ranges from 4096 to 65536. In Figure 3.7, the timing, storage, and flops are shown and they are consistent with the complexity estimates.

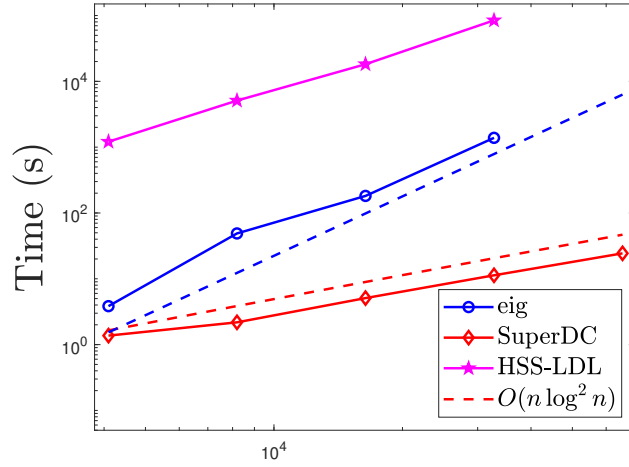
We also demonstrate the advantage of our flexible deflation strategy (Section 3.4.1). During the conquering stage, let ζ be the ratio of the total number of deflated intermediate eigenvalues to the total number of intermediate eigenvalues. The ratio ζ (see Table 3.3) is around 99% for all sizes, which indicates that most of the intermediate eigenvalues get deflated. This brings significant speed-ups to SuperDC. For example, at $n = 32,768$, `eig` takes 1385.5 seconds, while SuperDC only needs 11.3 seconds, which is a difference of about 123 times. Also, the memory saving is about 15 times.

The accuracy is reported in Table 3.3. SuperDC computes the eigendecomposition $A \approx Q\Lambda Q^T$ accurately, as shown by γ , θ , and δ_{rs} . We do not include the maximum relative error $\delta_{mr} = \max_{1 \leq k \leq n} \frac{|\lambda_k^* - \lambda_k|}{|\lambda_k^*|}$ here, since A is highly singular and has many eigenvalues of order $O(\epsilon_{mach})$. The magnitudes of these tiny eigenvalues are below the deflation tolerance so that the errors introduced by deflation may contaminate their accuracies. In addition, the backward errors $\Delta \tilde{D}_k$ (see (3.66)) introduced at leaf nodes may also contaminate their accuracies.

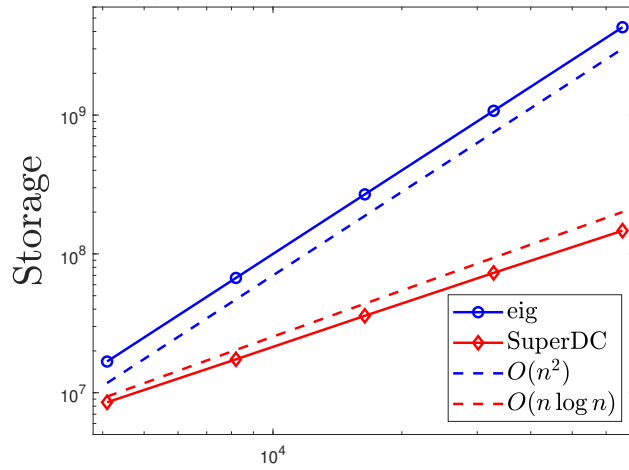
We also test SuperDC on random Toeplitz matrix. Since the associated Cauchy-like matrix \mathcal{C} has off-diagonal rank $r = O(\log n)$, the complexity of SuperDC is $O(n \log^4 n)$. In addition, we find in experiments that for random Toeplitz matrix, deflation rarely happens (i.e., $\zeta \approx 0$). So we have no speed-up from deflation. As a result, we expect that for random Toeplitz matrix, the breakeven point between SuperDC and `eig` is larger and the speed-up will not be as significant as in the Prolate matrix case.

Example 4

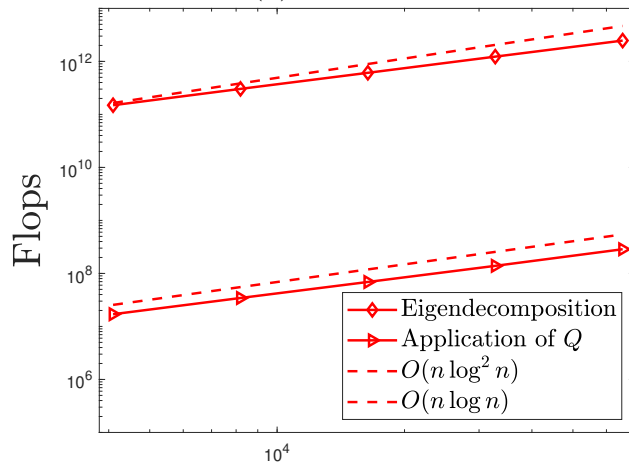
The last example is a discretized kernel matrix A in [57] which is the evaluation of the function $\sqrt{|s-t|}$ at the Chebyshev points $\cos(\frac{2i-1}{2n}\pi)$, $i = 1, 2, \dots, n$. The HSS construction



(a) Eigendecomposition timing



(b) Storage



(c) Flops of SuperDC

Figure 3.7. Example 3. Timing and storage of SuperDC and eig and flops of SuperDC.

Table 3.3. *Example 3.* Accuracy of SuperDC, where the error (δ) for $n = 65,536$ is not reported since `eig` runs out of memory. Note that for $n = 4,096$, A has a zero eigenvalue.

n	4,096	8,192	16,384	32,768	65,536
γ	$2.3e - 11$	$4.4e - 11$	$1.8e - 10$	$3.5e - 10$	$1.4e - 9$
θ	$2.2e - 15$	$8.6e - 15$	$6.0e - 15$	$4.2e - 15$	$3.0e - 15$
δ_{rs}	$5.5e - 12$	$1.4e - 11$	$4.5e - 10$	$9.3e - 10$	
δ_{rm}	$1.1e - 10$	$7.3e - 10$	$2.2e - 8$	$6.1e - 8$	
ζ	98.8%	99.2%	99.5%	99.6%	99.4%

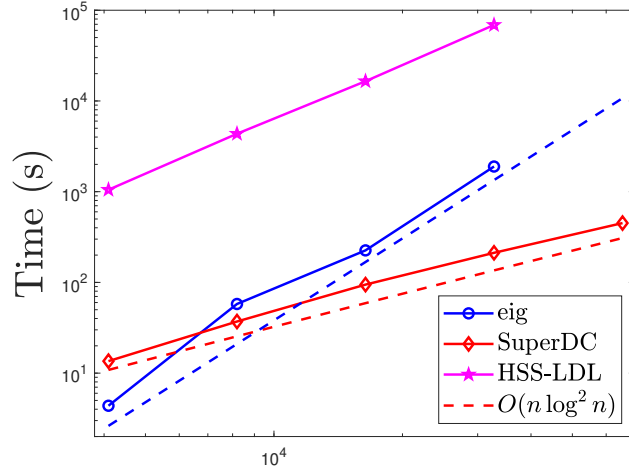
may be based on direct off-diagonal compression or efficient analytical methods like in [36]. We use an existing routine based on the former one for simplicity. To show the flexibility of accuracy controls, we aim for moderate accuracy in this test by using a compression tolerance 10^{-6} in the HSS construction, which is same as the deflation tolerance τ .

For this example, we still set the HSS leaf-level diagonal block size to be 2048. We can observe similar complexity results as in the previous examples, see Figure 3.8. With the larger tolerance than in the previous examples, we still achieve reasonable eigenvalue errors and residuals with numerically orthogonal eigenvectors, see Table 3.4.

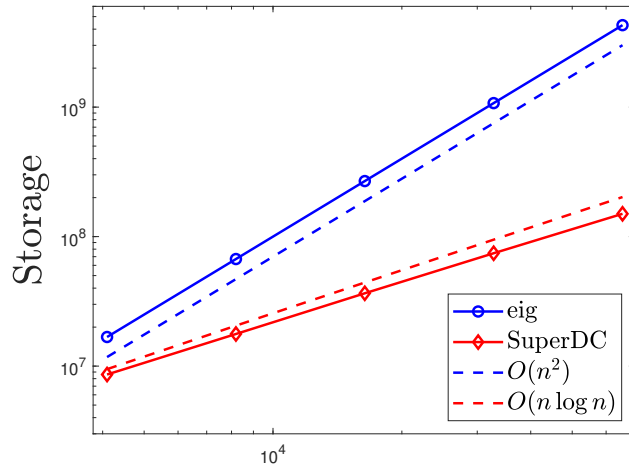
Table 3.4. *Example 4.* Accuracy of SuperDC, where the error (δ) for $n = 65,536$ is not reported since `eig` runs out of memory.

n	4,096	8,192	16,384	32,768	65,536
γ	$7.4e - 9$	$2.7e - 9$	$2.2e - 9$	$1.6e - 9$	$1.1e - 9$
θ	$1.4e - 13$	$2.6e - 13$	$2.5e - 13$	$2.4e - 13$	$3.8e - 13$
δ_{rs}	$3.8e - 8$	$5.5e - 8$	$5.7e - 8$	$8.7e - 8$	
δ_{rm}	$2.9e - 8$	$3.2e - 8$	$2.8e - 8$	$5.6e - 8$	
δ_{mr}	$1.2e - 4$	$4.2e - 4$	$4.7e - 4$	$5.8e - 4$	

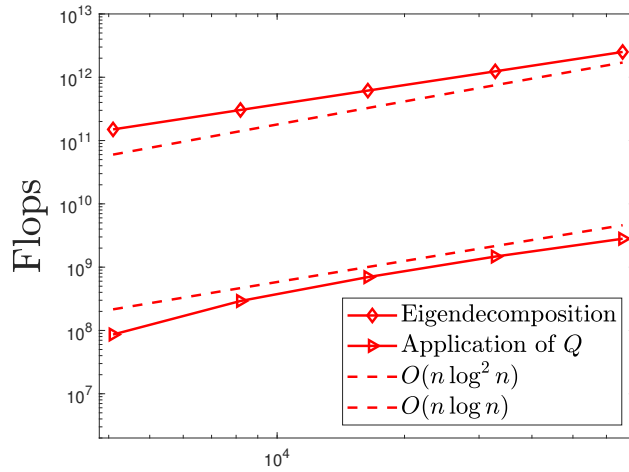
In the previous Example 2, it is shown that local shifting helps avoid cancellations so that FMM accelerations can be applied reliably. In fact, even if there is no cancellation in the original secular equation solution, our local shifting strategy (for triangular FMM-



(a) Eigendecomposition timing



(b) Storage



(c) Flops of SuperDC

Figure 3.8. *Example 4.* Timing and storage of SuperDC and eig and flops of SuperDC.

accelerated solution of the shifted secular equation) can also significantly benefit the rate of convergence of the roots. To illustrate this, we perform the following count. Suppose the low-rank update associated with the root node of the HSS tree \mathcal{T} has size r , so that r secular equations are solved when the conquering stage proceeds to the root. When solving the j th secular equation, let μ_j be the percentage of eigenvalues that have *not* converged after 5 Newton’s iterations. Let $\mu = \max_{1 \leq j \leq r} \mu_j$. Table 3.5 reports this maximum percentage μ with varying n . With local shifting, a vast majority of those eigenvalues (about 99% or more) converges within 5 iterations. This is significantly better than the case without local shifting (i.e., when the standard secular equation is solved with FMM accelerations). We also observe similar or even smaller μ ’s in other examples.

Table 3.5. *Example 4.* Maximum percentage (μ) of eigenvalues not converged within 5 iterations for solving the r secular equations associated with $\text{root}(\mathcal{T})$.

n	4,096	8,192	16,384	32,768	65,536
With local shifting	1.00%	0.88%	0.34%	0.38%	0.33%
Without local shifting	62.5%	57.6%	57.2%	58.5%	57.7%

Following Propositions 3.3.1 and 3.3.2, we also show the norm growth of the B, D generators after the dividing stage. For the initial B, D generators of the original HSS form, let \tilde{B}, \tilde{D} denote the updated generators after the entire dividing stage is finished. Let

$$\rho_B = \max_{i < \text{root}(\mathcal{T})} \|B_i\|_2, \quad \rho_D = \max_{i: \text{leaf}} \|D_i\|_2, \quad \rho_{\tilde{B}} = \max_{i < \text{root}(\mathcal{T})} \|\tilde{B}_i\|_2, \quad \rho_{\tilde{D}} = \max_{i: \text{leaf}} \|\tilde{D}_i\|_2.$$

In order for better demonstration of the norm growth after multilevel dividing, we set the leaf-level diagonal block size to be 256 here to have more levels. For each n , Table 3.6 shows the number of levels in the HSS approximation. When n increases, the HSS tree \mathcal{T} grows deeper. Table 3.6 shows that $\|A\|_2$ and $\rho(B)$ grow roughly linearly with n . However, $\rho_{\tilde{B}}$ and $\rho_{\tilde{D}}$ grow exponentially with the original dividing stage in [39], as predicted by Proposition 3.3.1. This poses a stability risk. When n grows beyond a certain size, overflow happens. (Note that $\rho_{\tilde{D}}$ has a larger magnitude than $\rho_{\tilde{B}}$, which is consistent with Proposition 3.3.1. In addition, since the backward error $\|\Delta\tilde{D}_k\|_2$ in (3.66) is proportional to $\rho_{\tilde{D}}$, the accuracy

of the eigendecomposition will deteriorate.) In contrast, the growth of $\rho_{\tilde{D}}$ and $\rho_{\tilde{B}}$ with our new dividing strategy is much slower and roughly follows the growth pattern of $\rho(B)$, as predicted by Proposition 3.3.2. Accordingly, our algorithm can handle much larger n much more reliably.

Table 3.6. *Example 4.* Norms of the D, B generators after the dividing stage, where ∞ means overflow.

n		4,096	8,192	16,384	32,768	65,536
Number of levels		5	6	7	8	9
$\ A\ _2$		$3.4e3$	$6.8e3$	$1.4e4$	$2.7e4$	$5.4e4$
Initial	ρ_B	$2.3e3$	$4.6e3$	$9.2e3$	$1.8e4$	$3.7e4$
	ρ_D	$6.1e1$	$4.3e1$	$3.1e1$	$2.2e1$	$1.5e1$
After the original dividing strategy	$\rho_{\tilde{B}}$	$5.5e24$	$9.9e53$	$2.1e117$	$4.2e253$	∞
	$\rho_{\tilde{D}}$	$3.0e49$	$9.9e107$	$4.5e234$	∞	∞
After the new dividing strategy	$\rho_{\tilde{B}}$	$2.3e3$	$4.6e3$	$9.2e3$	$2.1e4$	$5.5e4$
	$\rho_{\tilde{D}}$	$4.7e3$	$1.2e4$	$3.4e4$	$8.6e4$	$2.4e5$

3.6 Pseudocodes and algorithms

In this section, we present the pseudocodes and procedures that can help understand the major algorithms in SuperDC.

- Algorithm 2: the HSS dividing stage.
- Algorithm 3: solving the secular equation for the eigenvalues with triangular FMM accelerations and local shifting.
- Algorithm 4: the conquering stage for producing the eigendecomposition.
- Algorithm 5: application of the a local eigenmatrix Q_i or its transpose to a vector. This is used in Algorithm 4 and also can be used to apply the global eigenmatrix Q or its transpose to a vector when $i = \text{root}(\mathcal{T})$.

For notational convenience, we use r to represent the column sizes of all Z_i matrices in the pseudocodes. \mathcal{T}_i is also used to denote the subtree of \mathcal{T} rooted at node $i \in \mathcal{T}$. $Z(:, j)$ means the j^{th} column of Z .

The following utility routines are used in the algorithms. To save space, we are not showing pseudocodes for these routines.

- **updhss**(D_i, U_i, H): for an HSS block D_i corresponding to the subtree \mathcal{T}_i , update its D, B generators to get those of $D_i - U_i H U_i^T$ using Lemma 3.2.1.
- **trifmm**($\mathbf{d}, \mathbf{x}, \mathbf{y}, \mathbf{w}, \kappa$): compute a matrix-vector product $K\mathbf{w}$ with the triangular FMM and local shifting as in Sections 3.4.2 and 3.4.3, where $K = (\kappa(d_i, x_j))_{d_i \in \mathbf{d}, x_j \in \mathbf{x}}$ is a kernel matrix and \mathbf{y} is the gap vector (for accurately evaluating $\mathbf{x} - \mathbf{d}$). Note that the triangular FMM is used to multiply the lower triangular part of K with \mathbf{w} and the strictly upper triangular part of K with \mathbf{w} and the final result is the sum of the two products.
- **mnewton**($\boldsymbol{\psi}, \boldsymbol{\phi}, \boldsymbol{\psi}', \boldsymbol{\phi}'$): use the modified Newton's method to compute corrections to the current approximate gap as in (3.47), where $\boldsymbol{\psi}, \boldsymbol{\phi}, \boldsymbol{\psi}', \boldsymbol{\phi}'$ look like (3.37) and (3.38).
- **iniguess**(\mathbf{d}, \mathbf{w}): compute the initial guess as in [73] for the solution of the secular equation (3.13).
- **deflate**($\mathbf{d}, \mathbf{v}, \tau$): apply deflation with the criterion in Section 3.4.1.

3.7 Some improvements on implementations of SuperDC

In this section, we describe several improvements on the implementations of SuperDC. When properly implemented (in MATLAB), these techniques can speed up the SuperDC routine by a factor over 2, as compared with Section 3.5.

3.7.1 Rank-revealing factorization in dividing stage

As discussed in Section 3.5, the conquering stage is much more time-consuming than the dividing stage, since it needs to solve $\text{colsize}(Z_p)$ secular equations at each non-leaf node

Algorithm 2 SuperDC dividing stage

```

1: procedure divide( $\{D_i\}_{i \in \mathcal{T}}, \{U_i\}_{i \in \mathcal{T}}, \{R_i\}_{i \in \mathcal{T}}, \{B_i\}_{i \in \mathcal{T}}$ )
2:   for node  $i = \text{root}(\mathcal{T}), \dots, 1$  do  $\triangleright$  Dividing  $D_i$  in a top-down traversal
3:     if  $i$  is a non-leaf node then
4:       if  $\text{colsize}(B_{c_1}) \leq \text{rowsize}(B_{c_1})$  then  $\triangleright c_1, c_2$ : children of  $i$ 
5:          $D_{c_1} \leftarrow \text{updhss}(D_{c_1}, U_{c_1}, \frac{1}{\|B_{c_1}\|_2} B_{c_1} B_{c_1}^T)$   $\triangleright$  Update generators of  $D_{c_1}$ 
6:           to get those of  $D_{c_1} - \frac{1}{\|B_{c_1}\|_2} U_{c_1} B_{c_1} B_{c_1}^T U_{c_1}^T$  like in Lemma 3.2.1
7:          $D_{c_2} \leftarrow \text{updhss}(D_{c_2}, U_{c_2}, \|B_{c_1}\|_2 I)$   $\triangleright$  Update generators of  $D_{c_2}$ 
8:           to get those of  $D_{c_2} - \|B_{c_1}\|_2 U_{c_2} U_{c_2}^T$  like in Lemma 3.2.1
9:       else
10:         $D_{c_1} \leftarrow \text{updhss}(D_{c_1}, U_{c_1}, \|B_{c_1}\|_2 I)$   $\triangleright$  Update generators of  $D_{c_1}$ 
11:          to get those of  $D_{c_1} - \|B_{c_1}\|_2 U_{c_1} U_{c_1}^T$  like in Lemma 3.2.1
12:         $D_{c_2} \leftarrow \text{updhss}(D_{c_2}, U_{c_2}, \frac{1}{\|B_{c_1}\|_2} B_{c_1}^T B_{c_1})$   $\triangleright$  Update generators of  $D_{c_2}$ 
13:          to get those of  $D_{c_2} - \frac{1}{\|B_{c_1}\|_2} U_{c_2} B_{c_1}^T B_{c_1} U_{c_2}^T$  like in Lemma 3.2.1
14:       end if
15:     end if
16:   end for
17:   for node  $i = 1, \dots, \text{root}(\mathcal{T})$  do  $\triangleright$  Form  $Z_i$  in a bottom-up traversal
18:     if  $i$  is a non-leaf node then
19:       if  $\text{colsize}(B_{c_1}) \leq \text{rowsize}(B_{c_1})$  then  $\triangleright c_1, c_2$ : children of  $i$ 
20:          $Z_i \leftarrow \begin{pmatrix} \frac{1}{\sqrt{\|B_{c_1}\|_2}} U_{c_1} B_{c_1} \\ \sqrt{\|B_{c_1}\|_2} U_{c_2} \end{pmatrix}$   $\triangleright$  Local update  $Z$  matrix like in (3.25)
21:       else
22:          $Z_i \leftarrow \begin{pmatrix} \sqrt{\|B_{c_1}\|_2} U_{c_2} \\ \frac{1}{\sqrt{\|B_{c_1}\|_2}} U_{c_2} B_{c_1}^T \end{pmatrix}$   $\triangleright$  Local update  $Z$  matrix like in (3.28)
23:       end if
24:     if  $i \neq \text{root}(\mathcal{T})$  then
25:        $U_i \leftarrow \begin{pmatrix} U_{c_1} R_{c_1} \\ U_{c_2} R_{c_2} \end{pmatrix}$   $\triangleright$  Assemble  $U_i$  for parent node of  $i$ 
26:     end if
27:   end for
28:   return updated generators  $\{D_i\}_{i \in \mathcal{T}}, \{B_i\}_{i \in \mathcal{T}}, \{Z_i\}_{i \in \mathcal{T}}$ 
29: end procedure

```

Algorithm 3 Secular equation solution for eigenvalues (of $\text{diag}(\mathbf{d}) + \mathbf{v}\mathbf{v}^T$)

```

1: procedure secular( $\mathbf{d}, \mathbf{v}$ )
     $\triangleright$  Eigenvalue solution via the solution of the shifted secular equation (3.43)
2:    $\mathbf{w} \leftarrow \mathbf{v} \odot \mathbf{v}$ 
3:    $\mathbf{x}^{(0)} \leftarrow \text{iniguess}(\mathbf{d}, \mathbf{w})$   $\triangleright$  Computation of the initial guess as in [73]
4:    $\mathbf{y}^{(0)} \leftarrow \mathbf{x}^{(0)} - \mathbf{d}$ 
5:   for  $j = 0, 1, \dots$  do
6:      $[\boldsymbol{\psi}, \boldsymbol{\phi}] \leftarrow \text{trifmm}(\mathbf{d}, \mathbf{x}^{(j)}, \mathbf{y}^{(j)}, \mathbf{w}, \frac{1}{s-t})$   $\triangleright$  Computation of  $\boldsymbol{\psi}, \boldsymbol{\phi}$  in (3.37)
7:      $[\boldsymbol{\psi}', \boldsymbol{\phi}'] \leftarrow \text{trifmm}(\mathbf{d}, \mathbf{x}^{(j)}, \mathbf{y}^{(j)}, \mathbf{w}, \frac{1}{(s-t)^2})$   $\triangleright$  Computation of  $\boldsymbol{\psi}', \boldsymbol{\phi}'$  in (3.38)
8:      $\mathbf{f} \leftarrow \mathbf{e} + \boldsymbol{\psi} + \boldsymbol{\phi}$ 
9:     if  $|\mathbf{f}| < cn(\mathbf{e} + |\boldsymbol{\psi}| + |\boldsymbol{\phi}|)\epsilon$  then  $\triangleright$  Stopping criterion
10:      break
11:    end if
12:     $\Delta \mathbf{x}^{(j)} \leftarrow \text{mnewton}(\boldsymbol{\psi}, \boldsymbol{\phi}, \boldsymbol{\psi}', \boldsymbol{\phi}')$ 
     $\triangleright$  Computation of root update with modified Newton's method
13:     $\mathbf{y}^{(j+1)} \leftarrow \mathbf{y}^{(j)} + \Delta \mathbf{x}^{(j)}$   $\triangleright$  Updated gap approximation as in (3.47)
14:     $\mathbf{x}^{(j+1)} \leftarrow \mathbf{y}^{(j+1)} + \mathbf{d}$   $\triangleright$  Updated eigenvalue approximation
15:  end for
16:   $\boldsymbol{\lambda} \leftarrow \mathbf{x}^{(j)}, \boldsymbol{\eta} \leftarrow \mathbf{y}^{(j)}$   $\triangleright$  Eigenvalue and gap upon convergence
17:  return  $\boldsymbol{\lambda}, \boldsymbol{\eta}$ 
18: end procedure

```

Algorithm 4 SuperDC conquering stage

```

1: procedure conquer( $\{D_i\}_{i \in \mathcal{T}}, \{U_i\}_{i \in \mathcal{T}}, \{R_i\}_{i \in \mathcal{T}}, \{B_i\}_{i \in \mathcal{T}}, \{Z_i\}_{i \in \mathcal{T}}, \tau$ )
     $\triangleright$  The  $D_i, B_i$  generators have been updated in the dividing stage
2:   for node  $i = 1, \dots, \text{root}(\mathcal{T})$  do  $\triangleright$  Conquering in a postordered traversal
3:     if  $i$  is a leaf node then  $\triangleright$  Leaf-level eigendecomposition
4:        $(\boldsymbol{\lambda}_i, \hat{Q}_i) \leftarrow \text{eig}(D_i)$   $\triangleright$  Via Matlab eig function
5:     else
6:        $\begin{pmatrix} Z_{i,1} \\ Z_{i,2} \end{pmatrix} \leftarrow Z_i$   $\triangleright$  Partitioning following the sizes of  $D_{c_1}$  and  $D_{c_2}$ 
7:        $Z_{i,1} \leftarrow \text{superdcmv}(Q_{c_1}, Z_{i,1}, 1)$   $\triangleright Q_{c_1}^T Z_{i,1}$ 
8:        $Z_{i,2} \leftarrow \text{superdcmv}(Q_{c_2}, Z_{i,2}, 1)$   $\triangleright Q_{c_2}^T Z_{i,2}$ 
9:        $Z_i \leftarrow \begin{pmatrix} Z_{i,1} \\ Z_{i,2} \end{pmatrix}$   $\triangleright \hat{Z}_i$  like in (3.9)
10:       $(\boldsymbol{\lambda}_i^{(0)}, P_i) \leftarrow \text{sort}(\boldsymbol{\lambda}_{c_1}, \boldsymbol{\lambda}_{c_2})$   $\triangleright$  Ordering of all the diagonal entries
        of  $\boldsymbol{\lambda}_{c_1}, \boldsymbol{\lambda}_{c_2}$  together, with  $P_i$  the permutation matrix
11:      for  $j = 1, 2, \dots, r$  do  $\triangleright r = \text{colsize}(Z_i)$ 
12:         $(\mathbf{d}_i^{(j)}, Z_i(:, j)) \leftarrow \text{deflate}(\boldsymbol{\lambda}_i^{(j-1)}, Z_i(:, j), \tau)$   $\triangleright$  Deflation (Section 3.4.1)
13:         $(\boldsymbol{\lambda}_i^{(j)}, \boldsymbol{\eta}_i^{(j)}) \leftarrow \text{secular}(\mathbf{d}_i^{(j)}, Z_i(:, j))$   $\triangleright$  Secular equation solution
14:         $\mathbf{v}_1 \leftarrow \text{trifmm}(\mathbf{d}_i^{(j)}, \boldsymbol{\lambda}_i^{(j)}, \boldsymbol{\eta}_i^{(j)}, \mathbf{e}, \log |s - t|)$   $\triangleright G_1 \mathbf{e}$  as needed in (3.56)
15:         $\mathbf{v}_2 \leftarrow \text{trifmm}(\mathbf{d}_i^{(j)}, \mathbf{d}_i^{(j)}, \mathbf{0}, \mathbf{e}, \log |s - t|)$   $\triangleright G_2 \mathbf{e}$  as needed in (3.56)
16:         $\hat{\mathbf{v}}_i^{(j)} \leftarrow \exp(\frac{\mathbf{v}_1 - \mathbf{v}_2}{2})$   $\triangleright$  Löwner's formula for  $\hat{\mathbf{v}}$  as in (3.54)–(3.56)
17:         $\mathbf{b}_i^{(j)} \leftarrow (\text{trifmm}(\mathbf{d}_i^{(j)}, \boldsymbol{\lambda}_i^{(j)}, \boldsymbol{\eta}_i^{(j)}, \hat{\mathbf{v}}_i^{(j)} \odot \hat{\mathbf{v}}_i^{(j)}, \frac{1}{(s-t)^2}))^{-1/2}$ 
         $\triangleright$  Normalization factor as in (3.58)
18:         $\hat{Q}_i^{(j)} \leftarrow \{\hat{\mathbf{v}}_i^{(j)}, \mathbf{b}_i^{(j)}, \mathbf{d}_i^{(j)}, \boldsymbol{\lambda}_i^{(j)}, \boldsymbol{\eta}_i^{(j)}\}$   $\triangleright$  Cauchy-like structured
        representation of the local eigenmatrix as in (3.57)
19:        for  $k = j + 1, j + 2, \dots, r$  do  $\triangleright$  Multiplication of  $\hat{Q}_i^{(j)}$ 
        to the remaining columns of  $Z_i$  via (3.60)
20:           $Z_i(:, k) \leftarrow \hat{\mathbf{v}}_i^{(j)} \odot Z_i(:, k)$ 
21:           $Z_i(:, k) \leftarrow \text{trifmm}(\mathbf{d}_i^{(j)}, \boldsymbol{\lambda}_i^{(j)}, \boldsymbol{\eta}_i^{(j)}, Z_i(:, k), \frac{1}{s-t})$ 
22:           $Z_i(:, k) \leftarrow \mathbf{b}_i^{(j)} \odot Z_i(:, k)$ 
23:        end for
24:      end for
25:       $\boldsymbol{\lambda}_i \leftarrow \boldsymbol{\lambda}_i^{(r)}$   $\triangleright$  Local eigenvalues associated with node  $i$ 
26:    end if
27:  end for
28:   $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}_{\text{root}(\mathcal{T})}, Q \leftarrow \{\{\hat{Q}_i^{(j)}\}_{j=1}^r, P_i\}_{i \in \mathcal{T}}$   $\triangleright$  Final eigenvalues
    and eigenmatrix  $Q$  in (3.65), with  $\hat{Q}_i$  in (3.64) given by  $\prod_{j=1}^r \hat{Q}_i^{(j)}$ 
29:  return  $\boldsymbol{\lambda}, Q$ 
30: end procedure

```

Algorithm 5 SuperDC eigenmatrix-vector multiplication

```

1: procedure superdcmv( $Q_i, \mathbf{x}, \text{transpose}$ )  $\triangleright$  Application of a local eigenmatrix  $Q_i$ 
   or its transpose to a vector  $\mathbf{x}$ , depending on whether ‘transpose’ is 0 or 1
2:    $i_1 \leftarrow$  smallest descendant of  $i$ 
3:   if transpose = 0 then  $\triangleright \mathbf{y} = Q_i \mathbf{x}$ 
4:      $\mathbf{y}_i \leftarrow \mathbf{x}$ 
5:     for  $k = i, i - 1, \dots, i_1$  do  $\triangleright$  Reverse postordered traversal of  $\mathcal{T}_i$ 
6:       if  $k$  is leaf then
7:          $\mathbf{y}_k \leftarrow Q_k \mathbf{y}_k$   $\triangleright$  Dense  $Q_k$  at the leaf level
8:       else
9:         for  $j = r, r - 1, \dots, 1$  do  $\triangleright$  Multiplication of  $\hat{Q}_k^{(j)}$  via (3.60)
10:           $\mathbf{y}_k \leftarrow \mathbf{b}_k^{(j)} \odot \mathbf{y}_k$ 
11:           $\mathbf{y}_k \leftarrow \text{trifmm}(\mathbf{d}_k^{(j)}, \boldsymbol{\lambda}_k^{(j)}, \boldsymbol{\eta}_k^{(j)}, \mathbf{y}_k, \frac{1}{s-t})$ 
12:           $\mathbf{y}_k \leftarrow \hat{\mathbf{v}}_k^{(j)} \odot \mathbf{y}_k$ 
13:        end for
14:         $\mathbf{y}_k \leftarrow P_k^T \mathbf{y}_k$   $\triangleright$  Permutation like in (3.63)
15:         $\begin{pmatrix} \mathbf{y}_{c_1} \\ \mathbf{y}_{c_2} \end{pmatrix} \leftarrow \mathbf{y}_k$   $\triangleright$  Partitioning following the sizes of  $Q_{c_1}, Q_{c_2}$ ,
   with  $c_1, c_2$  the children of  $k$ 
16:      end if
17:    end for
18:  else  $\triangleright \mathbf{y} = Q_i^T \mathbf{x}$ 
19:    Partition  $\mathbf{x}$  into  $\mathbf{x}_k$  pieces following the leaf-level  $Q_k$  sizes
20:    for  $k = i_1, i_1 + 1, \dots, i$  do  $\triangleright$  Postordered traversal of  $\mathcal{T}_i$ 
21:      if  $k$  is leaf then
22:         $\mathbf{y}_k \leftarrow Q_k^T \mathbf{x}_k$   $\triangleright$  Dense  $Q_k$  at the leaf level
23:      else
24:         $\mathbf{y}_k \leftarrow \begin{pmatrix} \mathbf{y}_{c_1} \\ \mathbf{y}_{c_2} \end{pmatrix}$   $\triangleright c_1, c_2$ : children of  $k$ 
25:         $\mathbf{y}_k \leftarrow P_k \mathbf{y}_k$   $\triangleright$  Permutation like in (3.63)
26:        for  $j = 1, 2, \dots, r$  do  $\triangleright$  Multiplication of  $(\hat{Q}_k^{(j)})^T$  via (3.60)
27:           $\mathbf{y}_k \leftarrow \hat{\mathbf{v}}_k^{(j)} \odot \mathbf{y}_k$ 
28:           $\mathbf{y}_k \leftarrow -\text{trifmm}(\boldsymbol{\lambda}_k^{(j)}, \mathbf{d}_k^{(j)}, \boldsymbol{\eta}_k^{(j)}, \mathbf{y}_k, \frac{1}{s-t})$   $\triangleright$  The negative sign
   and the switch of  $\boldsymbol{\lambda}_k^{(j)}$  and  $\mathbf{d}_k^{(j)}$  are because of the transpose
29:           $\mathbf{y}_k \leftarrow \mathbf{b}_k^{(j)} \odot \mathbf{y}_k$ 
30:        end for
31:      end if
32:    end for
33:  end if
34:  return  $\mathbf{y}$ 
35: end procedure

```

$p \in \mathcal{T}$. In order to make SuperDC more efficient, in Section 3.3, we use an adaptive dividing strategy to minimize $\text{colsize}(Z_p) = \min(\text{rowsize}(B_i), \text{colsize}(B_i))$.

However, in some cases, the matrix B_i may be rank-deficient. For example, in the HSS construction (see, e.g., [77]) of a symmetric banded matrix with bandwidth w , the size of B_i is $2w \times 2w$ while its rank is just w . In this case, we can take advantage of the rank-deficiency by computing a rank-revealing factorization

$$B_i = X_i Y_i^T, \quad \text{such that} \quad (3.68)$$

$$\|X_i\|_2 = \|Y_i\|_2 = \sqrt{\|B_i\|_2}, \quad \text{colsize}(X_i) = \text{rank}(B_i). \quad (3.69)$$

Then we can divide D_p as

$$D_p = \begin{pmatrix} D_i - U_i X_i X_i^T U_i^T & \\ & D_j - U_j Y_i Y_i^T U_j^T \end{pmatrix} + \begin{pmatrix} U_i X_i \\ U_j Y_i \end{pmatrix} \begin{pmatrix} X_i^T U_i^T & Y_i^T U_j^T \end{pmatrix}, \quad (3.70)$$

so that the low-rank update Z_p now becomes

$$Z_p = \begin{pmatrix} U_i X_i \\ U_j Y_i \end{pmatrix}, \quad \text{colsize}(Z_p) = \text{rank}(B_i) < \min(\text{rowsize}(B_i), \text{colsize}(B_i)).$$

Because $\|X_i\|_2 = \|Y_i\|_2 = \sqrt{\|B_i\|_2}$, we can analogously show that the dividing strategy (3.70) satisfies Proposition 3.3.2.

3.7.2 Precomputations in triangular FMM

When solving the shifted secular equation (3.43) via modified Newton's method, we need to use triangular FMM (with local shifting) to compute the matrix-vector products (3.39) at each iteration to update y_k 's like (3.47), where $y_k = x_k - d_k$ are the approximations to the exact differences $\eta_k = \lambda_k - d_k$.

For this purpose, a straightforward way is just to repeat the triangular FMM algorithm at each individual iteration. That is, at each iteration, (i) we first partition the interval covering $\mathbf{x} = \{x_k\}_{k=1}^n$ and $\mathbf{d} = \{d_j\}_{j=1}^n$; (ii) then compute the corresponding low-rank factors of the

far-field blocks like (2.14), (2.15), (2.17), and the translation matrices like (2.53),(2.60), as well as the near-field blocks; (iii) then multiply via Algorithm 1 the triangular FMM matrices with the vector $\mathbf{w} = (v_1^2 \ \dots \ v_n^2)^T$ to get $\boldsymbol{\psi}$, $\boldsymbol{\phi}$, $\boldsymbol{\psi}'$, and $\boldsymbol{\phi}'$.

However, via a more careful design, most of the computations in the triangular FMM can be done in a precomputation step, so that when the approximations y_k 's in (3.49) are updated, only a small amount of extra work is needed to get $\boldsymbol{\psi}$, $\boldsymbol{\phi}$, $\boldsymbol{\psi}'$, and $\boldsymbol{\phi}'$ for next iteration.

- (i) *Precomputations.* We construct a hierarchical partition (see Section 2.5 of Chapter 2) of the interval $\mathbf{I} = [d_1, d_{n+1}]$ just based on $\tilde{\mathbf{d}} = \{d_k\}_{k=1}^{n+1}$, where $d_{n+1} = d_n + \|\mathbf{v}\|_2^2$. Note that this partition shall be independent of \mathbf{x} . In the telescoping expansion of the triangular FMM matrix (2.61), the matrices $R^{(l)}$, $B^{(l)}$ and $V^{(L)}$ can be precomputed since they *do not* depend on \mathbf{x} (see their explicit formulas (2.15), (2.17), (2.53)). Then in Algorithm 1, these matrices are multiplied with appropriate slices of the vector \mathbf{w} . These computations are *independent* of y_k 's and x_k 's, so that they can be reused in the following update step.
- (ii) *Efficient Newton iterations.* Thanks to the interlacing property (3.33), the partition of the interval \mathbf{I} will also partition $\mathbf{x} = \{x_k\}_{k=1}^n$ into small subsets. When y_k 's (equivalently, x_k 's) are updated, in order to get $\boldsymbol{\psi}$, $\boldsymbol{\phi}$, $\boldsymbol{\psi}'$, and $\boldsymbol{\phi}'$ for next iteration, we only need to compute the local expansion matrix $U^{(L)}$, as well as the near-field interaction matrix K_n in the telescoping expansion (2.61). Moreover, the diagonal blocks of $U^{(L)}$ can be computed more accurately via $y_k = x_k - d_k$,

$$U_{\mathbf{i}} = \left(\left(\frac{x_k - o_{\mathbf{i}}}{\delta_{\mathbf{i}}} \right)^j \right)_{x_k \in \mathbf{i}, 0 \leq j \leq r-1} = \left(\left(\frac{y_k + (d_k - o_{\mathbf{i}})}{\delta_{\mathbf{i}}} \right)^j \right)_{x_k \in \mathbf{i}, 0 \leq j \leq r-1}, \quad (3.71)$$

where we can compute $d_k - o_{\mathbf{i}}$ accurately since the interval \mathbf{i} is produced in the hierarchical partition of $\tilde{\mathbf{d}} = \{d_k\}_{k=1}^{n+1}$. We refer (3.71) as *shifting in the far-field block* or *far-field shifting*.

The precomputation strategy can significantly reduce the amount of computations in the modified Newton's iteration. Moreover, these precomputations can be also reused for com-

puting the vectors $\hat{\mathbf{v}}$ and \mathbf{b} . In our experiments in MATLAB, SuperDC can be twice faster when the precomputation strategy is implemented.

3.7.3 Improved convergence criteria

In this subsection, we reexamine the convergence criteria

$$|f(d_k + y_k)| < cn(1 + |\psi_k(d_k + y_k)| + |\phi_k(d_k + y_k)|)\epsilon_{\text{mach}}. \quad (3.72)$$

This bound is first proposed in [16], and has the advantage that it comes for free after computing ψ and ϕ via triangular FMM. On the other hand, as pointed out in [16], [73], it might allow too much error when n is extremely large. While it works well in the tests for the large matrices in Section 3.5, a tighter and more accurate bound will still be desired. Indeed, Li proposes in [73] to compute $\psi_k(d_k + y_k) = \sum_{j=1}^k \frac{v_j^2}{\delta_{jk} - y_k}$ by summing up each term from $j = 1$ to k and $\phi_k(d_k + y_k) = \sum_{j=n}^{k+1} \frac{v_j^2}{\delta_{jk} - y_k}$ from n to $k + 1$, so that a backward error bound can be obtained on the way at the cost of about $2n$ extra additions

$$|f(d_k + y_k)| \leq \left(2 + \sum_{j=1}^k \frac{(k-j+6)v_j^2}{|\delta_{jk} - y_k|} + \sum_{j=n}^{k+1} \frac{(j-k+5)v_j^2}{|\delta_{jk} - y_k|} + |f(d_k + y_k)| + |y_k f'(d_k + y_k)| \right) \epsilon_{\text{mach}}. \quad (3.73)$$

This backward error bound would be more accurate and reasonable than (3.72) if $\psi_k(d_k + y_k)$ and $\phi_k(d_k + y_k)$ are computed via Li's method. However, it would not be applicable in SuperDC because (i) $\psi_k(d_k + y_k)$ and $\phi_k(d_k + y_k)$ are computed via the triangular FMM matrix-vector products $\psi = \tilde{C}_L \mathbf{w}$ and $\phi = \tilde{C}_U \mathbf{w}$; (ii) it is difficult to compute the bounds (3.73) simultaneously for all $k = 1, \dots, n$ in $O(n)$ operations.

Therefore, an alternative bound based on the backward error of triangular FMM matrix-vector product will be more suitable for SuperDC. In particular, the backward error bound

in Theorem 2.6.6 of Chapter 2 suggests that the following convergence criteria shall be a good candidate

$$|f(d_k + y_k)| < \tilde{c} \log^2 n (1 + |\psi_k(d_k + y_k)| + |\phi_k(d_k + y_k)|) \epsilon_{\text{mach}}, \quad (3.74)$$

where we relax the factor $\log n$ in Theorem 2.6.6 to $\log^2 n$ in order to accommodate the approximation error ϵ there. Here, \tilde{c} is moderate constant and $\tilde{c} = 256$ works well in our tests. In our numerical experiments, the new bound (3.74), together with the far-field shifting (3.71), can improve the accuracy of some eigenvalues by two digits.

3.8 Generalization to SVD solver

In this section, we generalize SuperDC to compute the SVD of a nonsymmetric HSS matrix A . We show two ways for doing this.

A straightforward way is to follow the idea in [21] to embed the matrix A into a symmetric HSS matrix, then apply the SuperDC eigensolver. To be more specific, we can embed A into the symmetric matrix

$$\bar{A} = \begin{pmatrix} & A^T \\ A & \end{pmatrix}.$$

Moreover, \bar{A} will be a symmetric HSS matrix after symmetric permutation. For example, for a two-level HSS matrix $A = \begin{pmatrix} D_1 & U_1 B_1 V_2^T \\ U_2 B_2 V_1^T & D_2 \end{pmatrix}$ and a suitable permutation matrix P , then $P\bar{A}P^T$ has the form

$$\begin{aligned}
P\bar{A}P^T &= P \left(\begin{array}{cc|cc} & & D_1^T & V_1B_2^TU_2^T \\ & & V_2B_1^TU_1^T & D_2^T \\ D_1 & U_1B_1V_2^T & & \\ U_2B_2V_1^T & D_2 & & \end{array} \right) P^T \\
&= \left(\begin{array}{c|c} D_1^T & V_1B_2^TU_2^T \\ \hline D_1 & U_1B_1V_2^T \\ V_2B_1^TU_1^T & D_2^T \\ U_2B_2V_1^T & D_2 \end{array} \right) \equiv \begin{pmatrix} \bar{D}_1 & \bar{U}_1\bar{B}_1\bar{U}_2^T \\ \bar{U}_2\bar{B}_1^T\bar{U}_1^T & \bar{D}_2 \end{pmatrix},
\end{aligned}$$

where $\bar{D}_i = \begin{pmatrix} D_i^T \\ D_i \end{pmatrix}$, $\bar{U}_i = \begin{pmatrix} V_i \\ U_i \end{pmatrix}$, $i = 1, 2$, and $\bar{B}_1 = \begin{pmatrix} B_2^T \\ B_1 \end{pmatrix}$. Then the SVD $A = X\Sigma Y^T$ can be obtained from the eigenvalue decomposition $\bar{A} = Q\Lambda Q^T$. In particular, the singular values of A are the nonnegative eigenvalues of \bar{A} , and the left and right singular matrices X and Y are given in terms of the structured eigenmatrix Q so that their matrix-vector product routines can be computed via that of Q , see [21] for more details. The overall complexity is $O((2r)^2 \cdot 2n \log^2(2n)) = O(8r^2n \log^2 n)$ and storage is $O(2r \cdot 2n \log(2n)) = O(4rn \log n)$, where the $2r$ and $2n$ are because the sizes of \bar{B}_1 and \bar{A} double compared to the sizes of B_1 and A , respectively. Although the complexity is still quasilinear, it needs eight times more work and four times more storage than the symmetric case.

The alternative is to directly exploit the HSS structure of A , but in a different way from the symmetric case. For the convenience of presentation, we assume all B generators of A have row size r . To illustrate the idea, suppose $A = D_3 = \begin{pmatrix} D_1 & U_1B_1V_2^T \\ U_2B_2V_1^T & D_2 \end{pmatrix}$ is a two-level $n \times n$ HSS matrix, where $n = n_1 + n_2$. Compute the QL factorizations $U_i = Q_i \begin{pmatrix} 0 \\ \tilde{U}_i \end{pmatrix}$ and partition $Q_i^T D_i$ accordingly

$$Q_1^T \left(D_1 \mid U_1 \right) = \left(\begin{array}{c|c} \hat{D}_1 & \\ \hline \tilde{D}_1 & \tilde{U}_1 \end{array} \right), \quad Q_2^T \left(U_2 \mid D_2 \right) = \left(\begin{array}{c|c} & \hat{D}_2 \\ \hline \tilde{U}_2 & \tilde{D}_2 \end{array} \right),$$

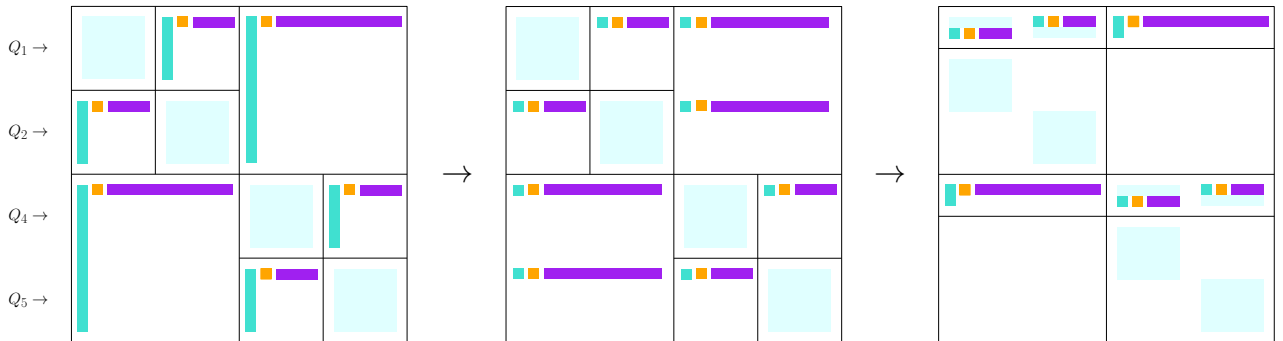
where $\tilde{U}_i \in \mathbb{R}^{r \times r}$, $\tilde{D}_i \in \mathbb{R}^{r \times n_i}$, $\hat{D}_i \in \mathbb{R}^{(n_i-r) \times n_i}$, $i = 1, 2$. Define $\Pi_3 = P_3 \cdot \text{diag}(Q_1^T, Q_2^T)$ where P_3 is a suitable permutation matrix, then

$$\Pi_3 A = P_3 \left(\begin{array}{c|c} \hat{D}_1 & \\ \hline \tilde{D}_1 & \tilde{U}_1 B_1 V_2^T \\ \hline \tilde{U}_2 B_1 V_1^T & \hat{D}_2 \\ & \tilde{D}_2 \end{array} \right) = \left(\begin{array}{c|c} \tilde{D}_1 & \tilde{U}_1 B_1 V_2^T \\ \hline \tilde{U}_2 B_1 V_1^T & \hat{D}_2 \\ \hline \hat{D}_1 & \\ & \hat{D}_2 \end{array} \right) \equiv \begin{pmatrix} T_1 & T_2 \\ \hat{D}_1 & \\ & \hat{D}_2 \end{pmatrix}. \quad (3.75)$$

The right-hand-side of (3.75) is said to have *block broken-arrowhead form*.

Next, suppose $A = D_7 = \begin{pmatrix} D_3 & U_3 B_3 V_6^T \\ U_6 B_6 V_3^T & D_6 \end{pmatrix}$ is a three-level HSS matrix, where D_3 and D_6 are two-level HSS matrices. Let Π_6 be defined similarly as in (3.75) with children nodes 4 and 5, then

$$\begin{pmatrix} \Pi_3 \\ \Pi_6 \end{pmatrix} A = \begin{pmatrix} T_1 & T_2 & \hat{U}_3 B_3 V_6^T \\ \hat{D}_1 & & \\ \hat{D}_2 & & \\ \hline \hat{U}_6 B_6 V_3^T & T_4 & T_5 \\ & \hat{D}_4 & \\ & & \hat{D}_5 \end{pmatrix}, \text{ where } \hat{U}_3 = \begin{pmatrix} \tilde{U}_1 R_1 \\ \tilde{U}_2 R_2 \end{pmatrix}, \hat{U}_6 = \begin{pmatrix} \tilde{U}_4 R_4 \\ \tilde{U}_5 R_5 \end{pmatrix} \in \mathbb{R}^{2r \times r}.$$

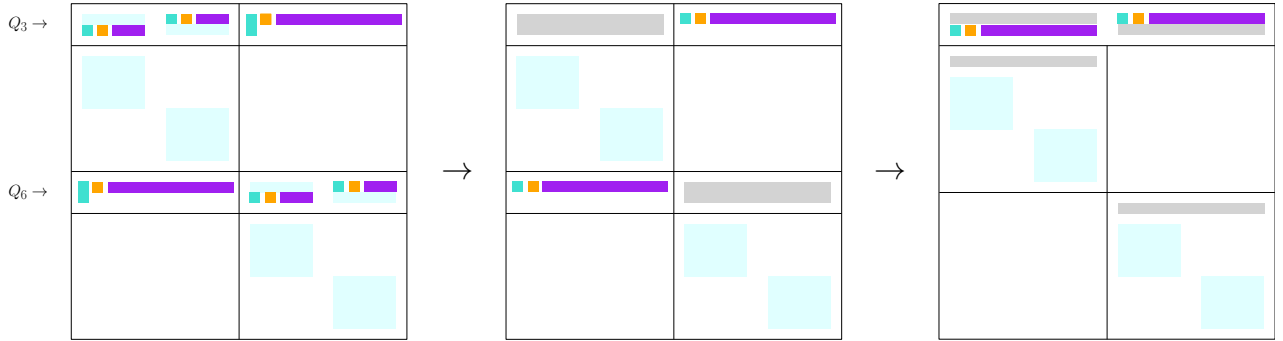


Compute the QL factorizations $\hat{U}_i = Q_i \begin{pmatrix} 0 \\ \tilde{U}_i \end{pmatrix}$, $\tilde{U}_i \in \mathbb{R}^{r \times r}$, $i = 3, 6$, and partition accordingly

$$Q_3^T \left(T_1 \ T_2 \mid \hat{U}_3 \right) = \left(\begin{array}{cc|c} \hat{T}_1 & \hat{T}_2 & \\ \tilde{T}_1 & \tilde{T}_2 & \tilde{U}_3 \end{array} \right), \quad Q_6^T \left(\hat{U}_6 \mid T_4 \ T_5 \right) = \left(\begin{array}{c|cc} & \hat{T}_4 & \hat{T}_5 \\ \tilde{U}_6 & \tilde{T}_4 & \tilde{T}_5 \end{array} \right).$$

Define $\Pi_7 = P_7 \cdot \text{diag}(Q_3^T, I_{n_3-2r}, Q_6^T, I_{n_6-2r})$ where P_7 is a suitable permutation and I_{n_i-2r} 's are the identity matrices, then

$$\Pi_7 \begin{pmatrix} \Pi_3 & \\ & \Pi_6 \end{pmatrix} A = \left(\begin{array}{cc|cc} \tilde{T}_1 & \tilde{T}_2 & \tilde{U}_3 B_3 V_6^T & \\ \tilde{U}_6 B_6 V_3^T & \tilde{T}_4 & \tilde{T}_5 & \\ \hline \hat{T}_1 & \hat{T}_2 & & \\ \hat{D}_1 & & & \\ & \hat{D}_2 & & \\ \hline & & \hat{T}_4 & \hat{T}_5 \\ & & \hat{D}_4 & \\ & & & \hat{D}_5 \end{array} \right) \equiv \begin{pmatrix} T_3 & T_6 \\ \hat{D}_3 & \\ & \hat{D}_6 \end{pmatrix}, \quad \hat{D}_i \in \mathbb{R}^{(n_i-r) \times n_i}. \quad (3.76)$$



As a result, the matrix A is reduced to block broken-arrowhead form (3.76), with its subblocks \hat{D}_3 and \hat{D}_6 also in block broken-arrowhead form.

The above procedure can be generalized to multilevel in an obvious way, such that a multilevel HSS matrix can be reduced to *multilevel block broken-arrowhead form* via a sequence of orthogonal matrices $\{\Pi_i\}_{i \in \mathcal{T}}$, see Figure 3.9. Therefore, the SVD of A can be computed recursively if we know how to compute the SVD of a block broken-arrowhead



Figure 3.9. Multilevel block broken-arrowhead form

matrix. We show how to do this for the rectangular matrix $\hat{D}_p = \begin{pmatrix} \hat{T}_i & \hat{T}_j \\ \hat{D}_i & \\ & \hat{D}_j \end{pmatrix}$ where

$\hat{D}_i \in \mathbb{R}^{(n_i-r) \times n_i}$, $\hat{D}_j \in \mathbb{R}^{(n_j-r) \times n_j}$, $\begin{pmatrix} \hat{T}_i & \hat{T}_j \end{pmatrix} \in \mathbb{R}^{r \times (n_i+n_j)}$. Without loss of generality, we can assume $\begin{pmatrix} \hat{T}_i & \hat{T}_j \end{pmatrix}$ has full row rank. Otherwise, we can just compute the QL factorization

$\begin{pmatrix} \hat{T}_i & \hat{T}_j \end{pmatrix} = G_p \begin{pmatrix} 0 & 0 \\ \check{T}_i & \check{T}_j \end{pmatrix}$ where $G_p \in \mathbb{R}^{r \times r}$ is orthogonal, and then solve the SVD of

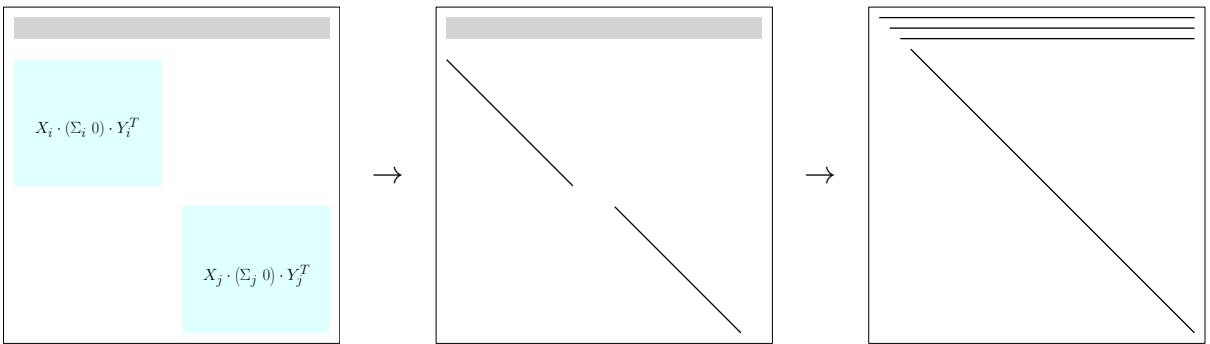
the smaller matrix $\begin{pmatrix} \check{T}_i & \check{T}_j \\ \hat{D}_i & \\ & \hat{D}_j \end{pmatrix}$. Suppose we have computed the SVDs of \hat{D}_i and \hat{D}_j

$$\begin{aligned} \hat{D}_i &= X_i \begin{pmatrix} \Sigma_i & 0 \end{pmatrix} Y_i^T, & \Sigma_i &\in \mathbb{R}^{(n_i-r) \times (n_i-r)}, \\ \hat{D}_j &= X_j \begin{pmatrix} \Sigma_j & 0 \end{pmatrix} Y_j^T, & \Sigma_j &\in \mathbb{R}^{(n_j-r) \times (n_j-r)}, \end{aligned}$$

$$\begin{aligned}
\begin{pmatrix} \hat{T}_i & \hat{T}_j \\ \hat{D}_i & \hat{D}_j \end{pmatrix} &= \begin{pmatrix} I_r & & & \\ & X_i & & \\ & & X_j & \\ & & & \end{pmatrix} \begin{pmatrix} \hat{T}_i Y_i & \hat{T}_j Y_j \\ (\Sigma_i \ 0) & (\Sigma_j \ 0) \end{pmatrix} \begin{pmatrix} Y_i^T \\ Y_j^T \end{pmatrix} \\
&= \begin{pmatrix} I_r & & & \\ & X_i & & \\ & & X_j & \\ & & & \end{pmatrix} \begin{pmatrix} F_i & F_j & H_i & H_j \\ & & \Sigma_i & \\ & & & \Sigma_j \end{pmatrix} P \begin{pmatrix} Y_i^T \\ Y_j^T \end{pmatrix}
\end{aligned} \tag{3.77}$$

where $\hat{T}_i Y_i = \begin{pmatrix} H_i & F_i \end{pmatrix}$, $\hat{T}_j Y_j = \begin{pmatrix} H_j & F_j \end{pmatrix}$ and P is a permutation matrix. Compute the RQ factorization of the $r \times 2r$ matrix $\begin{pmatrix} F_i & F_j \end{pmatrix} = \begin{pmatrix} 0 & Z \end{pmatrix} G$, where $Z \in \mathbb{R}^{r \times r}$ is upper triangular and $G \in \mathbb{R}^{2r \times 2r}$ is orthogonal, then

$$\begin{aligned}
\begin{pmatrix} \hat{T}_i & \hat{T}_j \\ \hat{D}_i & \hat{D}_j \end{pmatrix} &= \begin{pmatrix} I_r & & & \\ & X_i & & \\ & & X_j & \\ & & & \end{pmatrix} \begin{pmatrix} 0 & Z & H_i & H_j \\ 0 & & \Sigma_i & \\ 0 & & & \Sigma_j \end{pmatrix} \begin{pmatrix} G & \\ & I_{n_i+n_j-2r} \end{pmatrix} P \begin{pmatrix} Y_i^T \\ Y_j^T \end{pmatrix} \\
&= \begin{pmatrix} I_r & & & \\ & X_i & & \\ & & X_j & \\ & & & \end{pmatrix} \begin{pmatrix} Z & H_i & H_j & 0 \\ & \Sigma_i & & 0 \\ & & \Sigma_j & 0 \end{pmatrix} \tilde{P} \begin{pmatrix} G & \\ & I_{n_i+n_j-2r} \end{pmatrix} P \begin{pmatrix} Y_i^T \\ Y_j^T \end{pmatrix}
\end{aligned}$$



Therefore we only need to compute the SVD of the following upper triangular square matrix

$$M = \left(\begin{array}{c|cc} Z & H_i & H_j \\ \hline & \Sigma_i & \\ & & \Sigma_j \end{array} \right) \equiv \begin{pmatrix} Z^{(0)} & H^{(0)} \\ & \Sigma^{(0)} \end{pmatrix} \in \mathbb{R}^{(n_i+n_j-r) \times (n_i+n_j-r)}. \tag{3.78}$$

For this purpose, we follow the strategy in [20], [21], [47] to get the SVD of M . To be more specific, partition

$$Z^{(0)} = \begin{pmatrix} Z^{(1)} & \mathbf{z} \\ & v_0 \end{pmatrix}, \quad H^{(0)} = \begin{pmatrix} \tilde{H}^{(0)} \\ \mathbf{v}^T \end{pmatrix},$$

$$M = \left(\begin{array}{c|cc} Z^{(1)} & \mathbf{z} & \tilde{H}^{(0)} \\ \hline & v_0 & \mathbf{v}^T \\ & & \Sigma^{(0)} \end{array} \right) \equiv \begin{pmatrix} Z^{(1)} & \tilde{H}^{(1)} \\ & M^{(1)} \end{pmatrix},$$

so that $M^{(1)}$ is a broken arrowhead matrix. Let $M^{(1)} = X^{(1)}\Sigma^{(1)}(Y^{(1)})^T$ be the SVD computed according to Lemmas 3.8.1 and 3.8.2, then

$$M = \begin{pmatrix} I_{r-1} & \\ & X^{(1)} \end{pmatrix} \begin{pmatrix} Z^{(1)} & H^{(1)} \\ & \Sigma^{(1)} \end{pmatrix} \begin{pmatrix} I_{r-1} & \\ & Y^{(1)} \end{pmatrix}^T.$$

The matrix in the middle has similar form to M but with a smaller leading block $Z^{(1)}$. The above steps are repeated r times to get the SVD of M

$$M = \hat{X}_p \Sigma_p \hat{Y}_p^T, \quad \text{where}$$

$$\hat{X}_p = \prod_{k=1}^r \begin{pmatrix} I_{r-k} & \\ & X^{(k)} \end{pmatrix}, \quad \hat{Y}_p = \prod_{k=1}^r \begin{pmatrix} I_{r-k} & \\ & Y^{(k)} \end{pmatrix}.$$

Then the SVD of \hat{D}_p will be given by

$$\hat{D}_p = X_p \begin{pmatrix} \Sigma_p & 0 \end{pmatrix} Y_p^T,$$

where the left and right singular matrices are

$$X_p = \begin{pmatrix} I_r & \\ & X_i \\ & & X_j \end{pmatrix} \prod_{k=1}^r \begin{pmatrix} I_{r-k} & \\ & X^{(k)} \end{pmatrix},$$

$$Y_p = \begin{pmatrix} Y_i & \\ & Y_j \end{pmatrix} P^T \begin{pmatrix} G^T & \\ & I_{n_i+n_j-2r} \end{pmatrix} \tilde{P}^T \begin{pmatrix} \prod_{k=1}^r \begin{pmatrix} I_{r-k} & \\ & Y^{(k)} \end{pmatrix} \\ & I_r \end{pmatrix}.$$

As a result, the problem is boiled down to computing the SVD of a broken arrowhead matrix like $M^{(1)}$, which has been well studied in [20], [21], [47]. The singular values and vectors of $M^{(1)}$ can be found explicitly via the following Lemma 3.8.1 from [21]. Note that, a standard deflation step need be first applied to $M^{(1)}$ so that the prerequisites of Lemma 3.8.1 are satisfied. This is similar to the symmetric case. More discussions about deflation can be found in [18], [20], [21], [34] as well as Section 3.4.1.

Lemma 3.8.1 ([20], [21], [47]). *Suppose $0 < d_2 < \dots < d_n$ and $v_i \neq 0$. Let $U\Sigma W^T$ be the SVD of the broken-arrowhead matrix*

$$\begin{pmatrix} v_1 & v_2 & \cdots & v_n \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{pmatrix} = U\Sigma W^T, \quad \text{with}$$

$$U = \begin{pmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_n \end{pmatrix}, \quad \Sigma = \text{diag}(\sigma_1, \dots, \sigma_n), \quad W = \begin{pmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_n \end{pmatrix},$$

where $0 < \sigma_1 < \dots < \sigma_n$. Then the singular values $\{\sigma_k\}_{k=1}^n$ satisfies the interlacing property

$$0 \equiv d_1 < \sigma_1 < d_2 < \dots < d_n < \sigma_n < d_n + \|v\|_2,$$

and the secular equation

$$f(\sigma) = 1 + \sum_{k=1}^n \frac{v_k^2}{d_k^2 - \sigma^2} = 0. \quad (3.79)$$

And the singular vectors are

$$\mathbf{u}_k = \left(-1 \quad \frac{d_2 v_2}{d_2^2 - \sigma_k^2} \quad \cdots \quad \frac{d_n v_n}{d_n^2 - \sigma_k^2} \right)^T / \sqrt{1 + \sum_{i=2}^n \frac{d_i^2 v_i^2}{(d_i^2 - \sigma_k^2)^2}}, \quad (3.80)$$

$$\mathbf{w}_k = \left(\frac{v_1}{d_1^2 - \sigma_k^2} \quad \cdots \quad \frac{v_n}{d_n^2 - \sigma_k^2} \right)^T / \sqrt{\sum_{i=1}^n \frac{v_i^2}{(d_i^2 - \sigma_k^2)^2}}. \quad (3.81)$$

Similar to the symmetric rank-one update problem, SuperDC can find the gap $\eta_k = \sigma_k - d_k$ efficiently and accurately via modified Newton's method and triangular FMM. Note that since $d_i \geq 0, \sigma_k > 0$, we can take advantage of the identity $d_i^2 - \sigma_k^2 = ((d_i - d_k) - \eta_k)(d_i + \sigma_k)$ to avoid cancellation in FMM.

However, the singular vectors given by (3.80) and (3.81) may lose orthogonality even if the computed singular values $\hat{\sigma}_k$ have double precision accuracy. In order for the computed singular vectors to be numerically orthogonal, Gu uses Löwner theorem [20], [79] to compute \hat{v}_i to replace v_i in (3.80) and (3.81).

Lemma 3.8.2 ([20], [79]). *If $\{d_k\}_{k=1}^n$ and $\{\hat{\sigma}_k\}_{k=1}^n$ satisfy the interlacing property*

$$0 = d_1 < \hat{\sigma}_1 < d_2 < \cdots < d_n < \hat{\sigma}_n,$$

then there exists a broken-arrowhead matrix $\hat{M} = \begin{pmatrix} \hat{v}_1 & \hat{v}_2 & \cdots & \hat{v}_n \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{pmatrix}$ whose singular values

are $\{\hat{\sigma}_k\}_{k=1}^n$. The components of the vector $\hat{\mathbf{v}} = (\hat{v}_1 \quad \cdots \quad \hat{v}_n)^T$ are uniquely determined up to a sign

$$\log |\hat{v}_i| = \frac{1}{2} \left(\sum_{j=1}^n \log |d_i^2 - \hat{\sigma}_j^2| - \sum_{j=1, j \neq i}^n \log |d_i^2 - d_j^2| \right).$$

Again, similar to the symmetric case, FMM can be adapted to accelerate the computations of $\hat{\mathbf{v}}$ and the normalization factors of \mathbf{u}_k and \mathbf{w}_k , as well as the matrix-vector product routines of the singular matrices U and W .

3.9 Conclusions

In this chapter, we have designed a more reliable SuperDC eigensolver. It significantly improves the divide-and-conquer algorithm in [39] in terms of stability and efficiency. A series

of stability enhancements is built into the different stages of the algorithm. In particular, we avoid an exponential norm growth risk in the dividing stage via a balancing strategy. And we are able to combine FMM accelerations with several key stability safeguards that have been used in practical divide-and-conquer algorithms. We also give a variety of algorithm designs and structure studies that have been missing or unclear in [39]. The comprehensive numerical tests confirm the nearly linear complexity and much higher efficiency than the Matlab `eig` function for the eigendecomposition of different types of HSS matrices. Nice accuracy and eigenvector orthogonality have been observed. Comparisons also illustrate the benefits of our stability techniques.

The SuperDC eigensolver makes it feasible to use full eigendecompositions to solve various challenging numerical problems as mentioned at the beginning of this chapter. In addition, we expect that the novel local shifting strategy and triangular FMM accelerations are also useful for other FMM-related matrix computations when stability and accuracy are crucial. In our future work, we plan to provide the proof of backward stability, as well as a high-performance parallel implementation, which will extend the applicability of the algorithm to large-scale numerical computations.

We also describe a novel precomputation strategy that can reuse the computations in triangular FMM. We also extend the SuperDC to compute the SVD of HSS matrices. This can accommodate more situations when matrices are nonsymmetric.

4. SUPERFAST FACTORIZATION UPDATE FOR DIAGONALLY-SHIFTED SPARSE DISCRETIZED MATRICES

In this chapter, we consider some important discretized matrices, and develop a series of techniques that enable us to quickly obtain a factorization of A and then perform factorization update for multiple shifted matrices $A - s_j I$. We first compute a structured partial factorization of A , which can be reused to obtain new factorizations for free for multiple s_j . This idea is feasible because of several innovative ideas. One is to compute a fast structured eigenvalue decomposition for large a pivot submatrix A_{11} . This eigenvalue decomposition can be updated for free for different shifts. Then we use structured HSS form to approximate the Schur complements. This is done via randomization and matrix-vector multiplication. A key idea is to assemble all the matrix-vector multiplications for all the shifts together in a highly structured matrix-matrix multiplication. We fully utilize all the sparsity and structures in this process.

By carefully designing all the steps, most of the operations can be done in a precomputation step. The update is essentially only limited to a small subproblem (such as that correspond to boundary mesh points). For two dimensional elliptic problems and Helmholtz problems with certain coefficients and discretizations, the algorithm requires a precomputation cost of roughly $O(n)$ flops. The factorization update for each new shift needs only $O(\sqrt{n} \log n)$. The factorization update is thus said to be superfast and is significantly more efficient than redoing factorizations.

4.1 Introduction

In recent years, there have been extensive developments on fast direct factorizations of sparse discretized Helmholtz equations. Such direct solvers provide fast and reliable solutions that are very attractive for applications such as full waveform inversion (FWI). With low or medium frequencies, they can usually reach nearly $O(n)$ complexity in two dimensions and $O(n) \sim O(n^{4/3})$ complexity in three dimensions.

However, in FWI, it often involves the solution of Helmholtz equations with multiple frequencies like in

$$-\Delta u - \omega_j^2 c(x)^{-2} u = f, \quad j = 1, 2, \dots, m. \quad (4.1)$$

It is well known that, in matrix computation, an LU factorization of A generally cannot be reused to get that of $A - s_j I$ for a new shift $s_j I$. Thus, it is usually not possible to perform fast shifted LU factorization update.

In this chapter, we are interested in the fast factorization of shifted sparse matrices, such as matrices arising from some discretized equations. We suppose that, after appropriate permutations and preprocessing, the discretized matrix has the following form:

$$A - s_j I \equiv \begin{pmatrix} A_{11} - s_j I & A_{12} \\ A_{21} & A_{22} - s_j I \end{pmatrix}, \quad (4.2)$$

where $\{s_j\}_{j=1}^m$ are m scalars, and A is partitioned so that A_{11} is Hermitian. We suppose A_{11} is $n \times n$ and A_{22} is $N \times N$. Examples of such problems are shifted Laplacians and certain discretized Helmholtz equations corresponding to varying frequencies [28], [80], [81].

For some Helmholtz problems like the one in [28], we choose the permutation so that A_{11} correspond to the interior points. A_{11} is Hermitian and has a form like in a usual finite difference matrix with Dirichlet boundary conditions. Later, we refer to A_{11} as the interior problem. A_{22} corresponds to the boundary points with boundary conditions such as PML. Sometimes, the discretized matrix does not appear like (4.2) but can be converted into such a form (see Section 4.2).

We seek to develop a superfast (sublinear cost) factorization update algorithm for such problems. For the form in (4.2), we first compute an eigenvalue decomposition for A_{11} . Such a decomposition is usually prohibitively expensive for large sparse matrices. However, for problems like the one in [28], where the interior part A_{11} is related to a separable problem, we can quickly perform eigenvalue decompositions. The main idea is to utilize SuperDC in Chapter 3. The cost to factorize A_{11} is nearly $O(n)$. Note that the eigendecomposition of A_{11} can be updated for free for each new shift $s_j I$. That is, we can quickly eliminate $A_{11} - s_j I$ from each $A - s_j I$.

The next challenge is to deal with the Schur complement S_j resulting from the elimination of $A_{11} - s_j I$. The Schur complement problem is a smaller problem. Nevertheless, it has a significant cost to form the Schur complement explicitly and factorize it for different shifts. The reason is that S_j is dense due to the update from the partial elimination. This update involves $(A_{11} - s_j I)^{-1}$ and is very costly to form directly.

Here, we apply a structured approximation to the Schur complement, and then compute a structured factorization. To save the costs for multiple shifts, we apply some innovative strategies. One is a randomized construction of the structured approximation to S_j . The main cost of this construction is the multiplication of S_j and random vectors z . Thus, we assemble all the matrix-vector multiplications for different shifts into a large structured intermediate matrix. Each column of this structured intermediate matrix corresponds to one s_j value. This structured form is a product of multiple matrices that are sparse or highly structured. We take full advantage of the sparsity and structures to quickly assemble the intermediate matrix. This enables us to get the needed matrix-vector products for all s_j simultaneously. With the use of multiple z , it is convenient to construct structured approximations such as hierarchically semiseparable (HSS) forms [14], [57] for S_j .

For discretized Poisson equations and certain discretized Helmholtz equations in two dimensions, the entire algorithm involves some precomputations that cost about $O(n)$. Then for each new shift, the cost to obtain a new factorization is significantly lower than refactoring $A - s_j I$. In fact, the factorization update costs only $O(\sqrt{n} \log n)$ (sublinear) and is thus said to be *superfast*.

In the following, we discuss the fast eigenvalue decomposition of A_{11} in Section 4.2. Section 4.3 gives the main algorithm for the shifted factorization. Section 4.4 discusses the complexity and some extensions. Section 4.5 presents some numerical experiments. Some concluding remarks are drawn in Section 4.6.

4.2 Fast eigenvalue decomposition for the interior problem

For (4.1), the discretized matrix looks like $S + \omega_j^2 M$, where S is the stiffness matrix and M is the mass matrix. We need convert it into the form in (4.2). Here, we are interested in

a type of problems where the fast eigendecomposition of A_{11} in (4.2) is possible. To illustrate the idea, we use the 2D case, and the idea can be similarly understood for 3D cases. We assume S and M have the following forms:

$$S = \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix}, \quad M = \begin{pmatrix} M_{11} & \\ & M_{22} \end{pmatrix}, \quad \text{with} \quad (4.3)$$

$$S_{11} = I \otimes G + H \otimes I, \quad M_{11} = I \otimes D^{-2}, \quad (4.4)$$

where S and M are partitioned conformably as in (4.2), G and H correspond to discretized problems in one dimensions, and D is an invertible diagonal matrix with the same size as G . This is the case for the problem in [28]. For such a case, A_{11} in (4.2) looks like

$$A_{11} = (I \otimes D)(I \otimes G + H \otimes I)(I \otimes D),$$

where D is a diagonal matrix, and G and H are 1D Hermitian discretized matrices in the x and y directions, respectively. Suppose G and D are $K \times K$ and H is $M \times M$. A_{11} can be rewritten as

$$A_{11} = I \otimes \tilde{G} + H \otimes D^2,$$

where $\tilde{G} = DGD$. We can apply SuperDC in Chapter 3 to compute the eigendecomposition

$$H = Q_h \Lambda_h Q_h^*, \quad (4.5)$$

where we abuse notation and still use H to mean its HSS approximation. (We will also do so for later cases.) Then

$$\begin{aligned} A_{11} &= I \otimes \tilde{G} + (Q_h \Lambda_h Q_h^*) \otimes D^2 \\ &= (Q_h Q_h^*) \otimes \tilde{G} + (Q_h \Lambda_h Q_h^*) \otimes D^2 \\ &= (Q_h \otimes I)[I \otimes \tilde{G} + \Lambda_h \otimes D^2](Q_h \otimes I)^*. \end{aligned}$$

Next, we find an eigendecomposition for $I \otimes \tilde{G} + \Lambda_h \otimes D^2$, which is a block diagonal matrix with diagonal blocks

$$T_j \equiv \tilde{G} + \lambda_{h,j} D^2 = D(G + \lambda_{h,j} I)D, \quad j = 1, 2, \dots, M, \quad (4.6)$$

where $\Lambda_h = \text{diag}(\lambda_{h,1}, \dots, \lambda_{h,M})$ and M is the size of H . Since G is a 1D Hermitian discretized matrix, it can be approximated by a structured matrix such as the hierarchically semiseparable (HSS) form [14], [57]. Since D is diagonal, T_j can also be approximated by an HSS form. Note that there is no need to form T_j in (4.6) explicitly since we can directly update the HSS approximation to $G + \lambda_{h,j} I$. Such updates can be performed quickly by updating G 's generators. As a result, we can again use SuperDC to quickly find an eigendecomposition for T_j as

$$T_j = Q_{t,j} \Lambda_{t,j} Q_{t,j}^*. \quad (4.7)$$

Then A_{11} has an eigendecomposition

$$A_{11} = Q \Lambda Q^*, \quad \text{with} \quad (4.8)$$

$$\Lambda = \text{diag}(\Lambda_{t,1}, \dots, \Lambda_{t,M}), \quad Q = (Q_h \otimes I) \text{diag}(Q_{t,1}, \dots, Q_{t,M}). \quad (4.9)$$

We make some remarks regarding this structured eigenmatrix.

- Note that Q_h, Q_g and all $Q_{t,j}$ are all in structured forms and can be multiplied with a vector in nearly linear complexity via the fast multipole method (see Chapters 2 and 3). The overall complexity to multiply Q or Q^* is roughly linear.
- These structured eigenmatrices are very memory-efficient. If traditional eigensolvers were used and eigenmatrices were dense, storing $O(M)$ such dense $K \times K$ matrices would be a challenge, especially when mesh sizes M and K are large.
- These two perspectives justifies our use of the SuperDC, which is highly efficient in both speed and space.

4.3 Fast diagonally-shifted factorization update

Then, we consider the factorization update problem for (4.2). According to (4.8),

$$A - s_j I = \begin{pmatrix} A_{11} - s_j I & A_{12} \\ A_{21} & A_{22} - s_j I \end{pmatrix} \begin{matrix} Interior \\ Boundary \end{matrix} \quad (4.10)$$

$$\approx \begin{pmatrix} Q & \\ L_{21} & I \end{pmatrix} \begin{pmatrix} \Lambda - s_j I & \\ & S_j \end{pmatrix} \begin{pmatrix} Q^* & R_{12} \\ & I \end{pmatrix}, \quad (4.11)$$

where

$$\begin{aligned} L_{21} &= A_{21}Q(\Lambda - s_j I)^{-1}, & R_{12} &= (\Lambda - s_j I)^{-1}Q^*A_{12}, \\ S_j &= A_{22} - s_j I - A_{21}Q(\Lambda - s_j I)^{-1}Q^*A_{12}. \end{aligned} \quad (4.12)$$

L_{21} and R_{12} are not explicitly formed since they admit quick matrix-vector multiplications, which is how they are used in the solution stage. Q in (4.9) is structured and can be quickly applied to a vector in nearly linear complexity. The remaining challenge is to apply hierarchical structured methods to *all* S_j . We use HSS methods.

4.3.1 Structured approximation of Schur complements with multiple shifts

The HSS approximation can be done by randomized construction [12], [13], [30], where the dominate cost is to multiply S_j and some random vectors z like

$$S_j z = A_{22}z - s_j z - A_{21}Q(\Lambda - s_j I)^{-1}Q^*A_{12}z. \quad (4.13)$$

The products $A_{22}z$ and $\tilde{z} \equiv Q^*(A_{12}z)$ can be quickly computed based on the sparsity of A_{22}, A_{12} and the structure of Q . The major challenge is to evaluate $A_{21}Q(\Lambda - s_j I)^{-1}\tilde{z}$ for *all* j . Naive multiplication costs at least $O(n)$ for each s_j , resulting in a total cost at least $O(mn)$ for m shifts.

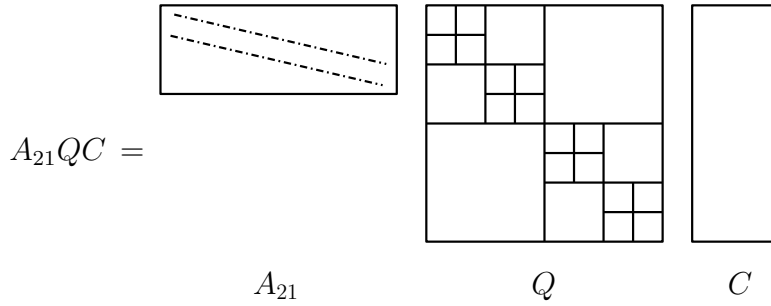
A more intuitive way is to formulate $(\Lambda - s_j I)^{-1} \tilde{z}$ for all s_j values together as the columns of a Cauchy-like matrix:

$$C = \left(\frac{\tilde{z}_i}{\lambda_i - s_j} \right)_{n \times m}, \quad (4.14)$$

where $\tilde{z} = \left(\tilde{z}_1 \ \dots \ \tilde{z}_n \right)^T$. We then compute $A_{21}Q(\Lambda - s_j I)^{-1} \tilde{z}$ for all s_j together since

$$\left(A_{21}Q(\Lambda - s_1 I)^{-1} \tilde{z} \ \dots \ A_{21}Q(\Lambda - s_m I)^{-1} \tilde{z} \right) = A_{21}QC.$$

We just need to find the columns of $A_{21}QC$. If m is very small, then we directly form $A_{21}QC$ by using the sparsity of A_{21} and the structure of Q . Thus, we focus on the case where m is not too small.

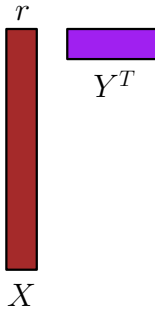


A convenient strategy is to form $A_{21}Q$ first in precomputations and then use Cauchy-like matrix-vector multiplications or the FMM to compute $A_{21}QC$. However, this precomputation cannot make use of the sparsity of A_{21} and costs $O(n^{3/2})$ in two dimensions (and $O(n^{5/3})$ in three dimensions) if $m = O(N)$.

We thus design a fully structured strategy that can take advantage of all the structures: the sparsity in A_{21} , the structures of Q , and the structures of C . Note that the Cauchy-like matrix C in (4.14) is highly structured.

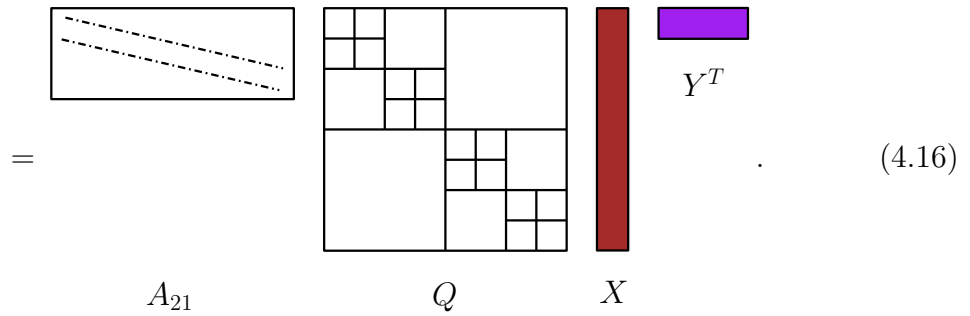
- If the interior eigenvalues $\{\lambda_i\}$ and the shifts $\{s_j\}$ are well separated, then C is numerically low rank. We can obtain a separable low-rank approximation $C \approx XY^T$ via

Taylor expansions, or even better, a recent analytical compression method in [36] with the Cauchy kernel function.

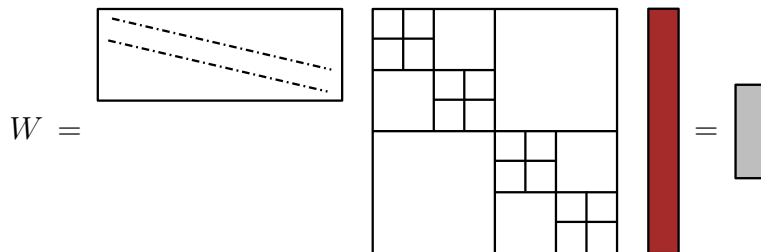
$$C = \left(\frac{\tilde{z}_i}{\lambda_i - s_j} \right)_{n \times m} \approx X_{n \times r} Y_{m \times r}^T =$$


Here X only depends on $\{\lambda_i, \tilde{z}_i\}$ and Y only depends on $\{s_j\}$. At this point,

$$A_{21}QC \approx A_{21}QXY^T \tag{4.15}$$



Note that when we use the method in [36], the column basis matrix X essentially only depends on the $\{\lambda_i\}$ values and is *independent of the $\{s_j\}$ values*. This means that the $N \times r$ matrix $A_{21}QX$ can be *precomputed*. Even such a precomputation is very fast based on the sparsity of A_{21} and the structure of Q . After this, it is quick to evaluate $A_{21}QC$ since Y is just an $m \times r$ matrix. That is, the matrix $W = A_{21}QX$ is independent of the shifts and can be *precomputed* in $O(rn \log n)$.



After precomputing W ,

$$A_{21}QC \approx WY^T = \begin{array}{|c|c|} \hline \text{gray rectangle} & \text{purple rectangle} \\ \hline \end{array}$$

can be computed in $O(rmN)$. Each shift $S_j z$ now costs $O(rN)$, rather than $O(n)$ in direct evaluation.

Thus, we can quickly evaluate $S_j z$ in (4.13) for all $j = 1, \dots, m$. For convenience, suppose r_1 is the maximum off-diagonal numerical rank (called HSS rank [14]) of S_j 's. To get HSS approximations to all S_j based on the matrix-vector products, we can use either a partially matrix-free HSS construction [12], [13], [30] or a fully matrix-free HSS construction algorithms in [12], [82]. The previous version requires fewer matrix-vector multiplications but needs to form $O(r_1 N)$ entries of S_j with unknown locations. This is then impractical. We thus use the fully matrix-free version, which requires the multiplication of each S_j with $O(r_1 \log N)$ random vectors z . That is, the multiplication procedure needs to be repeated for all these z vectors. After that, we have the products of each S_j with $O(r_1 \log N)$ random vectors z . Then we apply the algorithm in [82] to construct the HSS approximations.

4.3.2 Factorization update

With the HSS approximation to each S_j , we can then compute an ULV-type HSS factorization [14], [57] of the following form:

$$S_j \approx \mathcal{U}_j \mathcal{L}_j \mathcal{V}_j^*,$$

where \mathcal{U}_j (\mathcal{V}_j) represents a sequence of unitary factors and \mathcal{L}_j represents a sequence of triangular factors. Plugging this into (4.11) yields a structured factorization:

$$A - s_j I \approx \begin{pmatrix} Q & \\ L_{21} & \mathcal{U}_j \end{pmatrix} \begin{pmatrix} \Lambda - s_j I & \\ & \mathcal{L}_j \end{pmatrix} \begin{pmatrix} Q^* & R_{12} \\ & \mathcal{V}_j^* \end{pmatrix}.$$

When s_j varies, L_{21} and R_{12} have the forms in (4.10) and need no cost to update. Thus, the only actual update is to construct and factorization the HSS approximation to S_j . That is, with our elaborate design of the algorithm, we essentially restrict the actual factorization update to just the Schur complement problem (or the subproblem corresponding to the boundary).

4.4 Complexity and discussions

Algorithm 6 Precomputation stage

Input $G, H, D, A_{12}, A_{21}, A_{22}$
Output W, U, Λ, Q, Z

- 1: Compute $[Q_h, \Lambda_h] = \text{superdc}(H)$
- 2: **for** $j = 1, \dots, M$ **do**
- 3: $T_j = D(G + \lambda_{h,j}I)D$
- 4: Compute $[Q_{t,j}, \Lambda_{t,j}] = \text{superdc}(T_j)$
- 5: **end for**
- 6: $\Lambda = \text{diag}(\Lambda_{t,1}, \dots, \Lambda_{t,M}), \quad Q = (Q_h \otimes I) \text{diag}(Q_{t,1}, \dots, Q_{t,M})$
- 7: Generate $O(r_1 \log N)$ random vectors Z
- 8: $W = [], U = []$
- 9: **for** z : column of Z **do**
- 10: Compute $\tilde{z} = Q^* A_{12} z$
- 11: Generate matrix X from $\{\lambda_i\}$ and \tilde{z} via Taylor expansion
- 12: Compute $w = A_{21} Q X, u = A_{22} z$
- 13: $W = [W; w]$
- 14: $U = [U, u]$
- 15: **end for**

Algorithm 7 Online computation stage

Input $W, U, \Lambda, Q, Z, s = \{s_j\}_{j=1}^m$
Output $\mathcal{S}_j, \{\mathcal{U}_j, \mathcal{L}_j, \mathcal{V}_j\}_{j=1}^m$

- 1: Generate matrix Y from $\{s_j\}$ via Taylor expansion
- 2: $W = W \cdot Y$
- 3: **for** $j = 1, \dots, M$ **do**
- 4: Randomized matrix-free HSS construction $\mathcal{S}_j = \text{randhssmf}(W, U, Z, s_j)$
- 5: HSS ULV factorization $\{\mathcal{U}_j, \mathcal{L}_j, \mathcal{V}_j\} = \text{hssulv}(\mathcal{S}_j)$
- 6: **end for**

Procedures of precomputation stage, as well as online factorization update stage, are summarized in Algorithm 6 and 7. We then inspect the complexity of the overall factorization update algorithm. Suppose r_1 is a uniform bound for the HSS ranks of all S_j 's. For Helmholtz equations with small frequencies, r_1 is expected to be small. Also suppose r_0 is the bound of the HSS ranks of G and H in (4.4). Since G and H essentially correspond to 1D discretizations, r_0 is small. (For example, G and H may be w -banded or even tridiagonal, then $r_0 = 2w$.) For convenience, we count the cost of forming $A_{21}QC$ as in (4.15) where C can be approximated by a low-rank form. The costs of some basic operations involved in the algorithm are given in Table 4.1. The main operations for the factorization updates for all m frequencies are given in Table 4.2.

Table 4.1. Costs of some basic operations.

Operation	Cost
SuperDC of an order K HSS matrix with HSS rank r_0	$O(r_0^2 K \log^2 K)$
Multiplication of Q and a vector	$O(r_0(KM \log M + KM \log K))$
Multiplication of A_{21} and a vector	$O(N)$
Structured computation of $A_{21}QC$	$O(rr_0(KM \log M + KM \log K) + rN + rmN)$
Fully matrix-free HSS construction for each S_j	$O(r_1^2 N \log N)$
HSS ULV factorization for each S_j	$O(r_1^2 N)$

Thus, the total cost for m shifts is

$$\begin{aligned}
& O(K^2 + r_0^2 K + M^2 + r_0^2 M) + O(r_0^2 M \log^2 M) + O(r_0 KM) + O(r_0^2 MK \log^2 K) \\
& \quad + O(rr_1 r_0 (KM \log M + KM \log K) \log N + rr_1 N \log N) \\
& \quad + O(mr_1^2 N \log N) + O(mr_1^2 N \log N) + O(mr_1^2 N) \\
= & \underbrace{O(r_0^2 n \log n + rr_1 r_0 n \log^2 n)}_{\text{Precomputation}} + \underbrace{O(mr_1^2 N \log n)}_{m \text{ updates}},
\end{aligned}$$

Table 4.2. Costs of the main operations in the precomputation and factorization update.

	Operation	Cost
Precompute	HSS approximations to G and H	$O(K^2 + r_0^2 K + M^2 + r_0^2 M)$
	Eigendecomposition (4.5)	$O(r_0^2 M \log^2 M)$
	HSS approximation to (4.6) for all j	$O(r_0 K M)$
	Eigendecomposition (4.7) for all j	$O(r_0^2 M K \log^2 K)$
	Forming $A_{21} Q X$ for $O(r_1 \log N)$ vectors z	$O(r r_1 r_0 (K M \log M + K M \log K) \log N + r_1^2 N \log N)$
m updates	Form $(A_{21} Q X) Y^T$ for $O(r_1 \log N)$ vectors z	$O(m r r_1 N \log N)$
	Matrix-free HSS construction for all S_j	$O(m r_1^2 N \log N)$
	HSS ULV factorization for all S_j	$O(m r_1^2 N)$

where $O(r_1^2 N \log n)$ is the cost to update the factorization for each new shift. For the 2D case, we have $r_0 = O(1), r_1 = O(1), r = O(1), N = O(\sqrt{n})$. Thus, the precomputation cost is $O(n \log^2 n)$, and the cost for each new update is

$$O(r_1^2 N \log n) = O(\sqrt{n} \log n).$$

Such a cost is essentially roughly proportional to the number of boundary mesh points. Therefore, the factorization update is significantly faster than refactorizations.

Note that when we compute the product of Q and a vector, it needs to multiple $Q_h \otimes I$ and a vector, say, x . It is preferred to use the relationship

$$(Q_h \otimes I)x = \text{vec}(X Q_h^T),$$

where X is formed by reshaping x into a matrix through the so-called unvec operation. In this way, it naturally utilizes BLAS3 instead of BLAS2 operations.

We may also consider the 3D case, where the approximation of S_j should be based on \mathcal{H}^2 matrices [5] since r_1 would be too large if HSS approximation is used.

4.5 Numerical experiments

In this section, we shall test our new factorization update algorithm on some numerical examples to demonstrate its efficiencies. Throughout this section, the following notations are used

- **SMF**: unsymmetric structured multifrontal method in [25],
- **NEW**: our new multiple shifts factorization update algorithm,
- e_2 : the relative 2-norm solution error $\frac{\|x-x_{\text{true}}\|_2}{\|x_{\text{true}}\|_2}$,
- **SuperDC**: superfast divide-and-conquer eigensolver in Chapter 3

Example 1

Helmholtz equation with PML boundary condition

$$\Delta u + \frac{w^2}{c^2}u = f, \quad \Omega = [0, 1]^2,$$

where c and f are velocity field and force term respectively. The frequency w is set to be $w = 5Hz$. The following linear systems

$$(A - s_j I)x = b_j$$

are solved, where A is finite difference discretization matrix. Details of discretization and PML formulation can be found in [83]. The shifts s_j 's are randomly picked within a circle away from the real axis

$$s_j \stackrel{rand.}{\in} \{z \in \mathbb{C} : |z - c| \leq r\}, \quad |c| > r.$$

We fix $w = 5Hz$ and vary the mesh sizes $n = 256^2, 512^2, 1024^2, 2048^2, 4096^2$. The tolerance for the SuperDC eigensolver is $\text{tol} = 10^{-3}$, and the bound for the HSS ranks of all S_j is set

to be $r_1 = 20$. The flops counts for precomputation stage is reported in Table 4.3. To see the scaling, we also plotted the result in Figure 4.1. The flops count scaling is consistent with our estimate $O(n \log^2 n)$. Note that we did not include the timing here since n is not large enough to show the theoretical scaling. We also include the storage of NEW. In the precomputation stage, we need to store

- eigenmatrix Q , which consists of a sequence of structured eigenmatrices from SuperDC,
- intermediate matrix W , which is used for randomized matrix-free HSS construction of boundary Schur complements in the online computation stage.

Table 4.3. Precomputation flops count for NEW, $\tau_{ol}=10^{-3}$.

Matrix size	256^2	512^2	1024^2	2048^2	4096^2
Precomputation flops	3.0e11	1.8e12	1.0e13	5.2e13	2.8e14
Storage of Q	3.8e7	1.9e8	1.3e9	5.7e9	2.4e10
Storage of W	1.1e8	2.6e8	6.0e8	1.3e9	2.9e9

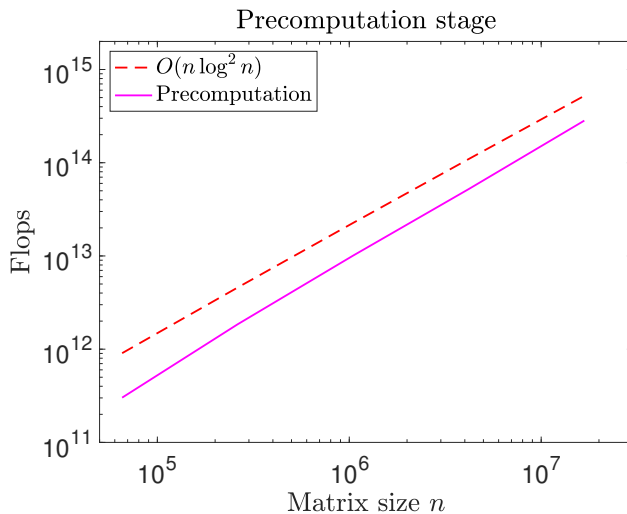


Figure 4.1. Precomputation flops count of NEW, $\tau_{ol}=10^{-3}$.

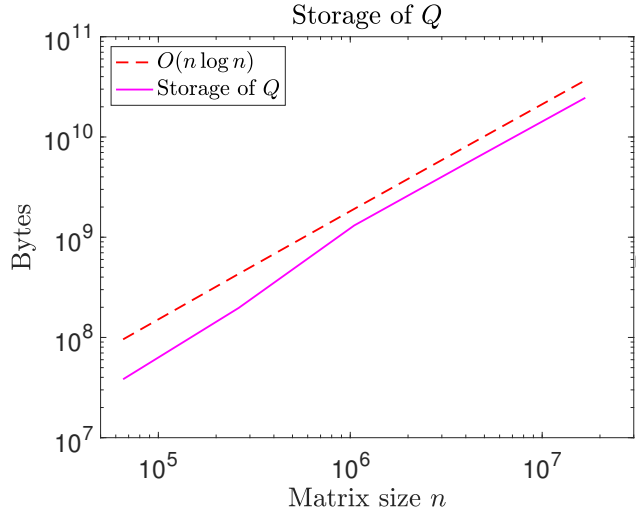


Figure 4.2. Storage of Q , $\text{tol}=10^{-3}$

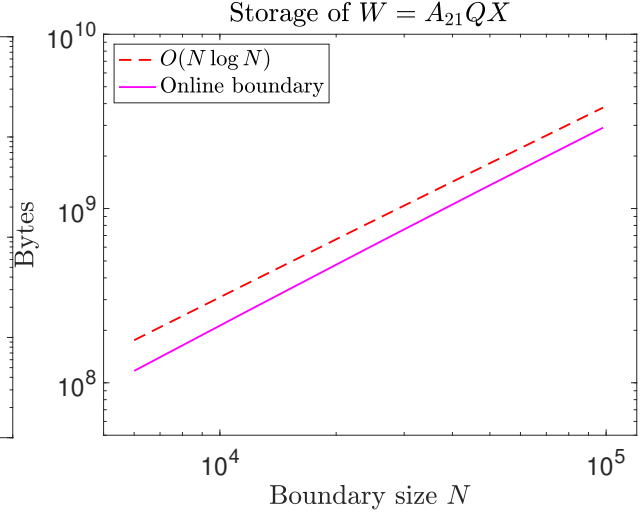


Figure 4.3. Storage of W , $\text{tol}=10^{-3}$

Next, we compare **NEW** and **SMF** in terms of factorization update flops counts and storage per shift in Table 4.4, Figure 4.4 and 4.5. Here for **SMF**, we refactorize $A - s_j I$ for each shift s_j . Note that the flops count and timing of **NEW** is significantly lower than that of **SMF**. For each shift, **NEW** is sublinear $O(\sqrt{n} \log^2 n)$ in flops and storage, since we restrict the update problem to the boundary.

Table 4.4. Factorization update flops count and storage, NEW v.s. SMF

Matrix size		256 ²	512 ²	1024 ²	2048 ²	4096 ²
Flops	SMF	1.4e10	6.3e10	2.6e11	1.1e12	5.0e12
	NEW	5.4e8	1.3e9	3.1e9	7.3e9	1.7e10
Time	SMF	1.6e0	5.6e0	2.5e1	3.5e2	4.0e3
	NEW	9.5e-1	1.7e0	3.9e0	8.9e0	2.3e1
Storage	SMF	3.0e7	1.6e8	7.8e8	4.0e9	1.9e10
	NEW	1.9e7	4.0e7	8.2e7	1.7e8	3.5e8

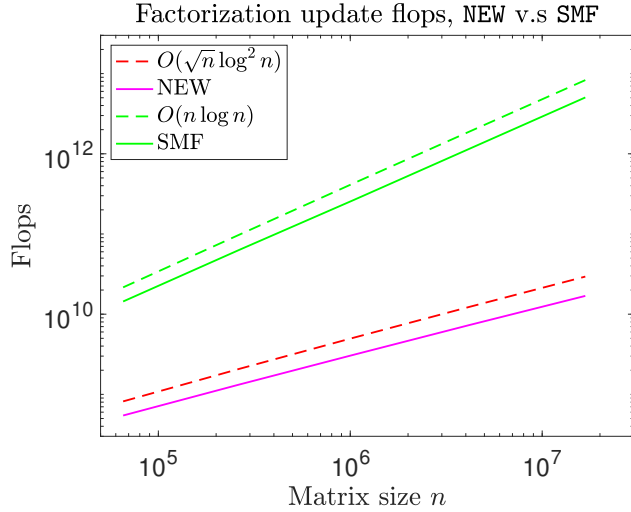


Figure 4.4. Factorization update flops

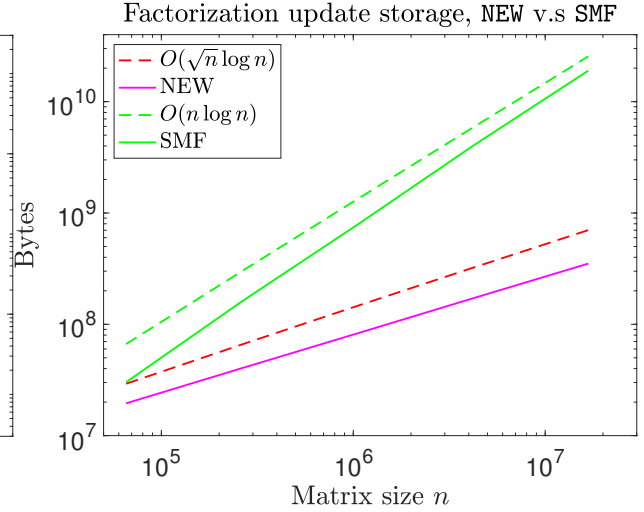


Figure 4.5. Factorization update storage

We then inspect the solution stage in terms of flops count and accuracy. We report the results in Table 4.5 and Figure 4.6. Note that the solution stage flops count for NEW is roughly linear with respect to the matrix size, which is consistent with our estimate. Although NEW costs approximately 3 times more than SMF here, the extra cost is almost negligible compared

to the gainz in the factorization update stage. Therefore, the overall performance of **NEW** is still way superior than **SMF** refactorization. Furthermore, the solution error of **NEW** is one or two digits better than **SMF**. This may be due to the orthogonal eigenmatrix and that each update is restricted to the smaller boundary problem.

Table 4.5. Solution stage, **SMF** v.s. **NEW**

Matrix size		256 ²	512 ²	1024 ²	2048 ²	4096 ²
Flops	SMF	1.5e8	6.2e8	2.5e9	1.0e10	4.3e10
	NEW	5.1e8	2.7e9	1.3e10	6.1e10	3.0e11
e_2	SMF	2.1e-5	1.5e-4	3.9e-4	1.2e-3	3.4e-3
	NEW	3.9e-4	1.1e-4	4.2e-5	4.2e-5	4.9e-5

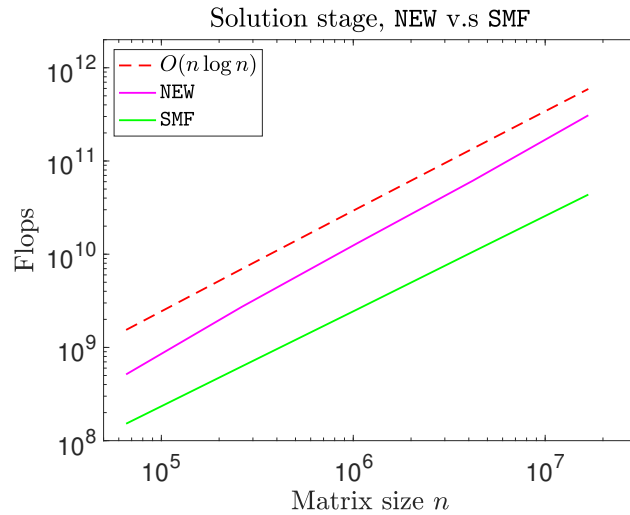


Figure 4.6. Solution flops, **SMF** v.s. **NEW**

4.6 Conclusions

We have considered a challenging problem of performing factorization update for diagonally-shifted sparse discretized matrices. A series of structured matrix techniques is used to first

compute a fast partial factorization of A . The partial factorization needs no cost to be updated for $A - s_j I$. For different s_j , we then only need to update the structured approximation of the Schur complement and perform a structured factorizations. The main operation in the structured approximation is the multiplication of the Schur complement with vectors. We assemble these multiplications corresponding to all s_j so as to perform fast structured matrix-matrix multiplications. We have shown that the majority of the computations can be done in a precomputation stage. For Helmholtz equations with certain coefficients and discretizations, it is superfast to perform the update for each new shift, and the cost is essentially roughly proportional to the number of boundary mesh points.

REFERENCES

- [1] X. Sun and N. P. Pitsianis, “A matrix version of the fast multipole method,” *Siam Review*, vol. 43, no. 2, pp. 289–300, 2001.
- [2] V. Rokhlin, “Rapid solution of integral equations of scattering theory in two dimensions,” *Journal of Computational Physics*, vol. 86, no. 2, pp. 414–439, 1990.
- [3] L. Greengard and V. Rokhlin, “A fast algorithm for particle simulations,” *Journal of computational physics*, vol. 73, no. 2, pp. 325–348, 1987.
- [4] L. Greengard and V. Rokhlin, “On the efficient implementation of the fast multipole algorithm,” YALE UNIV NEW HAVEN CT DEPT OF COMPUTER SCIENCE, Tech. Rep., 1988.
- [5] W. Hackbusch and S. Börm, “Data-sparse approximation by adaptive \mathcal{H}^2 -matrices,” *Computing*, vol. 69, no. 1, pp. 1–35, 2002.
- [6] W. Hackbusch, “A sparse matrix arithmetic based on \mathcal{H} -matrices. part i: Introduction to \mathcal{H} -matrices,” *Computing*, vol. 62, no. 2, pp. 89–108, 1999.
- [7] S. Börm, L. Grasedyck, and W. Hackbusch, “Introduction to hierarchical matrices with applications,” *Engineering analysis with boundary elements*, vol. 27, no. 5, pp. 405–422, 2003.
- [8] S. Börm, L. Grasedyck, and W. Hackbusch, “Hierarchical matrices,” *Lecture notes*, vol. 21, p. 2003, 2003.
- [9] Y. Eidelman, I. Gohberg, and V. Olshevsky, “The QR iteration method for hermitian quasiseparable matrices of an arbitrary order,” *Linear Algebra and its Applications*, vol. 404, pp. 305–324, 2005.
- [10] Y. Eidelman, I. Gohberg, and I. Haimovici, *Separable type representations of matrices and fast algorithms*. Springer, 2014.
- [11] S. Wang, X. S. Li, J. Xia, Y. Situ, and M. V. De Hoop, “Efficient scalable algorithms for solving dense linear systems with hierarchically semiseparable structures,” *SIAM Journal on Scientific Computing*, vol. 35, no. 6, pp. C519–C544, 2013.
- [12] X. Liu, J. Xia, and M. V. De Hoop, “Parallel randomized and matrix-free direct solvers for large structured dense linear systems,” *SIAM Journal on Scientific Computing*, vol. 38, no. 5, S508–S538, 2016.

- [13] P.-G. Martinsson, “A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix,” *SIAM Journal on Matrix Analysis and Applications*, vol. 32, no. 4, pp. 1251–1274, 2011.
- [14] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, “Fast algorithms for hierarchically semiseparable matrices,” *Numerical Linear Algebra with Applications*, vol. 17, no. 6, pp. 953–976, 2010.
- [15] P. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J.-Y. l’Excellent, and C. Weisbecker, “Improving multifrontal methods by means of block low-rank representations,” *SIAM Journal on Scientific Computing*, vol. 37, no. 3, A1451–A1474, 2015.
- [16] M. Gu and S. C. Eisenstat, “A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem,” *SIAM Journal on Matrix Analysis and Applications*, vol. 15, no. 4, pp. 1266–1276, 1994.
- [17] J. J. Cuppen, “A divide and conquer method for the symmetric tridiagonal eigenproblem,” *Numerische Mathematik*, vol. 36, no. 2, pp. 177–195, 1980.
- [18] J. J. Dongarra and D. C. Sorensen, “A fully parallel algorithm for the symmetric eigenvalue problem,” *SIAM Journal on Scientific and Statistical Computing*, vol. 8, no. 2, s139–s154, 1987.
- [19] M. Gu and S. C. Eisenstat, “A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem,” *SIAM Journal on Matrix Analysis and Applications*, vol. 16, no. 1, pp. 172–191, 1995.
- [20] M. Gu and S. C. Eisenstat, “A divide-and-conquer algorithm for the bidiagonal svd,” *SIAM Journal on Matrix Analysis and Applications*, vol. 16, no. 1, pp. 79–92, 1995.
- [21] E. R. Jessup and D. C. Sorensen, “A parallel algorithm for computing the singular value decomposition of a matrix,” *Siam Journal on Matrix Analysis and Applications*, vol. 15, no. 2, pp. 530–548, 1994.
- [22] I. Duff, “A multifrontal approach for solving sparse linear equations,” in *Numerical Methods*, Springer, 1983, pp. 87–98.
- [23] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. Liu, “A supernodal approach to sparse partial pivoting,” *SIAM Journal on Matrix Analysis and Applications*, vol. 20, no. 3, pp. 720–755, 1999.
- [24] J. W. Liu, “The multifrontal method for sparse matrix solution: Theory and practice,” *SIAM review*, vol. 34, no. 1, pp. 82–109, 1992.

- [25] J. Xia, “Efficient structured multifrontal factorization for general large sparse matrices,” *SIAM Journal on Scientific Computing*, vol. 35, no. 2, A832–A860, 2013.
- [26] J. Xia, “Randomized sparse direct solvers,” *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 1, pp. 197–227, 2013.
- [27] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, “Superfast multifrontal method for large structured linear systems of equations,” *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 3, pp. 1382–1411, 2010.
- [28] S. Wang, V. Maarten, and J. Xia, “Acoustic inverse scattering via helmholtz operator factorization and optimization,” *Journal of Computational Physics*, vol. 229, no. 22, pp. 8445–8462, 2010.
- [29] N. Halko, P.-G. Martinsson, and J. A. Tropp, “Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions,” *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.
- [30] Y. Xi, J. Xia, S. Cauley, and V. Balakrishnan, “Superfast and stable structured solvers for toeplitz least squares via randomized sampling,” *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 1, pp. 44–72, 2014.
- [31] S. Amini and A. Profit, “Multi-level fast multipole solution of the scattering problem,” *Engineering Analysis with Boundary Elements*, vol. 27, no. 5, pp. 547–564, 2003.
- [32] D. Cai and J. Xia, “A stable matrix version of the fast multipole method: Stabilization strategies and examples,” *ETNA-Electronic Transactions on Numerical Analysis*, vol. 54, 2020.
- [33] L. Greengard and J. Strain, “The fast gauss transform,” *SIAM Journal on Scientific and Statistical Computing*, vol. 12, no. 1, pp. 79–94, 1991.
- [34] X. Ou and J. Xia, “Superdc: Superfast divide-and-conquer eigenvalue decomposition for rank-structured matrices with improved stability,” *submitted to SIAM Journal on Scientific Computing*, 2021.
- [35] W. Fong and E. Darve, “The black-box fast multipole method,” *Journal of Computational Physics*, vol. 228, no. 23, pp. 8712–8725, 2009.
- [36] X. Ye, J. Xia, and L. Ying, “Analytical low-rank compression via proxy point selection,” *SIAM Journal on Matrix Analysis and Applications*, vol. 41, no. 3, pp. 1059–1085, 2020.

- [37] L. Ying, G. Biros, and D. Zorin, “A kernel-independent adaptive fast multipole algorithm in two and three dimensions,” *Journal of Computational Physics*, vol. 196, no. 2, pp. 591–626, 2004.
- [38] E. Darve, “The fast multipole method: Numerical implementation,” *Journal of Computational Physics*, vol. 160, no. 1, pp. 195–240, 2000.
- [39] J. Vogel, J. Xia, S. Cauley, and V. Balakrishnan, “Superfast divide-and-conquer method and perturbation analysis for structured eigenvalue solutions,” *SIAM Journal on Scientific Computing*, vol. 38, no. 3, A1358–A1382, 2016.
- [40] Y. Xi and J. Xia, “On the stability of some hierarchical rank structured matrix algorithms,” *SIAM Journal on Matrix Analysis and Applications*, vol. 37, no. 3, pp. 1279–1303, 2016.
- [41] G. N. Watson, *A treatise on the theory of Bessel functions*. Cambridge university press, 1995.
- [42] M. Abramowitz, I. A. Stegun, and R. H. Romer, *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, 1988.
- [43] S. Amini and A. Profit, “Analysis of the truncation errors in the fast multipole method for scattering problems,” *Journal of Computational and Applied Mathematics*, vol. 115, no. 1-2, pp. 23–33, 2000.
- [44] W. Meng and L. Wang, “Bounds for truncation errors of graf’s and neumann’s addition theorems,” *Numerical Algorithms*, vol. 72, no. 1, pp. 91–106, 2016.
- [45] W. Meng, “Bound on global error of the fast multipole method for helmholtz equation in 2-d,” *arXiv preprint arXiv:1806.08512*, 2018.
- [46] N. J. Higham, *Accuracy and stability of numerical algorithms*. SIAM, 2002.
- [47] X. Liao, S. Li, L. Cheng, and M. Gu, “An improved divide-and-conquer algorithm for the banded matrices with narrow bandwidths,” *Computers & Mathematics with Applications*, vol. 71, no. 10, pp. 1933–1943, 2016.
- [48] P.-G. Martinsson, V. Rokhlin, and M. Tygert, “A fast algorithm for the inversion of general toeplitz matrices,” *Computers & Mathematics with Applications*, vol. 50, no. 5-6, pp. 741–752, 2005.
- [49] J. Xia, Y. Xi, and M. Gu, “A superfast structured solver for toeplitz linear systems via randomized sampling,” *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 3, pp. 837–858, 2012.

- [50] P.-G. Martinsson and V. Rokhlin, “A fast direct solver for boundary integral equations in two dimensions,” *Journal of Computational Physics*, vol. 205, no. 1, pp. 1–23, 2005.
- [51] J. Shen, Y. Wang, and J. Xia, “Fast structured direct spectral methods for differential equations with variable coefficients, i. the one-dimensional case,” *SIAM Journal on Scientific Computing*, vol. 38, no. 1, A28–A54, 2016.
- [52] A. Cortinovis, D. Kressner, and S. Massei, “Divide-and-conquer methods for functions of matrices with banded or hierarchical low-rank structure,” *SIAM Journal on Matrix Analysis and Applications*, vol. 43, no. 1, pp. 151–177, 2022.
- [53] J. Shen, T. Tang, and L.-L. Wang, *Spectral methods: algorithms, analysis and applications*. Springer Science & Business Media, 2011, vol. 41.
- [54] Y. Wang, “Fast structured spectral methods,” Ph.D. dissertation, Purdue University, 2017.
- [55] M. Tygert, “Recurrence relations and fast algorithms,” *Applied and Computational Harmonic Analysis*, vol. 28, no. 1, pp. 121–128, 2010.
- [56] G. H. Golub and J. H. Welsch, “Calculation of gauss quadrature rules,” *Mathematics of computation*, vol. 23, no. 106, pp. 221–230, 1969.
- [57] S. Chandrasekaran, M. Gu, and T. Pals, “A fast ULV decomposition solver for hierarchically semiseparable representations,” *SIAM Journal on Matrix Analysis and Applications*, vol. 28, no. 3, pp. 603–622, 2006.
- [58] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, X. Sun, A.-J. van der Veen, and D. White, “Some fast algorithms for sequentially semiseparable representations,” *SIAM Journal on Matrix Analysis and Applications*, vol. 27, no. 2, pp. 341–364, 2005.
- [59] R. Vandebril, M. Van Barel, and N. Mastronardi, *Matrix computations and semiseparable matrices: linear systems*. JHU Press, 2007, vol. 1.
- [60] S. Ambikasaran and E. Darve, “An $O(n \log n)$ fast direct solver for partial hierarchically semi-separable matrices,” *Journal of Scientific Computing*, vol. 57, no. 3, pp. 477–501, 2013.
- [61] S. Chandrasekaran and M. Gu, “A divide-and-conquer algorithm for the eigendecomposition of symmetric block-diagonal plus semiseparable matrices,” *Numerische Mathematik*, vol. 96, no. 4, pp. 723–731, 2004.
- [62] Y. Eidelman and I. Haimovici, “Divide and conquer method for eigenstructure of quasiseparable matrices using zeroes of rational matrix functions,” in *A Panorama of Modern Operator Theory and Related Topics*, Springer, 2012, pp. 299–328.

- [63] P. Arbenz, “Divide and conquer algorithms for the bandsymmetric eigenvalue problem,” *Parallel computing*, vol. 18, no. 10, pp. 1105–1128, 1992.
- [64] A. Šušnjara and D. Kressner, “A fast spectral divide-and-conquer method for banded matrices,” *Numerical Linear Algebra with Applications*, vol. 28, no. 4, e2365, 2021.
- [65] D. A. Bini, L. Gemignani, and V. Y. Pan, “Fast and stable QR eigenvalue algorithms for generalized companion matrices and secular equations,” *Numerische Mathematik*, vol. 100, no. 3, pp. 373–408, 2005.
- [66] S. Chandrasekaran, M. Gu, J. Xia, and J. Zhu, “A fast QR algorithm for companion matrices,” in *Recent advances in matrix and operator theory*, Springer, 2007, pp. 111–143.
- [67] M. Van Barel, R. Vandebril, P. Van Dooren, and K. Frederix, “Implicit double shift QR-algorithm for companion matrices,” *Numerische Mathematik*, vol. 116, no. 2, pp. 177–212, 2010.
- [68] P. Benner and T. Mach, “Computing all or some eigenvalues of symmetric \mathcal{H}_l -matrices,” *SIAM Journal on Scientific Computing*, vol. 34, no. 1, A485–A496, 2012.
- [69] D. Bini and V. Pan, “Parallel complexity of tridiagonal symmetric eigenvalue problem,” in *Proceedings of the second annual ACM-SIAM symposium on Discrete algorithms*, 1991, pp. 384–393.
- [70] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, *et al.*, *LAPACK Users’ guide*. SIAM, 1999.
- [71] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen, “Rank-one modification of the symmetric eigenproblem,” *Numerische Mathematik*, vol. 31, no. 1, pp. 31–48, 1978.
- [72] D. O’leary and G. Stewart, “Computing the eigenvalues and eigenvectors of symmetric arrowhead matrices,” *Journal of Computational Physics*, vol. 90, no. 2, pp. 497–505, 1990.
- [73] R.-C. Li, “Solving secular equations stably and efficiently,” CALIFORNIA UNIV BERKELEY DEPT OF ELECTRICAL ENGINEERING and COMPUTER SCIENCES, Tech. Rep., 1993.
- [74] G. H. Golub, “Some modified matrix eigenvalue problems,” *SIAM Review*, vol. 15, no. 2, pp. 318–334, 1973.
- [75] J. W. Demmel, *Applied numerical linear algebra*. SIAM, 1997.

- [76] Y. Xi, J. Xia, and R. Chan, “A fast randomized eigensolver with structured ldl factorization update,” *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 3, pp. 974–996, 2014.
- [77] J. Xia, “Fast direct solvers for structured linear systems of equations,” Ph.D. dissertation, 2006.
- [78] J. Varah, “The prolate matrix,” *Linear algebra and its applications*, vol. 187, pp. 269–278, 1993.
- [79] K. Löwner, “Über monotone matrixfunktionen,” *Mathematische Zeitschrift*, vol. 38, no. 1, pp. 177–216, 1934.
- [80] G. Pan, R. A. Phinney, and R. I. Odom, “Full-waveform inversion of plane-wave seismograms in stratified acoustic media: Theory and feasibility,” *Geophysics*, vol. 53, no. 1, pp. 21–31, 1988.
- [81] J. Virieux and S. Operto, “An overview of full-waveform inversion in exploration geophysics,” *Geophysics*, vol. 74, no. 6, WCC1–WCC26, 2009.
- [82] J. Xia, Z. Li, and X. Ye, “Effective matrix-free preconditioning for the augmented immersed interface method,” *Journal of Computational Physics*, vol. 303, pp. 295–312, 2015.
- [83] B. Engquist and L. Ying, “Sweeping preconditioner for the helmholtz equation: Moving perfectly matched layers,” *Multiscale Modeling & Simulation*, vol. 9, no. 2, pp. 686–710, 2011.
- [84] A. F. Nikiforov and V. B. Uvarov, *Special functions of mathematical physics*. Springer, 1988, vol. 205.
- [85] S. Ohnuki and W. C. Chew, “Truncation error analysis of multipole expansion,” *SIAM Journal on Scientific Computing*, vol. 25, no. 4, pp. 1293–1306, 2004.
- [86] S. Amini and A. Profit, “Analysis of a diagonal form of the fast multipole algorithm for scattering theory,” *BIT Numerical Mathematics*, vol. 39, no. 4, pp. 585–602, 1999.
- [87] E. Darve, “The fast multipole method i: Error analysis and asymptotic complexity,” *SIAM Journal on Numerical Analysis*, vol. 38, no. 1, pp. 98–128, 2000.
- [88] S. Wang, X. S. Li, F.-H. Rouet, J. Xia, and M. V. De Hoop, “A parallel geometric multifrontal solver using hierarchically semiseparable structure,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 42, no. 3, pp. 1–21, 2016.

- [89] S. Chandrasekaran, P. Dewilde, M. Gu, W. Lyons, and T. Pals, “A fast solver for hss representations via sparse matrices,” *SIAM Journal on Matrix Analysis and Applications*, vol. 29, no. 1, pp. 67–81, 2007.
- [90] S. Chandrasekaran, M. Gu, X. Sun, J. Xia, and J. Zhu, “A superfast algorithm for toeplitz systems of linear equations,” *SIAM Journal on Matrix Analysis and Applications*, vol. 29, no. 4, pp. 1247–1266, 2008.
- [91] J. Xia, “On the complexity of some hierarchical structured matrix algorithms,” *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 2, pp. 388–410, 2012.
- [92] J. Xia, “Multi-layer hierarchical structures,” *CSIAM Transaction of Applied Mathematics*, vol. 2, pp. 263–296, 2021.
- [93] M. Gu, B. Parlett, D. Z.-Y. Ting, and J. Xia, “Low-rank update eigensolver for supercell band structure calculations,” *Journal of Computational Electronics*, vol. 1, no. 3, pp. 411–414, 2002.

VITA

Xiaofeng Ou (born in April, 1995) is a Ph.D candidate in the Department of Mathematics, Purdue University, West Lafayette, Indiana, United States. He was born in Foshan, Guangdong, China. He obtained his bachelor degree in Mathematics and Applied Mathematics from Sun Yat-sen University, Guangzhou, Guangdong, China in June 2017. He became a graduate student at Purdue University since August 2017. His academic advisor during his Ph.D career is Professor Jianlin Xia.