8-2018

Analytical Methods for Structured Matrix Computations

Xin Ye
Purdue University

ANALYTICAL METHODS FOR STRUCTURED MATRIX COMPUTATIONS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Xin Ye

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2018

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Jianlin Xia, Chair

Department of Mathematics

Dr. Venkataramanan Balakrishnan

School of Electrical and Computer Engineering

Dr. David F. Gleich

Department of Computer Science

Dr. Jie Shen

Department of Mathematics

Approved by:

Dr. David Goldberg

Associate Head for Graduate Studies

To my parents

who are always caring and supportive

ACKNOWLEDGMENTS

Writing a long acknowledgment is completely non-characteristic for me for the following two reasons: firstly, I have always hated to do non-scientific writing since I ever started to learn writing; secondly, I'm unwilling to express my own emotions except happy to anyone. But writing a thesis will happen only once in a life time, so why not?

The summer of 2013 surely marks a giant leap of my life, some major changes have happened at that moment: being a undergraduate student majoring in Economics versus pursuing a Ph.D. in Mathematics, having been living in my home city and speaking my native tongue for 22 years versus starting a new life in another side of the planet and learning a completely new language. I believe that anyone shares a similar experience with me will feel the same, the five-year journey will not be possible without the help and company of my mentors, colleagues, families and friends.

First of all, I would like to acknowledge all the fundings during my Ph.D. study. The department of Mathematics is continuously supporting everyone with teaching assistantship whenever we need it, seven semesters of research assistantship were provided by Prof. Jianlin Xia and Prof. Venkataramanan Balakrishnan in School of Electrical and Computer Engineering. The last year was supported by a dissertation fellowship from Purdue Research Foundation. I've also received some travel supports from Duke University, SIAM and IMA.

I would like to express my deep gratitude to Prof. Jianlin Xia, my Ph.D. advisor, for his guidance not only on mathematics and research but also many aspects of my life outside of academia. He led me into the world of numerical linear algebra and continues to encourage me whenever I feel hesitant, I'm always so inspired by his passionate and meticulous attitude towards his work.

I've always enjoyed taking courses with Prof. David Gleigh, Prof. Ahmed Sameh and Prof. Steven Bell, we also had many discussions on various topics outside of class which helped me broaden my view and make connections between different areas in Mathematics. Prof. Raymond H. Chan and Prof. Venkataramanan Balakrishnan collaborated with me and provided many helpful advises that eventually leads to my first major published work. I would also like to thank Prof. Jie Shen for being my thesis committee member.

My academic big brothers Yuanzhe Xi, Xiao Liu and Zixin Xin all played important roles especially during the first few months when I joined the group, I lost count of how many times I interrupted you and sought for help. For Yue Zhao, Jian Zhai, Ling Xu, Qinfeng Li and Difeng Cai, we all started our graduate study at the same time and we would often get together and chat about everything during the first two years we were here. Frankie Chan and Joan Ponce merit a special mention, they kept sharing news, gossips and internet memes with me which is the sole reason I could improve my spoken English so quickly, as for introducing you to hot pot, you are welcome.

Jiawei Zhou and Tianshuo Zhang are without doubt the two most important friends outside of the department, we've almost lived together for the entirety of our Ph.D. lives. They are good roommates, good cooks, good gamers and all these make our apartment at Beau Jardin 20–8 more like a home than only a place for sleep. Ruibo Wang, Xinyu Kong and I became friends when we were around 12, it's a miracle that we are only 80 miles from each other after traveling such a long distance from our hometown to midwest US. We were able to visit each other very often and later I attended their wedding as the best man. There are still many more names that I don't have enough space to list here, but I believe our friendships will be remembered and cherished forever.

Last, but definitely not the least, is my parents. They made me who I am today, a person with enough intelligence to know the world and, most importantly, a good

heart to treat others. The appreciation to them is beyond any words can express, I'm so happy I can share this memorable moment of my life with you.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Ye, Xin Ph.D., Purdue University, August 2018. Analytical Methods for Structured Matrix Computations. Major Professor: Jianlin Xia.

The design of fast algorithms is not only about achieving faster speeds but also about retaining the ability to control the error and numerical stability. This is crucial to the reliability of computed numerical solutions. This dissertation studies topics related to structured matrix computations with an emphasis on their numerical analysis aspects and algorithms. The methods discussed here are all based on rich analytical results that are mathematically justified. In chapter 2, we present a series of comprehensive error analyses to an analytical matrix compression method and it serves as a theoretical explanation of the proxy point method. These results are also important instructions on optimizing the performance. In chapter 3, we propose a non-Hermitian eigensolver by combining HSS matrix techniques with a contour-integral based method. Moreover, probabilistic analysis enables further acceleration of the method in addition to manipulating the HSS representation algebraically. An application of the HSS matrix is discussed in chapter 4 where we design a structured preconditioner for linear systems generated by AIIM. We improve the numerical stability for the matrix-free HSS construction process and make some additional modifications tailored to this particular problem.

1. INTRODUCTION

Matrix computation always serve as a core component of computational mathematics, as matrices can represent the discretization of a real world continuous problem. In the current "big data" age, there has been an urgent need for large scale computational tools to keep up with the rapid development of computation hardwares and the increasing amount of data collected.

This dissertation is built around computational methods for structured matrices which has been an active research area in numerical linear algebra and matrix computation in the last two decades. Generally speaking, the core idea behind this class of methods is the ability to explore low-rankness of certain matrices or submatrices. The discussion on this low rank feature is two-fold.

The theoretical aspect of why certain matrices admit low rank representations. This is mostly motivated by physical backgrounds governed by certain mathematical equations when performing real world simulations. The matrices being considered carry more information than just a list of numbers arranged in a rectangle manner. Depending on the type of problems in consideration, the analysis may require knowledges from partial differential equations (PDEs), integral equations, or matrix analysis tools. Being able to identify all the rank patterns is extremely important for designing effective solvers for a specific problem.

The computational aspect of how to utilize this property to produce fast and stable algorithms. Sophisticated algorithms are all based on the simple fact that when a dense or large sparse matrix is compressed into a low rank form, the computation tasks associated with it will generally have lower computational costs, memory, and storage usages. Various hierarchical matrix representations are developed, and they result in many fast and superfast solvers. In addition to achieving faster speeds, people also discuss their numerical stability and parallel performance.

Many effective and fast algorithms have been developed and successfully applied to a variety of problems. They meet the need for finding numerical solutions with limited resources. The situation now reaches a point where many different algorithms have similar computational efficiency, so in addition to getting lower cost, more efforts have been made to stabilize the algorithms and achieve better accuracy. This requires rigorous mathematical proofs. With these analytical results we know exactly what will happen numerically when our algorithms are applied and the computed solutions can then be more reliably used for other purposes.

Among the many different hierarchical representations such as $\mathcal{H}$, $\mathcal{H}^2$, fast multipole (FMM), sequentially semiseparable (SSS) and hierarchically off-diagonal low-rank (HODLR), we will work with hierarchically semiseparable (HSS) matrices in this work. Compared with its competitors, the HSS representation is noted to be a 1D structure. It thus has limited application when used alone, but this can be overcome by adding other layers of structures on top of HSS to obtain a more complex and general purpose structure such as multi-layer hierarchical (MHS) structures. With this one limitation comes several unique features. Firstly, HSS admits a ULV-type factorization which can be used for direct solution of linear systems. The solution process follows a binary tree associated with the HSS matrix hierarchically instead of column-by-column or block-by-block sequentially; secondly, all related algorithms have proven numerical stability which guarantees the quality of the computed solution.

We start with discussions in chapter 2 on low rank approximation of a matrix which is the foundation of any rank-structured techniques. As a key component of the HSS construction process, we rely heavily on the speed and error of the compression process. We will propose an analytical compression method based on contour integration and proxy point selection. All discussions on the algorithm design are accompanied by error analysis results. In particular, our analysis can serve as the missing theoretical explanation of the widely used proxy point method. The theo-

ries developed here are well align with heuristics observed by many previous related works.

In chapter 3, we develop a non-Hermitian eigensolver based on HSS algorithms and a contour-integral eigensolver. The core idea of a contour-integral eigensolver is the implicit use of a filter function to only maintain the desired spectrum information. We will first consider the filter design problem by examining the decay properties for different quadrature rules. Two acceleration techniques are applied. The first one is more straightforward and is done by HSS factorization update. The second one is by noticing that in the quadsection stage of the eigensolver, a low accuracy representation already gives good enough estimation for the eigenvalue count. We also provide theoretical justification to this strategy by analyzing the probability for miscounting.

In chapter 4, we present an application to preconditioning the linear systems resulting from the augmented immersed interface method (AIIM). We bring several improvements to the matrix-free HSS construction algorithm tailored to this specific problem. The generic matrix-free HSS construction algorithm is also updated with much better numerical stability by completely avoiding the use of pseudoinverse.

Conclusion remarks and some future works are discussed in chapter 5.

2. ANALYTICAL COMPRESSION VIA PROXY POINT SELECTION AND CONTOUR INTEGRATION

The content in this chapter is from a preprint under preparation [124]: X. Ye, J. Xia, and L. Ying, *Analytical compression via proxy point selection and contour integration*, to be submitted, 2018.

2.1 Introduction

In this chapter, we focus on the low-rank approximations of kernel matrices: those generated by a kernel function $k(x, y)$ and two sets of points $X = \{x_j\}_{j=1}^m$ and $Y = \{y_j\}_{j=1}^n$. This type of problem frequently arises in a wide range of computational tasks, e.g., in numerical solution to PDEs where an operator or the Green's function acts as the kernel and the points are from the discretization method of the domain, or in machine learning where there are a covariance function and a set of data points. Especially when rank-structured techniques such as $\mathcal{H}$ [42], $\mathcal{H}^2$ [43] and HSS [12,117] are applied, low-rank approximations are computed throughout the structured matrix construction, whether or not the chosen technique can successfully solve the problem relies heavily on the quality and efficiency of such low-rank approximation.

As is proven by the famous Eckhart-Young theorem, the best low-rank approximation is given by the truncated singular value decomposition (SVD), but too much computation cost has to be paid for this optimal solution. The SVD approach can be categorized into the *algebraic compression method*, other popular algorithms also falls into this category are strong rank-revealing QR factorization (SRRQR) [39], strong rank-revealing LU factorization (SRRLU) [83], interpolative decomposition (ID) [15] and spectrum-revealing Cholesky factorization (SRCH) [121]. These generic methods work well for all matrices but the costs are all greater or equal to $\mathcal{O}(mnr)$ with r be-

ing the target rank, when integrated into the structured matrix construction process they all result to at least quadratic costs with respect to the problem size thus are still not suitable for modern large-scale applications.

A remedy to the algebraic methods is the introduction of randomization. The most general such approach [26] is done by randomized sampling on the rows or columns of the matrix with certain weight, the selected submatrix can capture the majority of the actions in the original matrix with very low failure probability; when a fast matrix-vector product scheme is available, column or row spaces information can be also extracted from the product in a matrix-free fashion [37, 46, 78, 121]. With the assumption that each entries of the matrix can be accessed in $\mathcal{O}(1)$ cost or the mat-vec can be conducted in less than $\mathcal{O}(mn)$ cost, randomized methods generally bring down the compression cost to at most $\mathcal{O}((m+n)r)$ with some possible polylogarithmic term in m or n.

Not like the purely algebraic methods which focus on the desired row or column subspaces, a series of *analytical compression methods* is developed based on entry-wise approximation with the appreciation of certain form of the kernel function $k(x, y)$. The low-rank approximation of the matrix is a direct result of a degenerate approximation to the function that separates the variables:

$$k(x, y) \approx \sum_{j=1}^{r} \alpha_j \phi_j(x) \psi_j(y) \tag{2.1}$$

where $\{\phi_j\}$ and $\{\psi_j\}$ are appropriate basis functions and $\{\alpha_j\}$ are coefficients independent of x or y. The fast multipole method (FMM) [36] falls into this category where the degenerate approximation is obtained by multipole expansion, others forms of approximation can also be generated by using spherical harmonic basis functions [99], Fourier transform with Poisson's formula [2, 76], Laplace transform with Cauchy integral formula [60], Chebyshev interpolation [24] and Taylor expansion [11], various other polynomial basis functions are also discussed in [86].

One typical downside of the purely analytical approach is that usually the basis $\{\phi_j\}$ and $\{\psi_j\}$ are of different form compared to the kernel function $k(x, y)$, as a result,

none of the factors in the low-rank approximation will inherit the same structure of the original matrix while a structure preserving compression is proven to be beneficial to structured matrix technique, e.g. in [73, 120].

Recently, a new class of compression methods called *proxy point methods* tailored for kernel matrices have been developed and are receiving more and more attentions due to their exceptional simplicity and reliability, some discussion can be found in [24, 79, 122, 126, 127] when dealing with different kernels and different geometries of points. While the methods vary from one to another, they all share the same basic idea and can be summarized in algorithm 1, all the details are omitted here and will be discussed later in section 2.3. Note that an explicit degenerate form eq. (2.1) is not needed and the algorithm only requires kernel function evaluation when forming the matrix $K^{X,Z}$ in line 2, this feature enables some classical methods to be applicable to more general cases, examples include the recursive skeletonization [47, 79, 82] and kernel independent FMM [80, 126, 127].

Algorithm 1 *Basic proxy point method*

Input: $k(x, y)$, X, Y

Output: low-rank approximation $K^{X,Y} \approx UV$

1: Pick *proxy surface* Γ and a set of *proxy points* $Z \subset \Gamma$

2: Form and compress $K^{X,Z} \approx \hat{U}\hat{V}$

3: Set $U = \hat{U}$ and obtain V according to $\hat{V}$

Compared to their great successes on various applications in practice, results from the theoretical aspect are still lacking in the literature. Potential theory [55, Chapter 6] can be used to explain the choice of proxy surface Γ in line 1 when dealing with PDE kernels ($k(x, y)$ is the fundamental solution of some PDE), little is known about the approximation error introduced by picking proxy points Z and then combining with an algebraic compression in line 2.

Our work here focuses on the kernel

$$k(x, y) = \frac{1}{(x - y)^d} \tag{2.2}$$

where $d \in \mathbb{Z}^+$, $x, y \in \mathbb{C}$ and $x \neq y$, we will propose our own variant of the analytical method for this problem based on contour integration and conduct a systematical analysis on different types of approximation error. Our main contribution contains two major pieces:

(i) In section 2.2, we rationalize the use of proxy points Z and present an explicit approximation error bound associated with it, then we discuss the optimal choice for Z based on the analysis results.

(ii) In section 2.3, the above technique is then incorporated into a hybrid method like algorithm 1 and the corresponding compound error analysis is also presented.

Some notations we use frequently in the paper are listed here:

- Intergers m, n, N denote the sizes of sets X, Y and Z, respectively;
- $\mathcal{C}(a;\gamma)$, $\mathcal{D}(a;\gamma)$ and $\bar{\mathcal{D}}(a;\gamma)$ denote the circle, open disk and closed disk with center $a \in \mathbb{C}$ and radius $\gamma > 0$, respectively;
- $\mathcal{A}(a;\gamma_1,\gamma_2) = \{z : \gamma_1 < |z-a| < \gamma_2\}$ $(0 < \gamma_1 < \gamma_2)$ is a open annulus region;
- for a function in small letter, the corresponding capital letter means a matrix generated by this function, e.g., for the kernel function $k(x,y)$ and two sets of points $X = \{x_j\}_{j=1}^m$ and $Y = \{y_j\}_{j=1}^n$, $K^{X,Y}$ represents a matrix of size $m \times n$ where the (i,j) entry is $k(x_i, y_j)$;
- the operation denotes entry-wise multiplication of two matrices.

2.2 Analytical compression

In this section, we will introduce the analytical approximation method and the main analysis results.

We will follow the convention to call set $X = \{x_j\}_{j=1}^m$ the *target points* and $Y = \{y_j\}_{j=1}^n$ the *source points*. Note that the kernel function eq. (2.2) is translation invariant, i.e., $k(x-z, y-z) = k(x, y)$ for any $x \neq y$ and $z \in \mathbb{C}$, the target points can be moved to be clustered around the origin without loss of any generality. To be more specific, unless otherwise stated in the context, we will always assume $X \subset \mathcal{D}(0; \gamma_1)$ and $Y \subset \mathcal{A}(0; \gamma_2, \gamma_3)$ where the radii satisfy $0 < \gamma_1 < \gamma_2 < \gamma_3$.

2.2.1 The derivation via Cauchy integral formula

Let $x \in X$ and $y \in Y$ be any two points, draw a Jordan curve Γ that encloses x while excludes y and let $R > 0$ be large enough so that the circle $\mathcal{C}(0; R)$ encloses both Γ and y; see fig. 2.1.

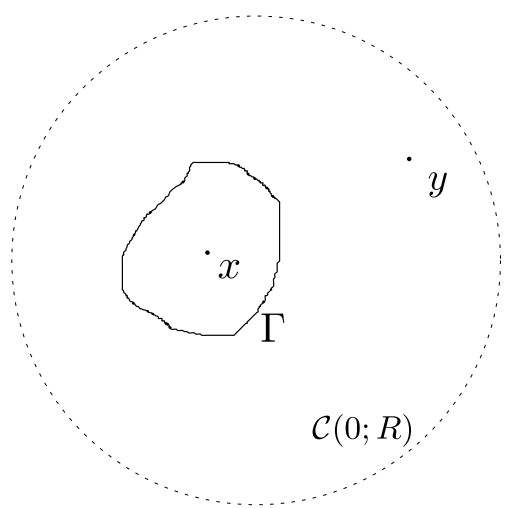


Figure 2.1. Points x and y, curve Γ and circle $\mathcal{C}(0; R)$

Define the domain Ω_R to be the open region inside of $\mathcal{C}(0; R)$ and outside of Γ, then the boundary $\partial\Omega_R$ consists of $\mathcal{C}(0; R)$ and $-\Gamma$ (curve Γ with negative direction). Now consider the function $f(z) := k(x, z)$ on the bounded closed domain $\bar{\Omega}_R = \Omega_R \cup \partial\Omega_R$,

the only singularity of $f(z)$ is at $z = x \notin \bar{\Omega}_R$, thus it's analytic (or holomorphic) on $\bar{\Omega}_R$ and by the Cauchy integral formula [98] we get

$$\begin{aligned} k(x,y) = & f(y) = \frac{1}{2\pi i} \oint_{\partial\Omega_R} \frac{f(z)}{z-y} \mathrm{d}z \\ & = \frac{1}{2\pi i} \oint_{\mathcal{C}(0;R)} \frac{k(x,z)}{z-y} \mathrm{d}z - \frac{1}{2\pi i} \oint_{\Gamma} \frac{k(x,z)}{z-y} \mathrm{d}z, \end{aligned} \tag{2.3}$$

note that

$$\oint_{\mathcal{C}(0;R)} \frac{k(x,z)}{z-y} \mathrm{d}z \ \leq 2\pi R \cdot \max_{z \in \mathcal{C}(0;R)} \ \frac{1}{(z-y)(x-z)^d} \ \leq \frac{2\pi R}{(R-|x|)^d (R-|y|)}$$

and the right-hand side goes to zero when $R \to \infty$, thus

$$\lim_{R \to \infty} \oint_{\mathcal{C}(0;R)} \frac{k(x,z)}{z-y} \mathrm{d}z = 0.$$

Since eq. (2.3) holds for any large enough R as long as $\mathcal{C}(0;R)$ encloses Γ and y, by taking the limit on it for $R \to \infty$, the first term vanishes and (2.3) becomes the key formula for our entry-wise approximation

$$k(x,y) = \frac{1}{2\pi i} \oint_{\Gamma} \frac{k(x,z)}{y-z} \mathrm{d}z. \tag{2.4}$$

Remark 2.2.1 *This result is very similar to the Cauchy integral formula except that the point y under consideration is outside of the contour Γ in the integral and it might be possible to generalize the original formula to obtain a more general conclusion that implies what we get here, but, nonetheless, we will give our own short proof since it's not a trivial result.*

To approximate the above contour integration, pick an N-point quadrature rule $\{(z_j, \omega_j)\}_{j=1}^N$ on Γ where $Z = \{z_j\}_{j=1}^N \subset \Gamma$ is the set of quadrature points and $\{\omega_j\}_{j=1}^N$ is the corresponding quadrature weight. Denoted by $k_N(x,y)$ the approximation induced by such a numerical integration, then

$$k_N(x,y) = \frac{1}{2\pi i} \sum_{j=1}^N \omega_j \frac{k(x,z_j)}{y-z_j} = \sum_{j=1}^N k(x,z_j) \frac{\omega_j}{2\pi i (y-z_j)},$$

define $w_N(z_j, y) = \omega_j / 2\pi i(y - z_j)$ (since ω_j can be viewed as a function in z_j and N) and we get

$$k_N(x, y) = \sum_{j=1}^{N} k(x, z_j) w_N(z_j, y) = K^{x,Z} W_N^{Z,y}. \tag{2.5}$$

As shown above, we have separated the two variables x and y in $k(x, y)$ by the approximation and it acts just like the degenerate approximation eq. (2.1) but with one additional property: the basis functions $\{\phi_j\}$ in this case is $\{k(\cdot, z_j)\}$, the exact same kernel with the original entry $k(x, y)$ that is being approximated. This alone already gives an intuitive explanation of the use of proxy surface: the interaction between x and y can be well approximated by interaction between x and some proxy points Z, the weight function w_N is used to make them equivalent (in terms of computing potential, this is to place equivalent charges on the proxy surface). A pictorial illustration is shown in fig. 2.2.

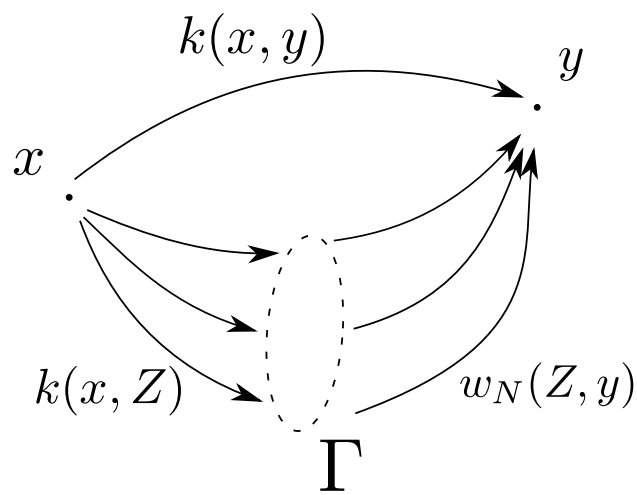


Figure 2.2. The upper path represents the original interaction $k(x, y)$, the lower path represents the approximation $k_N(x, y)$ through the proxy surface Γ.

2.2.2 Approximation error analysis

Although the approximation eq. (2.5) holds for any proxy surface Γ satisfying the given conditions and for any quadrature rule, we still need to make specific choice in order to obtain a more practical error bound. Firstly, we will assume the proxy surface $\Gamma = \mathcal{C}(0;\gamma)$ to be a circle, this is the most popular choice in related works and is also consistent with our assumption on the geometry of X and Y at the start of section 2.2. For now, the proxy surface Γ is only assumed to be between X and Y, i.e., $\gamma_1 < \gamma < \gamma_2$, we will come back to discuss more on this matter later in this section. Secondly, the quadrature is chosen to be the composite trapezoidal rule, in particular we pick

$$z_j = \gamma \exp\left(\frac{2\pi i j}{N}\right) \text{and } \omega_j = \frac{2\pi i z_j}{N}, \quad j = 1, 2, \ldots, N. \tag{2.6}$$

This choice can be justified by noting that trapezoidal rule converges exponentially fast if applied to a periodic integrand [103], in our case it is a contour integral, our results later also align with this knowledge. Moreover, if no specific direction is more important that others, trapezoidal rule performs uniformly well on all directions of the complex plane $\mathbb{C}$; some related discussions on this issue can be found in [53, 123].

As a result of the above assumptions, the weight function w_N in eq. (2.5) becomes

$$w_N(z, y) = \frac{1}{N}\frac{z}{y - z}, \quad y \neq z,$$

also we define $g(z) = 1/(z - 1)$ as a complex function for $z \in \mathbb{C}$ and $z \neq 1$ or a real function for $z \in \mathbb{R}$ and $z > 1$ depending on the context. In the following proposition, we derive an analytical expression for the low-rank approximation and it's error.

Proposition 2.2.1 *For $x \neq 0$, let $k_N(x, y)$ be defined in eq. (2.5) and quadrature is the composite trapezoidal rule eq. (2.6), then*

$$k_N(x, y) = k(x, y)\left[1 + g\left((y/\gamma)^N\right) + \sum_{j=0}^{d-1} \frac{(y - x)^j}{j!} \frac{\mathrm{d}^j}{\mathrm{d}x^j} g\left((\gamma/x)^N\right)\right] \tag{2.7}$$

and the relative approximation error is

$$\begin{aligned}\varepsilon_N(x,y) &:= \frac{k_N(x,y)-k(x,y)}{k(x,y)} \\ &= g\left((y/\gamma)^N\right) + \sum_{j=0}^{d-1} \frac{(y-x)^j}{j!}\frac{\mathrm{d}^j}{\mathrm{d}x^j} g\left((\gamma/x)^N\right)\end{aligned} \tag{2.8}$$

First we need the following lemma in the proof of the above results.

Lemma 2.2.1 *Let* $\{z_j\}_{j=1}^N$ *be the points defined in eq.* (2.6)*, then*

$$\sum_{j=1}^{N} \frac{z_j}{z-z_j} = \frac{N\gamma^N}{z^N-\gamma^N} \tag{2.9}$$

holds for all $z \in \mathbb{C} - \{z_j\}_{j=1}^N$.

Proof We will state without proving the following result that is used in this proof:

$$\sum_{j=1}^{N} z_j^p = \begin{cases} N\gamma^p, & \text{if } p \text{ is a multiple of } N, \\ 0, & \text{otherwise.} \end{cases} \tag{2.10}$$

Suppose $|z| < \gamma$ so $|z/z_j| < 1$ for any $j = 1, 2, \ldots, N$, then

$$\begin{aligned}\sum_{j=1}^{N} \frac{z_j}{z-z_j} &= -\sum_{j=1}^{N} \frac{1}{1-z/z_j} = -\sum_{j=1}^{N}\sum_{l=0}^{\infty}\left(\frac{z}{z_j}\right)^l = -\sum_{l=0}^{\infty} z^l \sum_{j=1}^{N} z_j^{-l} \\ &= -\sum_{s=0}^{\infty} z^{sN} N\gamma^{-sN} \quad \text{(apply eq. (2.10), only } l = sN \text{ terms are left)} \\ &= -N\sum_{s=0}^{\infty}\left(\frac{z^N}{\gamma^N}\right)^s = -\frac{N}{1-z^N/\gamma^N} = \frac{N\gamma^N}{z^N-\gamma^N}.\end{aligned}$$

Similarly we can also prove it when $|z| > \gamma$ using the fact that $|z_j/z| < 1$. At last, since both sides of eq. (2.9) are analytic functions on $\mathbb{C} - \{z_j\}_{j=1}^N$ and they agree on $|z| \neq \gamma$, by continuity, they must also agree on $|z| = \gamma$ and $z \notin \{z_j\}_{j=1}^N$ which completes the proof. ■

Proof [Proof of proposition 2.2.1] We prove this proposition by induction on d. Consider the case when $d = 1$, substituting eq. (2.6) into eq. (2.5) yields

$$
\begin{aligned}
k_N(x,y) &= \sum_{j=1}^{N} k(x,z_j) w_N(z_j,y) = \frac{1}{N}\sum_{j=1}^{N} \frac{z_j}{(x-z_j)(y-z_j)} \\
&= \frac{1}{N(x-y)} \sum_{j=1}^{N} \frac{x-y}{(x-z_j)(y-z_j)} z_j = \frac{1}{N(x-y)} \sum_{j=1}^{N} \frac{(x-z_j)-(y-z_j)}{(x-z_j)(y-z_j)} z_j \\
&= \frac{1}{N(x-y)} \quad \sum_{j=1}^{N} \frac{z_j}{y-z_j} - \sum_{j=1}^{N} \frac{z_j}{x-z_j} \Bigg) \\
&= \frac{1}{N(x-y)} \left(\frac{N\gamma^N}{y^N-\gamma^N} - \frac{N\gamma^N}{x^N-\gamma^N} \right) \Bigg(\quad \text{(apply theorem 2.2.1)} \\
&= \frac{1}{x-y} \left(\frac{\gamma^N}{y^N-\gamma^N} - \frac{\gamma^N}{x^N-\gamma^N} \right) \Bigg(\\
&= \frac{1}{x-y} \left[1 + g\left((y/\gamma)^N \right) + g\left((\gamma/x)^N \right) \right],
\end{aligned}
$$

thus eq. (2.7) holds for $d = 1$.

Now suppose eq. (2.7) is true for $d = l$ for some $l \in \mathbb{Z}$ and $l \geq 1$, equating eq. (2.5) and eq. (2.7) and plug in $k(x,y)$ explicitly we get

$$
\sum_{j=1}^{N} \frac{1}{(x-z_j)^l} w_N(z_j,y) = \frac{1}{(x-y)^l} \left[1 + g\left((y/\gamma)^N \right) + \sum_{j=0}^{l-1} \frac{(y-x)^j}{j!} \frac{\mathrm{d}^j}{\mathrm{d}x^j} g\left((\gamma/x)^N \right) \right] \Bigg(
$$

We are to take the derivative on both sides of eq. (2.7) with respect to x, note that

$$
\frac{\mathrm{d}}{\mathrm{d}x} \frac{1}{(x-z)^l} = -\frac{l}{(x-z)^{l+1}}
$$

for $z = y$ or $z = z_j$ $(j = 1, 2, \cdots, N)$, so

$$
\frac{\mathrm{d}}{\mathrm{d}x}\mathrm{LHS} = -l \quad \sum_{j=1}^{N} \frac{1}{(x-z_j)^{l+1}} w_N(z_j,y) \Bigg) \Bigg(
$$

and

$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}x}\mathrm{RHS} &= \frac{-l}{(x-y)^{l+1}} \left[1 + g\left((y/\gamma)^N \right) + \sum_{j=0}^{l-1} \frac{(y-x)^j}{j!} \frac{\mathrm{d}^j}{\mathrm{d}x^j} g\left((\gamma/x)^N \right) \right] \Bigg(\\
&+ \frac{1}{(x-y)^l} \left[\sum_{j=0}^{l-1} \frac{(y-x)^j}{j!} \frac{\mathrm{d}^{j+1}}{\mathrm{d}x^{j+1}} g\left((\gamma/x)^N \right) - \sum_{j=1}^{l-1} \frac{(y-x)^{j-1}}{(j-1)!} \frac{\mathrm{d}^j}{\mathrm{d}x^j} g\left((\gamma/x)^N \right) \right] \Bigg(
\end{aligned}
$$

$$= \frac{-l}{(x-y)^{l+1}} \left[1 + g\left((y/\gamma)^N\right) + \sum_{j=0}^{l-1} \frac{(y-x)^j}{j!} \frac{\mathrm{d}^j}{\mathrm{d}x^j} g\left((\gamma/x)^N\right) \right]$$

$$+ \frac{1}{(x-y)^l} \frac{(y-x)^{l-1}}{(l-1)!} \frac{\mathrm{d}^l}{\mathrm{d}x^l} g\left((\gamma/x)^N\right) \quad \text{(all terms cancel except for } j = l-1\text{)}$$

$$= \frac{-l}{(x-y)^{l+1}} \left[1 + g\left((y/\gamma)^N\right) + \sum_{j=0}^{l} \frac{(y-x)^j}{j!} \frac{\mathrm{d}^j}{\mathrm{d}x^j} g\left((\gamma/x)^N\right) \right],$$

the above two equations yield

$$\sum_{j=1}^{N} \frac{1}{(x-z_j)^{l+1}} w_N(z_j, y) = \frac{1}{(x-y)^{l+1}} \left[1 + g\left((y/\gamma)^N\right) + \sum_{j=0}^{l} \frac{(y-x)^j}{j!} \frac{\mathrm{d}^j}{\mathrm{d}x^j} g\left((\gamma/x)^N\right) \right],$$

thus eq. (2.7) holds for $d = l + 1$.

By induction, eq. (2.7) is proved for any $d \geq 1$. The relative approximation error eq. (2.8) can be easily derived using this result. ■

With this analytical expression we can give the first upper bound for the approximation error with this contour integral technique.

Theorem 2.2.2 *With all assumptions in proposition 2.2.1, there exists an $N_1 > 0$ such that for any $N > N_1$, the approximation error eq.* (2.8) *is bounded by*

$$|\varepsilon_N(x,y)| \leq g\left(|y/\gamma|^N\right) + Cg\left(|\gamma/x|^N\right) \tag{2.11}$$

where C is a constant depends on N, d and $|y/x|$.

Proof Since

$$g\left((y/\gamma)^N\right) = \frac{1}{|(y/\gamma)^N - 1|} \leq \frac{1}{|y/\gamma|^N - 1} = g\left(|y/\gamma|^N\right)$$

we only need to prove

$$\sum_{j=0}^{d-1} \frac{(y-x)^j}{j!} \frac{\mathrm{d}^j}{\mathrm{d}x^j} g\left((\gamma/x)^N\right) \leq Cg\left(|\gamma/x|^N\right) \tag{2.12}$$

for some constant C. When $d = 1$, it's easy to verify that the above inequality holds for $C = 1$ and any N. Thus now we only consider the case when $d \geq 2$.

Note that for any $l \geq 1$

$$\frac{\mathrm{d}}{\mathrm{d}x} g^l\left((\gamma/x)^N\right) = \frac{lN}{x}\left[g^l\left((\gamma/x)^N\right) + g^{l+1}\left((\gamma/x)^N\right)\right], \tag{2.13}$$

here g^l denotes function g raise to power l, hence the derivatives appear in eq. (2.12) all have the following form

$$\frac{\mathrm{d}^l}{\mathrm{d}x^l} g\left((\gamma/x)^N\right) = \frac{1}{x^l}\sum_{j=1}^{l+1} \alpha_j^{(l)} g^j\left((\gamma/x)^N\right) \tag{2.14}$$

where $\alpha_j^{(l)}$ $(1 \leq l \leq d-1,\ 1 \leq j \leq l+1)$ are constants.

We claim that when $N > d$ and for any $1 \leq l \leq d-1$, there exits constants $c^{(l)}$ dependent on d so that $|\alpha_j^{(l)}| \leq c^{(l)} N^l$ for all $1 \leq j \leq l+1$, this is proved by induction on l. When $l = 1$, we see from eq. (2.13) that

$$\frac{\mathrm{d}}{\mathrm{d}x} g\left((\gamma/x)^N\right) = \frac{N}{x}\left[g\left((\gamma/x)^N\right) + g^2\left((\gamma/x)^N\right)\right],$$

so $\alpha_1^{(1)} = \alpha_2^{(1)} = N$ and there exists $c^{(1)} = 1$ that satisfies the claim. Suppose the claim holds for $l = s$ where $1 \leq s \leq d-2$ (additionally assume $d > 2$ otherwise the proof for the claim is already finished), then

$$\begin{aligned}
\frac{\mathrm{d}^{s+1}}{\mathrm{d}x^{s+1}} g\left((\gamma/x)^N\right) &= \frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{1}{x^s}\sum_{j=1}^{s+1}\alpha_j^{(s)} g^j\left((\gamma/x)^N\right)\right) \\
&= -\frac{s}{x^{s+1}}\sum_{j=1}^{s+1}\alpha_j^{(s)} g^j\left((\gamma/x)^N\right) + \frac{1}{x^s}\sum_{j=1}^{s+1}\alpha_j^{(s)}\frac{jN}{x}\left[g^j\left((\gamma/x)^N\right) + g^{j+1}\left((\gamma/x)^N\right)\right] \\
&= \frac{1}{x^{s+1}}\Bigg[(N-s)\alpha_1^{(s)} g\left((\gamma/x)^N\right) + \sum_{j=2}^{s+1}\left((jN-s)\alpha_j^{(s)} + N(j-1)\alpha_{j-1}^{(s)}\right) g^j\left((\gamma/x)^N\right) \\
&\qquad + N(s+1)\alpha_{s+1}^{(s)} g^{s+2}\left((\gamma/x)^N\right)\Bigg],
\end{aligned}$$

thus the coefficients satisfy the following recurrence relation

$$\alpha_j^{(s+1)} = \begin{cases} (N-s)\alpha_1^{(s)}, & j = 1, \\ (jN-s)\alpha_j^{(s)} + N(j-1)\alpha_{j-1}^{(s)}, & 2 \leq j \leq s+1, \\ N(s+1)\alpha_{s+1}^{(s)}, & j = s+2. \end{cases}$$

We can pick $c^{(s+1)} = (2d+1)c^{(s)}$ and prove $|\alpha_j^{(s+1)}| \leq c^{(s+1)}N^{s+1}$ with the above relations, the claim holds for $l = s+1$ and this finishes the induction.

Now we go back to prove eq. (2.12), substitute eq. (2.14) into it we get

$$\begin{aligned}
\sum_{j=0}^{d-1} \frac{(y-x)^j}{j!} \frac{\mathrm{d}^j}{\mathrm{d}x^j} g\left((\gamma/x)^N\right) &= \sum_{j=0}^{d-1} \frac{(y-x)^j}{j!} \frac{1}{x^j} \sum_{l=1}^{j+1} \alpha_l^{(j)} g^l\left((\gamma/x)^N\right) \\
&\leq \sum_{j=0}^{d-1} \frac{(|y/x|-1)^j}{j!} \sum_{l=1}^{j+1} |\alpha_l^{(j)}| \cdot g^l\left(|\gamma/x|^N\right) \\
&\leq \sum_{j=0}^{d-1} \frac{(|y/x|-1)^j}{j!} c^{(j)} N^j \sum_{l=1}^{j+1} g^l\left(|\gamma/x|^N\right).
\end{aligned} \tag{2.15}$$

When $|\gamma/x|^N > 3$ so that $g\left(|\gamma/x|^N\right) < 1/2$, we get for any j,

$$\sum_{l=1}^{j+1} g^l\left(|\gamma/x|^N\right) \leq 2g\left(|\gamma/x|^N\right).$$

Continue on eq. (2.15), for $N > N_1 := \max\{d, \log 3/\log|\gamma/x|\}$,

$$\sum_{j=0}^{d-1} \frac{(y-x)^j}{j!} \frac{\mathrm{d}^j}{\mathrm{d}x^j} g\left((\gamma/x)^N\right) \leq g\left(|\gamma/x|^N\right) \cdot 2\sum_{j=0}^{d-1} \frac{(|y/x|-1)^j}{j!} c^{(j)} N^j,$$

thus eq. (2.12) holds for

$$C = 2\sum_{j=0}^{d-1} \frac{(|y/x|-1)^j}{j!} c^{(j)} N^j \tag{2.16}$$

which is a constant only depends on N, d and $|y/x|$. ■

Remark 2.2.2 *We can set $c^{(0)} = 1$ and $c^{(j)} = (2d+1)^{j-1}$ for $1 \leq j \leq d-1$ base on the above proof, so the constant C is a low order polynomial in N, d and $|y/x|$. Another minor issue is we defined $N_1 = \max\{d, \log 3/\log|\gamma/x|\}$ which is depend on γ, we can drop this dependency by enforcing γ_1 to be slightly larger than $|x|$, then $\log 3/\log|\gamma/x| \leq \log 3/\log|\gamma_1/x|$ and so we can instead set $N_1 = \max\{d, \log 3/\log|\gamma_1/x|\}$, this is crucial in deriving our next result.*

The last remaining piece of the error analysis is focused on the radius γ of the proxy surface Γ which is the only variable in eq. (2.11). Although the upper bound

eq. (2.11) always decays exponentially as N increases, different choice of γ can result to significant difference in magnitude on its actual value for a fixed N. This is summarized in the following proposition.

Theorem 2.2.3 *With all assumptions in theorem 2.2.2, if the upper bound eq.* (2.11) *is viewed as a real function in γ on the interval $(|x|, |y|)$, then there exists $N_2 > 0$ such that if $N > N_2$,*

1. *the function has a unique minimizer γ^*,*
2. *the minimum decays as $\mathcal{O}\left(|y/x|^{-N/2}\right)$.*

Proof This is equivalent to consider the real function

$$h(t) = \frac{1}{b/t - 1} + \frac{C}{t/a - 1}$$

on $t \in (a, b)$ where $a = |x|^N$, $b = |y|^N$ and C is defined in eq. (2.16). The derivative of the function is

$$h'(t) = \frac{(b - aC)t^2 + 2ab(C - 1)t + ab(a - bC)}{(t - a)^2(t - b)^2},$$

now consider the numerator $p(t) = (b - aC)t^2 + 2ab(C - 1)t + ab(a - bC)$ on the closed interval $[a, b]$, it is a quadratic function in t with the following properties:

- the coefficient of the second order term is

  $$b - aC = |x|^N \left(|y/x|^N - C\right)$$

  as discussed in remark 2.2.2, for any $N > N_1 \geq d$, C is low order polynomial in N and $|y/x|$, there must be some $N_2 > N_1$ such that $|y/x|^N > C$ holds for any $N > N_2$ and thus $b - aC > 0$;

- the discriminant is $\Delta = 4abC(a - b)^2 > 0$;

- when evaluated at $t = a$ and $t = b$, the function gives

  $$p(a) = -aC(a - b)^2 < 0,$$
  $$p(b) = b(a - b)^2 > 0.$$

All the above combined indicates that $p(t)$ has one root $t_0 \in (a, b)$ and $h'(t) < 0$ on $t \in (a, t_0)$ and $h'(t) > 0$ on $t \in (t_0, b)$, thus $\gamma^* = \sqrt[N]{t_0}$ is the unique minimizer of the upper bound eq. (2.11).

To prove the second part of the theorem, we compute explicitly the root t_0 by solving $h'(t) = 0$ (or $p(t) = 0$) so that

$$t_0 = \frac{-ab(C-1) + (b-a)\sqrt{abC}}{b - aC}$$

and then substitute into $h(t)$ to get

$$h(t_0) = \frac{2\sqrt{Cb/a} + (C+1)}{b/a - 1} = \frac{2\sqrt{C}|y/x|^{N/2} + (C+1)}{|y/x|^N - 1} \sim \mathcal{O}\left(|y/x|^{-N/2}\right),$$

the computation is purely tedious algebra thus is omitted here. ■

Remark 2.2.3 *The requirements for picking N_2 are $N_2 > N_1$ and $|y/x|^{N_2} > C$, hence N_2 is a constant only depends on d and $|y/x|$ and, again, is independent of γ. The minimizer γ^* is in the interval $(|a|, |b|)$ which is slightly larger than (γ_1, γ_2), for now let's suppose we don't care γ^* is inside the interval (γ_1, γ_2) or not, then similar to previous discussions, γ^* is only dependent on N, d and $|y/x|$.*

2.2.3 Block-wise error and practical issue

The analyses in section 2.2.2 can be seen as the entry-wise approximation error estimate for a matrix block if each entries in it is approximated by the same method in section 2.2.1, thus now we shift our attention to the block-wise counterparts.

Apply the approximation to each entries of the matrix $K^{X,Y}$, then eq. (2.5) together with the approximation error term eq. (2.8) give the form of the low-rank approximation

$$K^{X,Y} = K_N^{X,Y} + K^{X,Y} \quad E_N^{X,Y} = K^{X,Z} W_N^{X,Y} + K^{X,Y} \quad E_N^{X,Y} \tag{2.17}$$

where entries of $E_N^{X,Y}$ satisfy the bound eq. (2.11). Note that by replacing $|x|$ and $|y|$ with γ_1 and γ_2 in eq. (2.11), respectively, the upper bound is no longer associated

with a single source-target combination $(x_i, y_j) \in X \times Y$ and is instead uniform for all points in X and Y. This immediately leads to a corollary regarding the block-wise approximation error, the proof is omitted.

Corollary 2.2.4 *With all assumptions in theorem 2.2.2 and $\gamma \in (\gamma_1, \gamma_2)$, the F-norm relative approximation error is bounded by*

$$\frac{\|K_N^{X,Y} - K^{X,Y}\|_F}{\|K^{X,Y}\|_F} \leq g\left((\gamma_2/\gamma)^N\right) + Cg\left((\gamma/\gamma_1)^N\right) \tag{2.18}$$

where C is as defined in eq. (2.16) with $|y/x|$ replaced by γ_2/γ_1.

Similarly, theorem 2.2.3 can also be adopted to a block version by conducting the same analysis on the upper bound eq. (2.18), we will not state this result here and instead only give the following corollary which is a direct extension regarding the (numerical) rank of kernel matrix.

Corollary 2.2.5 *The numerical rank of the kernel matrix $K^{X,Y}$ is*

$$rank\left(K^{X,Y}\right) = \mathcal{O}\left(\log(1/\varepsilon)/\log(\gamma_2/\gamma_1)\right)$$

given any relative tolerance $0 < \varepsilon < 1$.

Remark 2.2.4 *To achieve certain accuracy by this analytical compression method, the number of quadrature nodes N depends only on the ratio between γ_2 and γ_1, i.e., the separation of the two sets, while the number of points in X or Y has no effect on the low-rank compression error. This analysis is consistent with the conclusions in the FMM context [11, 99].*

When the number of quadrature nodes N is fixed, we always want to pick γ that gives the smallest approximation error so we can simply set $\gamma = \gamma^*$ and this choice is justified by the omitted block version of theorem 2.2.3. But we only know the existence of such an optimal choice, not its value since there is no explicit expression in the theorem. Now we will discuss how to obtain γ^* or at least a good approximation of γ^* in practice.

2.2.3.1 When $d = 1$

In this case, many equations will have nice, analytical expressions so that we are able to draw much stronger conclusions compared to those for the general case in section 2.2.2.

As shown in proposition 2.2.1 for $d = 1$, the approximation error has this explicit expression

$$\varepsilon_N(x, y) = g\left((\gamma/x)^N\right) + g\left((y/\gamma)^N\right)$$

and hence

$$|\varepsilon_N(x, y)| \leq g\left(|\gamma/x|^N\right) + g\left(|y/\gamma|^N\right).$$

A version of the results in theorems 2.2.2 and 2.2.3 is summarized below.

Proposition 2.2.2 *When $d = 1$, for any $N > 0$ and $\gamma \in (\gamma_1, \gamma_2)$, the block-wise approximation error is bounded by*

$$\frac{\|K_N^{X,Y} - K^{X,Y}\|_F}{\|K^{X,Y}\|_F} \leq g\left((\gamma/\gamma_1)^N\right) + g\left((\gamma_2/\gamma)^N\right). \tag{2.19}$$

If viewed as a function in γ, this upper bound has a unique minimizer $\gamma^* = \sqrt{\gamma_1\gamma_2}$ and the optimal upper bound is $2g\left((\gamma_2/\gamma_1)^{N/2}\right)$.

Proof The approximation error eq. (2.19) is an extension of the entry-wise error, simply replace $|x|$ and $|y|$ by γ_1 and γ_2, respectively, and apply to each entries of the matrix $K^{X,Y}$. Then

$$\begin{aligned} g\left((\gamma/\gamma_1)^N\right) + g\left((\gamma_2/\gamma)^N\right) &= \frac{1}{(\gamma/\gamma_1)^N - 1} + \frac{1}{(\gamma_2/\gamma)^N - 1} \\ &= \frac{\left((\gamma/\gamma_1)^N + (\gamma_2/\gamma)^N\right) - 2}{(\gamma_2/\gamma_1)^N + 1 - \left((\gamma/\gamma_1)^N + (\gamma_2/\gamma)^N\right)} \\ &= -1 + \frac{(\gamma_2/\gamma_1)^N - 1}{(\gamma_2/\gamma_1)^N + 1 - \left((\gamma/\gamma_1)^N + (\gamma_2/\gamma)^N\right)}, \end{aligned}$$

note that $(\gamma/\gamma_1)^N + (\gamma_2/\gamma)^N \geq 2(\gamma_2/\gamma_1)^{N/2}$ and equality is reached when $\gamma^* = \sqrt{\gamma_1\gamma_2}$, thus the upper bound reaches its minimum $2g\left((\gamma_2/\gamma_1)^{N/2}\right)$ at $\gamma^* = \sqrt{\gamma_1\gamma_2}$. ■

Now we show a simple numerical example to illustrate the theoretical results in this section. We set $m = 200$, $n = 300$, $\gamma_1 = 0.5$, $\gamma_2 = 2$ and $\gamma_3 = 5$, the points in X and Y are uniformly chosen from their corresponding regions and are plotted in fig. 2.3(a). In fig. 2.3(b), we fix the number of quadrature nodes to be $N = 20$ and let γ vary from 0.55 to 1.9, both lines are of V-shape and the exact relative error reaches its minimum near the optimal radius we picked at $\gamma^* = \sqrt{\gamma_1\gamma_2} = 1$; in fig. 2.3(c), we fix γ to be at its optimal $\gamma^* = 1$ while let N vary from 6 to 29, both the exact error and its bound decay exponentially when N increases; the two vertical dashed line in the two graphs correspond to the optimal case, the proxy points Z at their optimal positions are plotted in fig. 2.3(a).

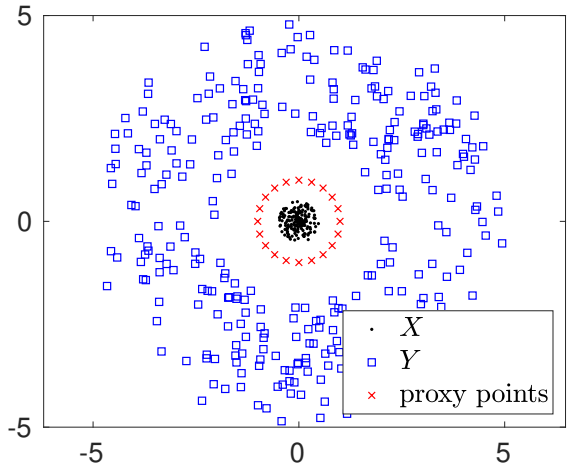
2.2.3.2 When $d > 1$

In this case, unfortunately we are unable to derive an explicit formula for γ^* and the optimal upper bound for the approximation error as in section 2.2.3.1, but we can still utilize the theoretical bound eq. (2.18) to approximate γ^* in practice to achieve better approximation error.

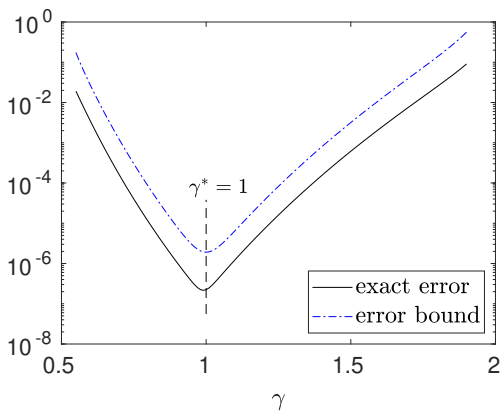
As was discussed in remark 2.2.3, the optimal $\gamma^* \in (\gamma_1, \gamma_2)$ is only dependent on N, d and γ_2/γ_1 (replacing $|y/x|$ in the entry-wise analysis) and, most importantly, is independent of the number of points in X and Y and their distribution. This feature motivates the idea to pick $X_0 \subset \mathcal{D}(0; \gamma_1)$ and $Y_0 \subset \mathcal{A}(0; \gamma_2, \gamma_3)$, then for any fixed N and d we would expect the following two quantities

$$E_N^0(\gamma) := \frac{\|K_N^{X_0,Y_0} - K^{X_0,Y_0}\|_F}{\|K^{X_0,Y_0}\|_F} \quad \text{and} \quad E_N(\gamma) := \frac{\|K_N^{X,Y} - K^{X,Y}\|_F}{\|K^{X,Y}\|_F} \tag{2.20}$$

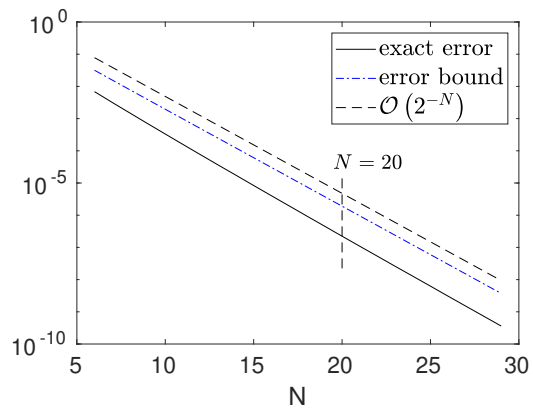
to have similar behavior when γ varies in (γ_1, γ_2), hence $E_N^0(\gamma)$ can be used as an indicator of the exact approximation error $E_N(\gamma)$ when they are viewed as a function in γ. Note that K^{X_0,Y_0} and $K_N^{X_0,Y_0}$ are computable through eq. (2.2) and eq. (2.5), respectively, thus $E_N^0(\gamma)$ can be computed explicitly and the cost is extremely small if $|X_0| \ll |X|$ and $|Y_0| \ll |Y|$.



(a) Sets X, Y and proxy points Z at their optimal positions



(b) varying proxy surface radius γ

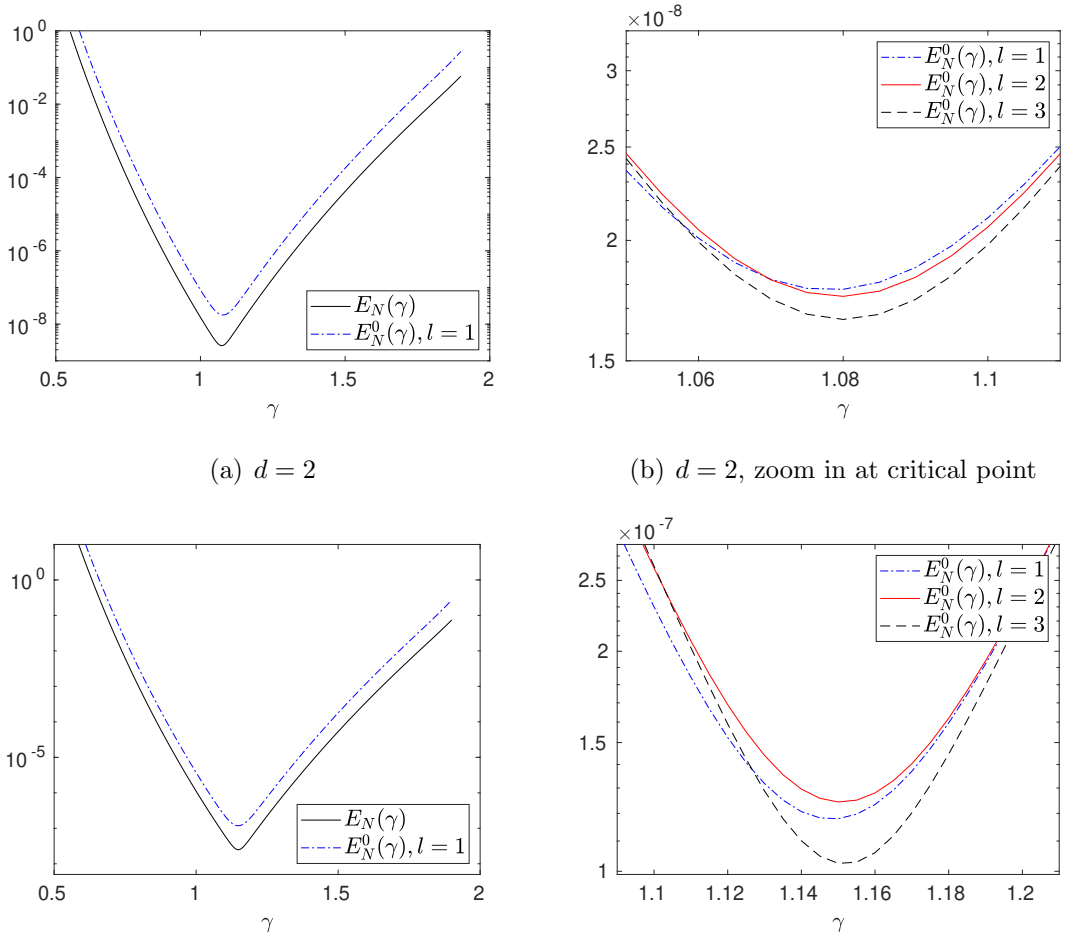


(c) varying quadrature number N

Figure 2.3. For $d = 1$, the exact F-norm relative error compared with its upper bound for different γ and N.

Here we show another numerical test. Consider $d = 2, 3$ and the two sets X and Y are the same as in fig. 2.3(a), N is fixed at $N = 30$. We set $l = |X_0| = |Y_0|$ to be 1, 2 or 3, more specifically, X_0 is uniformly distributed on the circle $\mathcal{C}(0; \gamma_1)$ and we make sure $x = \gamma_1 \in \mathbb{C}$ is always in X_0, likewise for Y_0. This way of picking X_0 and Y_0 on the boundaries of X and Y, respectively, corresponds to the worst case scenario

of any error bounds developed before, so that $E_N^0(\gamma)$ is more likely to capture the behavior of $E_N(\gamma)$.



(a) $d = 2$

(b) $d = 2$, zoom in at critical point

(c) $d = 3$

(d) $d = 3$, zoom in at critical point

Figure 2.4. For $d = 2$ and 3, compare $E_N(\gamma)$ with the indicator $E_N^0(\gamma)$ for $l = 1, 2$ and 3.

We can see from figs. 2.4(a) and 2.4(c) that, $E_N^0(\gamma)$ already gives a good estimate of the behavior of $E_N(\gamma)$ for both cases when $d = 2$ or 3. Then in figs. 2.4(b) and 2.4(d) we plot $E_N^0(\gamma)$ for $l = 1, 2, 3$ and zoom in at around the minimum since they almost coincide with each other when moving away from this interval, this indicates that

setting $l = 1$ suffices in practice and we won't gain much by increasing l and paying more computation cost.

To conclude the above discussion, for any given $d > 1$, N and γ_2/γ_1, let $X_0 = \{\gamma_1\}$ and $Y_0 = \{\gamma_2\}$ and construct the function $E_N^0(\gamma)$ in eq. (2.20), then solve the minimization problem on the interval (γ_1, γ_2) and γ^* is set to be the minimizer of it.

Remark 2.2.5 *$E_N^0(\gamma)$ is a non-linear smooth function in γ on (γ_1, γ_2), for stability reason, we can instead solve for the minimizer of* $\log\left(E_N^0(\gamma)\right)$.

In most of the hierarchical matrix techniques, a prespecified separation parameter or admissible condition is used to determine the compressibility of certain submatrix. In the case of kernel matrices, this separation parameter is characterized by the ratio γ_2/γ_1 thus the radius combination $\{\gamma_1, \gamma_2\}$ will be shared by all submatrices that need to be compressed, which also means the process of approximating the optimal value γ_* only needs to be run once. We are trading very little pre-computation cost with much better compression accuracy. We summarize the overall strategy for picking γ_* in algorithm 2.

2.3 Hybrid method

In most of the applications, the purely analytical approach like the one introduced in section 2.2 is not used on its own, it is usually combined with another algebraic step to further compress the matrix, the resulting algorithm is called a *hybrid method*. In this section, we will analyze in details of the proxy point method algorithm 1 which falls in the class of hybrid method, theoretical background and error estimation will be provided.

Algorithm 2 *Approximating the optimal radius γ*

Input: γ_1, γ_2, d, τ (given compression tolerance)

Output: γ^*, N

```
if d = 1 then
    Set γ* = √(γ1γ2)
    Compute N with 2g((γ2/γ1)^(N/2)) = τ        ▷ error bound in proposition 2.2.2
else
    Initialize N = 0 and ε = 1                  ▷ ε is the computed error
    while ε > τ do
        N ← N + 10                              ▷ increase N when accuracy is not enough
        Form the function E_N^0(γ)              ▷ by eq. (2.20)
        Minimize log(E_N^0(γ)) to get γ* and set ε = E_N^0(γ*)
    end while
end if
```

2.3.1 Review of SRRQR factorization and ID

For matrix A of size $m \times n$ and a target rank $r \leq \max(m, n)$, the interpolative decomposition (ID) [15] is $A \approx UA|_J$ where the column basis U has the form

$$U = P \begin{pmatrix} I \\ E \end{pmatrix} \tag{2.21}$$

and P is a permutation matrix, I is a $r \times r$ identity matrix and E is an $(m-r) \times r$ residual matrix, the row basis $A|_J$ is a submatrix of A with selected rows in the index set J and $|J| = r$. When ID is applied to a kernel matrix $K^{X,Y}$, the index set J corresponds to a subset of X denoted by $\tilde{X}$ and is referred to as the *representative points* of X, then the compression can be written as $K^{X,Y} \approx UK^{\tilde{X},Y}$.

Although the column basis U is not orthogonal, the numerical stability of such a factorization is guaranteed if the ID is obtained by SRRQR in the sense that entries of E can be bounded by a constant $f \geq 1$, typically the process costs $\mathcal{O}(mnr)$ if f is

set to be a small constant larger then 1 or small power in n, e.g., 2 or $\sqrt{n}$; see [39] for details.

2.3.2 Hybrid strategy and error analysis

When the column size n of the kernel matrix $K^{X,Y}$ is too large, directly applying the ID is prohibitively expensive. We can introduce an analytical compression step via contour integration (abbreviated as CI) at the start to greatly reduce the cost, this hybrid strategy is summarized as follows:

$$K^{X,Y} \approx K_N^{X,Y} = K^{X,Z} W_N^{Z,Y} \qquad \text{(by CI on } K^{X,Y}\text{)} \tag{2.22}$$

$$\approx U K^{\tilde{X},Z} W_N^{Z,Y} \qquad \text{(by ID on } K^{X,Z}\text{)} \tag{2.23}$$

$$= U K_N^{\tilde{X},Y} \approx U K^{\tilde{X},Y} \qquad \text{(by CI on } K^{\tilde{X},Y}\text{)}. \tag{2.24}$$

Note that eqs. (2.22) and (2.24) are done analytically with no cost, the only actual computation needed in this process is to form $K^{X,Z}$ and apply ID to it in eq. (2.23) which is bounded by $\mathcal{O}(mN) + \mathcal{O}(mNr) = \mathcal{O}(mNr)$. As we have discussed before, N is only a constant independent of m and n and, furthermore, is much smaller than the column size n, thus the hybrid method is extremely efficient compared to applying ID on the original kernel matrix.

An obvious similarity between the above hybrid method and the proxy point method algorithm 1 can be observed, the contour Γ and the quadrature nodes Z in eqs. (2.4) and (2.5) correspond to the proxy surface and proxy points in line 1, this fact helps to explain the connection between the bases of $K^{X,Y}$ and $K^{X,Z}$. In eq. (2.23) which corresponds to line 2, the algebraic compression with a structure-preserving method ID enables us to reuse the factors U and $\tilde{X}$ in the final low-rank approximation without any additional computation cost. At last in line 3, there is also an implicit analytical compression step as shown in eq. (2.24).

The next theorem concerns the approximation error for our variant of the hybrid method with CI and ID.

Theorem 2.3.1 *The kernel function is as defined in eq.* (2.2)*, sets X, Y satisfy the assumptions at the beginning of section 2.2, the N proxy points Z are at the optimal positions according to the discussions in section 2.2.3 and the compression error τ_{CI} is defined by eq.* (2.18) *when $\gamma = \gamma^*$, the relative tolerance (in F-norm) used in ID is τ_{ID} and the constant in SRRQR is $f > 1$ and the compression rank is $r < N$, then a rank-r approximation of the kernel matrix $K^{X,Y}$ by the hybrid method eqs.* (2.22) *to* (2.24) *satisfies*

$$\|K^{X,Y} - UK^{\tilde{X},Y}\|_F \leq (C_{CI}\tau_{CI} + C_{ID}\tau_{ID})\,\|K^{X,Y}\|_F \tag{2.25}$$

where

$$C_{CI} = 1 + \sqrt{r + (m-r)rf^2}\sqrt{1 - \frac{(m-r)(\gamma_2-\gamma_1)^{2d}}{m(\gamma_1+\gamma_3)^{2d}}},$$
$$C_{ID} = \frac{\gamma^*(\gamma_1+\gamma_3)^d}{(\gamma_2-\gamma^*)(\gamma^*-\gamma_1)^d}.$$

Proof Note that

$$\begin{aligned}
&\|K^{X,Y} - UK^{\tilde{X},Y}\|_F \\
&\quad\leq \|K^{X,Y} - K_N^{X,Y}\|_F + \|K_N^{X,Y} - UK^{\tilde{X},Y}\|_F \\
&\quad\leq \|K^{X,Y} - K_N^{X,Y}\|_F + \|U\|_F\|K^{\tilde{X},Y} - K_N^{\tilde{X},Y}\|_F + \|K_N^{X,Y} - UK_N^{\tilde{X},Y}\|_F \\
&\quad\leq \|K^{X,Y} - K_N^{X,Y}\|_F + \|U\|_F\|K^{\tilde{X},Y} - K_N^{\tilde{X},Y}\|_F + \\
&\qquad\qquad\qquad\qquad \|K^{X,Z} - UK^{\tilde{X},Z}\|_F\|W_N^{Z,Y}\|_F,
\end{aligned} \tag{2.26}$$

now we will derive upper bounds for the three terms in the last line separately.

For the first term, it's CI applied on the original kernel matrix so directly we have

$$\|K^{X,Y} - K_N^{X,Y}\|_F \leq \tau_{\mathrm{CI}}\|K^{X,Y}\|_F. \tag{2.27}$$

For the second term,

$$\|U\|_F = \left\| P\begin{pmatrix} I \\ E \end{pmatrix} \right\|_F = \left\| \begin{pmatrix} I \\ E \end{pmatrix} \right\|_F \leq \sqrt{r + (m-r)rf^2}$$

and similar to eq. (2.27)

$$\|K^{\tilde{X},Y} - K_N^{\tilde{X},Y}\|_F \leq \tau_{\mathrm{CI}}\|K^{\tilde{X},Y}\|_F,$$

note that entries of $K^{X,Y}$ satisfy

$$\frac{1}{(\gamma_1+\gamma_3)^d} \le |k(x,y)| \le \frac{1}{(\gamma_2-\gamma_1)^d}$$

hence

$$\begin{aligned}\frac{\|K^{\tilde{X},Y}\|_F^2}{\|K^{X,Y}\|_F^2} &= 1 - \frac{\|K^{X\setminus\tilde{X},Y}\|_F^2}{\|K^{X,Y}\|_F^2} \\ &\le 1 - \frac{(m-r)n\frac{1}{(\gamma_1+\gamma_3)^{2d}}}{mn\frac{1}{(\gamma_2-\gamma_1)^{2d}}} = 1 - \frac{(m-r)(\gamma_2-\gamma_1)^{2d}}{m(\gamma_1+\gamma_3)^{2d}},\end{aligned}$$

then the second term can be bounded with

$$\begin{aligned}&\|U\|_F\|K^{\tilde{X},Y} - K_N^{\tilde{X},Y}\|_F \\ &\qquad \le \tau_{\mathrm{CI}}\sqrt{n(1+(m-r)rf^2)}\sqrt{\left(1-\frac{(m-r)(\gamma_2-\gamma_1)^{2d}}{m(\gamma_1+\gamma_3)^{2d}}\right)}\|K^{X,Y}\|_F. \qquad (2.28)\end{aligned}$$

For the third term, $K^{X,Z}$ is approximated by ID so

$$\|K^{X,Z} - UK^{\tilde{X},Z}\|_F \le \tau_{\mathrm{ID}}\|K^{X,Z}\|_F,$$

also notice that entries of $K^{X,Z}$ and $W_N^{Z,Y}$ are bounded by

$$|k(x,z)| \le \frac{1}{(\gamma^*-\gamma_1)^d} \quad \text{and} \quad |w_N(z,y)| \le \frac{\gamma^*}{N(\gamma_2-\gamma^*)},$$

respectively, hence

$$\|W_N^{Z,Y}\|_F \le \sqrt{Nn}\frac{\gamma^*}{N(\gamma_2-\gamma^*)} = \sqrt{\frac{n}{N}}\frac{\gamma^*}{\gamma_2-\gamma^*}$$

and

$$\frac{\|K^{X,Z}\|_F^2}{\|K^{X,Y}\|_F^2} \le \frac{mN\frac{1}{(\gamma^*-\gamma_1)^{2d}}}{mn\frac{1}{(\gamma_1+\gamma_3)^{2d}}} = \frac{N}{n}\frac{(\gamma_1+\gamma_3)^{2d}}{(\gamma^*-\gamma_1)^{2d}},$$

thus the third term is bounded by

$$\begin{aligned}\|K^{X,Z} - UK^{\tilde{X},Z}\|_F\|W_N^{Z,Y}\|_F &\le \tau_{\mathrm{ID}}\|K^{X,Z}\|_F\|W_N^{Z,Y}\|_F \\ &\le \tau_{\mathrm{ID}}\sqrt{\frac{n}{N}}\frac{\gamma^*}{\gamma_2-\gamma^*}\sqrt{\frac{N}{n}\frac{(\gamma_1+\gamma_3)^d}{(\gamma^*-\gamma_1)^d}}\|K^{X,Y}\|_F \qquad (2.29) \\ &\le \tau_{\mathrm{ID}}\frac{\gamma^*(\gamma_1+\gamma_3)^d}{(\gamma_2-\gamma^*)(\gamma^*-\gamma_1)^d}\|K^{X,Y}\|_F.\end{aligned}$$

At last, the error bound eq. (2.25) is obtained by substituting eqs. (2.27) to (2.29) into eq. (2.26). ■

Remark 2.3.1 *Once the annulus region $\mathcal{A}(0;\gamma_2,\gamma_3)$ is fixed, source points Y are completely irrelevant to the hybrid method on both algorithm and error analysis parts. The basis U and representative points $\tilde{X}$ can be obtained with only target points X, then they can be applied to $K^{X,Y'}$ with arbitrary $Y' \subset \mathcal{A}(0;\gamma_2,\gamma_3)$ and still achieve the same compression accuracy. Although $\gamma_3 > 0$ is needed so that Y is on a bounded domain in order to derive a theoretical error bound, we believe such an limitation is not needed in practice. The analytical compression tends to be more accurate when a source point y and a target point x moves far away from each other.*

Remark 2.3.2 *Usually the constant governing the numerical stability in the SRRQR factorization is set to be $f = 2$ so that the algorithm produces basis with enough stability and won't take exponential time to run. In this case, the two constants in eq. (2.25) scale roughly as $C_{CI} \sim \mathcal{O}(\sqrt{m})$ and $C_{ID} \sim \mathcal{O}(1)$. The error τ_{CI} introduced by analytical compression seems more sensitive than τ_{ID} by the algebraic part. As a result, to achieve a desired accuracy in the hybrid method, one should ask for higher accuracy in the analytical step.*

2.4 Conclusions

In this chapter, we have discussed an analytical compression method based on Cauchy integral and numerical quadrature. Several novel analysis results are presented regarding the approximation error introduced by this process, we also analyzed strategies on picking proxy points for optimal performance. The method is then combined with an algebraic compression step ID to produce a hybrid compression method that usually referred to as proxy point method, the theories developed here serve as an intuitive as well as mathematical explanation of the effective of such a method.

There are two immediate follow-ups on this work in the future. Our compression method can be applied to HSS construction for Cauchy-like matrices and Toeplitz matrices, by further exploiting the structure in the matrices, we can potentially reduce the HSS construction cost to be lower than linear which can greatly reduce the overall

computation cost for solving linear systems. We are also interested in extending the analysis to more kernels and higher dimensionality so that the framework proposed here become a general purpose method for more applications.

3. FAST CONTOUR-INTEGRAL EIGENSOLVER FOR NON-HERMITIAN MATRICES

The content in this chapter is from a published paper [123]: X. Ye, J. Xia, Raymond H. Chan, S. Cauley, and V. Balakrishnan, *A fast contour-integral eigensolver for non-Hermitian matrices*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 1268–1297. A permission letter from SIAM can be found in appendix A.

3.1 Introduction

In this chapter, we consider the eigenvalue solution for a non-Hermitian matrix A:

$$Ax = \lambda x, \quad A \in \mathbb{C}^{n\times n}, \tag{3.1}$$

where $\lambda \in \mathbb{C}$ is an eigenvalue and x is the corresponding eigenvector. We study a type of contour-integral eigensolvers and propose a series of acceleration techniques. We suppose an eigenvalue decomposition of A exists:

$$A = X\Lambda X^{-1}, \tag{3.2}$$

where $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is a diagonal matrix for the eigenvalues, and $X = (x_1, x_2, \ldots, x_n)$ is the eigenvector matrix.

Classical methods for solving (3.1) include power iterations, inverse iterations, and QR iterations. The main operations involve matrix-vector multiplications, linear system solutions, or QR factorizations. In QR iterations, A is usually first reduced to an upper Hessenberg form.

Recently, a class of contour-integral based eigensolvers have been developed to find a partial spectrum. They have some very appealing features, such as the robustness in terms of convergence rates, the natural accommodation of eigenvalue multiplicity,

and the nice scalability. In [94, 95], the Sakurai-Sugiura (SS) method is proposed to reduce a generalized eigenvalue problem to a smaller one with Hankel matrices, and later in [96] a stable version called CIRR is introduced by combining the contour-integral technique with the Rayleigh-Ritz procedure. The FEAST algorithm is first proposed in [90] for Hermitian matrices, where a spectral projector is constructed via the integration of the resolvent of a matrix, followed by projected subspace iterations. Some non-Hermitian FEAST methods can be found in [53, 57, 100, 125]. Contour-integral eigensolvers utilize a filter function, whose quality is a key factor of the effectiveness of the eigenvalue solutions. Rational filter functions are discussed in [41, 101]. Other types of filter functions can be obtained via the solution of optimization or least-squares problems [104, 110].

The basic idea of the FEAST algorithm is as follows. Suppose $\lambda_i,\ i = 1, 2, \ldots, s \leq n$ are all the eigenvalues inside a Jordan curve Γ on the complex plane. Consider the contour integral

$$\phi(z) = \frac{1}{2\pi \mathbf{i}} \int_\Gamma \frac{1}{\mu - z} \mathrm{d}\mu, \quad z \notin \Gamma, \quad \mathbf{i} = \sqrt{-1}. \tag{3.3}$$

A spectral projector to the desired eigenspace $\mathrm{span}\{x_1, x_2, \ldots, x_s\}$ is constructed based on Cauchy's residue theorem [98] in complex analysis:

$$\begin{aligned} \Phi \equiv \phi(A) &= \frac{1}{2\pi \mathbf{i}} \int_\Gamma (\mu I - A)^{-1} \mathrm{d}\mu = \frac{1}{2\pi \mathbf{i}} \int_\Gamma (\mu I - X\Lambda X^{-1})^{-1} \mathrm{d}\mu \\ &= X \left(\frac{1}{2\pi \mathbf{i}} \int_\Gamma (\mu I - \Lambda)^{-1} \mathrm{d}\nu \right) X^{-1} = X \begin{pmatrix} I_s & \\ & 0 \end{pmatrix} X^{-1}. \end{aligned} \tag{3.4}$$

In practice, the spectral projector Φ is not explicitly formed. Instead, the basis of the eigenspace can be extracted with randomization, where the product of Φ and an appropriately chosen random matrix Y is computed:

$$Z = \Phi Y = \frac{1}{2\pi \mathbf{i}} \int_\Gamma (\mu I - A)^{-1} Y \mathrm{d}\mu. \tag{3.5}$$

This needs to evaluate the contour integral, which is done by numerical quadratures. In the process, linear systems are solved for $(\mu I - A)^{-1} Y$. After Z is evaluated, it is

used as starting vectors in projected subspace iterations to compute the desired eigenpairs. The accuracy of the quadrature approximation is essential to the convergence rate of the subspace iterations.

In the FEAST algorithm and other contour-integral eigensolvers, it needs an estimate of the number of eigenvalues of A inside Γ, denoted $\#_\Lambda(A, \Gamma)$, which is sometimes assumed to be known in advance. Some estimation methods have been proposed in [85, 94, 125] based on stochastic strategies.

In both the eigenvalue count and the projected subspace iterations, it needs to evaluate the numerical quadrature by solving linear systems with multiple shifts μI and multiple right-hand sides. This poses challenges to both direct and iterative linear solvers. For example, direct solvers are suitable for multiple right-hand sides, but each additional shift typically requires a new factorization. If A is a general dense matrix, each factorization costs $\mathcal{O}(n^3)$ flops. The total eigenvalue solution cost may be quite high, depending on the number of eigenvalues desired and the accuracy.

Here, we seek to design a fast contour-integral eigensolver. There are three major tasks. (1) One task is to analyze some numerical quadrature rules for the design of filter functions and understand their quality. This helps us choose an appropriate quadrature with justified optimality for the contour integration. (2) The next task is to show why some low-accuracy approximations can be used to quickly and reliably count the eigenvalues inside Γ. Both deterministic and probabilistic justifications are included. (3) The third task is to present a fast algorithm to find selected or all eigenpairs of A based on the analysis and a type of fast shifted factorizations. Some tools to use include structured matrices and shifted structured factorization update. The matrices we consider include some rank structured ones and more general cases. Previously, for non-Hermitian rank-structured eigenvalue problems, fast QR iterations are designed for special cases such as companion matrices [9, 14, 21, 105]. Here, we consider more general cases.

Our first task is to perform some analysis on the quality of some commonly used quadrature rules for approximating (3.3). The quadrature approximation is expected

to be not too far away from 1 for z inside Γ and not too close to Γ, and to decay quickly for z outside and away from Γ. Existing FEAST algorithms usually use the Gauss-Legendre rule [57, 90, 125], though recent numerical observations find that the Trapezoidal rule may be preferable [100, 104]. Here, we analytically show that the Trapezoidal rule is much superior in the sense that it yields quadrature approximations with nearly optimal decay outside the unit circle (as the mapped contour Γ) in the complex plane. Thus, the Trapezoidal rule will be used in our eigensolver. We would like to mention that interesting analysis has been performed for approximating the operator exponent by contour integration of the matrix resolvent [29], where an exponentially convergent sinc quadrature rule is proposed and is also applicable to other common kernel functions such as $1/|x-y|$ and $\log|x-y|$ [45].

The next task is to show the feasibility of using low-accuracy matrix approximations for the quick and reliable estimate of $\#_\Lambda(A,\Gamma)$. The eigenvalue count involves quadrature approximations similar to (3.5) and needs linear solutions with multiple shifts and multiple right-hand sides. Certain low-accuracy matrix approximations with fast solutions enable us to quickly estimate $\#_\Lambda(A,\Gamma)$, as long as the count is unchanged or remains close. We show that, when Γ is not too close to the eigenvalues inside it, A can be approximated by a matrix $\tilde{A}$ with a low accuracy so that the eigenvalue count remains the same ($\#_\Lambda(A,\Gamma) = \#_\Lambda(\tilde{A},\Gamma)$). The farther away Γ is from the eigenvalues, the lower the approximation accuracy of $\tilde{A}$ is allowed to be. On the other hand, if there are eigenvalues close to Γ, we use probabilistic methods to justify the reliability of $\#_\Lambda(\tilde{A},\Gamma)$. We show that, for some situations, with high probabilities, the eigenvalue count is off by only a very small number α. Roughly speaking, the probability of miscounting the eigenvalues by α decays exponentially with α. This is sufficient for us since we do not need the count to be exact.

Our choice of $\tilde{A}$ is based on rank structured forms, since it is convenient to control the approximation accuracy and also quick to perform structured direct solutions with multiple right-hand sides and even multiple shifts. (Note that the approximation analysis for the eigenvalue count is not restricted to rank structured forms.) The

rank structured forms involve low-rank approximations of some off-diagonal blocks. Examples of such forms include $\mathcal{H}$ [42], $\mathcal{H}^2$ [10, 43], and hierarchically semiseparable (HSS) [12, 117] representations. For matrices with small off-diagonal ranks or numerical ranks, fast direct solvers exist. Such matrices widely appear in numerical computations, such as polynomial root finding, Toeplitz problems, and some discretized problems. Here, even if A itself is not rank structured, we may still use a rank structured approximation $\tilde{A}$ to quickly count the eigenvalues.

Our third task is then to design a fast contour-integral eigensolver for rank structured A and even more general cases. This is based on fast factorizations of rank structured approximations, as well as fast factorization update with varying shifts for the quadrature evaluations. We will adaptively choose the approximation accuracy to balance the efficiency and the accuracy. Previously, for Hermitian HSS matrices, a shifted structured LDL factorization update is designed [7, 113]. Here, we further show that, even for non-Hermitian HSS matrices, we can still update the factorization for varying shifts so as to save nearly 40% of the factorization cost for each shift.

To find the eigenvalues inside a search region, our eigensolver recursively partitions the region into subregions, until the number of eigenvalues inside each subregion is smaller than a threshold k. This process can be organized into a quadtree. For subregions corresponding to the leaf nodes, we then increase the approximation accuracy of $\tilde{A}$ and switch to projected subspace iterations. The shifted structured factorization update can benefit both the eigenvalue count and the subspace iteration. The saving in the eigenvalue count is especially significant, since the count is done for each intermediate subregion or each node of the quadtree and the subspace iteration is done just for the leaf nodes. Additionally, deflation is incorporated into the eigensolver.

In particular, if A itself is rank structured and has maximum off-diagonal rank or numerical rank r, we show that the optimal threshold for the eigenvalue count in the leaf level subregions is $k = \mathcal{O}(r)$. This minimizes the total complexity for finding all the eigenpairs, which is $\mathcal{O}(rn^2) + \mathcal{O}(r^2 n)$ under a modest assumption.

Various applications are then discussed. We also discuss the choice of the initial search region. Numerical tests are done for some problems. We can clearly see the benefits of shifted factorization update and low-accuracy matrix approximation. The cost for the eigenvalue counts has been reduced to a very small portion of the total.

The outline of the remaining presentation is as follows. In Section 3.2, we show our analysis of the quadrature rules for the filter function design. The idea of low-accuracy matrix approximations for the eigenvalue count is given in Section 3.3. Our fast contour-integral eigensolver is presented in Section 3.4. Section 3.5 gives the numerical experiments to illustrate the efficiency and accuracy.

The following notation is used throughout the paper:

- $\mathcal{C}_\gamma(z)$ denotes the circle centered at z with radius γ;
- $\mathcal{D}_\gamma(z)$ denotes the open disk centered at z with radius γ;
- $\mathcal{A}_{\gamma,\delta}(z) = \{\omega \in \mathbb{C} : \gamma - \delta < |\omega - z| < \gamma + \delta\}$ is the open annulus region centered at z with outer radius $\gamma + \delta$ and inner radius $\gamma - \delta$.

3.2 Analysis of quadrature rules for filter function design

In contour-integral eigensolvers, the quality of the quadrature approximation is critical for the accuracy of the eigenvalue computation. Here, we first perform analysis on some quadrature rules so as to choose an appropriate one for (3.3) and (3.5).

If the contour Γ has a parametrization $\Gamma = \{h(t) : t \in [a, b)\}$, then (3.3) becomes

$$\phi(z) = \frac{1}{2\pi \mathbf{i}} \int_a^b \frac{h'(t)}{h(t) - z} \mathrm{d}t. \tag{3.6}$$

A q-point quadrature rule can be used to approximate $\phi(z)$ by the filter function

$$\tilde{\phi}(z) = \frac{1}{2\pi \mathbf{i}} \sum_{j=1}^{q} w_j \frac{h'(t_j)}{h(t_j) - z}, \tag{3.7}$$

where t_j's and w_j's are the quadrature nodes and weights, respectively.

We focus on the case when Γ is a circle $\mathcal{C}_\gamma(z_0)$. Specifically when $\mathcal{C}_\gamma(z_0)$ is the unit circle $\mathcal{C}_1(0)$, write $\phi(z)$ in (3.3) and (3.6) as $\phi_0(z)$ and write $\tilde{\phi}(z)$ in (3.7) as $\tilde{\phi}_0(z)$. Then $\phi(z)$ can be transformed directly into $\phi_0(z)$ by

$$\begin{aligned}\phi(z) &= \frac{1}{2\pi\mathbf{i}}\int_{\mathcal{C}_\gamma(z_0)} \frac{1}{\mu - z}\mathrm{d}\mu = \frac{1}{2\pi\mathbf{i}}\int_{\mathcal{C}_1(0)} \frac{\gamma}{z_0 + \gamma\nu - z}\mathrm{d}\nu \\ &= \frac{1}{2\pi\mathbf{i}}\int_{\mathcal{C}_1(0)} \frac{1}{\nu - (z - z_0)/\gamma}\mathrm{d}\nu = \phi_0\left(\frac{z - z_0}{\gamma}\right)\end{aligned}$$

Thus, it is sufficient to focus only on $\phi_0(z)$ and its approximation $\tilde{\phi}_0(z)$. Let the parametrization of the unit circle $\mathcal{C}_1(0)$ be $h(t) = e^{\mathbf{i}\pi t},\ -1 \le t < 1$. Then

$$\phi_0(z) = \frac{1}{2}\int_{-1}^{1} \frac{e^{\mathbf{i}\pi t}}{e^{\mathbf{i}\pi t} - z}\mathrm{d}t, \tag{3.8}$$

$$\tilde{\phi}_0(z) = \frac{1}{2}\sum_{j=1}^{q} w_j \frac{e^{\mathbf{i}\pi t_j}}{e^{\mathbf{i}\pi t_j} - z} \equiv \frac{1}{2}\sum_{j=1}^{q} \frac{w_j z_j}{z_j - z}, \tag{3.9}$$

where z_j's are the mapped quadrature nodes on $\mathcal{C}_1(0)$:

$$z_j = e^{\mathbf{i}\pi t_j}, \quad j = 1, 2, \ldots, q. \tag{3.10}$$

Rewrite (3.9) as a rational form

$$\tilde{\phi}_0(z) \equiv \frac{f(z)}{g(z)} \quad \text{with} \quad g(z) = \prod_{j=1}^{q}(z - z_j), \tag{3.11}$$

where $g(z)$ is a polynomial of degree $d_g \equiv q$ and with roots $z_j,\ j = 1, 2, \ldots, q$, and $f(z)$ is a polynomial uniquely determined by the choice of the quadrature rule (t_j's and w_j's). The degree d_f of $f(z)$ satisfies $0 \le d_f \le q - 1$. For $\tilde{\phi}_0(z)$ to be a good approximation of the exact function $\phi_0(z)$, we would expect:

- $|\tilde{\phi}_0(z)|$ is not too far away from 1 when z is inside $\mathcal{C}_1(0)$ and not too close to z_j's;
- $|\tilde{\phi}_0(z)|$ decays fast when z is outside $\mathcal{C}_1(0)$ and moves away from it.

The following proposition indicates that the first criterion is always satisfied when an interpolatory quadrature is used.

Proposition 3.2.1 *For $z \in \mathbb{C}$ and $\tilde{\phi}_0(z)$ in (3.9) resulting from any interpolatory quadrature rule applied to (3.8),*

- $\tilde{\phi}_0(0) = 1$;
- $|\tilde{\phi}_0(z)| > \frac{1}{2}$ *when* $|z| < 1$;
- $|\tilde{\phi}_0(z)| < \frac{1}{\delta}$ *when* $|z_j - z| > \delta > 0,\ j = 1, 2, \ldots, q$.

Proof For any interpolatory quadrature rule, the weights $\{w_j\}_{j=1}^q$ satisfy $\sum_{j=1}^q w_j = 2$. Then directly from (3.9), we get

$$\tilde{\phi}_0(0) = \frac{1}{2}\sum_{j=1}^{q} \frac{w_j z_j}{z_j} = \frac{1}{2}\sum_{j=1}^{q} w_j = 1.$$

When $|z| < 1$, we have

$$\begin{aligned}\mathrm{Re}(\tilde{\phi}_0(z)) &= \frac{1}{2}\left(\tilde{\phi}_0(z) + \overline{\tilde{\phi}_0(z)}\right) = \frac{1}{4}\sum_{j=1}^{q}\left(\frac{w_j z_j}{z_j - z} + \frac{w_j \overline{z_j}}{\overline{z_j} - \overline{z}}\right) \\ &= \frac{1}{4}\sum_{j=1}^{q} w_j \frac{z_j(\overline{z_j} - \overline{z}) + \overline{z_j}(z_j - z)}{(z_j - z)(\overline{z_j} - \overline{z})} = \frac{1}{4}\sum_{j=1}^{q} w_j \frac{2 - (z_j\overline{z} + \overline{z_j}z)}{1 + |z|^2 - (z_j\overline{z} + \overline{z_j}z)}.\end{aligned}$$

Note that $z_j\overline{z} + \overline{z_j}z \in \mathbb{R}$ and $|z_j\overline{z} + \overline{z_j}z| \le 2|z| < 1 + |z|^2 < 2$. Then,

$$\mathrm{Re}(\tilde{\phi}_0(z)) > \frac{1}{4}\sum_{j=1}^{q} w_j \frac{2 - (z_j\overline{z} + \overline{z_j}z)}{2 - (z_j\overline{z} + \overline{z_j}z)} = \frac{1}{4}\sum_{j=1}^{q} w_j = \frac{1}{2}.$$

This yields $|\tilde{\phi}_0(z)| \ge \mathrm{Re}(\tilde{\phi}_0(z)) > 1/2$.

Finally, when $|z_j - z| > \delta > 0,\ j = 1, 2, \ldots, q$,

$$|\tilde{\phi}_0(z)| < \frac{1}{2}\sum_{j=1}^{q} \frac{w_j}{\delta} = \frac{1}{\delta}.$$

■

The proposition means, if z is inside $\mathcal{D}_1(0)$, then $|\tilde{\phi}_0(z)|$ has a lower bound $1/2$. It also has an upper bound $1/\delta$ for z not within a distance δ of any mapped quadrature point z_j. If z is too close to any z_j, then it is possible for $|\tilde{\phi}_0(z)|$ to be large. This can also be observed from Figure 3.1 below.

We then study the decay property of $|\tilde{\phi}_0(z)|$. From the rational form (3.11), we can see that $|\tilde{\phi}_0(z)|$ decays as $\mathcal{O}(|z|^{d_f-d_g})$ for $|z|>1$. This means, the smaller the degree d_f is, the faster $|\tilde{\phi}_0(z)|$ decays outside $\mathcal{C}_1(0)$ and thus the better the quadrature approximation is. The next theorem compares two popular quadrature rules: the Trapezoidal rule and the Gauss-Legendre quadrature, in terms of the degree d_f.

Theorem 3.2.1 *For $\tilde{\phi}_0(z)$ in (3.9)–(3.11), the degree d_f of $f(z)$ satisfies:*

1. *If the Trapezoidal rule is used, where $t_j=-1+\frac{2(j-1)}{q}$ and $w_j=\frac{2}{q}$, then*

$$d_f=0 \quad (\textit{in fact, } f(z)=(-1)^{q+1}).$$

   That is, the Trapezoidal rule gives the optimal d_f.

2. *If the Gauss-Legendre quadrature is used, where $\{t_j\}_{j=1}^q$ are the roots of the Legendre polynomial of degree q and $\{w_j\}_{j=1}^q$ are the corresponding weights, then*

$$d_f\geq 1.$$

Proof Comparing (3.9) and (3.11) yields

$$f(z)=-\frac{1}{2}\sum_{j=1}^{q}w_jz_j\prod_{i\neq j}(z-z_i).$$

Let the coefficient of the term z^{q-k} in $f(z)$ be C_{q-k} for $1\leq k\leq q$, which has the following form:

$$C_{q-k}=\frac{(-1)^k}{2}\sum_{1\leq i_1<i_2<\cdots<i_k\leq q}(w_{i_1}+w_{i_2}+\cdots+w_{i_k})z_{i_1}z_{i_2}\cdots z_{i_k}. \tag{3.12}$$

For the Trapezoidal rule, the mapped quadrature nodes z_j in (3.10) satisfy

$$z_j^q=e^{\mathbf{i}\pi qt_j}=e^{\mathbf{i}\pi(2j-2-q)}=(-1)^q, \quad 1\leq j\leq q.$$

Hence, z_j's are the roots of $z^q-(-1)^q$, so that

$$g(z)=z^q-(-1)^q. \tag{3.13}$$

Since all the weights w_j are equal, (3.12) can be simplified as

$$C_{q-k} = \frac{k}{q}\Big[(-1)^k \sum_{1\le i_1<i_2<\cdots<i_k\le q} z_{i_1} z_{i_2} \cdots z_{i_k}\Big], \quad 1 \le k \le q.$$

Note that the part in parenthesis in the above equation is the coefficient of the term z^{q-k} in the polynomial $g(z)$ in (3.11) and also in (3.13). Thus,

$$C_{q-k} = 0,\ 1 \le k \le q-1, \quad C_0 = (-1)^{q+1}.$$

Therefore, $f(z) = (-1)^{q+1}$ and $d_f = 0$.

For the Gauss-Legendre quadrature, we prove the result by contradiction. Suppose $d_f = 0$. Some well-known properties of the Gauss-Legendre quadrature are

$$\sum_{j=1}^{q} t_j = 0, \quad t_j + t_{q+1-j} = 0, \quad w_j = w_{q+1-j}, \quad 1 \le j \le q, \tag{3.14}$$

where we assume $t_1 < t_2 < \cdots < t_q$. As a result, the mapped nodes satisfy

$$\prod_{j=1}^{q} z_j = 1, \quad z_j z_{q+1-j} = 1, \quad 1 \le j \le q. \tag{3.15}$$

Define $\mathcal{S}_k = \{(i_1, i_2, \ldots, i_k) : 1 \le i_1 < i_2 < \cdots < i_k \le q\}$ to be the set of index sequences of the summation in (3.12). Then for any $1 \le k \le q-1$, the two sets $\mathcal{S}_k$ and $\mathcal{S}_{q-k}$ have a one-to-one correspondence in the sense that, for any sequence $\sigma \in \mathcal{S}_k$, there is a unique sequence $\beta \in \mathcal{S}_{q-k}$ such that $\sigma \cup \beta = \{1, 2, \ldots, q\}$ and $\sigma \cap \beta = \varnothing$. Therefore, for any $1 \le k \le q-1$, similar to (3.12),

$$C_k = \frac{(-1)^{q-k}}{2} \sum_{(i_1,i_2,\ldots,i_{q-k})\in\mathcal{S}_{q-k}} (w_{i_1} + w_{i_2} + \cdots + w_{i_{q-k}}) z_{i_1} z_{i_2} \cdots z_{i_{q-k}}.$$

We can then use (3.14) and (3.15) to get

$$\begin{aligned}
C_k &= \frac{(-1)^{q-k}}{2} \sum_{(i_1,i_2,\ldots,i_k)\in\mathcal{S}_k} (2 - (w_{i_1} + w_{i_2} + \cdots + w_{i_k})) \frac{1}{z_{i_1} z_{i_2} \cdots z_{i_k}} \\
&= \frac{(-1)^{q-k}}{2} \sum_{(i_1,i_2,\ldots,i_k)\in\mathcal{S}_k} (2 - (w_{q+1-i_1} + \cdots + w_{q+1-i_k})) z_{q+1-i_1} \cdots z_{q+1-i_k} \\
&= \frac{(-1)^{q-k}}{2} \sum_{(i_1,i_2,\ldots,i_k)\in\mathcal{S}_k} (2 - (w_{i_1} + w_{i_2} + \cdots + w_{i_k})) z_{i_1} z_{i_2} \cdots z_{i_k}
\end{aligned}$$

$$
\begin{aligned}
&= \left((-1)^{q-k} \sum_{(i_1,i_2,\ldots,i_k)\in\mathcal{S}_k} z_{i_1} z_{i_2} \cdots z_{i_k} \right) \\
&\quad - \left(\frac{(-1)^{q-k}}{2} \sum_{(i_1,i_2,\ldots,i_k)\in\mathcal{S}_k} (w_{i_1} + w_{i_2} + \cdots + w_{i_k}) z_{i_1} z_{i_2} \cdots z_{i_k} \right) \\
&= \left((-1)^{q-k} \sum_{(i_1,i_2,\ldots,i_k)\in\mathcal{S}_k} z_{i_1} z_{i_2} \cdots z_{i_k} \right) - (-1)^{q-2k} C_{q-k}.
\end{aligned}
$$

By assumption, we have $C_k = 0$, $C_{q-k} = 0$ for $1 \le k \le q-1$, so

$$
\sum_{(i_1,i_2,\ldots,i_k)\in\mathcal{S}_k} z_{i_1} z_{i_2} \cdots z_{i_k} = 0, \quad 1 \le k \le q-1.
$$

The above equation together with $\prod_{j=1}^{q} z_j = 1$ indicate that z_j in (3.10) must be roots of the polynomial $z^q + (-1)^q$. Thus, the roots of Legendre polynomial must be

$$
t_j = -1 + \frac{2j-1}{q}, \quad j = 1, 2, \ldots, q.
$$

This is clearly a contradiction, and hence $d_f \ge 1$. ■

This theorem indicates that the filter function $\tilde{\phi}_0(z)$ from the Trapezoidal rule decays as

$$
|\tilde{\phi}_0(z)| \sim \mathcal{O}(|z|^{-q}) \quad \text{for large } |z|,
$$

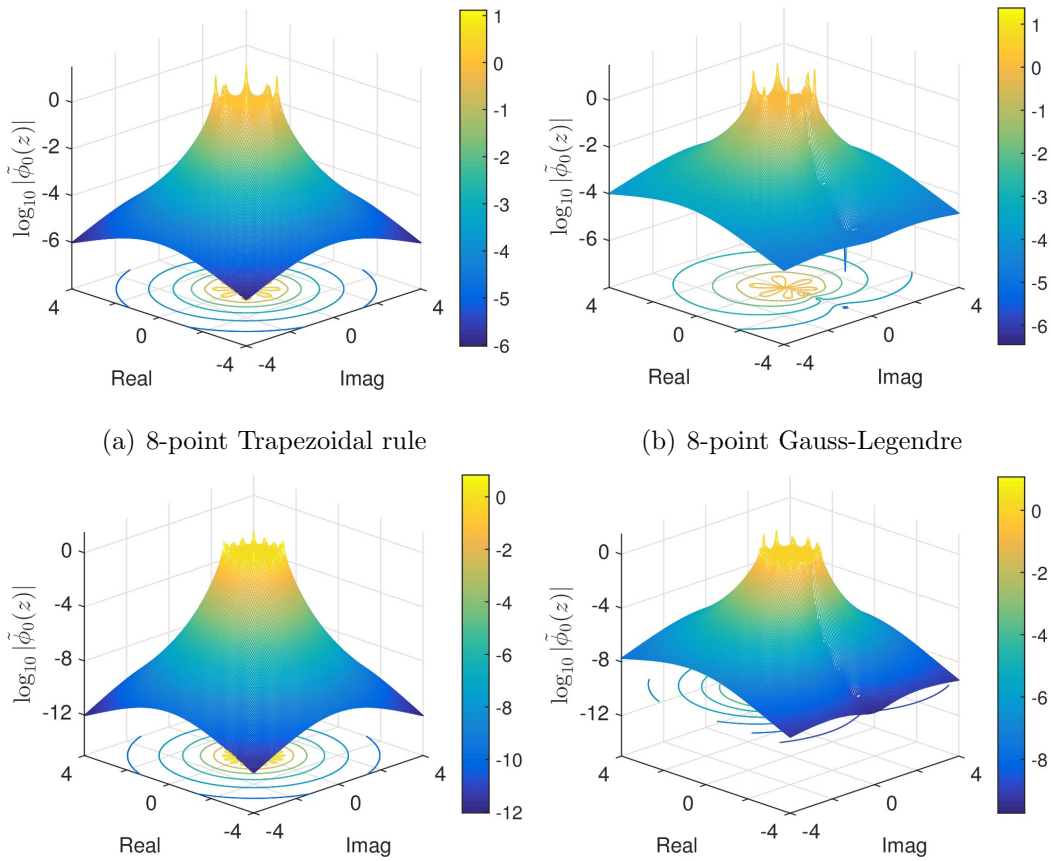
Thus, the Trapezoidal rule yields nearly *optimal decay.* The decay in the Gauss-Legendre case is at best $\mathcal{O}(|C_1||z|^{1-q})$, where

$$
C_1 = \sum_{j=1}^{q} (1 + w_j) \cos(\pi t_j).
$$

It can be verified numerically, though not analytically yet, that $|C_1|$ is not small and actually increases when q increases.

To illustrate the decay, we plot $|\tilde{\phi}_0(z)|$ from the two quadrature rules with $q = 8$ and 16 in Figure 3.1. Outside $\mathcal{C}_1(0)$, $|\tilde{\phi}_0(z)|$ decays quickly when $|z|$ increases, and moreover, the Trapezoidal rule yields much faster decay than the Gauss-Legendre quadrature. For $q = 8$, $|\tilde{\phi}_0(z)|$ from the Trapezoidal rule is about two orders of magnitude smaller at the corners of the mesh (with $|z|$ not even very large). For $q = 16$, this difference increases to over four orders of magnitude.

Theorem 3.2.1 and Figure 3.1 also align with the numerical observations in [100, 104]. In [104], an optimization method is used to design filter functions, and in the unit circle case, the best filter function is observed to be precisely the one obtained by applying the Trapezoidal rule. Our analysis provides a theoretical justification.



(a) 8-point Trapezoidal rule

(b) 8-point Gauss-Legendre

(c) 16-point Trapezoidal rule

(d) 16-point Gauss-Legendre

Figure 3.1. $\log_{10}|\tilde{\phi}_0(z)|$ on the $[-4,4]\times[-4,4]$ mesh obtained with the Trapezoidal rule and the Gauss-Legendre rule.

Therefore, unlike in [57, 90, 125], our eigensolver below uses the Trapezoidal rule to evaluate (3.5) in both the eigenvalue counts and the subspace iterations.

3.3 Low-accuracy matrix approximation for fast eigenvalue counts

3.3.1 Motivations

In contour-integral eigensolvers, it usually requires to know the eigenvalue count $\#_\Lambda(A, \Gamma)$ inside a contour Γ in advance. In our eigensolver in the next section, we may need to estimate eigenvalue counts for many subregions, so it is essential to quickly perform the estimation. Some methods to estimate eigenvalue counts have been proposed in [28, 85, 94, 125] based on stochastic evaluations of the rank or trace [50] of Φ in (3.4). The basic idea is as follows.

According to (3.4), the trace and also the rank of Φ give the exact eigenvalue count $\#_\Lambda(A, \Gamma)$. To estimate the trace, we can pick a small number of random vectors to form an $n \times m$ matrix Y, and compute $Y^T \Phi Y = Y^T Z$, where Z looks like (3.5). Then

$$\#_\Lambda(A, \Gamma) \equiv \text{trace}(\Phi) \approx \frac{1}{m} \text{trace}(Y^T Z). \tag{3.16}$$

Theoretically, a small number m can lead to a high probability of accurate estimation. However, since Z in (3.5) is approximated by numerical quadratures, m may not be too small. We can start from a very small m and gradually include more random vectors in Y until a reliable estimate is reached.

In the eigenvalue counts with quadrature approximations, it needs to solve linear systems for $\mu I - A$ with multiple shifts μI, multiple right-hand sides, and for possibly many contours, which amounts to a significant computational cost. However, notice the following important aspects:

1. Since we are just interested in the eigenvalue count (at this stage) instead of the precise eigenvalues, as long as the eigenvalues are not too close to Γ, a small perturbation to A does not alter the eigenvalue count.

2. Moreover, in our eigensolver, we will quadsect a search region containing the eigenvalues and only need to know whether the eigenvalue count inside each subregion is much larger than a threshold k or not. Thus, the eigenvalue count does not even have to be very accurate.

As a result, we can use a matrix $\tilde{A}$ that approximates A and satisfies the following two requirements:

1. $\#_\Lambda(\tilde{A}, \Gamma) \approx \#_\Lambda(A, \Gamma)$ and it is convenient to control how accurately $\tilde{A}$ approximates A;

2. $\#_\Lambda(\tilde{A}, \Gamma)$ can be quickly estimated, i.e., the linear systems with multiple shifts and right-hand sides in the quadrature approximation of (3.5) (with A replaced by $\tilde{A}$) can be quickly solved.

A natural tool that satisfies both requirements is the rank structure, which allows fast direct factorizations. (Note that the fundamental approximation analysis for the eigenvalue count in this section is not restricted to rank structured forms.) In particular, HSS type methods is a convenient algebraic tool with systematic error control, stability analysis, and fast factorizations. In the next section, we will further show the feasibility of updating the factorization for multiple shifts. More general $\mathcal{H}$-matrix representations may be used to accommodate even broader applications, though it is not clear how to perform fast factorization updates for varying shifts.

Before justifying the reliability of our low-accuracy matrix approximation for fast eigenvalue counts, we briefly review HSS representations. The reader is referred to [12, 117] for more details. An HSS matrix $\tilde{A}$ can be recursively bipartitioned following a postordered binary tree T (called HSS tree) with nodes $i = 1, 2, \ldots, t$, where t is the root. Initially, let $D_t = \tilde{A}$. For any nonleaf node i of T, the partition of D_i looks like $D_i = \begin{pmatrix} D_{c_1} & U_{c_1} B_{c_1} V_{c_2}^T \\ U_{c_2} B_{c_2} V_{c_1}^T & D_{c_2} \end{pmatrix}$, where c_1 and c_2 are the children of i. Here, the off-diagonal basis matrices U, V also satisfy a nested property: $U_i = \begin{pmatrix} U_{c_1} R_{c_1} \\ U_{c_2} R_{c_2} \end{pmatrix}$, $V_i = \begin{pmatrix} V_{c_1} W_{c_1} \\ V_{c_2} W_{c_2} \end{pmatrix}$. All such matrices D, U, V, R, W, B are called HSS generators that define $\tilde{A}$. The block row or column corresponding to D_i but excluding D_i is called an HSS block. The HSS matrix has l levels of partition if the HSS tree T has l levels, where the root is at level 0 and the leaves are at level l. The maximum rank (or numerical rank) of all the HSS blocks at all the levels is called the *HSS rank*.

A matrix is rank structured if all its off-diagonal blocks have small ranks or numerical ranks. That is, the singular values of the off-diagonal blocks decay quickly. Here to be more specific, by saying a matrix is rank structured, we mean it can be accurately approximated by a compact HSS form.

3.3.2 Reliability of eigenvalue count with low-accuracy matrix approximation

In our eigensolver, we will use an HSS form $\tilde{A}$ to approximate A. To see how such an approximation perturbs the eigenvalues, we give following lemma that extends a Hermitian version in [107].

Lemma 3.3.1 *Suppose A has simple eigenvalues, and $\tilde{A}$ is an l-level HSS approximation to A in (3.2), so that each off-diagonal block $U_i B_i V_j^T$ of $\tilde{A}$ approximates the corresponding block in A to an accuracy τ which is sufficiently small. Let λ be a simple eigenvalue of A, then there exists an eigenvalue $\tilde{\lambda}$ of $\tilde{A}$ such that*

$$|\lambda - \tilde{\lambda}| \leq \kappa(\lambda) l\tau + \mathcal{O}((l\tau)^2), \tag{3.17}$$

where $\kappa(\lambda)$ is the 2-norm condition number of λ.

The lemma follows directly from the HSS approximation error $\|A - \tilde{A}\|_2 \leq l\tau$ [107] and standard eigenvalue perturbation analysis [4, 19].

Throughout this section, we will assume that all eigenvalues λ_i of A are simple and the perturbation to the matrix is sufficiently small, so as to identify a one-to-one correspondence between the eigenvalues of A and those of its approximation $\tilde{A}$. More specifically, Lemma 3.3.1 indicates that for any eigenvalue λ_i, there must be a perturbed eigenvalue $\tilde{\lambda}$ within a disk centered at λ_i and with radius $\kappa(\lambda_i) l\tau + \mathcal{O}((l\tau)^2)$. $\tilde{\lambda}$ is unique if this disk is isolated from all the other such disks which yields the desired one-to-one correspondence, we can guarantee this by enforcing the following sufficient condition:

$$\tilde{\kappa} l\tau + \mathcal{O}((l\tau)^2) \leq \frac{1}{2} \min_{1 \leq i,j \leq n, i \neq j} |\lambda_i - \lambda_j|, \tag{3.18}$$

where $\tilde{\kappa}$ is a sharp upper bound for all $\kappa(\lambda_i)$. In more general cases when any approximation $\tilde{A}$ is used, the following analogous condition is assumed:

$$\tilde{\kappa}\|A - \tilde{A}\| + \mathcal{O}(\|A - \tilde{A}\|^2) \leq \frac{1}{2} \min_{1 \leq i,j \leq n, i \neq j} |\lambda_i - \lambda_j|. \tag{3.19}$$

The following theorem shows when $\tilde{A}$ can be used to obtain the exact eigenvalue count inside $\mathcal{C}_\gamma(z)$, and also gives a necessary condition for the eigenvalue count to be off by a certain number. We assume the perturbations to the eigenvalues are strictly bounded by δ, which is related to the perturbation in the matrix according to the discussions above.

Theorem 3.3.2 *Suppose A has simple eigenvalues λ with $|\lambda| < \rho$, $\tilde{A}$ is an approximation to A satisfying (3.19), and any eigenvalue λ of A and the corresponding eigenvalue $\tilde{\lambda}$ of $\tilde{A}$ satisfy*

$$|\lambda - \tilde{\lambda}| < \delta < \gamma < \rho. \tag{3.20}$$

1. *If A has no eigenvalue inside $\mathcal{A}_{\gamma,\delta}(z)$, then*

$$\#_\Lambda(A, \mathcal{C}_\gamma(z)) = \#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z)).$$

2. *If $|\#_\Lambda(A, \mathcal{C}_\gamma(z)) - \#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z))| \geq \alpha$ for an integer $\alpha > 0$, then there must be at least α eigenvalues of A inside $\mathcal{A}_{\gamma,\delta}(z)$.*

Proof Figure 3.2 can be used to assist in the understanding of the results and proof.

The first statement can be shown as follows. Since no eigenvalue of A lies inside $\mathcal{A}_{\gamma,\delta}(z)$, any eigenvalue λ satisfies $|\lambda - z| \geq \gamma + \delta$ or $|\lambda - z| \leq \gamma - \delta$. If $|\lambda - z| \geq \gamma + \delta$, according to (3.20),

$$|\tilde{\lambda} - z| = |\lambda - z - (\lambda - \tilde{\lambda})| \geq |\lambda - z| - |\lambda - \tilde{\lambda}| > \gamma + \delta - \delta = \gamma.$$

Thus, $\tilde{\lambda}$ is outside $\mathcal{C}_\gamma(z)$, just like λ. If $|\lambda - z| \leq \gamma - \delta$, then

$$|\tilde{\lambda} - z| = |\tilde{\lambda} - \lambda + \lambda - z| \leq |\tilde{\lambda} - \lambda| + |\lambda - z| < \delta + \gamma - \delta = \gamma.$$

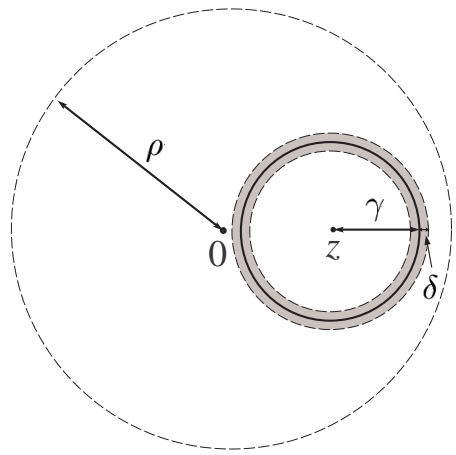


Figure 3.2. The annulus region $\mathcal{A}_{\gamma,\delta}(z)$ (shaded area) related to a circle $\mathcal{C}_\gamma(z)$, where the outer disk $\mathcal{D}_\rho(0)$ is where all the eigenvalues are located.

Thus, $\tilde{\lambda}$ is inside $\mathcal{C}_\gamma(z)$, just like λ. That is, λ and $\tilde{\lambda}$ must be both inside or outside $\mathcal{C}_\gamma(z)$. Then A and $\tilde{A}$ have the same number of eigenvalues inside $\mathcal{C}_\gamma(z)$, and the first statement holds.

We then show the second statement by contradiction. Suppose there are less than α eigenvalues of A inside $\mathcal{A}_{\gamma,\delta}(z)$. If $\#_\Lambda(A, \mathcal{C}_\gamma(z)) \geq \#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z))$, let n_1 be the number of eigenvalues of A satisfying $|\lambda - z| \leq \gamma - \delta$. Then $\#_\Lambda(A, \mathcal{C}_\gamma(z)) < n_1 + \alpha$. Also according to the proof above, $\#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z)) \geq n_1$. Thus,

$$|\#_\Lambda(A, \mathcal{C}_\gamma(z)) - \#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z))| = \#_\Lambda(A, \mathcal{C}_\gamma(z)) - \#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z)) < n_1 + \alpha - n_1 = \alpha.$$

Thus, we get a contradiction. Similarly, if $\#_\Lambda(A, \mathcal{C}_\gamma(z)) < \#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z))$, let n_1 be the number of eigenvalues of $\tilde{A}$ satisfying $|\tilde{\lambda} - z| \leq \gamma - \delta$. Then $\#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z)) < n_1 + \alpha$, $\#_\Lambda(A, \mathcal{C}_\gamma(z)) \geq n_1$, and we similarly get $\#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z)) - \#_\Lambda(A, \mathcal{C}_\gamma(z)) < \alpha$ and thus a contradiction. ■

Theorem 3.3.2 means, if the contour is not too close to the eigenvalues, then $\tilde{A}$ can be used to obtain the exact eigenvalue count. The farther away the contour is from the eigenvalues, the lower accuracy of $\tilde{A}$ can be used. This is especially effective if the eigenvalues are scattered. On the other hand, if the eigenvalue count with $\tilde{A}$ is

off by α or more, then there must be at least α eigenvalues within a distance δ of the contour. We then use probabilistic methods to study the error in the count based on the relation between the eigenvalues and $\mathcal{A}_{\gamma,\delta}(z)$.

Lemma 3.3.3 *Suppose the eigenvalues λ of A are uniformly i.i.d. in $\mathcal{D}_\rho(0)$. Then for any fixed $z \in \mathbb{C}$ and $\gamma, \delta \in (0, \rho)$, the probability for any λ to lie inside $\mathcal{A}_{\gamma,\delta}(z)$ satisfies*

$$\Pr\{\lambda \in \mathcal{A}_{\gamma,\delta}(z)\} \leq \mathcal{P} \equiv \frac{4\delta \max(\gamma, \delta)}{\rho^2}. \tag{3.21}$$

Proof The probability density function for λ has the form $\psi(\hat{\lambda}) = \begin{cases} \frac{1}{\pi\rho^2}, & |\hat{\lambda}| < \rho, \\ 0, & |\hat{\lambda}| \geq \rho. \end{cases}$

If $\gamma \geq \delta$,

$$\Pr\{\lambda \in \mathcal{A}_{\gamma,\delta}(z)\} \leq \iint_{\gamma-\delta<|\hat{\lambda}-z|<\gamma+\delta} \psi(\hat{\lambda})\mathrm{d}\hat{\lambda} = \frac{\pi(\gamma+\delta)^2 - \pi(\gamma-\delta)^2}{\pi\rho^2} = \frac{4\delta\gamma}{\rho^2}.$$

If $\gamma < \delta$,

$$\Pr\{\lambda \in \mathcal{A}_{\gamma,\delta}(z)\} \leq \iint_{|\hat{\lambda}-z|<\gamma+\delta} \psi(\hat{\lambda})\mathrm{d}\hat{\lambda} = \frac{\pi(\gamma+\delta)^2}{\pi\rho^2} < \frac{4\delta^2}{\rho^2}.$$

The result holds in both cases. (Note that the bounds may highly overestimate the probability when $\mathcal{A}_{\gamma,\delta}(z)$ is not fully inside $\mathcal{D}_\rho(0)$.) ■

Lemma 3.3.3 gives a probability bound for λ to fall inside $\mathcal{A}_{\gamma,\delta}(z)$ when the eigenvalues are random and uniformly distributed in $\mathcal{D}_\rho(0)$. Thus, if A is approximated by $\tilde{A}$ as in Theorem 3.3.2, then the probability of incorrectly counting λ for $\#_\Lambda(A, \mathcal{C}_\gamma(z))$ is at most $\mathcal{P}$. If $\tilde{A}$ is an HSS approximation as in Lemma 3.3.1, δ can be chosen to be a strict upper bound for the error in (3.17).

In addition, Lemma 3.3.3 means that, the larger ρ is or the smaller δ and γ are, the less likely λ falls inside $\mathcal{A}_{\gamma,\delta}(z)$. In particular, later in our eigensolver, since the search region is recursively partitioned into smaller ones, γ gets smaller along the partition and so does the probability $\mathcal{P}$. This combined with Theorem 3.3.2 means that it is more likely to get reliable eigenvalue counts based on $\tilde{A}$.

Lemma 3.3.3 assumes the circle $\mathcal{C}_\gamma(z)$ is fixed and the eigenvalues are random. We can also assume an eigenvalue λ is fixed and $\mathcal{C}_\gamma(z)$ is random, and study the probability of $\mathcal{A}_{\gamma,\delta}(z)$ to include λ.

Lemma 3.3.4 *Suppose λ is a fixed point in the complex plane, z is uniformly i.i.d. in $\mathcal{D}_\rho(0)$, γ is random and uniformly distributed on $(0,\rho)$, and z and γ are independent. Then for any $\delta \in (0,\rho)$,*

$$\Pr\{\lambda \in \mathcal{A}_{\gamma,\delta}(z)\} < 2\frac{\delta}{\rho} + \frac{1}{3}\left(\frac{\delta}{\rho}\right)^3.$$

Proof The probability density function for γ has the form $\varphi(\hat{\gamma}) = \begin{cases} \frac{1}{\rho}, & 0 < \hat{\gamma} < \rho, \\ 0, & \text{otherwise.} \end{cases}$

By the law of total probability,

$$\begin{aligned} \Pr\{\lambda \in \mathcal{A}_{\gamma,\delta}(z)\} &= \int_0^\rho \Pr\{\lambda \in \mathcal{A}_{\gamma,\delta}(z) \mid \gamma = \hat{\gamma}\}\varphi(\hat{\gamma})\mathrm{d}\hat{\gamma} \\ &= \frac{1}{\rho}\int_0^\delta \Pr\{|z-\lambda| < \hat{\gamma}+\delta\}\mathrm{d}\hat{\gamma} + \frac{1}{\rho}\int_\delta^\rho \Pr\{\hat{\gamma}-\delta < |\lambda - z| < \hat{\gamma}+\delta\}\mathrm{d}\hat{\gamma}. \end{aligned}$$

Similarly to the proof of Lemma 3.3.3, we can get

$$\Pr\{|z-\lambda| < \hat{\gamma}+\delta\} \le \frac{(\hat{\gamma}+\delta)^2}{\rho^2}, \quad \Pr\{\hat{\gamma}-\delta < |\lambda - z| < \hat{\gamma}+\delta\} \le \frac{4\delta\hat{\gamma}}{\rho^2}.$$

Thus,

$$\Pr\{\lambda \in \mathcal{A}_{\gamma,\delta}(z)\} \le \frac{1}{\rho}\int_0^\delta \frac{(\hat{\gamma}+\delta)^2}{\rho^2}\mathrm{d}\hat{\gamma} + \frac{1}{\rho}\int_\delta^\rho \frac{4\delta\hat{\gamma}}{\rho^2}\mathrm{d}\hat{\gamma} = 2\frac{\delta}{\rho} + \frac{1}{3}\left(\frac{\delta}{\rho}\right)^3.$$

■

We then give the probability for miscounting $\#_\Lambda(A, \mathcal{C}_\gamma(z))$ when the eigenvalues of A are random and A is approximated by an HSS form $\tilde{A}$.

Theorem 3.3.5 *Suppose the eigenvalues of A are uniformly i.i.d. in $\mathcal{D}_\rho(0)$, and $\tilde{A}$ is an l-level HSS approximation to A as in Lemma 3.3.1 and satisfies (3.18). Also, suppose $\delta < \rho$ is a strict upper bound for the right-hand side in (3.17) for all the*

eigenvalues. Let $\mathcal{P}$ be given in (3.21). Then for any integer $\alpha \geq n\mathcal{P}$ and any fixed $z \in \mathbb{C}$ and $\gamma \in (0, \rho)$,

$$\Pr\{|\#_\Lambda(A, \mathcal{C}_\gamma(z)) - \#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z))| \geq \alpha\} \leq \frac{(\alpha+1)}{\alpha+1-(n+1)\mathcal{P}} \binom{n}{\alpha} \mathcal{P}^\alpha (1-\mathcal{P})^{n-\alpha+1}. \tag{3.22}$$

Proof According to Theorem 3.3.2,

$$\begin{aligned} &\Pr\{|\#_\Lambda(A, \mathcal{C}_\gamma(z)) - \#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z))| \geq \alpha\} \\ \leq &\Pr\{\text{there are at least } \alpha \text{ eigenvalues of } A \text{ in } \mathcal{A}_{\gamma,\delta}(z)\}. \end{aligned} \tag{3.23}$$

Now from Lemmas 3.3.1 and 3.3.3, the eigenvalues satisfy

$$\Pr\{\lambda \in \mathcal{A}_{\gamma,\delta}(z)\} \equiv \widehat{\mathcal{P}} \leq \mathcal{P}. \tag{3.24}$$

Let $\hat{y}$ be the number of eigenvalues inside $\mathcal{A}_{\gamma,\delta}(z)$. Since the eigenvalues are i.i.d., $\hat{y}$ has a binomial distribution with parameters $\widehat{\mathcal{P}}$ and n. Also, let y be a binomial random variable with parameters $\mathcal{P}$ and n. Thus, (3.23) and (3.24) yield

$$\Pr\{|\#_\Lambda(A, \mathcal{C}_\gamma(z)) - \#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z))| \geq \alpha\} \leq \Pr\{\hat{y} \geq \alpha\} \leq \Pr\{y \geq \alpha\}.$$

Since $\alpha \geq n\mathcal{P}$, by [54, Proposition 1], the tail probability of the binomial random variable y is bounded by

$$\Pr\{y \geq \alpha\} \leq \frac{(\alpha+1)(1-\mathcal{P})}{\alpha+1-(n+1)\mathcal{P}} \binom{n}{\alpha} \mathcal{P}^\alpha (1-\mathcal{P})^{n-\alpha}.$$

The result then follows. ■

The theorem can be understood as follows. Due to the term $\mathcal{P}^\alpha$, roughly speaking, the probability of miscounting the eigenvalues by α decays exponentially with α for reasonably small $\mathcal{P}$. Thus, the probability is very small even for modest α. This is sufficient for us since we only need an estimate of the count.

To give an idea of this probability bound in (3.22), we show it with different eigenvalue perturbation errors δ. See Table 3.1, where the parameters correspond to

Table 3.1.
Bounds for the probability of miscounting the number of eigenvalues inside $\mathcal{C}_\gamma(z)$ by α or more, where $n = 1600$, $\rho = 4000$.

γ	δ	Bound for $\Pr\{\lvert\#_\Lambda(A, \mathcal{C}_\gamma(z)) - \#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z))\rvert \geq \alpha\}$				
		$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$	$\alpha = 5$
	$1e-1$	$3.99e-3$	$7.97e-6$	$1.06e-8$	$1.06e-11$	$8.45e-15$
100	$1e-2$	$4.00e-4$	$7.99e-8$	$1.06e-11$	$1.06e-15$	$8.48e-20$
	$1e-3$	$4.00e-5$	$7.99e-10$	$1.06e-14$	$1.06e-19$	$8.48e-25$
	$1e-1$	$3.92e-2$	$7.79e-4$	$1.03e-5$	$1.03e-7$	$8.20e-10$
1000	$1e-2$	$3.99e-3$	$7.97e-6$	$1.06e-8$	$1.06e-11$	$8.45e-15$
	$1e-3$	$3.99e-4$	$7.99e-8$	$1.06e-11$	$1.06e-15$	$8.48e-20$

a matrix in Example 1 below. Clearly, even though δ is not very small, the probability of miscounting the number of eigenvalues by $\alpha > 2$ is extremely low. When α slightly increases and/or δ decreases, the probability decreases rapidly.

We would also like to mention that Theorem 3.3.5 is still a very conservative estimate. For example, consider A to be the matrix with size $n = 1600$ in Example 1 below. Let $\tilde{A}$ be an HSS approximation obtained with a relative tolerance $\tau = 10^{-1}, 10^{-2}, \ldots, 10^{-5}$. We run the eigenvalue counts for 100 randomly selected circles. For 57 of the cases, we get the exact counts for all these τ's. For the other cases, $\#_\Lambda(A, \mathcal{C}_\gamma(z_0))$ and $\#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z_0))$ differ by a very small number with $\tau = 10^{-1}$ or 10^{-2}. With smaller τ, exact counts are obtained for almost all the cases. Table 3.2 shows some of the results.

Table 3.2 also shows the HSS ranks r. Note that when τ reduces from 10^{-1} to 10^{-4} (and all the counts then become exact), the HSS rank increases from $r = 4$ to $r = 11$. Since HSS factorization and solution have asymptotic complexities $\mathcal{O}(r^2 n)$ and $\mathcal{O}(rn)$ [117], respectively, using $\tau = 10^{-1}$ makes the factorization about 7.6 times

Table 3.2.
Eigenvalue counts of A and $\tilde{A}$ inside some circles $\mathcal{C}_\gamma(z)$, where A is a Cauchy-like matrix corresponding to $n = 1600$ in Example 1 below, τ is the relative tolerance in a randomized HSS construction, and r is the HSS rank.

z	γ	$\#_\Lambda(A, \mathcal{C}_\gamma(z))$	$\lvert\#_\Lambda(A, \mathcal{C}_\gamma(z)) - \#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z))\rvert$				
			$\tau = 10^{-1}$	10^{-2}	10^{-3}	10^{-4}	10^{-5}
			$r = 4$	7	9	11	14
$976.8517 - 596.6716\mathbf{i}$	109.5545	2	0	0	0	0	0
$122.4701 + 395.7090\mathbf{i}$	221.7331	42	1	0	0	0	0
$-250.9437 + 91.2499\mathbf{i}$	395.2032	147	1	0	0	0	0
$-1029.6903 - 1599.1273\mathbf{i}$	986.0082	127	1	1	0	0	0
$1646.1010 + 2850.7448\mathbf{i}$	1315.6815	10	0	0	0	0	0
$-493.2565 + 1022.0571\mathbf{i}$	1526.3885	865	0	0	0	0	0
$115.6055 - 2472.7009\mathbf{i}$	2063.6158	400	2	0	0	0	0
$-1014.5968 + 1995.9028\mathbf{i}$	3004.7346	1220	1	0	0	0	0
$660.5523 + 507.5861\mathbf{i}$	3954.0531	1596	0	0	0	0	0

faster and the solution about 2.8 times faster than using $\tau = 10^{-4}$. For examples where the HSS ranks are higher, the difference is even bigger. See Example 2 below. This clearly demonstrates the benefit of low-accuracy matrix approximations for the eigenvalue count.

3.4 Our fast contour-integral eigensolver

In this section, we show the design of our fast contour-integral eigensolver for finding a partial spectrum or the full spectrum of A. We will start from an initial contour that encloses the desired eigenvalues, and then repeatedly *quadsect* the search

region into smaller subregions. When the total number of desired eigenvalues is large, a significant amount of efforts is to make sure each subregion includes no more than a certain number of eigenvalues. Before a contour gets very close to the eigenvalues, the discussions in the previous section indicate that we can use low-accuracy approximations to A to obtain a reliable count of the eigenvalues enclosed by the contour.

We first briefly review the non-Hermitian FEAST algorithm, and then discuss our fast eigensolver based on several strategies for accelerating the FEAST algorithm.

3.4.1 Review of the non-Hermitian FEAST algorithm

The basic procedure of the FEAST algorithm for non-Hermitian matrices is as follows [53,57,90,100,125]. Consider the case when Γ in (3.4) is a circle $\mathcal{C}_\gamma(z_0)$. The matrix Z in (3.5) is used to extract an approximation to the eigenspace $\text{span}\{x_1, x_2, \ldots, x_s\}$. Φ in (3.4) can be approximated by numerical integration:

$$\widetilde{\Phi} = \frac{1}{2}\sum_{j=1}^{q} w_j(z_j - z_0)(z_j I - A)^{-1},$$

where $z_j = z_0 + \gamma e^{i\pi t_j},\ j = 1, 2, \ldots, q$ are the mapped quadrature nodes on $\mathcal{C}_\gamma(z_0)$. Then Z can be approximated by

$$\widetilde{Z} = \widetilde{\Phi} Y = \frac{1}{2}\sum_{j=1}^{q} w_j(z_j - z_0)(z_j I - A)^{-1} Y \equiv \frac{1}{2}\sum_{j=1}^{q} c_j S_j, \tag{3.25}$$

where $c_j = w_j(z_j - z_0)$ and S_j's are solutions to the shifted linear systems

$$(z_j I - A) S_j = Y, \quad j = 1, 2, \ldots, q. \tag{3.26}$$

Solve the linear systems and evaluate $\widetilde{Z}$, which is used to compute the desired eigenpairs in Rayleigh-Ritz iterations. This is summarized in Algorithm 3.

In steps 2–3 of Algorithm 3, it is sufficient for convergence when the initial subspace size $\hat{s}$ is not smaller than the actual eigenvalue count. To ensure a good overall convergence rate, it is preferable to make $\hat{s}$ a little larger than the actual eigenvalue

Algorithm 3 *Basic FEAST algorithm with projected subspace iteration* [90, 125]

Input: A, $\mathcal{C}_\gamma(z_0)$ (contour)

Output: $(\widehat{\Lambda}, \widehat{X})$ (eigenvalues inside $\mathcal{C}_\gamma(z_0)$ and the corresponding eigenvectors)

1: **procedure** FEAST
2: $\quad \hat{s} \leftarrow$ upper bound of $\#_\Lambda(A, \mathcal{C}_\gamma(z_0))$ ▷ *Initial subspace size*
3: $\quad Y \leftarrow n \times \hat{s}$ Gaussian random matrix
4: $\quad c_j = w_j(z_j - z_0),\ j = 1, \ldots, q$ ▷ *Weights w_j & nodes z_j in a quadrature rule*
5: $\quad$ **repeat**
6: $\quad\quad S_j \leftarrow (z_j I - A)^{-1} Y, \quad j = 1, \ldots, q$ ▷ *Solving* (3.26)
7: $\quad\quad \widetilde{Z} \leftarrow \frac{1}{2} \sum_{j=1}^{q} c_j S_j$ ▷ *Evaluating $\widetilde{Z} = \widetilde{\Phi} Y$ by* (3.25)
8: $\quad\quad \widetilde{Q} \leftarrow$ orthonormal basis of $\widetilde{Z}$ ▷ *This is important and is added in [125]*
9: $\quad\quad \hat{A} \leftarrow \widetilde{Q}^T A \widetilde{Q}$ ▷ *Reduced problem*
10: $\quad\quad \hat{A} = \widehat{X} \widehat{\Lambda} \widehat{X}^{-1}$ ▷ *Solving the reduced eigenvalue problem*
11: $\quad\quad Y \leftarrow \widetilde{Q} \widetilde{X}$ ▷ *Recovery of approximate eigenvectors of A*
12: $\quad$ **until** convergence
13: $\quad \widehat{X} \leftarrow Y$ ▷ *Convergent approximate eigenvectors of A*
14: **end procedure**

count [41, 90]. In the iterations, after step 11, $(\widehat{\Lambda}, \widehat{X})$ gives the Ritz pairs of A. It is easy to identify spurious eigenvalues by either checking whether they are inside Γ or computing the relative residuals. Discussions on the convergence criteria can be found in [41, 53, 125].

3.4.2 Fast contour-integral eigensolver

Our fast contour-integral eigensolver includes two major stages.

1. *Quadsection stage.* Start from an initial search region, estimate the number of eigenvalues inside. If the number is much larger than a given threshold, quadsect the region into subregions. Then repeat the procedure. This stage

involves eigenvalue counts with low-accuracy structured matrix approximations as discussed in Section 3.3. Fast structured matrix factorization, factorization update for varying shifts, and fast structured solution will be used.

2. *Subspace iteration stage.* In the subregions generated by the previous stage, apply projected subspace iteration as in the FEAST algorithm, where structured accelerations for the matrix factorizations and linear solutions also apply if A is rank structured.

We focus on rank structured A, and adaptively control the accuracy of its HSS approximation $\tilde{A}$. Lower accuracies are used for the eigenvalue count, and higher accuracies are used for the eigenvalue solution. For convenience, our discussions are based on search regions enclosed by circles.

3.4.2.1 Structured factorization update for varying shfits

Both the quadsection stage and the subspace iteration stage involve solutions of linear systems of the following form for multiple shifts μI:

$$(\mu I - \tilde{A})\tilde{S} = Y. \tag{3.27}$$

We precompute a ULV factorization for the HSS matrix $\tilde{A}$ with the algorithms in [12, 117, 120] and it costs $\mathcal{O}(r^2 n)$ flops, where r is the HSS rank of $\tilde{A}$. Then for each shifted matrix $\mu I - \tilde{A}$, we can update the ULV factorization, and the ULV factors are used to solve (3.27). If μ is set to be z_j in (3.26), we can get an approximation to S_j.

This shifted ULV factorization is an extension of the Hermitian version in [113]. Suppose the HSS generators of $\tilde{A}$ are $D_i, U_i, V_i, R_i, W_i, B_i$ as defined in Section 3.3.1. We briefly outline the ULV factorization procedure for $\tilde{A}$ in [12,117] without justification, and then show which steps can be updated to quickly get the factors of $\mu I - \tilde{A}$. For notational convenience, we present the update for $\tilde{A} - \mu I$.

First, for a leaf node i of the HSS tree, compute a QR factorization

$$U_i = Q_i \begin{pmatrix} 0 \\ \tilde{U}_i \end{pmatrix}, \tag{3.28}$$

and apply Q_i^T to the block row on the left. This needs to modify D_i as

$$\tilde{D}_i = Q_i^T D_i \equiv \begin{pmatrix} \tilde{D}_{i;1,1} & \tilde{D}_{i;1,2} \\ \tilde{D}_{i;2,1} & \tilde{D}_{i;2,2} \end{pmatrix}, \tag{3.29}$$

where the partition is done so that $\tilde{D}_{i;2,2}$ is a square matrix with the same row size as $\tilde{U}_i$.

Second, perform an LQ factorization of the first block row of $\tilde{D}_i$:

$$\begin{pmatrix} L_{i;1,1} & 0 \end{pmatrix} P_i = \begin{pmatrix} \tilde{D}_{i;1,1} & \tilde{D}_{i;1,2} \end{pmatrix},$$

and apply P_i^T to the corresponding block column on the right. This needs to update $\tilde{D}_i$ and V_i (with conformable partitions):

$$\tilde{D}_i P_i^T \equiv \begin{pmatrix} L_{i;1,1} & 0 \\ L_{i;2,1} & L_{i;2,2} \end{pmatrix}, \quad P_i V_i \equiv \begin{pmatrix} \hat{V}_i \\ \tilde{V}_i \end{pmatrix}.$$

Then $L_{i;1,1}$ can be eliminated, which corresponds to the elimination of node i. Similarly, eliminate the sibling node j of i. The parent node then becomes a new leaf corresponding to D, U, V generators

$$\begin{pmatrix} L_{i;2,2} & \tilde{U}_i B_i \tilde{V}_j^T \\ \tilde{U}_j B_j \tilde{V}_i^T & L_{j;2,2} \end{pmatrix}, \quad \begin{pmatrix} \tilde{U}_i R_i \\ \tilde{U}_j R_j \end{pmatrix}, \quad \begin{pmatrix} \tilde{V}_i W_i \\ \tilde{V}_j W_j \end{pmatrix}, \tag{3.30}$$

respectively. We can then repeat the steps on the parent node.

Now, when the shifted HSS matrix $\tilde{A} - \mu I$ is considered, a significant amount of computations can be saved:

- No HSS construction is need for $\tilde{A} - \mu I$, since all the generators remain the same except the D_i generators which just need to be shifted as:

$$D_i \leftarrow D_i - \mu I.$$

- In the ULV factorization, (3.28) remains unchanged.
- (3.29) can be quickly updated as

$$\tilde{D}_i \leftarrow \tilde{D}_i - \mu Q_i^T.$$

  This avoids a dense block multiplication.

- In (3.30), the following multiplications remain unchanged:

$$\tilde{U}_i B_i, \quad \tilde{U}_i R_i, \quad \tilde{U}_j B_j, \quad \tilde{U}_j R_j. \tag{3.31}$$

Thus, the entire HSS construction cost and part of the ULV factorization cost are saved. The steps (3.28), (3.29), and (3.31) can be precomputed. For convenience, we call these operations the *pre-shift factorization*. The remain operations are to be done for each shift μI in a *post-shift factorization*. Assuming the leaf level diagonal block size is $2r$ as often used [117], then the costs for the precomputations and the update are given in Table 3.3. Clearly, for each shift μI, we save about 40% of the HSS factorization cost (which is $\frac{116}{3}r^2n$ [114, Section 4.2]).

Table 3.3.
Costs of the precomputations for $\tilde{A}$ and the factorization update for $\tilde{A} - \mu I$.

	Precomputations		Factorization update
	Construction	Pre-shift factorization	(Post-shift factorization)
Flops	$\approx \mathcal{O}(r^2n) \sim \mathcal{O}(rn^2)$	$\frac{46}{3}r^2n$	$\frac{70}{3}r^2n$

A similar precomputation strategy can also be applied when a type of structure-preserving HSS construction in [120] is used. The corresponding pre-shift factorization cost is $6r^2n$, which is about 30% of the total factorization cost $\frac{58}{3}r^2n$ [120, Section 3.6]. The algorithm is similar to that of the one mentioned above is thus omitted.

3.4.2.2 Fast eigenvalue count

To count the eigenvalues inside a circle $\mathcal{C}_\gamma(z_0)$, we choose a random matrix Y with a small column size m and evaluate $\widetilde{Z}$ just like in (3.25). Then (3.16) becomes

$$\#_\Lambda(A, \mathcal{C}_\gamma(z_0)) \approx \frac{1}{m} \operatorname{trace}(Y^T \widetilde{Z}). \tag{3.32}$$

As in [125], we can start from m that is very small and gradually increase it. The algorithm stops if the estimate converges to a number s smaller than a prespecified threshold k or if the estimate is much larger than k. The selection of k will be discussed in Section 3.4.2.4 based on an optimality condition. In addition, we may even use a power method similar to [37] to improve the quality of this estimator.

As discussed in Section 3.3, we use a low-accuracy HSS approximation $\tilde{A} \approx A$ to evaluate $\widetilde{Z}$ in (3.32). $\tilde{A}$ may be constructed directly with an algorithm in [117] or via randomization [120]. The randomized HSS construction is used here. It is especially attractive when A can be quickly applied to vectors. In the construction, we first compute the product of A and a skinny random matrix (with column size equal to r plus a small constant). This product is adaptively modified to yield the product of each off-diagonal block and a certain random matrix, so as to apply randomized compression to produce the relevant basis matrices. The details can be found in [120, Section 3.3 and Algorithm 1]. The cost of this construction is $\mathcal{O}(r^2 n)$ plus the cost for matrix-vector multiplications. The matrix-vector multiplication cost ranges from $O(rn)$ to $\mathcal{O}(rn^2)$. The cost of $\mathcal{O}(rn^2)$ is the most general case when the construction is performed directly on a dense matrix A. Sometimes when A results from discretization of certain kernels, then an analytical construction can be done quickly [11].

The shifted factorization update in the previous subsection is then applied to $\tilde{A}$. This leads to the fast eigenvalue count method in Algorithm 4. Following the discussions in Section 3.2, the Trapezoidal rule is used for the numerical integration. In addition, Section 3.3 also means that we can use a smaller number of quadrature points in the eigenvalue counts than in the later subspace iterations.

Algorithm 4 *Fast eigenvalue count*

procedure $s = \mathsf{EigCount}(\tilde{A}, \mathcal{C}_\gamma(z_0), k)$

Input: HSS factors of $\tilde{A}$ (from precomputations); $\mathcal{C}_\gamma(z_0)$ (contour); k (threshold for eigenvalue count)

Output: $s \approx \#_\Lambda(A, \mathcal{C}_\gamma(z_0))$ if $\#_\Lambda(A, \mathcal{C}_\gamma(z_0))$ is not much larger than k

$m \leftarrow$ a small integer ▷ *Initial number of random vectors*

$Y \leftarrow n \times m$ random matrix

$t \leftarrow 0$ ▷ *Total trace*

$c_j = w_j(z_j - z_0),\ j = 1, \ldots, q$ ▷ *Weights w_j & nodes z_j in Trapezoidal rule*

Update the HSS factors of $\tilde{A}$ to get those of $z_j I - \tilde{A}, \quad j = 1, \ldots, q$

repeat ▷ *Adaptive estimate of the eigenvalue count*

$S_j \leftarrow (z_j I - \tilde{A})^{-1} Y,\ j = 1, \ldots, q$ ▷ *HSS ULV solution*

$\widetilde{Z} \leftarrow \frac{1}{2} \sum_{j=1}^{q} c_j S_j, \quad t \leftarrow t + \mathrm{trace}(Y^T \widetilde{Z})$

$s \leftarrow \frac{t}{m}$ ▷ *Current-step estimate of the eigenvalue count*

if s remains the same for some consecutive steps **then**

Return ▷ *Estimate count is identified*

else ▷ *Attaching one extra vector a time; multiple may be attached*

$Y \leftarrow$ random vector

$m \leftarrow m + 1$

end if

until s is much larger than k ▷ *Further partitioning of the region is needed*

end procedure

3.4.2.3 Structured FEAST eigenvalue solution with deflation

For a subregion, if the approximate eigenvalue count s is smaller than or near the threshold k, we then solve for the eigenpairs with a structured FEAST algorithm. The FEAST Algorithm 3 can be accelerated with a high-accuracy HSS approximation $\tilde{A}$ to A. Similarly to Algorithm 4, the factorizations and solutions can be performed in

HSS forms. In particular, the structured factorization update for varying shifts can greatly save the cost. Moreover, the matrix-vector multiplications needed to form the reduced problem (step 10 of Algorithm 5) can also be performed quickly in HSS forms.

In practice, due to different convergence rates of the eigenpairs in the subspace iteration, a deflation technique called locking [51, 93] is often used to save some computation costs. Those eigenpairs that have already converged to a desired accuracy can be locked and excluded from later iterations. This structured FEAST algorithm with deflation is summarized in Algorithm 5.

3.4.2.4 Algorithm for all eigenpairs, complexity, and optimal threshold for subregion eigenvalue count

To find a large number of eigenpairs or even all the eigenpairs of A, we recursively partition the search region into smaller subregions until each *target subregion* contains no more than k eigenvalues, where k is the eigenvalue count threshold. The structured FEAST algorithm is then applied to each target subregion to find the eigenpairs.

Discussion on the initial search region will be given in Section 3.4.3. For convenience, we assume all the intermediate search regions are squares. (In practice, depending on the actual problem, the regions may be made more flexible and more precise.) For each square, we estimate the number of eigenvalues based on $\#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z_0))$ in Algorithm 4, where $\mathcal{C}_\gamma(z_0)$ is the smallest circle that encloses the square. Since the area of $\mathcal{D}_\gamma(z_0)$ is about 1.57 times the area of the square, this gives an intuitive way of choosing the column size in step 5 of Algorithm 5, which is suggested in [41, 90] to be around 1.5 times the actual eigenvalue count. In practice, Algorithm 5 may then find eigenvalues belonging to neighbor subregions (squares). In this case, we can deflate those eigenvalues when the neighbor subregions are visited.

The complete algorithm of our fast eigensolver is summarized in Algorithm 6, where we assume A can be approximated accurately by an HSS form in step 14. Then

Algorithm 5 *Structured FEAST eigenvalue solution with subspace iteration and deflation*

1: **procedure** $[\widehat{\Lambda}, \widehat{X}] = \mathsf{SFEAST}(\tilde{A}, \mathcal{C}_\gamma(z_0), \tilde{k})$

Input: HSS factors of $\tilde{A}$ (high-accuracy approximation of A); $\mathcal{C}_\gamma(z_0)$ (contour); s (eigenvalue count)

Output: $(\widehat{\Lambda}, \widehat{X})$ (eigenvalues inside $\mathcal{C}_\gamma(z_0)$ and the corresponding eigenvectors)

2: $c_j = w_j(z_j - z_0),\ j = 1, \ldots, q$ ▷ *Weights w_j & nodes z_j in Trapezoidal rule*

3: Update the HSS factors of $\tilde{A}$ to get those of $z_j I - \tilde{A}, \quad j = 1, \ldots, q$

4: $\widehat{\Lambda} \leftarrow \varnothing, \quad \widehat{X} \leftarrow \varnothing, \quad \widehat{Q} \leftarrow \varnothing$ ▷ *$\widehat{Q}$: convergent eigenspace*

5: $Y \leftarrow n \times (\frac{3}{2}s)$ random matrix

▷ *More than s columns used for faster convergence*

6: **repeat**

7: $S_j \leftarrow (z_j I - \tilde{A})^{-1} Y, \quad j = 1, \ldots, q$ ▷ *HSS ULV solution*

8: $\widetilde{Z} \leftarrow \frac{1}{2} \sum_{j=1}^{q} c_j S_j$ ▷ *Approximating $\widetilde{Z} = \widetilde{\Phi} Y$ in (3.25) based on $\tilde{A}$*

9: $Q \leftarrow$ basis of $\widetilde{Z}$ orthonormalized with respect to $\widehat{Q}$

10: $\hat{A} \leftarrow Q^T A Q$ ▷ *Reduced problem via HSS matrix-vector multiplication*

11: $\hat{A} = \widetilde{X} \widetilde{\Lambda} \widetilde{X}^{-1}$ ▷ *Solving the reduced eigenvalue problem*

12: $Y \leftarrow Q \widetilde{X}$ ▷ *Recovery of approximate eigenvectors of A*

13: $\begin{pmatrix} \widehat{\Lambda}_1 & \widehat{\Lambda}_2 \end{pmatrix} \leftarrow \widehat{\Lambda}$ ▷ *Partition with convergent eigenvalues in $\widehat{\Lambda}_1$*

14: $\begin{pmatrix} Y_1 & Y_2 \end{pmatrix} \leftarrow Y$ ▷ *Partition with convergent eigenvectors in Y_1*

15: $\begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \leftarrow Q$ ▷ *Partition with convergent eigenspace in Q_1*

16: $\widehat{\Lambda} \leftarrow \mathrm{diag}(\widehat{\Lambda}, \widehat{\Lambda}_1), \quad \widehat{X} \leftarrow \begin{pmatrix} \widehat{X} & X_1 \end{pmatrix}, \quad \widehat{Q} \leftarrow \begin{pmatrix} \widehat{Q} & Q_1 \end{pmatrix}$

17: $Y \leftarrow Y_2$

18: **until** convergence

19: **end procedure**

the low-accuracy HSS approximation in step 3 can be simply obtained by appropriate truncations.

Algorithm 6 *Fast structured non-Hermitian contour-integral eigensolver*

procedure $[\Lambda, X] = \mathsf{FastEig}(\tilde{A}, \Gamma, k)$

Input: A (explicit or implicit via matrix-vector multiplications); Γ (contour that encloses desired eigenvalues); k (threshold for subregion eigenvalue count)

Output: (Λ, X) (partial or full spectrum of A)

▷ *2D Quadsection stage*

Push the initial search region enclosed by Γ onto a stack $\mathcal{S}$

$\tilde{A} \approx A$ ▷ *Low-accuracy HSS construction for A and ULV factorization*

while $\mathcal{S} \neq \varnothing$ **do**

Pop a subregion $\mathcal{R}_i$ from $\mathcal{S}$

Find the smallest circle $\mathcal{C}_\gamma(z_0)$ that encloses $\mathcal{R}_i$

$s_i = \mathsf{EigCount}(\tilde{A}, \mathcal{C}_\gamma(z_0), k)$ ▷ *Algorithm 4*

if $\tilde{k}_i \leq k$ **then** ▷ *No further quadsection is needed*

Mark $\mathcal{R}_i$ as a target subregion

else

Quadsect $\mathcal{R}_i$ into 4 subregions and push the subregions onto $\mathcal{S}$

end if

end while

▷ *Eigenpair solution stage*

$\tilde{A} \approx A$ ▷ *High-accuracy HSS construction for A and ULV factorization*

$\Lambda \leftarrow \varnothing, \quad X \leftarrow \varnothing$

for each target subregion $\mathcal{R}_i$ **do**

$[\widehat{\Lambda}, \widehat{X}] = \mathsf{SFEAST}(\tilde{A}, \mathcal{R}_i, s_i)$ ▷ *Algorithm 5 (with minor modifications)*

$\Lambda \leftarrow \mathrm{diag}(\Lambda, \widehat{\Lambda}), \quad X \leftarrow \begin{pmatrix} X & \widehat{X} \end{pmatrix}$

end for

end procedure

We now analyze the asymptotic complexity of Algorithm 6 for finding all the eigenpairs of a matrix A with maximum off-diagonal (numerical) rank r, and also decide the optimal threshold k. Due to the nature of quadsection, a quadtree can be used to organize the process. Each node of the tree represents a subregion, and the leaf nodes represent the target subregions with roughly k eigenvalues or less. Note that this tree may be unbalanced.

Due to the independence of the computations for non-overlapping subregions, the complexity is directly related to the number of nodes in the quadtree. Without loss of generality, suppose each leaf of the tree corresponds to a subregion with about k eigenvalues, so that the tree has $\mathcal{O}(\frac{n}{k})$ leaves and also $\mathcal{O}(\frac{n}{k})$ nodes. (This modest assumption just eliminates extreme cases where the eigenvalues are highly clustered so that the tree has too many empty nodes. In fact, as long as each node is nonempty, the quadtree has at most n leaves and the asymptotic complexity count would remain about the same for small r.) The computation costs of some basic operations are listed in Table 3.4.

Table 3.4.
Computation costs of some basic operations, where r is the HSS rank of A.

Operation	Flops
HSS construction	Up to $\mathcal{O}(r^2 n)$
ULV factorization/post-shift factorization update	$\mathcal{O}(r^2 n)$
HSS solution	$\mathcal{O}(rn)$
HSS matrix-vector multiplication	$\mathcal{O}(rn)$
Orthonormalization (QR factorization) of a tall $n \times k$ matrix	$\mathcal{O}(k^2 n)$

In the quadsection stage, the eigenvalue count Algorithm 4 is performed for every node of the quadtree. The HSS construction cost will be counted in the eigenvalue solution stage since the low-accuracy HSS approximation can be obtained from trun-

cation. We count the costs associated with each node. A smaller HSS rank ($\tilde{r} \leq r$) is used in the low-accuracy HSS approximation, and the pre-shift ULV factorization costs $\xi_{1,0} = \mathcal{O}(\tilde{r}^2 n)$. The post-shift factorization update costs

$$\xi_{1,1} = \mathcal{O}(q\tilde{r}^2 n) = \mathcal{O}(\tilde{r}^2 n),$$

where q is the number of quadrature nodes and is small (see Section 3.2). Approximating (3.25) needs to solve m systems and to add q solution matrices, where m is in (3.32). The cost is

$$\xi_{1,2} = \mathcal{O}(qm\tilde{r}n) + (q-1)mn = \mathcal{O}(\tilde{r}mn).$$

All the trace computations for (3.32) cost $\xi_{1,3} = \mathcal{O}(m^2 n)$. Thus, the total cost for the quadsection stage is

$$\begin{aligned}\xi_1 &= \xi_{1,0} + \mathcal{O}\left(\frac{n}{k}\right)(\xi_{1,1} + \xi_{1,2} + \xi_{1,3}) = \mathcal{O}\left(\frac{\tilde{r}^2 n^2}{k}\right) + \mathcal{O}(\frac{m\tilde{r}n^2}{k}) + \mathcal{O}(\frac{m^2 n^2}{k}) \\ &= \mathcal{O}\left(\frac{\tilde{r}^2 n^2}{k}\right) + \mathcal{O}(\tilde{r}n^2) + \mathcal{O}(kn^2),\end{aligned}$$

where we have relaxed m to be k, although m may be actually a very small constant and much smaller than k.

In the second stage, we use Algorithm 5 to solve for the eigenpairs in the subregions associated with all the leaves of the quadtree. A high-accuracy HSS approximation and the pre-shift ULV factorization cost no more than $\xi_{2,0} = \mathcal{O}(rn^2) + \mathcal{O}(r^2 n)$ in the precomputation. We then count the costs associated with each leaf. Similar to the above, the post-shift factorization update costs

$$\xi_{2,1} = \mathcal{O}(qr^2 n) = \mathcal{O}(r^2 n).$$

The linear system solutions for the quadrature approximation costs

$$\xi_{2,2} = \beta\left[\mathcal{O}(qkrn) + (q-1)kn\right] = \mathcal{O}(rkn),$$

where β is the number of iterations and is assumed to be bounded since it is usually small. Getting the orthonormal basis costs $\xi_{2,3} = \mathcal{O}(k^2 n)$. Forming the reduced matrix $\hat{A}$ via HSS matrix-vector multiplications costs

$$\xi_{2,4} = \beta[\mathcal{O}(rkn) + \mathcal{O}(k^2 n)] = \mathcal{O}(rkn) + \mathcal{O}(k^2 n).$$

Solving the reduced eigenvalue problem and recovering the eigenvectors of A costs

$$\xi_{2,5} = \beta[\mathcal{O}(k^3) + \mathcal{O}(k^2 n)] = \mathcal{O}(k^3) + \mathcal{O}(k^2 n).$$

The cost for this stage is then

$$\xi_2 = \xi_{2,0} + \mathcal{O}\left(\frac{n}{k}\right)(\xi_{2,1} + \cdots + \xi_{2,5}) = \mathcal{O}\left(\frac{r^2 n^2}{k}\right) + \mathcal{O}(rn^2) + \mathcal{O}(kn^2) + \mathcal{O}(k^2 n).$$

Therefore, due to $\tilde{r} \leq r$, the total computation cost is

$$\xi = \xi_1 + \xi_2 = \mathcal{O}(rn^2) + [\mathcal{O}(k^2 n) + \mathcal{O}(kn^2)] + \mathcal{O}\left(\frac{r^2 n^2}{k}\right) \tag{3.33}$$

We can then use this to decide the optimal threshold k that minimizes ξ.

Theorem 3.4.1 *If A has HSS rank r, then the optimal eigenvalue count threshold for the subregions in Algorithm 6 is $k = O(r)$, and the optimal cost of the algorithm to find all eigenpairs of A is*

$$\xi = \mathcal{O}(rn^2) + \mathcal{O}(r^2 n). \tag{3.34}$$

Proof In (3.33), the term $\mathcal{O}(k^2 n) + \mathcal{O}(kn^2)$ increases with k, and the term $\mathcal{O}\left(\frac{r^2 n^2}{k}\right)$ decreases with k. Clearly, the minimum of ξ is achieved when $k = \mathcal{O}(r)$. ■

In addition, the backward stability of relevant HSS construction and factorization algorithms has been studied in [111, 112].

3.4.3 Applications and initial search region

3.4.3.1 Applications and extensions

Our fast contour-integral eigensolver has a wide range of applications. One category of matrices is rank structured A, and selected examples include:

- Banded matrices, where the HSS rank r is the bandwidth and the HSS form can be obtained on the fly. If the bandwidth is finite, the cost (3.34) to find all

the eigenpairs is $\xi = \mathcal{O}(n^2)$. Non-Hermitian banded eigenvalue problems arise in many computations and applications. For example, tridiagonal eigenvalue solution is needed in some non-Hermitian eigensolvers that reduce more general matrices (such as complex symmetric ones) to tridiagonal forms. Banded non-Hermitian eigenvalue problems also appear in the study of some 1D PDEs and in sparse neural networks [1].

- Companion matrices, where the HSS rank is $r = 2$ and the HSS form can be directly written out. Companion eignenvalue solution is usually used to find the roots of univariate polynomial roots. Our algorithm can achieve the same asymptotic complexity $\mathcal{O}(n^2)$ as other fast QR-type companion eigensolvers (e.g., [14]). However, since the companion matrix has more delicate structures that are not fully utilized here, the actual cost is likely higher than that in [14]. On the other hand, the scalability is likely better due to the partitioning of the search region into independent subregions.

- Toeplitz matrices, which in Fourier space have HSS ranks $r = \mathcal{O}(\log n)$ and the HSS construction costs $\mathcal{O}(n \log^2 n)$ [120]. The cost (3.34) is $\xi = \mathcal{O}(n^2 \log n)$. Toeplitz eigenvalue problems are often involved in the studies of time series, wave phenomena in periodic lattices, quantum spin chains, and many other physics and engineering problems [16, 18, 25, 52, 84].

- Some kernel functions (e.g., $1/|x - y|$ and $\log |x - y|$) discretized on 1D curves, where $r = \mathcal{O}(\log n)$ and the HSS construction costs $\mathcal{O}(n \log n)$ [11]. The cost (3.34) is $\xi = \mathcal{O}(n^2 \log n)$. Related problems appear in the studies of radial basis functions and integral kernels, in data science areas such as spectral clustering and kernel principal component analysis [97], and in some physics areas such as entanglement theory [56].

For the last two examples, a much smaller HSS rank $\tilde{r}$ may be used for the eigenvalue counts. In addition, the matrix-vector multiplication needed in forming

the reduced eigenvalue problem can also be quickly conducted using FFTs or the fast multipole method (FMM) [36].

Another category is A with slowly decaying off-diagonal singular values, where a low-accuracy compact HSS approximation can be used to accelerate the eigenvalue count. Examples include some discretized kernel functions in two dimensions. Potential applications of our methods also include more general matrices where the eigenvalues are roughly uniformly distributed, so that a low-accuracy matrix approximation has a high probability of reliably counting the eigenvalues.

For some cases, extensions and modifications can be made to accommodate additional matrix properties. For some structured sparse problems [115, 116], we may extend our eigensolver by replacing the HSS methods by structured sparse factorizations, where low-accuracy HSS approximations are used for the intermediate fill-in. This is particularly effective for discretized elliptic PDEs. For cases such as those with tensor product structures, the structured approximation and factorization costs may be significantly reduced. See [30, 33] for some examples, where the structured approximation cost is sublinear in n. For such cases, when n is large, it may be more practical to use our method to extract selected eigenvalues. Tensor structured methods for the eigenvalue solution of these problems can be found in [44].

We can also adopt the eigensolver to extract certain specific types of eigenvalues, such as the real ones. This will be useful in applications such as control. The search for eigenvalues is then restricted to the real line. More effective filter functions can be designed by setting the contour close to the interval, e.g., with a flat ellipse [41].

3.4.3.2 Determining the initial search region

When Algorithm 6 is used to find the entire spectrum, it requires an initial search region. There are many strategies to obtain the region, such as the estimation of the spectral radius and the study of inclusion regions for the field of values. Depending on specific applications, efficient and effective estimations may be available. Here, we

just briefly mention the most basic and general method based on the spectral radius. To estimate the spectral radius, we may use the Gershgorin theorem, an estimate of certain matrix norms, or the following well-known result.

Lemma 3.4.2 *Let ρ be the spectral radius of $A \in \mathbb{C}^{n\times n}$ and $\|\cdot\|$ be a consistent matrix norm. Then $\rho = \lim_{j\to\infty} \|A^j\|^{1/j}$.*

For A with fast matrix-vector multiplications, we may choose an appropriate j and estimate $\|A^j\|_1$ using Hager's method or a randomized Hager's method [37].

For some matrices, it may be quick to find $||A||_1$ exactly. For example, if A is a Toeplitz matrix, let u be its first column and v^T be its first row. We can compute

$$c_1 = ||u||_1, \quad c_i = c_{i-1} - |u_{n-i+2}| + |v_i|, \quad i = 2, \ldots, n.$$

Then $||A||_1 = \max |c_i|$.

If A is a companion matrix, other than the bound from $||A||_1$, we can find a nearly optimal bound on the eigenvalues based on a result for the roots of a polynomial $p(\lambda) = \sum_{i=0}^{n} a_i\lambda^i$ $(a_n \neq 0)$ [27]:

$$|\lambda| \leq 2\max\left(\frac{a_{n-1}}{a_n}, \left(\frac{a_{n-2}}{a_n}\right)^{1/2}, \ldots, \left(\frac{a_1}{a_n}\right)^{1/(n-1)}, \left(\frac{a_0}{2a_n}\right)^{1/n}\right)$$

3.5 Numerical experiments

Now, we show the performance of our fast eigensolver (FastEig Algorithm 6) for some test examples. In order to observe how the complexity depends on the matrix size n, we use quadsection to find all the eigenpairs and report the total clock time. The structure-preserving HSS construction and the corresponding shifted factorization schemes mentioned at the end of Section 3.4.2.1 are used. Since our eigensolver uses structured direct linear solutions in the intermediate computations, some comparisons are performed with structured direct solutions without our acceleration techniques for one example. (Standard dense direct solvers are obviously much slower

and are thus not compared.) It will demonstrate the benefits of the shifted structured factorization update and the eigenvalue count with low-accuracy HSS approximations.

The maximum number of subspace iterations is set to be 10. We report several different accuracy measurements:

- $\mathbf{e}_i = \frac{|\lambda_i - \tilde{\lambda}_i|}{|\lambda_i|}$: relative error, where $\tilde{\lambda}_i$ is the computed eigenvalue and the eigenvalue returned by the Intel MKL subroutine ZGEEV is treated as the exact eigenvalue λ_i;
- $\hat{\mathbf{e}} = \frac{\sqrt{\sum_{i=1}^{n} |\lambda_i - \tilde{\lambda}_i|^2}}{n\sqrt{\sum_{i=1}^{n} |\lambda_i|^2}}$: relative error as used in [107];
- $\mathbf{r}_i = \frac{\|A\tilde{x}_i - \tilde{\lambda}_i \tilde{x}_i\|_2}{\|A\tilde{x}_i\|_2 + \|\tilde{\lambda}_i \tilde{x}_i\|_2}$: relative residual, where $\tilde{x}_i$ is the computed eigenvector;
- $\hat{\mathbf{r}}_i = \frac{\|A\tilde{x}_i - \tilde{\lambda}_i \tilde{x}_i\|_2}{n\|A\|_2}$: relative residual as used in [38].
- mean($\cdot$): geometric mean.

The algorithm is implemented in (sequential) Fortran using the Intel Math Kernel Library (MKL) and Intel Fortran compiler. All the tests are done on an Intel Xeon-E5 processor with 64 GB memory on Purdue's computing cluster Conte. In the first example, we also compare the performance of our eigensolver with the Intel MKL subroutine ZGEEV, which is based on QR iterations.

Example 1. First, consider a Cauchy-like matrix A of the form

$$A_{ij} = \frac{u_i v_j}{s_i - t_j},$$

where $s_i = e^{2i\pi\mathbf{i}/n}$ and $t_j = e^{(2j+1)\pi\mathbf{i}/n}$ are located on the unit circle, and $\{u_i\}_{i=1}^n$ and $\{v_j\}_{j=1}^n$ are random.

The matrix is related to the discretization of $G(s,t) = \frac{1}{s-t}$ and is known to be rank structured. Table 3.2 above includes the HSS ranks for one matrix size. That table already shows how low-accuracy HSS approximation can be used to reliably estimate the eigenvalue counts.

According to FMM, the maximum off-diagonal numerical rank is $\mathcal{O}(\log n)$. The complexity of the eigensolver is then expected to be $\mathcal{O}(n^2 \log n)$. In the test, we let the matrix size n range from $1,600$ to $25,600$. We use relative tolerance $\tau_1 = 10^{-1}$ for the HSS compression in the quadsection stage and $\tau_2 = 10^{-8}$ in the eigenvalue solution stage. The clock times are reported in Figure 3.3 for reaching modest accuracies in Table 3.5, and are compared with the runtimes of the Intel MKL subroutine ZGEEV. Two reference lines for $\mathcal{O}(n^2 \log n)$ and $\mathcal{O}(n^3)$ are also included. We can see that the CPU times are roughly consistent with the complexity analysis. In fact, the slope for the plot of FastEig is significantly lower. The crossover point between these two algorithms for this particular test can also be observed.

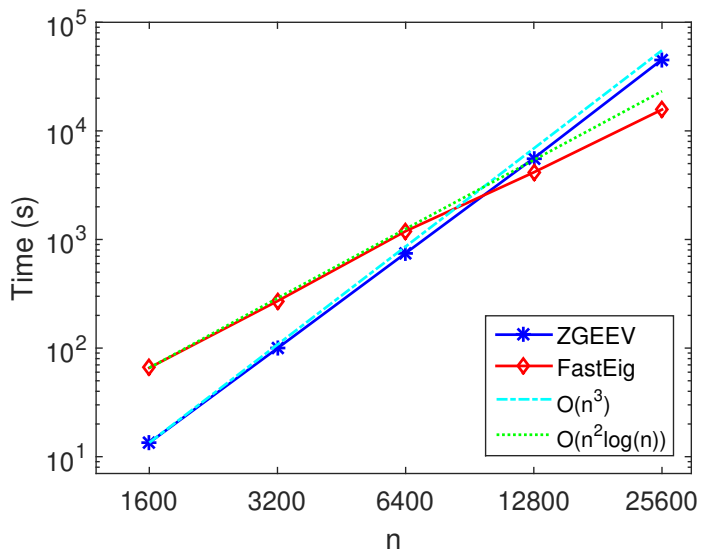


Figure 3.3. Example 1. Clock times of FastEig for finding all the eigenvalues.

Example 2. Next, consider A to be a discretized matrix from the Foldy-Lax formulation for studying scattering effects due to multiple point scatters [23, 58]. Let

$$A_{ij} = \begin{cases} 1, & \text{if } i = j, \\ -G(s_i, t_j)\sigma_j, & \text{otherwise,} \end{cases}$$

Table 3.5.
Example 1. Accuracies of the eigenvalue solution.

n	$1,600$	$3,200$	$6,400$	$12,800$	$25,600$
$\max(\mathbf{e}_i)$	$1.59e{-}7$	$9.47e{-}7$	$9.56e{-}7$	$9.99e{-}7$	$9.82e{-}7$
$\text{mean}(\mathbf{e}_i)$	$1.87e{-}9$	$2.08e{-}9$	$1.99e{-}9$	$3.08e{-}9$	$7.63e{-}9$
$\hat{\mathbf{e}}$	$3.96e{-}12$	$1.79e{-}10$	$1.83e{-}9$	$7.67e{-}10$	$1.58e{-}9$
$\max(\mathbf{r}_i)$	$2.27e{-}7$	$2.63e{-}5$	$3.00e{-}5$	$2.89e{-}5$	$2.99e{-}5$
$\text{mean}(\mathbf{r}_i)$	$1.89e{-}8$	$2.45e{-}8$	$2.79e{-}8$	$3.39e{-}8$	$5.46e{-}8$
$\max(\hat{\mathbf{r}}_i)$	$6.35e{-}11$	$2.91e{-}8$	$1.69e{-}7$	$7.85e{-}8$	$4.52e{-}8$
$\text{mean}(\hat{\mathbf{r}}_i)$	$1.13e{-}11$	$7.04e{-}12$	$4.43e{-}12$	$2.74e{-}12$	$2.12e{-}12$

where σ_j's are the scattering coefficients, and $G(s,t)$ is the Green's function of the 3D Helmholtz equation:

$$G(s,t) = \frac{e^{\mathbf{i}\omega|s-t|}}{4\pi|s-t|}, \quad s \neq t. \tag{3.35}$$

Here, $\omega = 4\pi$ and σ_j is random in $(0,1)$, as used in [3].

If the problem is discretized on one dimensional meshes, we observe performance similar to that in the previous example. Thus, we only consider A resulting from the discretization of (3.35) on $M \times N$ regular meshes with equidistance $h = 0.1$ in each direction. The matrix has order $n = MN$. Here, we fix $M = 20$ and let N increase from 80 to $1,280$. We use a rank bound 40 in the quadsection stage and a relative tolerance $\tau = 10^{-8}$ in the eigenvalue solution stage. In this case, the off-diagonal ranks are much higher than in the pervious example, so that our acceleration strategies make a significant difference.

Since our eigensolver involves direct linear solutions, we give some comparisons with different structured direct solution methods, depending on whether to use shifted factorization update in the linear solutions and/or low-accuracy approximation for

the eigenvalue count. The costs are given in Figure 3.4. We can observe the overall complexity of $\mathcal{O}(n^2 \log n)$. We can also see how the acceleration strategies help to improve the performance. In particular, we show in Table 3.6 the detailed time for one of the matrices. The shifted factorization update accelerates both the quadsection stage and the subspace iteration stage. By using low-accuracy HSS approximations for the eigenvalue count, the cost of the quadsection stage becomes significantly lower.

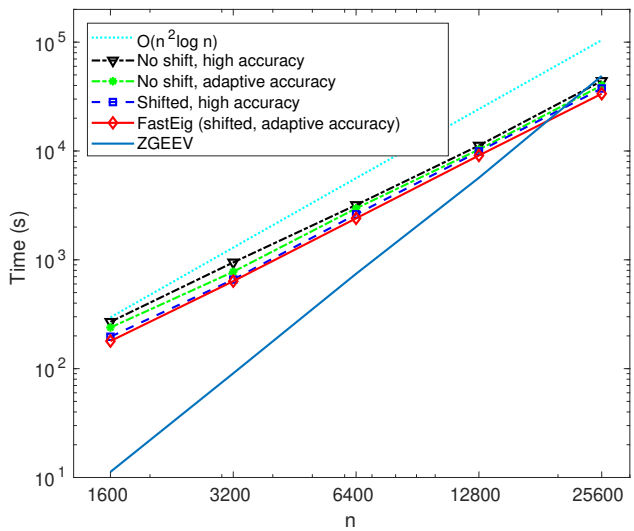


Figure 3.4. Example 2. Clock times of FastEig for finding all the eigenvalues, where "shifted" means structured linear solution with shifted factorization update, and "adaptive accuracy" means using low-accuracy HSS approximation for the eigenvalue count and high accuracy approximation for the later eigenvalue solution.

The benefit of the low-accuracy HSS approximation can also be seen from another aspect. Table 3.7 lists the HSS ranks of $\tilde{A}$ used in the two stages of FastEig. A small rank in the eigenvalue counts leads to significant savings.

The accuracies of the eigenpairs are given in Table 3.8. In addition, Figure 3.5 illustrates how the quadsection of the search region is performed.

Table 3.6.
Example 2. Detailed times for the matrix with $n = 6,400$ in Figure 3.4, depending on whether the acceleration strategies are used or not.

Shifted factorization update	Eigenvalue count with low-accuracy HSS	Quadsection stage	Subspace iteration stage
✗	✗	$1.30e3$	$1.89e3$
✗	✓	$7.40e2$	$1.88e3$
✓	✗	$1.27e3$	$1.70e3$
✓	✓	$6.84e2$	$1.72e3$

Table 3.7.
Example 2. HSS ranks of $\tilde{A}$ in the two stages of FastEig.

n (matrix size)	$1,600$	$3,200$	$6,400$	$12,800$	$25,600$
Quadsection/eigenvalue count stage	40	40	40	40	40
Subspace iteration stage	118	227	253	297	360

3.6 Conclusions

In this chapter, we have designed a fast contour-integral eigensolver based on a series of analytical and computational techniques. We show that the Trapezoidal rule is an ideal quadrature for constructing filter functions in contour-integral eigenvalue solutions. This is based on the study of the decay away from the unit circle. We then provide a strategy to use low-accuracy matrix approximations to achieve reliable eigenvalue counts. Such counts are either exact or only off by a small number with low probabilities under some assumptions. Probability estimates are given. In the eigenvalue count algorithm and the FEAST algorithm, rank structured methods are used to accelerate the computations, especially the factorization update for varying

Table 3.8.
Example 2. Accuracies of the eigenvalue solution.

n (matrix size)	$1,600$	$3,200$	$6,400$	$12,800$	$25,600$
$\max(\mathbf{e}_i)$	$3.27e-8$	$7.58e-6$	$3.77e-7$	$9.61e-6$	$9.58e-6$
$\text{mean}(\mathbf{e}_i)$	$3.31e-10$	$5.06e-10$	$6.13e-10$	$6.95e-10$	$7.20e-10$
$\hat{\mathbf{e}}$	$1.16e-12$	$2.23e-11$	$3.27e-9$	$1.71e-9$	$4.09e-10$
$\max(\mathbf{r}_i)$	$4.82e-8$	$1.32e-7$	$4.23e-7$	$7.69e-5$	$9.00e-5$
$\text{mean}(\mathbf{r}_i)$	$4.78e-9$	$1.07e-8$	$1.74e-8$	$2.44e-8$	$4.61e-8$
$\max(\hat{r}_i)$	$1.20e-11$	$1.09e-8$	$3.04e-8$	$3.34e-8$	$8.26e-9$
$\text{mean}(\hat{\mathbf{r}}_i)$	$1.22e-12$	$1.11e-12$	$6.09e-13$	$3.36e-13$	$2.63e-13$

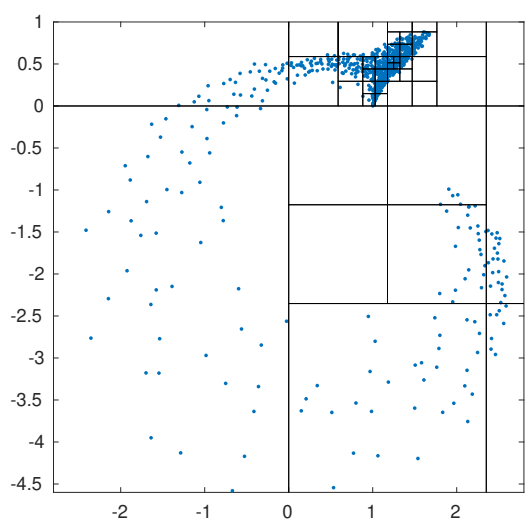


Figure 3.5. Example 2. Eigenvalue distribution and quadsection process for finding the eigenvalues of the matrix with $n = 800$.

shifts. The eigensolver may be used to find a large number of eigenvalues or the full spectrum in a quadsection framework, where we derive an optimal threshold for

the number of eigenvalues within each subregion. The eigensolver has nearly $\mathcal{O}(n^2)$ complexity for rank structured matrices, and some strategies can also benefit more general matrices. Due to the nice scalability of both contour-integral eigensolvers and HSS methods, our algorithms have a great potential to be parallelized. We plan to produce a scalable implementation. We are also in the process of extending the methods to more general matrix classes and matrices with clustered eigenvalues.

4. EFFECTIVE MATRIX-FREE PRECONDITIONING FOR THE AUGMENTED IMMERSED INTERFACE METHOD

The content in this chapter is from a published paper [119]: J. Xia, Z. Li, and X. Ye, *Effective matrix-free preconditioning for the augmented immersed interface method*, J. Comput. Phys., 303 (2015), pp. 295–312. Additional copyright information is provided in appendix A.

4.1 Introduction

In recent years, the augmented immersed interface method (AIIM) has been shown to be very effective for the solution of many interface problems and problems on irregular domains. The method is first developed for elliptic interface problems with discontinuous and piecewise constant coefficients [61]. Later, the idea is extended to moving interface problems on irregular domains [69] and incompressible Stokes equations with discontinuous viscosities [63]. We refer the readers to [62] for more information about AIIM.

Augmented strategies can be naturally used to design efficient and accurate algorithms based on existing fast solvers. As an example, for incompressible Stokes equations with a discontinuous viscosity across the interface, the discontinuous pressure and velocity can be decoupled by augmented strategies. The immersed interface method can then be applied conveniently with a fast Poisson solver in the iterative solution of the Schur complement system.

In augmented strategies, a large linear system is formed for the approximate solution $\mathbf{u}$ to the original problem together with an augmented variable $\mathbf{g}$ (which may

be a vector) of co-dimension one. Eliminating the block corresponding to $\mathbf{u}$ from the coefficient matrix yields a Schur complement system for $\mathbf{g}$, which is often much smaller compared with $\mathbf{u}$. Finding $\mathbf{g}$ can then make it convenient to solve for $\mathbf{u}$.

Thus, it is critical to quickly solve the Schur complement system, which can be done via direct or iterative solvers. Direct solvers can be used for some applications when the Schur complement matrix $\mathbf{A}$ is a constant matrix, for example, for a fixed interface or boundary. If $\mathbf{A}$ is not a constant matrix, typically for free boundary and moving interface problems, efficient iterative solvers may be preferred. Iterative solvers such as GMRES [92] do not require the explicit formation of the Schur complement matrix $\mathbf{A}$, and are thus often used. In addition, the solution often needs just modest accuracies, and iterative methods make it convenient to control the accuracy and the cost. Iterative solvers only need the product of $\mathbf{A}$ and vectors. This is usually done as follows. First, assume $\mathbf{g}$ is available and solve the original problem for an approximate solution $\mathbf{u}$. Next, use the approximate solution to compute the residual and thus the matrix-vector product.

AIIM can be considered as a generalized boundary integral method without explicitly using Green functions. The augmented variable can be considered as a source term. If the system behaves like an integral equation of the first kind, then GMRES converges slowly in general as described in several applications in this paper. For these applications, effective preconditioners are then needed.

Our work here is initially motivated by the application of AIIM to Navier-Stokes equations with a traction boundary condition. Explicit numerical tests indicate that, for some applications, the condition number of the Schur complement matrix $\mathbf{A}$ is of size $O(1)$, and $||\mathbf{A}^T\mathbf{A} - \mathbf{A}\mathbf{A}^T||_2$ is also very small. Nevertheless, GMRES still either takes too many steps to converge or simply stalls. Finding a preconditioner that can greatly improve the convergence of GMRES and is easy to apply becomes crucial for augmented methods.

However, there are some substantial difficulties in constructing suitable preconditioners. In fact, it may be difficult to even obtain a simple preconditioner such as

the diagonal of $\mathbf{A}$. The reasons are: (1) $\mathbf{A}$ is generally dense and the entries are not known explicitly in augmented methods, as mentioned above; (2) we can quickly multiply $\mathbf{A}$ and vectors, but not $\mathbf{A}^T$ and vectors. Preliminary attempts have been made, such as extracting few columns of $\mathbf{A}$ or multiplying $\mathbf{A}$ with special vectors, and then converting the results into a block diagonal preconditioner. The preconditioner may work for a particular problem or right-hand side, but often fails. Thus, it is necessary to design reliable *matrix-free* preconditioning techniques based on only matrix-vector products. Previously, some matrix-free preconditioners are designed for certain sparse problems and specific applications [6, 17, 32, 75, 102]. The ideas are to extract partial approximations or to compute incomplete factorizations via matrix-vector multiplications.

The objective of this work is to construct effective and efficient *structured matrix-free* preconditioners solely based on the products of $\mathbf{A}$ and vectors. This is achieved by taking advantage of some new preconditioning techniques introduced in recent years, such as rank structured preconditioning [22, 34, 40, 59, 67, 118] and randomized preconditioning [87, 88].

The idea of rank structured preconditioning is to obtain a preconditioner via the truncation of the singular values of appropriate off-diagonal blocks. If the off-diagonal singular values decay quickly (the problem is then said to have a *low-rank property*), it is known that this truncation strategy can be used to develop fast approximate direct solvers. On the other hand, if the off-diagonal singular values are aggressively truncated regardless of their decay rate, structured preconditioners can be obtained. The effectiveness is studied for some cases in [67, 118]. Among the most frequently used rank structures is the hierarchically semiseparable (HSS) form [12, 117]. Unlike standard dense matrix operations, structured methods in terms of compact HSS forms can achieve significantly better efficiency. In fact, the factorization and solution of an HSS matrix need only about $O(N)$ flops and $O(N)$ storage, where N is the matrix size.

In some latest developments, randomized sampling is combined with rank structures to obtain enhanced flexibility [71, 77, 113, 120]. That is, the construction of rank structures (e.g., HSS) may potentially use only matrix-vector products instead of the original matrix itself. The methods in [77,120] use both matrix-vector products and selected entries of the matrix. The one in [71] is matrix-free, although requiring slightly more matrix-vector products. In [113], a fully matrix-free and adaptive HSS construction scheme is developed. It uses an adaptive rank detection strategy in [46] to dynamically decide the off-diagonal ranks based on a pre-specified accuracy.

However, all the randomized methods in [71, 77, 113, 120] require the products of both $\mathbf{A}$ and $\mathbf{A}^T$ with vectors if $\mathbf{A}$ is nonsymmetric, and are thus not applicable to AIIM. Here, we seek to precondition $\mathbf{A}$ with an improved adaptive matrix-free scheme, using only the products of $\mathbf{A}$ and vectors. Due to the special features of AIIM as mentioned above, we construct a nearly symmetric HSS approximation H to $\mathbf{A}$ via randomized sampling. In the construction, $\mathbf{A}$ replaces $\mathbf{A}^T$ for the multiplication of $\mathbf{A}^T$ and vectors. Thus, the column basis of an off-diagonal block of $\mathbf{A}$ obtained by randomized sampling is used to approximate the row basis of the off-diagonal block at the symmetric position of $\mathbf{A}$. This enables us to find low-rank approximations to all the off-diagonal blocks. We explicitly specify a small ($O(1)$) rank or a low accuracy in the approximation. The off-diagonal approximations are then combined with the matrix-vector products to yield approximations of the diagonal blocks. This is done in a hierarchical fashion so that the overall HSS construction process needs only $O(\log N)$ matrix-vector products.

Several other improvements to the schemes in [71, 113] are made. We replace half of the randomized sampling by deterministic QR factorizations. We also design a strategy to reduce the number of matrix multiplications and avoid the use of pseudoinverses that are both expensive and potentially unstable.

H is then quickly factorized, and the factors are used as a preconditioner. Existing HSS algorithms are simplified to take advantage of the near symmetry, and the factorization and the application of the preconditioner have only $O(N)$ complexity

and $O(N)$ storage. Since $\mathbf{A}$ corresponds to the interface, its size N is much smaller than the Poisson solution needed to compute a matrix-vector product. Thus, the preconditioning cost is negligible as compared with the matrix-vector multiplication cost in the iterations. We also provide a simplified preconditioner given by the diagonal blocks of H that is easier to use and sometimes has comparable performance.

The effectiveness of the preconditioners is discussed in terms of several aspects of structured and randomized preconditioning, such as the benefits of low-accuracy rank structured preconditioners in reducing condition numbers. The preconditioners also share some ideas with the additive preconditioning techniques in [87, 88], where random low-rank matrices are added to the original matrix to provide effective preconditioners. In addition, since a low-accuracy HSS approximation tends to preserve well-separated eigenvalues [107], it can bring the eigenvalues together when used as a preconditioner. Although the convergence of GMRES does not necessarily rely on the eigenvalue distribution [35, 106], the eigenvalue redistribution provides an empirical explanation for the preconditioner, as used in practice.

The effectiveness and the efficiency are further demonstrated with a survey of several important applications, such as Navier-Stokes equations on irregular domains with traction boundary conditions, an incompressible interface in incompressible flow, a contact problem of drop spreading, and a mixed boundary problem. For the Schur complement matrix $\mathbf{A}$ in AIIM, GMRES (with restart) generally stalls. Even non-restarted GMRES barely converges unless the number of iterations reaches nearly N. However, after our matrix-free preconditioning, GMRES converges quickly. The convergence is also observed to be relatively insensitive to N and some physical parameters.

We also give a comprehensive test for a free boundary problem that involves multiple stages of GMRES solutions within the iterative solution of a nonlinear equation. The problem involves mixed boundary conditions, and the system behaves like integral equations of both first and second kinds. With our preconditioner, the overall performance GMRES solutions for all the stages is significantly improved.

The presentation is organized as follows. In Section 4.2, the features of the Schur complements and the motivation for our work are discussed, together with a brief review of the idea of AIIM. In particular, the linear system with the Schur complement matrix $\mathbf{A}$ is explained in detail, including the fast multiplication of $\mathbf{A}$ with vectors and the formation of the right-hand side. Section 4.3 presents the matrix-free structured preconditioner and the detailed algorithms, and the effectiveness is discussed. Section 4.4 lists the applications, summarizes the numerical tests, and discusses the generalizations. Some conclusions are drawn in Section 4.5.

4.2 Features of the Schur complement systems in AIIM and motivation for the work

AIIM usually has two discretizations. One is for the governing PDE with the assumption that the augmented variable is known. The second is for the augmented equation such as the boundary condition or the interface condition. In this section, as a preparation for our preconditioning techniques, we use the example of solving a Poisson equation on an irregular domain in [61, 62] to demonstrate the idea of AIIM, and show the form of the Schur complement $\mathbf{A}$ and its multiplication with vectors. We then discuss some useful features of the Schur complement system. These features are based on both mathematical and numerical observations, and provide a motivation for the development of our preconditioners.

4.2.1 Finite difference method for elliptic problems with singular source terms

In this subsection, we review the discretization of the governing PDE as in [61,62], assuming the augmented variable is known. Let $\mathbf{R} = \{(x, y),\ a < x < b,\ c < y < d\}$

be a rectangular domain. Consider an elliptic interface problem with a specified boundary condition on $\partial\mathbf{R}$:

$$\begin{aligned} \Delta u = f(x,y), \quad (x,y) \in \mathbf{R} - \partial\Omega, \\ [u] = w, \quad [u_n] = v, \end{aligned} \tag{4.1}$$

where $\partial\Omega = (\hat{x}(s), \hat{y}(s))$ is a curve or interface within $\mathbf{R}$ with a parameter s (such as the arc-length), $[u]$ is the jump of the solution across the boundary $\partial\Omega$, n is the unit direction pointing outward of $\partial\Omega$, and $[u_n] = [\nabla u \cdot n]$ is the jump in the normal derivative of u across $\partial\Omega$. If $w \equiv 0$, (4.1) can be rewritten as Peskin's model [89]

$$\Delta u = f(x,y) + \iint_{\partial\Omega} v(s)\delta(x - \hat{x}(s))\delta(y - \hat{y}(s))\, ds.$$

We have a single equation for the entire domain. The second term on the right-hand side involves the two-dimensional Dirac delta function, which is called a singular source term or a source distribution along the curve $\partial\Omega$. If $w \neq 0$, then it is called a double layer, similar to the derivative of the Dirac delta function, which is again called a singular source. To solve the equation above numerically, either the immersed boundary method or the immersed interface method (IIM) can be used. For example, following the standard five-point finite difference discretization on a uniform mesh, we can write both methods as

$$\frac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i+1,j} - 4u_{i,j}}{h^2} = f_{ij} + c_{ij},$$

where h is the uniform mesh size, $u_{ij} \approx u(x_i, y_j)$ is the discrete solution, $f_{ij} = f(x_i, y_j)$, and c_{ij} is the discrete delta function in the immersed boundary method or is chosen to achieve the second order accuracy in IIM. A matrix-vector form can be conveniently written for the scheme:

$$A\mathbf{u} = \mathbf{f} + B\mathbf{w} + C\mathbf{v},$$

where $\mathbf{w}$ and $\mathbf{v}$ are the discrete forms of w and v, respectively. The matrices B and C are sparse matrices that are related to the coordinates of grid points in the finite difference stencil and the boundary information including the first and the second

order partial derivatives of $\partial\Omega$. Usually, each row of B or C has $3 \sim 9$ nonzero entries, depending on the applications.

Here, we assume that we know w and v on a rectangular domain. For AIIM, one of them is unknown and should be chosen to satisfy a certain kind of interface/boundary conditions for different applications.

4.2.2 AIIM for Helmholtz/Poisson equations on irregular domains

Now we explain AIIM for the solution of Helmholtz/Poisson equations on an irregular interior or exterior domain Ω. See [48, 49, 69] for more details. Consider an Helmholtz/Poisson equation with a linear boundary condition $q(u, u_n)$ along the boundary $\partial\Omega$:

$$\begin{aligned} \Delta u - \omega u &= f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \\ q(u, u_n) &= 0, \quad \mathbf{x} \in \partial\Omega. \end{aligned}$$

In AIIM, Ω is embedded into a rectangle or cube domain $\mathbf{R}$, and $\partial\Omega$ becomes an interface. The PDE and the source term are then extended to the entire domain $\mathbf{R}$ as follows:

$$\begin{aligned} \Delta u - \lambda u &= \begin{cases} f, & \mathbf{x} \in \Omega, \\ 0, & \mathbf{x} \in \mathbf{R} - \Omega, \end{cases} \\ q(u, u_n) &= 0, \quad \mathbf{x} \in \partial\Omega. \end{aligned}$$

$$\begin{cases} [u] = g, \\ [u_n] = 0, \end{cases} \quad \text{or} \quad \begin{cases} [u] = 0, \\ [u_n] = g, \end{cases} \quad \mathbf{x} \in \partial\Omega.$$

If g is known, then we can find the solution u with a fast Poisson solver. In the discrete sense, this can be represented by a matrix-vector equation

$$A\mathbf{u} = \mathbf{f} - B\mathbf{g}. \tag{4.2}$$

To solve the original problem, the augmented variable g is determined so that $q(u(g), u_n(g)) = 0$ for $u(g)$ along the boundary/interface $\partial\Omega$. This is the second discretization in AIIM. One strategy of the discretization is to apply least squares interpolations [61, 62] in

terms of $\mathbf{u}$ and $\mathbf{g}$ at a set of discrete points along $\partial\Omega$, which leads to a matrix-vector equation

$$C\mathbf{u} + D\mathbf{g} - \mathbf{q} = 0. \tag{4.3}$$

The residue vector is

$$\mathcal{R}(\mathbf{g}) = C\mathbf{u}(\mathbf{g}) + D\mathbf{g} - \mathbf{q}.$$

$\mathcal{R}(\mathbf{g})$ here has dual meanings. It is not only the regular residual of the linear system (4.6) below, but is also the measurement of how the boundary condition is satisfied. $\mathbf{u}$ is the solution when $\mathcal{R}(\mathbf{g}) = \mathbf{0}$.

Combine (4.2) and (4.3) to get

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{g} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{q} \end{pmatrix}.$$

Compute a block LU factorization

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} A & \\ C & I \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ & \mathbf{A} \end{pmatrix}, \tag{4.4}$$

where $\mathbf{A}$ is the Schur complement

$$\mathbf{A} = D - CA^{-1}B. \tag{4.5}$$

We can then solve a much smaller system for $\mathbf{g}$:

$$\mathbf{A}\mathbf{g} = \mathbf{b}, \quad \text{with} \quad \mathbf{b} = \mathbf{q} - CA^{-1}\mathbf{f}. \tag{4.6}$$

Once we find $\mathbf{g}$, then we solve (4.2) for the solution $\mathbf{u}$. The right-hand side vector $\mathbf{b}$ in (4.6) can be found with the evaluation of $-\mathcal{R}(\mathbf{g})$ at $\mathbf{g} = \mathbf{0}$, since

$$-\mathcal{R}(\mathbf{0}) = -(C\mathbf{u}(\mathbf{0}) + D\mathbf{0} - \mathbf{q}) = -(CA^{-1}\mathbf{f} - \mathbf{q}) = \mathbf{b}.$$

In iterative solutions of (4.6), we need to evaluate the products of $\mathbf{A}$ with vectors $\tilde{\mathbf{g}}$. For this purpose, we first solve (4.2) with $\mathbf{g}$ replaced by $\tilde{\mathbf{g}}$:

$$A\tilde{\mathbf{u}}(\tilde{\mathbf{g}}) = \mathbf{f} - B\tilde{\mathbf{g}}.$$

Then

$$\begin{aligned}\mathbf{A}\tilde{\mathbf{g}} &= D\tilde{\mathbf{g}} - CA^{-1}B\tilde{\mathbf{g}} = D\tilde{\mathbf{g}} - CA^{-1}(\mathbf{f} - A\tilde{\mathbf{u}}(\tilde{\mathbf{g}})) \\ &= D\tilde{\mathbf{g}} - C\mathbf{u}(\mathbf{0}) + C\tilde{\mathbf{u}}(\tilde{\mathbf{g}}) = (C\tilde{\mathbf{u}}(\tilde{\mathbf{g}}) + D\tilde{\mathbf{g}}) - C\mathbf{u}(\mathbf{0}) \\ &= (C\tilde{\mathbf{u}}(\tilde{\mathbf{g}}) + D\tilde{\mathbf{g}} - \mathbf{q}) - (D\mathbf{0} + C\mathbf{u}(\mathbf{0}) - \mathbf{q}) \\ &= \mathcal{R}(\tilde{\mathbf{g}}) - \mathcal{R}(\mathbf{0}).\end{aligned} \tag{4.7}$$

That is, to compute $\mathbf{A}\tilde{\mathbf{g}}$, we can use an interpolation to get the residual for the boundary condition.

In general, the Schur complement matrix $\mathbf{A}$ is nonsymmetric. Even A may be nonsymmetric, such as in the other applications in this paper. $\mathbf{A}$ is generally a dense matrix since A^{-1} is. In the boundary integral method, $\mathbf{A}$ is close to the discretization of the second kind integral equation. In the examples presented in this paper, the matrices A, B, C, and D are not explicitly formed. The matrices C and D are sparse and rely on the interpolation scheme used to approximate the boundary condition. There are about $3 \sim 16$ entries each row, depending on the geometry of the boundary and its neighboring grid points for different applications. C and D are determined from the interpolation scheme to approximate the boundary condition. Hence, we generally do not have C^T and D^T available if the matrices are not formed. Thus in practice, $\mathbf{A}$ is not explicitly available, and the multiplication of $\mathbf{A}^T$ by vectors is not convenient.

4.2.3 Features of the Schur complement systems and challenges in GMRES solutions

As mentioned above, the Schur complement matrices $\mathbf{A}$ in AIIM such as (4.5) are usually dense and not explicitly formed. On the other hand, $\mathbf{A}$ can be multiplied by vectors quickly with the aid of fast solvers (e.g., Poisson solvers). Thus, iterative methods such as GMRES are usually used to solve the Schur complement system

$$\mathbf{A}\mathbf{g} = \mathbf{b}. \tag{4.8}$$

For the problems we consider, the following features or challenges are often observed.

- For many Schur complement systems resulting from AIIM, GMRES without preconditioning has difficulty in converging. By saying this, we mean that the restarted GMRES method stalls or the non-restarted GMRES method converges only when the number of iterations reaches nearly the size N of $\mathbf{A}$. Sometimes, this is due to the ill conditioning of $\mathbf{A}$. However, for various cases here, this happens even if $\mathbf{A}$ is well conditioned. Often, the eigenvalues of $\mathbf{A}$ are scattered around the origin, which is empirically observed to affect the convergence of GMRES. The physical background for the slow convergence is as follows. AIIM can be considered as a generalized boundary integral method without explicitly using Green functions. The augmented variable can be considered as a source term. Thus, if the discrete system corresponds to an integral equation of the second kind, then GMRES can converge quickly. (We refer the readers to [128] for the relation between an augmented approach and the boundary integral method.) One such an example is to solve a Poisson equation on an irregular domain with different boundary conditions. If the system behaves like an integral equation of the first kind, then GMRES converges slowly in general. One such example is to solve a Poisson equation on an irregular domain with both Dirichlet and Neumann boundary conditions defined along part of the boundary. Thus, an effective preconditioner is crucial for the convergence of GMRES.

- $\mathbf{A}$ is usually dense and it is costly to form it. (For example, A^{-1} is involved in (4.5) for a large sparse matrix A.) In fact, it is not convenient to even extract its diagonal. Even if we find the diagonal, it may have zero entries and cannot be used as a preconditioner in a straightforward way.

- $\mathbf{A}\tilde{\mathbf{g}}$ can be conveniently computed for a vector $\tilde{\mathbf{g}}$. Thus, we may extract few columns of $\mathbf{A}$ or multiply $\mathbf{A}$ by certain special vectors (e.g., vector of ones),

so as to construct diagonal or block diagonal preconditioners. However, the preconditioners may be close to singular or may perform poorly.

- $\mathbf{A}^T\tilde{\mathbf{g}}$ cannot be conveniently computed for a vector $\tilde{\mathbf{g}}$. (See the end of the previous subsection.) Thus, even the recent matrix-free direct solution techniques in [71,113] cannot be used to get a preconditioner, since they require the multiplication of both $\mathbf{A}$ and $\mathbf{A}^T$ by vectors to get an approximation to $\mathbf{A}$.
- In some cases, $\mathbf{A}$ is close to be normal or even symmetric.
- The singular values of the off-diagonal blocks of $\mathbf{A}$ have reasonable decay. In some cases, the decay is very fast so that such blocks have small numerical ranks.

As an example, we consider a 680×680 Schur complement $\mathbf{A}$ arising from AIIM for solving the Navier-Stokes equation with a traction boundary condition in Section 4.4.1.1. The mesh size is 240×240. Note that the 2-norm condition number of the matrix is $\kappa_2(\mathbf{A}) = 10.09$, and $\|\mathbf{A} - \mathbf{A}^T\|_2 = 0.02$, $\|\mathbf{A}^T\mathbf{A} - \mathbf{A}\mathbf{A}^T\|_2 = 1.98 \times 10^{-5}$. However, due to the traction boundary condition, restarted GMRES (with 50 inner iterations) applied to (4.8) fails to converge, as shown in Figure 4.1.

Thus, an effective preconditioner is needed. The diagonal of $\mathbf{A}$ contains zero entries and cannot be conveniently used as a preconditioner, even if we could extract it. We will seek to find a structured preconditioner. In fact, the off-diagonal blocks of $\mathbf{A}$ have relatively small numerical ranks. In Figure 4.2, we show the singular values of an $\frac{N}{2} \times \frac{N}{2}$ off-diagonal block of $\mathbf{A}$. Clearly, the block only has few large singular values. Thus, we can use a low-rank form to approximate this block to a reasonable accuracy. Overall, we can approximate $\mathbf{A}$ by a rank structured (e.g., HSS) matrix that can serve as an effective preconditioner. However, this would require either $\mathbf{A}$ explicitly or the multiplication of both $\mathbf{A}$ and $\mathbf{A}^T$ by vectors. In the next section, we address these issues.

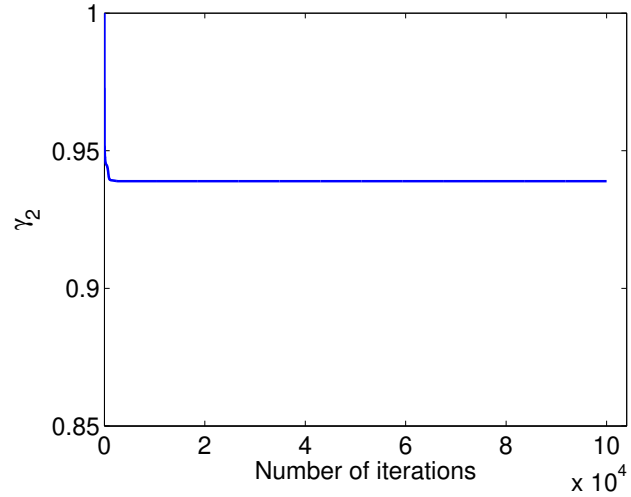


Figure 4.1. Convergence of restarted GMRES without preconditioning for (4.8) from AIIM for solving the Navier-Stokes equations with a traction boundary condition in Section 4.4.1.1, where $N = 680$, the mesh size is 240×240, and $\gamma_2 = \frac{||\mathbf{Ag}-\mathbf{b}||_2}{||\mathbf{b}||_2}$ is the relative residual.

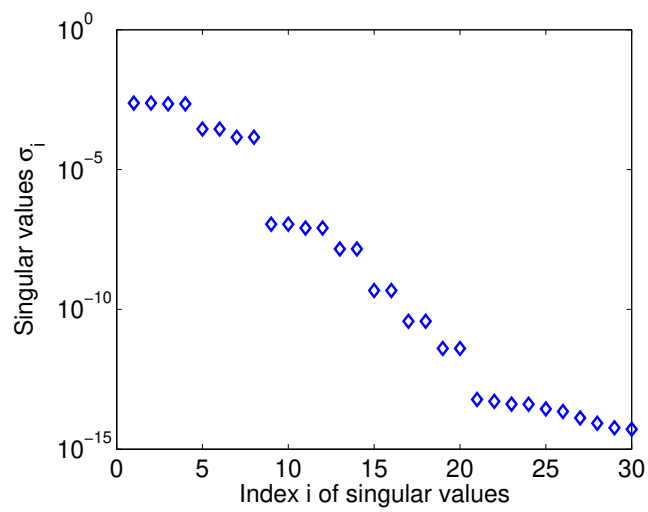


Figure 4.2. The first 30 singular values of $\mathbf{A}(1 : \frac{N}{2}, \frac{N}{2} + 1 : N)$ for the Schur complement $\mathbf{A}$ from AIIM for solving the Navier-Stokes equations with a traction boundary condition in Section 4.4.1.1, where $N = 680$ and the mesh size is 240×240.

4.3 Matrix-free preconditioning techniques for AIIM

The challenges and the tests in the previous section provide a motivation for this work. We show a matrix-free scheme that improves those in [71, 120] and produces a nearly symmetric approximation H to $\mathbf{A}$ in a structured form. H will be quickly factorized and used as a structured preconditioner.

4.3.1 Rank structures

Our preconditioning techniques employ rank structures, or more specifically, HSS representations [12, 117]. An HSS representation provides a convenient format to organize the off-diagonal blocks of a matrix H by low-rank forms. H is partitioned hierarchically, so that the off-diagonal blocks at all the hierarchical levels have low-rank forms and also share certain common bases. A comprehensive summary of HSS structures and algorithms can be found in [114]. Here, we briefly review its definition, with the aid of the following notation:

- $H|_{s\times t}$ is a submatrix of H formed by its entries corresponding to the row index set s and column index set t;
- $H|_s$ is a submatrix of H formed by its rows corresponding to the index set s;
- $H|_{:\times t}$ is a submatrix of H formed by its columns corresponding to the index set t.

A general HSS form looks like $D_k \equiv H|_{\mathcal{I}\times\mathcal{I}}$, where $\mathcal{I}$ is the index set $\{1, 2, \ldots, N\}$, and D_k is defined recursively following a full binary tree T with k nodes labeled as $i = 1, 2, \ldots, k$ in a postorder. Each node i of T is associated with an index set t_i such that

$$t_k = \mathcal{I}, \quad t_i = t_{c_1} \cup t_{c_2}, \quad t_{c_1} \cap t_{c_2} = \emptyset,$$

where c_1 and c_2 are the children of a non-leaf node i and $c_1 < c_2$ (c_1 ordered before c_2). Then for each such i, recursively define

$$D_i = \begin{pmatrix} D_{c_1} & U_{c_1} B_{c_1} V_{c_2}^T \\ U_{c_2} B_{c_2} V_{c_1}^T & D_{c_2} \end{pmatrix}, \quad U_i = \begin{pmatrix} U_{c_1} R_{c_1} \\ U_{c_2} R_{c_2} \end{pmatrix}, \quad V_i = \begin{pmatrix} V_{c_1} W_{c_1} \\ V_{c_2} W_{c_2} \end{pmatrix},$$

where the matrices D_i, U_i, etc. are called the *generators* that define the HSS form of H. Also, if we associate each node i of T with the corresponding generators, T is called an *HSS tree*. T can be used to conveniently organize the storage of H and perform HSS operations. The off-diagonal blocks we are interested in are $H|_{t_i \times (\mathcal{I} \setminus t_i)}$ and $H|_{(\mathcal{I} \setminus t_i) \times t_i}$, called *HSS block rows and columns*, respectively. U_i gives a column basis for $H|_{t_i \times (\mathcal{I} \setminus t_i)}$, and V_i gives a column basis for $(H|_{(\mathcal{I} \setminus t_i) \times t_i})^T$. For convenience, assume U_i and V_i have orthonormal columns. In standard HSS computations, the generators are usually dense. However, particular HSS construction algorithms may yield generators that have additional internal structures (see, e.g., [115, 120]).

Later, we use par(i) and sib(i) to denote the parent and the sibling of i, respectively. Also we say c_1 is a left node and c_2 is a right one.

4.3.2 Matrix-free structured preconditioning via randomized sampling

HSS preconditioning for symmetric positive definite problems have been discussed in [118]. Here, our matrices are generally nonsymmetric and indefinite. The primary idea of our preconditioning techniques is to construct a nearly-symmetric HSS approximation H to $\mathbf{A}$ with an improved matrix-free HSS construction scheme via only the products of $\mathbf{A}$ and vectors. The major components are as follows.

1. For an off-diagonal block of $\mathbf{A}$, use an adaptive randomized sampling method [46, 113] to find an approximate column basis, and a deterministic method to find an approximate row basis.
2. In a top-down traversal of the HSS tree, use the products of $\mathbf{A}$ and vectors to obtain hierarchically low-rank approximations to certain off-diagonal blocks.

Treat $\mathbf{A}$ as a symmetric matrix to approximate the other off-diagonal blocks. We avoid using pseudoinverses and some matrix multiplications in [71, 113].

3. This further enables us to approximate the diagonal blocks, and we then obtain a nearly symmetric HSS approximation H to $\mathbf{A}$.
4. Compute a structured ULV-type factorization [12, 117] of H. Use the factors as a preconditioner in a ULV solution scheme.

The detailed improvements over existing similar schemes are elaborated in the following subsections.

4.3.2.1 Improved randomized compression

Firstly, we explain briefly the adaptive randomized sampling ideas, and show our improvement to the randomized compression. For an $M \times N$ matrix Φ of rank or numerical rank r, we seek to find a low-rank approximation of the form

$$\Phi \approx UBV^T.$$

We multiply Φ and random vectors and use adaptive randomized sampling to find U as in [46, 108]. Then unlike the methods in [46, 71, 108, 113], we do not multiply Φ^T with random vectors when finding V. Furthermore, we do not use pseudoinverses to find B. These are detailed as follows.

Initially, let X be an $M \times \tilde{r}$ Gaussian random matrix, where $\tilde{r}$ is a conservative estimate of r. Compute the product

$$Y = \Phi X,$$

and a QR factorization

$$Y = U\tilde{Y}. \tag{4.9}$$

U is expected to provide a column basis matrix for Φ when $\tilde{r}$ is close to r. To quickly estimate how well $U(U^T\Phi)$ approximates Φ, the following bound can be used with high probability [46, 108]:

$$||\Phi - U(U^T\Phi)||_2 \leq \eta\sqrt{\frac{2}{\pi}} \max_{j=\tilde{r}-d+1,\ldots,\tilde{r}} \quad (I - UU^T)Y|_{:\times j} \quad {}_2\,, \tag{4.10}$$

where d is a small integer and η is a real number, and d and η determine the probability. If the desired accuracy is not reached, more random vectors are used. When this stops, we obtain the approximate basis matrix U and the rank estimate.

In existing randomized sampling methods, it then usually multiplies Φ^T with random vectors to find V in a similar way. That is, randomized sampling are used twice to extract both the row and the column basis. Here, instead, we compute

$$\tilde{\Phi} = U^T\Phi, \tag{4.11}$$

and then compute an RQ factorization

$$\tilde{\Phi} = BV^T. \tag{4.12}$$

That is, we use randomized sampling only once, but use the deterministic orthogonal way (4.11)–(4.12) to find V. This is potentially beneficial for the approximation. Furthermore, (4.12) provides both B and V without the need of pseudoinverses in [71, 113]. The RQ factorization is generally much faster and much more stable. This idea is similar to a deterministic compression strategy for HSS construction in [117], and a parallel implementation is later discussed in [73].

4.3.2.2 Nearly-symmetric matrix-free HSS construction with improvements

We then apply the improved randomized compression to the HSS blocks of $\mathbf{A}$:

$$\Phi = \mathbf{A}|_{t_i\times(\mathcal{I}\setminus t_i)} \quad \text{or} \quad \mathbf{A}|_{(\mathcal{I}\setminus t_i)\times t_i}, \tag{4.13}$$

The compression is done hierarchically for all the HSS blocks. Previous attempts to find accurate HSS approximations to problems with low-rank off-diagonal blocks are

made in [71,77,113,120], where the products of both $\mathbf{A}$ and $\mathbf{A}^T$ with random vectors are needed. As compared with the original matrix-free HSS constructions in [71,113], the following improvements and modifications are made:

- A nearly-symmetric HSS approximation is constructed. That is, for a node i of the HSS tree and $j = \mathrm{sib}(i)$, $V_i \equiv U_i$, $W_i \equiv R_i$, $B_j = B_i^T$, but the D generators are nonsymmetric.
- Randomized sampling is only used to find U_i for the left nodes i. To find U_i for the right nodes i, the deterministic way (4.11)–(4.12) is used instead. This reduces the number of matrix multiplications, and avoids expensive and potentially unstable pseudoinverses.
- Specifically for the purpose of preconditioning, a low approximation accuracy and small $\tilde{r}$ and d are used in (4.10).

The details are as follows. Assume that T is the desired HSS tree. As in [71,113], the nodes i of T are visited in a top-down way, so as to construct the column bases hierarchically for the HSS blocks $\mathbf{A}|_{t_i \times (\mathcal{I} \setminus t_i)}$. For convenience, we use $\tilde{T}_l$ and $\hat{T}_l$ to denote the sets of left and right nodes at a level l of T, respectively. Also, let the root of T be at level 0 and the leaves be at the largest level $l_{\max}$. For $1 \le l \le l_{\max}$, define

$$\tilde{\mathbf{t}}_l = \bigcup_{i \in \tilde{T}_l} t_i, \quad \hat{\mathbf{t}}_l = \bigcup_{j \in \hat{T}_l} t_j.$$

Let X be a Gaussian random matrix whose column size is decided via adaptive randomized sampling applied to $\mathbf{A}|_{t_i \times t_j}$ for each $i \in \tilde{T}_l$. Define $\tilde{X}$ so that

$$\tilde{X}|_{\tilde{\mathbf{t}}_l} = X|_{\tilde{\mathbf{t}}_l}, \quad \tilde{X}|_{\hat{\mathbf{t}}_l} = 0. \tag{4.14}$$

Compute

$$\tilde{Y} = \mathbf{A}\tilde{X} - \tilde{Z},$$

where $\tilde{Z}$ is the product of the partially computed HSS form (due to recursion at upper levels) and $\tilde{X}$, denoted

$$\tilde{Z} = \mathsf{hssmv}(\mathbf{A}, \tilde{X}, l-1).$$

($\tilde{Z}$ is 0 if $l = 1$.) Then compute a QR factorization

$$\tilde{Y}|_{t_i} = \bar{U}_i \bar{S}_i,$$

where $\bar{U}_i$ is a column basis matrix for $\mathbf{A}|_{t_i \times t_j}$ with $j = \mathrm{sib}(i)$.

To find a row basis matrix for $\mathbf{A}|_{t_i \times t_j}$, unlike the methods in [71, 113] that still uses randomized sampling, we use a deterministic way. Define a matrix $\hat{X}$, which is a zero matrix except for each $i \in \tilde{T}_l$,

$$\hat{X}|_{t_i} = \bar{U}_i.$$

Compute

$$\hat{Y} = \mathbf{A}\hat{X} - \mathsf{hssmv}(\mathbf{A}, \hat{X}, l-1).$$

Then for each right node j at level l, compute a QR factorization

$$\hat{Y}|_{t_j} = \bar{U}_j \bar{B}_j,$$

where $\bar{U}_j$ is a column basis matrix for $(\mathbf{A}|_{t_i \times t_j})^T$.

We are then ready to find the generators. If $l = 1$, simply set $U_i = \bar{U}_i$, $B_i = \bar{B}_j^T$. Otherwise, after $j = \mathrm{sib}(i)$ is also visited, let $p = \mathrm{par}(i)$ and partition U_p as $\begin{pmatrix} U_{p;1} \\ U_{p;2} \end{pmatrix}$ so that $U_{p;1}$ has the same row size as $\bar{U}_i$ (assuming i is a left node). Then compute QR factorizations

$$\begin{pmatrix} \bar{U}_i & U_{p;1} \end{pmatrix} = U_i \begin{pmatrix} S_i & R_i \end{pmatrix}, \quad \begin{pmatrix} \bar{U}_i & U_{p;2} \end{pmatrix} = U_j \begin{pmatrix} S_j & R_j \end{pmatrix}.$$

Also, set

$$B_i = S_j \bar{B}_j S_i^T.$$

In comparison, multiple pseudoinverses and a lot more matrix multiplications are needed in [71]. An improved version is given in [113], but still needs two more matrix multiplications and a pseudoinverse on top of two randomized sampling stages. Another strategy to avoid the pseudoinverse is later given in [73], and can potentially

better reveal the hierarchical structures. However, it needs additional randomized sampling and matrix-vector multiplications. Here, since our purpose is preconditioning, the strategy above is sufficient and preferable.

This process is repeated for the nodes of T in a top-down traversal. After the leaf level is traversed, approximations to all the HSS blocks are obtained. We can then approximate all the diagonal blocks $\mathbf{A}|_{t_i \times t_i}$ for the leaves i by subtracting the products of appropriate off-diagonal blocks of $\mathbf{A}$ and $\mathbf{I} = (I, I, \ldots, I)^T$ from $\mathbf{AI}$ [71]. (Some additional zero columns may be needed for certain block rows of $\mathbf{I}$.) More specifically,

$$\mathbf{A}|_{t_i \times t_i} \approx H|_{t_i \times t_i} \equiv \mathbf{AI} - \mathsf{hssmv}(\mathbf{A}, \mathbf{I}, l_{\max}).$$

By now, we have obtained an HSS approximation H to $\mathbf{A}$, where the off-diagonal blocks have a symmetric pattern, or

$$H|_{t_i \times t_j} = (H|_{t_j \times t_i})^T.$$

The diagonal blocks $D_i \equiv H|_{t_i \times t_i}$ are nonsymmetric. The accuracy of this HSS approximation may be low, but it suffices for our preconditioning purpose.

4.3.2.3 Nearly-symmetric HSS factorization

At this point, we can quickly factorize the HSS matrix H, and the factors are used for preconditioning. Since H is nearly symmetric, we use the fast ULV factorization in [117], with some simplifications to take full advantage of the off-diagonal symmetric pattern and with some modifications to accommodate the nonsymmetric diagonal blocks. The basic idea includes the following steps.

- Introduce zeros into $H|_{t_i \times (\mathcal{I} \setminus t_i)}$ by expanding U_i into an orthogonal matrix $\tilde{U}_i$ and multiplying $\tilde{U}_i$ to $H|_{t_i}$. This just needs to modify D_i as

$$D_i \leftarrow \tilde{U}_i D_i \tilde{U}_i^T.$$

- Partition D_i into a block 2×2 form $\begin{pmatrix} D_{i;1,1} & D_{i;1,2} \\ D_{i;2,1} & D_{i;2,2} \end{pmatrix}$, so that $D_{i;1,1}$ a square matrix with size equal to the column size of U_i. Partially LU factorize D_i:

$$D_i \equiv \begin{pmatrix} D_{i;1,1} & D_{i;1,2} \\ D_{i;2,1} & D_{i;2,2} \end{pmatrix} = \begin{pmatrix} L_{i;1,1} & \\ L_{i.2,1} & I \end{pmatrix} \begin{pmatrix} G_{i;1,1} & G_{i;1,2} \\ & G_{i;2,2} \end{pmatrix},$$

  where $D_{i;1,1} = L_{i;1,1}G_{i;1,1}$ is the LU factorization of $D_{i;1,1}$, and $G_{i;2,2} = D_{i;2,2} - L_{i.2,1}G_{i;1,2}$.

- After these steps for two sibling nodes i and j of T are finished, merge the remaining blocks. Here, this is to simply set

$$D_p \leftarrow \begin{pmatrix} G_{i;2,2} & B_i \\ B_i^T & G_{j;2,2} \end{pmatrix}.$$

  Note that no actual operations are performed. Then we remove i and j from T to obtain a smaller HSS form, called a reduced HSS matrix [115].

- Repeat the above steps on the reduced HSS matrix.

After the factorization, the ULV factors are a sequence of orthogonal and lower and upper triangular matrices and are used as a structured preconditioner. The ULV solution procedure for the preconditioning is similar to that in [117] and is skipped.

4.3.2.4 Algorithm and variation

The details are shown in Algorithm 7, which includes the HSS approximation and the factorization for constructing the preconditioner.

To make the preconditioner easier to use, we may also use a simplified variation by forming a block diagonal matrix $\mathbf{D}$ from the D_i generators:

$$\mathbf{D} = \text{diag}(D_i | i\text{: leaves of } T).$$

Then factorize $\mathbf{D}$ and use the triangular factors as the preconditioner.

To summarize, we have two types of preconditioners:

Algorithm 7 Constructing the matrix-free preconditioner

Input: HSS partition, HSS tree T, compression tolerance (and/or rank bound r), and mat-vec (Schur complement matrix-vector multiplication routine as in (4.7))

Output: HSS factors as a preconditioner

1: **procedure** `mfprec`

2: $\tilde{X} \leftarrow 0, \quad \hat{X} \leftarrow 0$

3: **for** level $l = 1, 2, \ldots, l_{\max}$ **do**

▷ *Improved nearly-symmetric matrix-free HSS construction*

4: Construct $\tilde{X}$ as in (4.14) via adaptive randomized sampling

5: $\tilde{Y} \leftarrow \mathsf{mat\text{-}vec}(\mathbf{A}, \tilde{X}) - \mathsf{hssmv}(\mathbf{A}, \tilde{X}, l-1)$ ▷ *hssmv returns* 0 *if* $l = 1$

6: **for** each left node i at level l **do**

7: $\tilde{Y}|_{t_i} = \bar{U}_i S_i$ ▷ *QR factorization*

8: $\hat{X}|_{t_i} \leftarrow \bar{U}_i$

9: **end for**

10: $\hat{Y} \leftarrow \mathsf{mat\text{-}vec}(\mathbf{A}, \hat{X}) - \mathsf{hssmv}(\mathbf{A}, \hat{X}, l-1)$

11: **for** each right node j at level l **do**

12: $\hat{Y}|_{t_j} = \bar{U}_j \bar{B}_j$ ▷ *QR factorization*

13: $i \leftarrow \mathrm{sib}(j)$

14: **if** $l = 1$ **then** ▷ *i is a child of the root*

15: $U_i \leftarrow \bar{U}_i, \quad U_j \leftarrow \bar{U}_j, \quad B_i \leftarrow \bar{B}_j^T$ ▷ *U, B generators*

16: **else**

17: $U_{\mathrm{par}(i)} = \begin{pmatrix} U_{\mathrm{par}(i);1} \\ U_{\mathrm{par}(i);2} \end{pmatrix}$ ▷ Partition so that $U_{\mathrm{par}(i);1}$ *and* $\bar{U}_i$ *have the same row size*

18: $\begin{pmatrix} \bar{U}_i & U_{\mathrm{par}(i);1} \end{pmatrix} = U_i \begin{pmatrix} S_i & R_i \end{pmatrix}, \quad \begin{pmatrix} \bar{U}_j & U_{\mathrm{par}(i);2} \end{pmatrix} = U_j \begin{pmatrix} S_j & R_j \end{pmatrix}$ ▷ *QR factorizations for U, R generators*

19: $B_i \leftarrow S_j \bar{B}_j S_i^T$ ▷ *B generator*

20: **end if**

▷ *Continue in Algorithm 8*

Algorithm 8 Constructing the matrix-free preconditioner (continued)

```
        end for
    end for
    Ỹ ← AI − hssmv(A, I, l_max)
    for each leaf i do
        D_i ← Ỹ|_{t_i}                                          ▷ D generator
    end for
    for node i = 1, 2, ..., k do
                                ▷ ULV factorization to generate the preconditioner
        Ũ_i ← U_i
    ▷ Extension of U_i to orthogonal Ũ_i; implicit zero introduction into A|_{t_i × (I \ t_i)}
        D_i ← Ũ_i D_i Ũ_i^T                                     ▷ Diagonal block update
        D_i = ( L_{i;1,1}  )( G_{i;1,1}  G_{i;1,2} )            ▷ Partial LU factorization
              ( L_{i.2,1}  I )( G_{i;2,2}          )
        if i is a nonleaf node then
            D_i ← ( G_{c_1;2,2}  B_{c_1} )                      ▷ c_1, c_2: children of i
                  ( B_{c_1}^T    G_{c_2;2,2} )
        end if
    end for
end procedure
```

- Preconditioner I: Structured preconditioner given by the ULV factors of the HSS approximation H to $\mathbf{A}$;
- Preconditioner II: Block-diagonal preconditioner given by the LU factors of the block diagonal matrix $\mathbf{D}$ formed by the D_i generators of H. This preconditioner is easier to apply, although it may lead to slightly slower convergence.

For convenience, we may simply say that H or $\mathbf{D}$ is the preconditioner.

4.3.3 Efficiency and effectiveness

The HSS construction costs $O(r^2N)$ flops plus the cost to multiply $\mathbf{A}$ with $O(r \log N)$ vectors, where r is the maximum numerical rank of the HSS blocks. The factorization and the preconditioning costs are $O(r^2N)$ and $O(rN)$ flops, respectively. We may also treat the diagonal blocks as symmetric ones so as to further save the costs. The storage for the preconditioner is $O(rN)$.

In the preconditioning, we use a low accuracy so that r is a very small integer. The preconditioning cost at each GMRES iteration is then $O(N)$. This cost is negligible as compared with the cost of multiplying $\mathbf{A}$ and a vector. The reason is that $\mathbf{A}$ corresponds to the interface and has a much smaller size than the entire problem. See, e.g., (4.4)–(4.5), where the multiplication of $\mathbf{A}$ and a vector needs to solve a Poisson problem represented by A. The matrix A has a size much larger than N.

Just like many other preconditioning techniques, a full analytical justification of the effectiveness of the preconditioner is not yet available. Especially, the complex nature of the rank structures makes the analysis a nontrivial issue. However, several aspects of rank structured approximation and randomized preconditioning are closely related and give useful heuristical explanations.

1. HSS representations recursively capture the algebraic structure of the discretization with the off-diagonal blocks corresponding to the interactions of subdomains. A low-accuracy off-diagonal approximation represents essential basic information within the subdomain interaction.

2. Keeping few largest off-diagonal singular values in rank structured approximation tends to have an effect of roughly preserving certain eigenvalues of the original matrix [107, 113]. In particular, if some eigenvalues are well separated from the others, then a low-accuracy HSS approximation can give a much more accurate approximation of these eigenvalues. This structured perturbation analysis is shown in [107]. Therefore, when the HSS approximation is used as a preconditioner, it can potentially help to bring the eigenvalues closer. Although this

does not necessarily guarantee the fast convergence of GMRES [35,106], it gives an empirical way to look into the behavior of the preconditioner.

3. Even if the problem may not have the low-rank property or the off-diagonal singular values only slowly decay, a structured approximation as a preconditioner can significantly accelerate the decay of the condition number κ [118], where the preconditioner is obtained with different numbers of off-diagonal singular values kept in the approximation.

4. In our preconditioner, multiplications of $\mathbf{A}$ by random matrices are used to approximate the off-diagonal blocks by low-rank forms, which are further used to approximate the diagonal blocks via matrix addition/subtraction. This is then similar to the idea of additive preconditioning [87, 88], where random low-rank matrices are added to the original matrix, which is shown to yield a well-conditioned preconditioner, and also improves the condition number of the original matrix.

4.4 Preconditioning AIIM for different applications and generalizations

In this section, we show several important applications where our preconditioning techniques for AIIM can be applied, particularly for flow problems. Numerical results are given to demonstrate the effectiveness and efficiency. We focus on the structured Preconditioner I given at the end of Section 4.3.2, and also briefly show the performance of Preconditioner II.

4.4.1 Preconditioning individual Schur complements in various applications

In this subsection, we show the preconditioning of some individual Schur complements in several applications. In our tests, different types of right-hand sides $\mathbf{b}$ are tried, such as random vectors, products of $\mathbf{A}$ and given vectors, and those actually

arising in the applications. Similar performance is observed. Thus for convenience in comparing different methods, we report the results with $\mathbf{b}$ to be the products of $\mathbf{A}$ and vectors of all ones. We point out that our preconditioners are not limited to any specific type of right-hand sides. For convenience, the following notation is used:

- $\kappa_2(\mathbf{A})$: 2-norm condition number of $\mathbf{A}$;
- $n_{\text{mv}}^{\text{pre}}$: number of matrix-vector multiplications for computing the preconditioner;
- n_{it}: number of GMRES iterations;
- $\gamma_2 = \frac{||\mathbf{Ag}-\mathbf{b}||_2}{||\mathbf{b}||_2}$: relative residual.

Before presenting the details of the individual problems, we give the basic properties of the matrices in Table 4.1, and summarize the convergence results in Table 4.2. The matrices include both well-conditioned and ill-conditioned ones. We can observe that, GMRES has difficulty to converge for all the cases. In fact, restarted GMRES fails to converge for all the cases except one (where it takes 8794 steps to converge for $N = 194$). Even non-restarted GMRES needs large numbers of iterations to converge, which are often close to N. On the other hand, preconditioned GMRES with our structured preconditioner gives quick convergence for all the cases. Both $n_{\text{mv}}^{\text{pre}}$ and n_{it} are relatively insensitive to the increase of N.

Remark 4.4.1 *Since* $\mathbf{A}$ *corresponds to the interface for the augmented variable, the size* N *of* $\mathbf{A}$ *is usually not very large. However, the number of variables in the original problem is much larger. See the mesh size in Table 4.1. Even if* $\mathbf{A}$ *may have a small size, the multiplication of* $\mathbf{A}$ *by a vector needs the solution of a much larger equation for* A*, e.g., with a Poisson solver.*

Remark 4.4.2 *For time dependent problems and multiple right-hand sides, we may use* $n_{\text{mv}}^{\text{pre}}$ *matrix-vector multiplications in a precomputation to find a preconditioner, and then apply the preconditioner to multiple time steps and right-hand sides. Thus, it sometimes makes sense to allow* $n_{\text{mv}}^{\text{pre}}$ *to be slightly larger so as to reduce* n_{it}*. This would be beneficial in reducing the total number of matrix-vector products.*

Table 4.1.
Properties of $\mathbf{A}$, including the corresponding mesh size for the entire domain.

Problem	Mesh	N	$\kappa_2(\mathbf{A})$	$\|\|\mathbf{A}-\mathbf{A}^T\|\|_2$	$\frac{\|\|\mathbf{A}-\mathbf{A}^T\|\|_2}{\|\|\mathbf{A}\|\|_2}$	$\|\|\mathbf{A}^T\mathbf{A}-\mathbf{A}\mathbf{A}^T\|\|_2$	$\frac{\|\|\mathbf{A}^T\mathbf{A}-\mathbf{A}\mathbf{A}^T\|\|_2}{\|\|\mathbf{A}^T\mathbf{A}\|\|_2}$
`Traction`	60×60	176	9.62	0.02	1.43	2.36×10^{-5}	0.10
	120×120	344	9.48	0.02	1.40	2.23×10^{-5}	0.09
	240×240	680	10.09	0.02	1.41	1.98×10^{-5}	0.09
	480×480	1360	13.71	0.02	1.42	1.66×10^{-5}	0.08
`Inextensible`	128×128	128	$1.55e5$	4.08	0.69	9.33	0.27
	256×256	256	$1.34e7$	8.05	0.70	39.05	0.29
	512×512	512	$2.31e6$	18.65	0.51	176.85	0.13
`Contact`	128×128	182	$6.79e2$	1.01	1.02	0.86	0.88
	256×256	362	$9.56e2$	1.02	1.03	0.86	0.87
	512×512	726	$4.76e3$	1.06	0.99	0.89	0.78
`Mix_Irregular`	64×64	194	$3.16e3$	1.03	0.98	0.91	0.82
	128×128	274	$3.46e3$	1.26	0.97	1.52	0.89
	256×256	514	$1.71e4$	2.21	0.99	4.83	0.97
	512×512	834	$2.54e5$	3.74	1.00	13.91	0.99
	1024×1024	1314	$8.59e6$	6.91	1.00	47.58	1.00

Table 4.2.
Summary of the convergence of GMRES without preconditioning and with our structured Preconditioner I, where τ is around $0.1 \sim 0.8$ in constructing the preconditioner.

Problem	GMRES				Preconditioned GMRES		
	Restarted		Non-restarted		(Non-restarted)		
	n_{it} (outer; inner)	γ_2	n_{it}	γ_2	$n_{\text{mv}}^{\text{pre}}$	n_{it}	γ_2
`Traction`	91750 (1835; 50)	Fail	175	$1.43e-4$	63	12	$3.73e-7$
	93650 (1873; 50)	Fail	343	$1.34e-5$	88	11	$5.83e-7$
	9500 (190; 50)	Fail	672	$3.11e-6$	85	12	$5.14e-7$
	3300 (66; 50)	Fail	1331	$1.71e-6$	106	13	$9.01e-7$
`Inextensible`	55750 (1115; 50)	Fail	127	$1.63e-4$	73	26	$7.90e-7$
	30200 (604; 50)	Fail	250	$8.91e-7$	92	42	$7.96e-8$
	100000 (2000; 50)	Fail	492	$4.63e-7$	110	68	$2.94e-7$
`Contact`	49250 (985; 50)	Fail	181	$6.98e-6$	71	21	$3.47e-7$
	81500 (1630; 50)	Fail	361	$5.20e-7$	83	30	$5.47e-7$
	3150 (63; 50)	Fail	725	$1.71e-6$	99	48	$8.01e-7$
`Mix_Irregular`	8794 (176; 50)	$1.00e-6$	118	$7.03e-7$	68	22	$7.82e-7$
	14150 (283; 50)	Fail	156	$8.14e-7$	79	39	$2.47e-7$
	95200 (1904; 50)	Fail	328	$9.48e-7$	81	64	$7.61e-7$
	7300 (146; 50)	Fail	633	$9.18e-7$	102	84	$9.60e-7$
	33950 (679; 50)	Fail	1045	$9.36e-7$	105	98	$9.09e-7$

Remark 4.4.3 *The measurements in Table 4.1 only give a partial way to look into the symmetry/normality of* $\mathbf{A}$*, and do not necessarily tell the actual symmetry/normality. In practice, we may not know if* $\mathbf{A}$ *is actually close to symmetric or not. For some of the examples, the eigenvalues of* $\mathbf{A}$ *are scattered around (say, in a disk) and are not close to the real axis. In general, the structure of the Schur complement matrix depends on the application and the representation of the interface. For our test ex-*

amples, it is hard to tell whether the matrix is eventually close to symmetric or not, but the preconditioner works well.

Remark 4.4.4 *The reasons why we are not reporting results for restarted GMRES are as follows. First, the numbers of iterations in our tests are usually small. Next, if restart is used, the convergence is slower than without, depending on the number of inner iterations. However, in AIIM with preconditioning, it seems preferable to store a little more intermediate iterates (as in non-restarted GMRES) so as to accelerate the convergence. This is because the sizes of the original discretized PDEs are much larger than the sizes of the Schur complement matrices* $\mathbf{A}$*. It is usually cheap to store some small intermediate iterates in GMRES, but is costly to compute matrix-vector products. Therefore, reducing the total number of matrix-vector products is more crucial (than saving a small amount of storage for the GMRES iterates). Of course, for large-scale practical computations, we may choose to balance the convergence and the storage via restart with appropriate numbers of inner iterations.*

We then explain the individual applications in Tables 4.1–4.2 and show additional specific tests in the following subsections.

4.4.1.1 Navier-Stokes equations on irregular domains with open and traction boundary conditions

This application is one of our motivations to develop the preconditioner. A detailed description of the problem and its solution with AIIM can be found in [68].

Let $\mathbf{R}$ be a rectangular domain with an inclusion Ω. Consider the Navier-Stokes equation

$$\rho\left(\frac{\partial u}{\partial t} + (u\cdot\nabla)u\right) + \nabla p = \mu\Delta u + g, \quad \nabla\cdot u = 0, \qquad \mathbf{x}\in\mathbf{R}\setminus\Omega, \tag{4.15}$$

where ρ, μ, u, p, and $g(x, y, t)$ are the fluid density, the viscosity, the fluid velocity, the pressure, and an external forcing term, respectively. There is a traction boundary condition [72] along the interior boundary $\partial\Omega$:

$$n^T \cdot \mu(\nabla u + \nabla u^T) \cdot n = p - \hat{p} - \gamma\xi, \quad \eta^T \cdot \mu(\nabla u + \nabla u^T) \cdot n = 0, \qquad (4.16)$$
$$\mathbf{x} \in \partial\Omega,$$

where ξ is the curvature, γ is the coefficient of the surface tension, η is the unit tangential direction of the interface, and $\hat{p}$ is the pressure of the air.

During the numerical solution, the procedure to advance from a time step t^k to the next one t^{k+1} includes two steps [68]. The first is to solve (4.15) with (4.16) for the velocity using AIIM. The second is to solve the momentum equation for the pressure p. The augmented variable is $q^{k+1} \equiv \frac{\partial u^{k+1}}{\partial n}$, which is determined so that u^{k+1} satisfies (4.16) with an approximation to p^{k+1}.

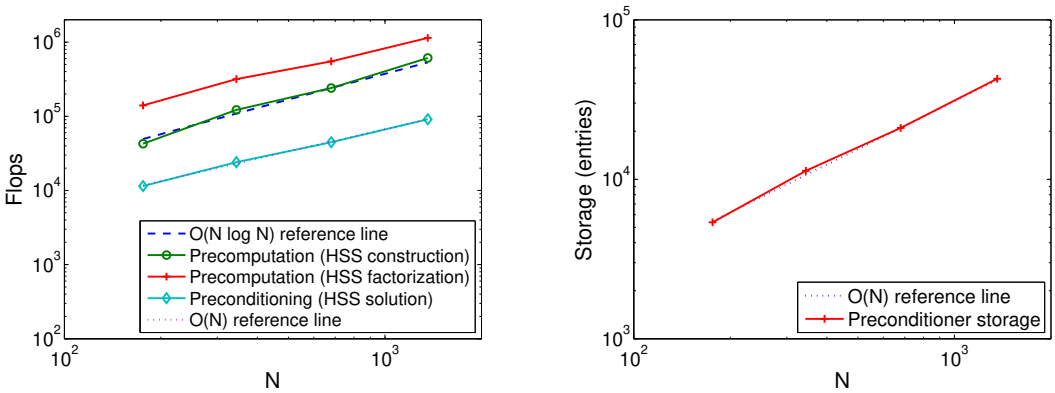
In a specific example, set the boundary $\partial\Omega$ to be a particular curve such as a circle defined by a function $\varphi(x, y)$. Let $\varphi_{i,j}$ be the discrete case of $\varphi(x, y)$. A grid point $\mathbf{x}_{ij}$ is irregular if

$$\max\{\varphi_{i-1,j}, \varphi_{i+1,j}, \varphi_{ij}, \varphi_{i,j-1}, \varphi_{i,j+1}\} \cdot \min\{\varphi_{i-1,j}, \varphi_{i+1,j}, \varphi_{ij}, \varphi_{i,j-1}, \varphi_{i,j+1}\} \leq 0.$$

The Schur complement system (4.8) for the augmented variable is defined at the orthogonal projections of such irregular $\mathbf{x}_{ij}$ on $\partial\Omega$ from the outside. If the time step size $\Delta t = t^{k+1} - t^k$ is fixed, then so is $\mathbf{A}$.

(4.8) is solved by GMRES with our preconditioning techniques. Preconditioned GMRES quickly converges. See the results in the row of '`Traction`' in Table 4.2, where we use time $t = 0$ and $\mu = 0.02$. Note that the number of matrix-vector products for constructing the preconditioner and the number of iterations increase very slowly with N. More specifically, Figure 4.3 shows the costs (excluding the matrix-vector multiplication which is problem dependent) and the storage. For varying N, we use a low accuracy $\tau = 0.4$ and a small off-diagonal rank. Then the construction of the preconditioner (including the HSS construction and the ULV factorization) and

the application of the preconditioner (the ULV solution) both cost about $O(N)$ flops and need about $O(N)$ storage. Such costs are negligible as compared with even one matrix-vector product with $\mathbf{A}$, which costs about $O(N^2 \log N)$.



(i) Costs of the algorithms

(ii) Storage for the preconditioner

Figure 4.3. Costs for precomputing or constructing the preconditioner (including HSS construction and ULV factorization) and applying the preconditioner (HSS solution), and the storage for the preconditioner.

In particular, for the example with $N = 680$ in Figure 4.1 in Section 4.2.3, the convergence of preconditioned GMRES is significantly faster than the standard non-restarted GMRES method. (Note that restarted GMRES simply stalls.) See Figure 4.4. It is also observed that most of the eigenvalues of the preconditioned matrix cluster around 1.

We also test the problem at time $t = 0.5$, and the results are given in Table 4.3. The convergence is very close to that for $t = 0$ in Table 4.2.

Furthermore, for the example with $N = 680$, we also test a wide range of μ values. See Table 4.4. Clearly, at both $t = 0$ and 0.5, the convergence preconditioned GMRES is similar for different μ. This illustrates the insensitivity of the convergence to different time steps and physical parameters.

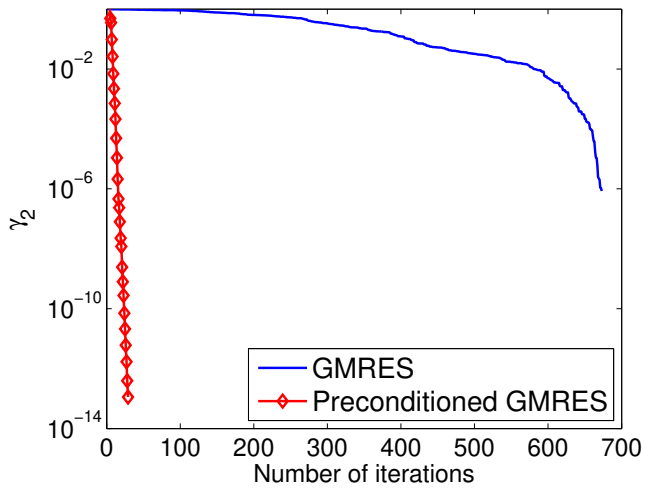


Figure 4.4. Convergence of non-restarted GMRES without preconditioning and with our structured preconditioning for the 680×680 Schur complement $\mathbf{A}$ (corresponding to Figure 4.1) from a 240×240 mesh.

Table 4.3.
Convergence of preconditioned GMRES for the problem '`Traction`' at time $t = 0.5$.

Mesh	$n_{\text{mv}}^{\text{pre}}$	n_{it}	γ_2
60×60	63	13	$4.29e-7$
120×120	88	12	$6.81e-7$
240×240	85	11	$6.37e-7$
480×480	106	13	$6.53e-7$

4.4.1.2 An extensible interface in an incompressible flow

Then we consider an inverse interface problem, which has a moving interface $\partial\Omega(t)$ in a shear flow within a rectangular domain $\mathbf{R}$ [64]. The problem is still modeled by the Navier-Stokes equation:

$$\frac{\partial u}{\partial t} + u \cdot \nabla u + \nabla p = \mu \, \Delta u + f(\mathbf{x}, t), \quad \nabla \cdot u = 0,$$

Table 4.4.
Convergence of preconditioned GMRES for the problem '`Traction`' at times $t = 0$ and 0.5 with different μ values.

μ	$t = 0$			$t = 0.5$		
	$n_{\text{mv}}^{\text{pre}}$	n_{it}	γ_2	$n_{\text{mv}}^{\text{pre}}$	n_{it}	γ_2
2	90	14	$5.87e-7$	90	14	$5.26e-7$
0.2	88	13	$4.04e-7$	88	14	$6.13e-7$
0.02	85	12	$5.14e-7$	85	11	$6.37e-7$
0.002	85	13	$3.63e-7$	85	12	$6.58e-7$

$$\mathbf{x} \in \mathbf{R},$$

where $f(\mathbf{x}, t)$ is the force term. Let η be the tangential direction of $\partial\Omega(t)$. The force term is

$$f(\mathbf{x}, t) = \iint_{\partial\Omega(t)} \frac{\partial}{\partial s} \left(\sigma(s,t)\, \eta(s,t)\right) \delta(\mathbf{x} - \omega(s,t))\, ds,$$

where $\omega(s,t)$ is a parametric representation of $\partial\Omega(t)$, and $\sigma(s,t)$ is the surface tension. The force term $f(\mathbf{x}, t)$ is also part of the unknown. We need to find $f(\mathbf{x}, t)$ such that the incompressible condition along the tangential direction is also satisfied. $\sigma(s,t)$ is chosen as the augmented variable so that

$$(\nabla_s \cdot u)_{\partial\Omega} = \frac{\partial u}{\partial \eta} \cdot \eta \Big|_{\partial\Omega} = 0.$$

That is, both the length of $\partial\Omega(t)$ and the enclosed area remain constant.

For this problem, the number of GMRES iterations is dramatically reduced with our preconditioner. See the results in the '`Inextensible`' row in Table 4.2, where the tests are done at time $t = 0$ with $\mu = 20$.

4.4.1.3 A contact problem of drop spreading

AIIM for modeling a moving contact line problem where a liquid drop spreads can be found in [65]. Just like in the previous problems, the fluid domain Ω with the free boundary $\partial\Omega$ is extended into a rectangular domain $\mathbf{R}$. The augmented variable is also the jump of the normal derivative of the velocity along $\partial\Omega$. The performance of GMRES with our preconditioner is given in the '`Contact`' row in Table 4.2, where the time is $t = 0$.

4.4.1.4 A boundary value problem with mixed boundary conditions on different parts of the boundary

Another important problem where an effective preconditioner is needed in AIIM is the Poisson equation with different types of boundary conditions on different parts of the boundary. As an example, consider the domain Ω enclosed by a half circle $x^2 + y^2 \leq 1$, $x \geq 0$ and the line segment $x = 0$, $-1 \leq y \leq 1$. Assume we have the following mixed boundary conditions:

$$\begin{aligned} u(x,y) &= g_1, \quad x^2 + y^2 = 1, \quad x \geq 0 \quad \text{(Dirichlet)}, \\ \frac{\partial u}{\partial n} &= g_2, \quad x = 0, \quad -1 \leq y \leq 1 \quad \text{(Neumann)}. \end{aligned}$$

Ω is extended to the rectangular domain $\mathbf{R} = [-2,\ 2]^2$, and AIIM is applied to the problem as explained in Section 4.2. The augmented variable is set to be $[\frac{\partial u}{\partial n}]$ along the half circle and $[u]$ along the line segment $x = 0$, $-1 \leq y \leq 1$. See the line of '`Mix_Irregular`' in Table 4.2 for the performance of our preconditioner.

In particular, we also try Preconditioner II given at the end of Section 4.3.2. See Table 4.5. We observe satisfactory convergence results too, though slightly slower than those in Table 4.2.

Table 4.5.
Convergence of GMRES with our Preconditioner II for the problem '`Mix_Irregular`' in Table 4.1.

Mesh	n_{it}	γ_2
64×64	24	$6.59e-7$
128×128	38	$9.12e-7$
256×256	66	$4.44e-7$
512×512	97	$8.34e-7$
1024×1024	111	$7.88e-7$

4.4.2 Comprehensive simulation in terms of a free boundary problem

We then show a comprehensive test in terms of a free boundary problem [66], including all the stages of an entire simulation.

This problem is to determine the position of the crack of certain minimizers. It is a mixed boundary value problem where we need to find a potential function $u(x, y)$ and a free boundary $\Gamma_1(x, y)$ such that

$$\Delta u = 0 \quad \text{in} \quad \Omega, \tag{4.17}$$

$$u|_{\Gamma_2} = w, \tag{4.18}$$

$$\frac{\partial u}{\partial \nu}_{\Gamma_1} = 0, \quad \frac{\pi}{2} c = \left[\nabla u^+ {}^2 - \nabla u^- {}^2 \right]_{\Gamma_1}, \tag{4.19}$$

where ν is the unit normal of Γ_1 and c is the curvature of Γ_1. The other boundary Γ_2 is fixed, and the Dirichlet boundary condition is defined on Γ_2. Note that the parts of free boundary Γ_1 below and above the x-axis are antisymmetric, and the free boundary has three fixed points: the top ($y > 0$) corner, the bottom ($y > 0$) corner, and the origin ($(0, 0)$). Figure 4.5 gives an illustration. Additional plots of the final shape of the boundary can be found in [66].

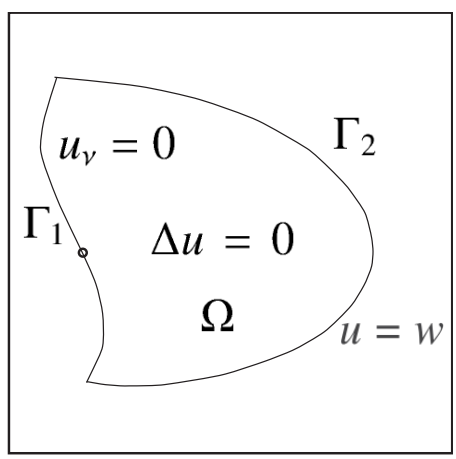


Figure 4.5. An illustration of the free boundary problem (4.17)–(4.19) in [66], where Γ_1 is a free boundary and Γ_2 is fixed.

As usual, we use an iterative scheme to solve this free boundary problem. The free boundary Γ_1 can be written as a perturbation to the line $(\hat{x}_0(t), \hat{y}_0(t)) = (a_1 t, a_2 t)$ connecting the origin and the top corner, where (a_1, a_2) is the direction of the line. Denote the free boundary above the x-axis as $(\hat{x}(t), \hat{y}(t)) = (t, g(t))$ with $g(0) = 0$, $g(d) = 0$, where d is the number such that $(\hat{x}_0(d), \hat{y}_0(d))$ is the fixed top corner.

From the analysis in [66], $g(t)$ is the solution to the following non-linear equation which depends on $u(x, y)$:

$$\frac{(t^2 + g^2(t))g''(t)}{(1 + g'^2(t))^{\frac{3}{2}}} + \frac{tg'(t) - g(t)}{((t^2 + g^2)(1 + g'^2(t)))^{\frac{1}{2}}} = G(t), \tag{4.20}$$

where $t \in (0, d)$, and

$$G(t) = \frac{1}{\pi}\left(|\partial_\tau u|^2(t, g(t)) - |\partial_\tau u|^2(-t, -g(t))\right).$$

Here, $\partial_\tau u$ is the tangential derivative of $u(x, y)$ along the free boundary Γ_1. A much simplified iterative method is proposed in [66] to solve the non-linear equation (4.20):

$$\frac{(t^2 + g_k^2(t))}{(1 + g_k'^2(t))^{\frac{3}{2}}} g_{k+1}''(t) + \frac{tg_k'(t) - g_k(t)}{((t^2 + g_k^2)(1 + g_k'^2(t)))^{\frac{1}{2}}} = G_k(t), \tag{4.21}$$

where

$$g_{k+1}(0) = g_{k+1}(d) = 0,$$

$$G_k(t) = \frac{1}{\pi}\left(|\partial_\tau u_k|^2 (t, g_k(t)) - |\partial_\tau u_k|^2 (-t, -g_k(t))\right),$$

and $(t, g_k(t))$ defines the boundary of the domain for (4.17)–(4.19) whose solution is u_k. This is much simpler than the Newton iterative method and has fast convergence.

At each stage k of the iteration (4.21), we use the augmented method described in Section 4.2 to solve the Poisson equation on Ω with Dirichlet and Neumann boundary conditions on different parts of the boundary. The augmented variable is the jump in the normal derivative. It is well known that the GMRES converges very slowly since the system is similar to one using the boundary integral method with integral equations of both first and second kinds.

In the simulation, we show the performance of GMRES as well as the preconditioned one. The initial guess of the free boundary is the line connecting the top-left corner and the origin. The Dirichlet boundary condition is

$$w(x,y) = \frac{\mathrm{sgn}(y)}{2}\left((x^2+y^2)^{\frac{1}{2}} - x\right)^{\frac{1}{2}} - 0.01. \tag{4.22}$$

Since multiple stages/iterations (4.21) are involved, the outcome of the previous stage GMRES iteration is used as the initial estimate of the next stage GMRES iteration. Thus, unlike the failure in the tests in the previous subsection, here, GMRES converges to modest accuracy after a certain number of steps.

Nevertheless, the preconditioned GMRES method with our preconditioner converges much faster. In Table 4.6, we show the total number of iterations n_{it} for GMRES at all stages, as well as the total number of matrix-vector multiplications $n_{\mathrm{mv}}^{\mathrm{pre}} + n_{\mathrm{it}}$ for the preconditioned GMRES method. We construct the preconditioner once and use it for all the iterations. The stopping criterion is when the difference in a certain measurement between two consecutive stages k and $k+1$ is smaller than 10^{-6}. The tolerance for the inner GMRES iterations is also 10^{-6}. We run the tests in Fortran on a 2.5 GHz Intel Core i7 Macbook Pro with 16 GB of memory. Both the total number of GMRES iterations and the total CPU time have been greatly reduced.

The numbers of GMRES and preconditioned GMRES iterations at each stage k for the iteration (4.21) is also given in Table 4.7. With preconditioned GMRES, the solution is not only faster, but also more reliable. In fact, with just GMRES, even for the small 40×40 mesh it takes an unusually large number of iterations. For the 160×160 mesh, it even fails to converge to the desired accuracy. However, with preconditioned GMRES, both the number of iterations (4.21) and the numbers of interior GMRES iterations are significantly reduced.

Table 4.6.
Comparison of the entire simulation for the free boundary problem (4.17)–(4.19) with GMRES and preconditioned GMRES for different mesh sizes.

Mesh size	N	With GMRES		With preconditioned GMRES	
		Total n_{it}	Total CPU time (sec)	Total $n_{\text{inv}}^{\text{pre}} + n_{\text{it}}$	Total CPU time (sec)
40×40	96	*1241*	$3.74e0$	150	$5.06e{-1}$
80×80	114	700	$2.90e0$	164	$7.75e{-1}$
160×160	150	*8801*	$7.40e1$	180	$1.57e0$
320×320	222	821	$2.08e1$	183	$4.77e0$
640×640	366	979	$9.25e1$	211	$2.10e1$
1280×1280	652	951	$4.07e2$	227	$9.20e1$
2560×2560	1224	937	$1.69e3$	257	$4.75e2$

Table 4.7.
Comparison of the numbers of iterations at all the stages k for the iteration (4.21).

Mesh	With GMRES	With preconditioned GMRES
40×40	$77, 79, 83, 73, 77, 81, 78, 77, \ldots, 77$ (9 times)	$24, 24, 24, 24, 24$
80×80	$70, 70, 70, 70, 70, 70, 70, 70, 70, 70$	$13, 13, 13, 13, 13, 13, 13, 13$
160×160	$89, 88, \ldots, 88$ (99 times) — divergence	$13, 17, 17, 17, 17, 17$
320×320	$83, 82, 82, 82, 82, 82, 82, 82, 82, 82$	$19, 20, 20, 20, 20, 20$
640×640	$97, 98, 98, 98, 98, 98, 98, 98, 98, 98$	$17, 23, 23, 23, 23$
1280×1280	$96, 95, 95, 95, 95, 95, 95, 95, 95, 95$	$24, 34, 34, 34$
2560×2560	$97, 94, 94, 94, 93, 93, 93, 93, 93, 93$	$25, 33, 33$

4.4.3 Generalizations

Other than AIIM, the preconditioning techniques can also be useful in other applications and methods such as saddle point problems, domain decomposition, hybrid solutions. See, e.g., [8, 31, 91]. In these cases, often a large problem is first solved by direct methods, and a much smaller Schur complement system corresponding to a small subdomain or interface is solved with iterative ones.

The matrix-free preconditioning techniques here are also useful for more general problems where it is difficult to form the matrix $\mathbf{A}$ explicitly or to evaluate the product of $\mathbf{A}^T$ with vectors. This also includes sparse problems such as the GeneRank problems in [109] which are nonsymmetric but have symmetric nonzero patterns or involve other types of symmetry.

4.5 Conclusions

AIIM has significant benefits for the fast and accurate solutions of some interface problems and problems defined on irregular domains. The efficient application of AIIM relies on the fast solution of the Schur complement system $\mathbf{Ag} = \mathbf{b}$. Since

the products of $\mathbf{A}$ with vectors can be quickly evaluated, but not the entries of $\mathbf{A}$ or the products of $\mathbf{A}^T$ with vectors, we propose matrix-free preconditioning techniques to accelerate the convergence of GMRES. Rank structured techniques are combined with adaptive randomized sampling. Several improvements to existing randomized and structured algorithms are made. Various advantages of the preconditioner are demonstrated, including the flexibility, efficiency, and effectiveness, which are supported by comprehensive tests on many difficult situations. In our future work, we would like to analytically study the effectiveness of the preconditioner, at least for certain simple cases (e.g., with few blocks). We also plan to develop a parallel implementation for large-scale tests.

5. Conclusions and future work

To conclude this dissertation, we have discussed several topics regarding the construction and application of HSS representation on both theoretical and computational aspects. On the micro scale of structured matrix computation, we analyzed in details on our own version of the proxy point method for a class of problems, theorems regarding different type of approximation errors are presented. On the macro scale, HSS representation has been used with other algorithms to obtain efficient eigensolver and preconditioner. Now we will list a couple of possible directions for future works related to this dissertation as well as other structured matrix techniques.

Chapter 2 is a starting point for a systematic error analysis for proxy point method in more general case, some settings one commonly encounters in the literature include 1) PDE kernels in 3 dimension, this is mostly seen in simulation of real world physics phenomenon, and 2) Gaussian or other symmetric positive definite (SPD) kernels with high dimension date, this is often seen in kernel methods for machine learning. This is an important part for understanding and controlling the approximation error when applying the structured matrix algorithms to these problems. The results obtained in chapter 2 themselves are also useful that it is closely related to Cauchy-like and Toeplitz systems. Previously we have produced a software package [74] for solving Toeplitz linear systems, now with the new and more efficient proxy point method as the main compression subroutine, we will be able to bring an update to the software for much better performance.

For the eigenvalue problem, the ultimate goal is to get an eigensolver with less than $\mathcal{O}(n^2)$ cost for computing all eigenpairs under the assumption that certain rank structure exists in the matrix. This is not likely to be possible without exploring the structure in eigenvector matrix like in [107], namely, we need to find a special permutation or eigenvalue ordering either explicitly or implicitly hidden in certain

procedure. Thus more work has to be done on the analysis part before diving into algorithm design. Specifically for contour-integral based eigensolvers, more investigations can be made on filter design and quadrature choice as we always need to weigh between more nodes for faster convergence and less nodes for less computation cost.

Systems arising from high frequency PDE problems are always hard to deal with as the they lose the low-rank property and many existing structured matrix techniques no longer have approximately linear complexity. Several successful attempts on this type of problem include high frequency fast multipole (HF-FMM) methods, some directional methods and butterfly factorization, but they are all concerned with fast matrix-vector multiplication which is useful for iterative solvers or preconditioners. We are interested in a direct solver for such problems, this means a ULV type factorization and the corresponding solution scheme like the HSS case.

High performance implementation is not mentioned at all in this work, but it is worth considering when doing real world applications. Structured matrix algorithms are born with some level of natural parallelism since they are usually associated with a tree structure, the remaining work is to modify the algorithms if necessary to get better communication.

REFERENCES

REFERENCES

[1] A. Amir, N. Hatano, and D. R. Nelson, *Non-Hermitian localization in biological networks*, Phys. Rev. E, 93 (2016), pp. 042310.

[2] C. R. Anderson, *An implementation of the fast multipole method without mltipoles*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 923–947.

[3] G. Bao, K. Huang, P. Li, and H. Zhao, *A direct imaging method for inverse scattering using the generalized Foldy-Lax formulation*, Contemp. Math., 615 (2014), pp. 49–70.

[4] F. Bauer and C. Fike, *Norms and exclusion theorems*, Numer. Math., 2 (1960), pp. 137–141.

[5] J. B. Bell, P. Colella, and H. M. Glaz, *A second-order projection method for the incompressible Navier-Stokes equations*, J. Comput. Phys., 85 (1989), pp. 257–283.

[6] S. Bellavia, J. Gondzio, and B. Morini, *A matrix-free preconditioner for sparse symmetric positive definite systems and least-squares problems*, SIAM J. Sci. Comput., 35 (2013), pp. A192–A211.

[7] P. Benner and T. Mach, *Computing all or some eigenvalues of symmetric $\mathcal{H}_l$-matrices*, SIAM J. Sci. Comput., 34-1 (2012), pp. A485–A496.

[8] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numerica, 14 (2005), pp. 1–137.

[9] D. A. Bini, L. Gemignani, and V. Y. Pan, *Fast and stable QR eigenvalue algorithms for generalized companion matrices and secular equations*, Numer. Math., 100 (2005), pp. 373–408.

[10] S. Börm and W. Hackbusch, *Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices*, Computing, 69 (2002), pp. 1–35.

[11] D. Cai and J. Xia, *Bridging the gap between the fast multipole method and fast stable structured factorizations*, preprint, 2016.

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

[13] S. Chandrasekaran, M. Gu, X. Sun, J. Xia, and J. Zhu, *A superfast algorithm for Toeplitz systems of linear equations*, SIAM J. Matrix Anal. Appl., 29 (2008), pp. 1247–1266.

[14] S. Chandrasekaran, M. Gu, J. Xia, and J. Zhu, *A fast QR algorithm for companion matrices*, Oper. Theory Adv. Appl., Birkhauser Basel, 179 (2007), pp. 111–143.

[15] H. Cheng, Z. Gimbutas, P. G. Martinsson, and V. Rokhlin, *On the compression of low rank matrices*, SIAM J. Sci. Comput., 26 (2005), pp. 1389–1404.

[16] I. Chremmos and G. Fikioris, *Spectral asymptotics in one-dimensional periodic lattices with geometric interaction*, SIAM J. Appl. Math., 76 (2016), pp. 950–975.

[17] J. Cullum and M. Tůma, *Matrix-free preconditioning using partial matrix estimation*, BIT Numer. Math., 46 (2006), pp. 711–729.

[18] H. Dai, Z. Geary, and L. P. Kadanoff, *Asymptotics of eigenvalues and eigenvectors of Toeplitz matrices*, J. Stat. Mech., (2009), pp. P05012.

[19] J. Demmel, *Applied Numerical Linear Algebra*, SIAM, 1997.

[20] A. Dutt, M. Gu, and V. Rokhlin, *Fast algorithms for polynomial interpolation, integration, and differentiation*, SIAM J. Numer. Anal., 33 (1996), pp. 1689–1711.

[21] Y. Eidelman, I. Gohberg, and V. Olshevsky, *The QR iteration method for Hermitian quasiseparable matrices of an arbitrary order*, Linear Alg. Appl., 404 (2005), pp. 305–324.

[22] B. Engquist and L. Ying, *Sweeping preconditioner for the Helmholtz equation: Hierarchical matrix representation*, Comm. Pure Appl. Math., 64 (2011), pp. 697–735.

[23] L. Foldy, *The multiple scattering of waves*, Phys. Rev., 67 (1945), pp. 107–119.

[24] W. Fong and E. Darve, *The black-box fast multipole method*, J. Comput. Phys., 228 (2009), pp. 8712–8725.

[25] P. J. Forrester and N. E. Frankel, *Applications and generalizations of Fisher–Hartwig asymptotics*, J. Math. Phys., 45 (2004), pp. 2003–2028.

[26] A. Frieze, R. Kannan, and S. Vempala, *Fast Monte-Carlo algorithms for finding low-rank approximations*, J. Assoc. Comput. Math., 51 (2004), pp. 1025–1041.

[27] M. Fujiwara, *Über die obere Schranke des absoluten Betrages der Wurzeln einer algebraischen Gleichung*, Tôhoku Math. J., 10 (1916), pp. 167–171.

[28] Y. Futamura, H. Tadano, and T. Sakurai, *Parallel stochastic estimation method of eigenvalue distribution*, JSIAM Letters, 2 (2010), pp. 127–130.

[29] I. P. Gavrilyuk, W. Hackbusch, and B. N. Khoromskij, *$\mathcal{H}$-matrix approximation for the operator exponential with applications*, Numer. Math., 92 (2002), pp. 83–111.

[30] I. P. Gavrilyuk, W. Hackbusch, and B. N. Khoromskij, *Hierarchical tensor-product approximation to the inverse and related operators for high-dimensional elliptic problems*, Computing, 94 (2005), pp. 131–157.

[31] L. Giraud and A. Haidar, *Parallel algebraic hybrid solvers for large 3D convection-diffusion problems*, Numer. Algor., 51 (2009), pp. 151–177.

[32] J. Gondzio, *Matrix-free interior point method*, Comput. Optim. Appl., 51 (2012), pp. 457–480.

[33] L. Grasedyck, *Existence and computation of low kronecker-rank approximations for large linear systems of tensor product structure*, Computing, 72 (2004), pp. 247–265.

[34] L. Grasedyck, R. Kriemann, and S. Le Borne, *Parallel black box $\mathcal{H}$-LU preconditioning for elliptic boundary value problems*, Comput. Visual Sci., 11 (2008), pp. 273–291.

[35] A. Greenbaum, V. Pták, and Z. Strakoš, *Any nonincreasing convergence curve is possible for GMRES*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 465–469.

[36] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comp. Phys., 73 (1987), pp. 325–348.

[37] M. Gu, *Subspace iteration randomization and singular value problems*, SIAM J. Sci. Comput., 37 (2015), pp. A1139–A1173.

[38] M. Gu and S. C. Eisenstat, *A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 79–92.

[39] M. Gu and S. C. Eisenstat, *Efficient algorithms for computing a strong rank-revealing QR factorization*, SIAM J. Sci. Comput., 17 (1996), pp. 848–869.

[40] M. Gu, X. S. Li, and P. S. Vassilevski, *Direction-preserving and Schur-monotonic semiseparable approximations of symmetric positive definite matrices*, SIAM. J. Matrix Anal. Appl., 31 (2010), pp. 2650–2664.

[41] S. Güttel, E. Polizzi, P. Tang, and G. Viaud, *Zolotarev quadrature rules and load balancing for the FEAST eigensolver*, SIAM J. Sci. Comput., 37 (2015), pp. 2100–2122.

[42] W. Hackbusch, *A Sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: introduction to $\mathcal{H}$-matrices*, Computing, 62 (1999), pp. 89–108.

[43] W. Hackbusch, B. Khoromskij, and S. Sauter, *On $\mathcal{H}^2$ matrices*, Lectures on Applied Mathematics, Springer-Verlag, Berlin, 2000, pp. 9–29.

[44] W. Hackbusch, B. Khoromskij, S. Sauter, and E. Tyrtyshnikov, *Use of tensor formats in elliptic eigenvalue problems*, Numer. Linear Algebra Appl., 19 (2012), pp. 133–151.

[45] W. Hackbusch, B. Khoromskij, and E. Tyrtyshnikov, *Hierarchical kronecker tensor-product approximations*, Numer. Math., 13 (2005), pp. 119–156.

[46] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288.

[47] K. L. Ho and L. Greengard, *A fast direct solver for structured linear systems by recursive skeletonization*, SIAM J. Sci. Comput., 34 (2012), pp. A2507–A2532.

[48] T. Hou, Z. Li, S. Osher, and H. Zhao, *A hybrid method for moving interface problems with application to the Hele-Shaw flow*, J. Comput. Phys., 134 (1997), pp. 236–252.

[49] J. Hunter, Z. Li, and H. Zhao, *Autophobic spreading of drops*, J. Comput. Phys., 183 (2002), pp. 335–366.

[50] M. F. Hutchinson, *A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines*, J. Commun. Statist. Simula., 19 (1990), pp. 433–450.

[51] A. Jennings and W. J. Stewart, *A simultaneous iteration algorithm for real matrices*, ACM Trans. of Math. Software, 7 (1981), pp. 184–198.

[52] B.-Q. Jin and V. E. Korepin, *Quantum spin chain, Toeplitz determinants and the Fisher-Hartwig conjecture*, J. Stat. Phys., 116 (2004), pp. 79–95.

[53] J. Kestyn, E. Polizzi, and P. Tang, *FEAST eigensolver for non-Hermitian problems*, SIAM J. Sci. Comput., 38 (2016), pp. S772–S799.

[54] B. Klar, *Bounds on tail probabilities of discrete distributions*, Prob. Eng. Inform. Sci., 14 (2000), pp. 161–171.

[55] K. Kress, *Linear integral equations, third edition*, Springer, 2014.

[56] J. I. Latorre and A. Riera, *A short review on entanglement in quantum spin systems*, J. Phys. A: Math. Theor., 42 (2009), pp. 504002.

[57] S. E. Laux, *Solving complex band structure problems with the feast eigenvalue algorithm*, Physical Review B, 86 (2012), pp. 075103.

[58] M. Lax, *Multiple scattering of waves*, Rev. Mod. Phys., 23 (1951), pp. 287–310.

[59] S. Le Borne and L. Grasedyck, *$\mathcal{H}$-matrix preconditioners in convection-dominated problems*, SIAM. J. Matrix Anal. Appl., 27 (2006), pp. 1172–1183.

[60] P.-D. Ltourneau, C. Cecka, and E. Darve, *Cauchy Fast Multipole Method for General Analytic Kernels*, SIAM J. Sci. Comput., 36 (2014), pp. A396–A426.

[61] Z. Li, *A fast iterative algorithm for elliptic interface problems*, SIAM J. Numer. Anal., 35 (1998), pp. 230–254.

[62] Z. Li and K. Ito, *The Immersed Interface Method: Numerical Solutions of PDEs Involving Interfaces and Irregular Domains*, SIAM, 2006.

[63] Z. Li, K. Ito, and M.-C. Lai, *An augmented approach for Stokes equations with a discontinuous viscosity and singular forces*, Comput. Fluids, 36 (2007), pp. 622–635.

[64] Z. Li and M.-C. Lai, *New finite difference methods based on IIM for inextensible interfaces in incompressible flows*, East Asian J. Appl. Math., 1 (2011), pp. 155–171.

[65] Z. Li, M-C. Lai, G. He, and H. Zhao, *An augmented method for free boundary problems with moving contact lines*, Comput. Fluids, 39 (2010), pp. 1033–1040.

[66] Z. Li and H. Mikayelyan, *Fine numerical analysis of the crack-tip position for a Mumford-Shah minimizer*, 2015, arXiv:1511.07733 [math.NA].

[67] R. Li and Y. Saad, *Divide and conquer low-rank preconditioners for symmetric matrices*, SIAM J. Sci. Comput., 35 (2013), pp. A2069–A2095.

[68] Z. Li, L. Xiao, Q. Cai, H. Zhao, and R. Luo, *A semi-implicit augmented IIM for Navier-Stokes equations with open, traction, or free boundary conditions*, J. Comput. Phys., 297 (2015), pp. 182–193.

[69] Z. Li, H. Zhao, and H. Gao, *A numerical study of electro-migration voiding by evolving level set functions on a fixed cartesian grid*, J. Comput. Phys., 152 (1999), pp. 281–304.

[70] E. Liberty, F. Woolfe, P. G. Martinsson, V. Rokhlin, and M. Tygert, *Randomized algorithms for the low-rank approximation of matrices*, Proc. Natl. Acad. Sci. USA, 104 (2007), pp. 20167–20172.

[71] L. Lin, J. Lu, and L. Ying, *Fast construction of hierarchical matrix representation from matrix-vector multiplication*, J. Comput. Phys., 230 (2011), pp. 4071–4087.

[72] J. Liu, *Open and traction boundary conditions for the incompressible Navier-Stokes equations*, J. Comput. Phys., 228 (2009), pp. 7250–7267.

[73] X. Liu, J. Xia, and, M. V. de Hoop, *Parallel randomized and matrix-free direct solvers for large structured dense linear systems*, SIAM J. Sci. Comput., 38 (2016), pp. S508–S538.

[74] X. Liu, Y. Xi, J. Xia, and X. Ye, *Superfast and stable Toeplitz linear and least squares solvers*, 2015, https://www.math.purdue.edu/xiaj (accessed 06/20/2018). Version 2.0.

[75] L. Lukšan and Jan Vlček, *Efficient tridiagonal preconditioner for the matrix-free truncated Newton method*, Appl. Math. Comput., 235 (2014), pp. 394–407.

[76] J. Makino, *Yet another fast multipole method without multipoles–pseudoparticle multipole method*, J. Comput. Phys., 151 (199), pp. 910–920.

[77] P. G. Martinsson, *A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix*, SIAM. J. Matrix Anal. Appl., 32 (2011), pp. 1251–1274.

[78] P.-G. Martinsson, G. Quintana Ort, N. Heavner, and R. van de Geijn, *Householder QR factorization with randomization for column pivoting (HQRRP)*, SIAM J. Sci. Comput., 39 (2017), pp. C96–C115.

[79] P. G. Martinsson and V. Rokhlin, *A fast direct solver for boundary integral equations in two dimensions*, J. Comput. Phys., 205 (2005), pp. 1–23.

[80] P. G. Martinsson and V. Rokhlin, *An Accelerated Kernel-Independent Fast Multipole Method in One Dimension*, SIAM J. Sci. Comput., 29 (2007), pp. 1160–1178.

[81] J. Mason and D. Handscomb, *Chebyshev polynomials*, Chapman & Hall/CRC, 2003.

[82] V. Minden, K. L. Ho, A. Damle, and L. Ying, *A recursive skeletonization factorization based on strong admissibility*, Multiscale Model. Simul., 15 (2017), pp. 768–796.

[83] L. Miranian and M. Gu, *Strong rank-revealing LU factorizations*, Linear Algebra Appl., 367 (2003), pp. 1–16.

[84] R. Movassagh and L. P. Kadanoff, *Eigenpairs of Toeplitz and disordered Toeplitz matrices with a Fisher-Hartwig symbol*, J. Stat Phys., 167 (2017), pp. 959–996.

[85] E. Napoli, E. Polizzi, and Y. Saad, *Efficient estimation of eigenvalue counts in an interval*, Numer. Linear Algebra Appl., 23 (2016), pp. 674–692.

[86] M. O'Neil and V. Rokhlin, *A new class of analysis-based fast transforms*, technical report, 2007.

[87] V. Y. Pan, D. Ivolgin, B. Murphy, R. E. Rosholt, Y. Tang, and X. Yan, *Additive preconditioning for matrix computations*, Linear Algebra Appl., 432 (2010), pp. 1070–1089.

[88] V. Y. Pan and G. Qian, *Randomized preprocessing of homogeneous linear systems of equations*, Linear Algebra Appl., 432 (2010), pp. 3272–3318.

[89] C. S. Peskin, *The immersed boundary method*, Acta Numerica, 11 (2002), pp. 479–517.

[90] E. Polizzi, *A density matrix-based algorithm for solving eigenvalue problems*, Phys. Rev. B, 79 (2009), pp. 115112.

[91] E. Polizzi and A. H. Sameh, *A parallel hybrid banded system solver: the SPIKE algorithm*, Parallel Comput., 32 (2006), pp. 177–194.

[92] Y. Saad, *GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[93] Y. Saad, *Numerical methods for large eigenvalue problems*, Second Edition, SIAM, 2011.

[94] T. Sakurai, Y. Futamura, and H. Tadano, *Efficient parameter estimation and implementation of a contour integral-based eigensolver*, J. Algorithms Comput. Technol., 7 (2013), pp. 249–270.

[95] T. Sakurai and H. Sugiura, *A projection method for generalized eigenvalue problems using numerical integration*, J. Comput. Appl. Math., 159 (2003), pp. 119–128.

[96] T. Sakurai and H. Tadano, *CIRR: a Rayleigh-Ritz type method with contour integral for generalized eigenvalue problems*, Hokkaido Math. J., 36 (2007), pp. 745–757.

[97] T. Shi, M. Belkin, and B. Yu, *Data spectroscopy: Eigenspaces of convolution operators and clustering*, Ann. Statist., 37 (2009), pp. 3960–3984.

[98] E. M. Stein and R. Shakarchi, *Complex analysis. Princeton Lectures in Analysis, II*, Princeton University Press, NJ, 2003.

[99] X. Sun and N. P. Pitsianis, *A matrix version of the fast multipole method*, SIAM Rev., 43 (2001), pp. 289–300.

[100] P. Tang, J. Kestyn, and E. Polizzi, *A new highly parallel non-hermitian eigensolver*, in Proceedings of the High Performance Computing Symposium, HPC '14, San Diego, CA, USA, 2014, Society for Computer Simulation International, pp. 1:1–1:9.

[101] P. Tang and E. Polizzi, *FEAST as a subspace iteration eigensolver accelerated be approximate spectral projection*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 354–390.

[102] J. D. Tebbens and M. Tůma, *Preconditioner updates for solving sequences of linear systems in matrix-free environment*, Numer. Linear Algebra Appl., 17 (2010), pp. 997–1019.

[103] L. N. Trefethen and J. A. C. Weideman, *The exponentially convergent trapezoidal rule*, SIAM Rev., 56 (2014), pp. 385–458.

[104] M. Van Barel, *Designing rational filter functions for solving eigenvalue problems by contour integration*, Linear Algebra Appl., 502 (2016), pp. 346–365.

[105] M. Van Barel, R. Vandebril, P. Van Dooren, and K. Frederix, *Implicit double shift QR-algorithm for companion matrices*, Numer. Math., 116 (2010), pp. 177–212.

[106] E. Vecharynski and J. Langou, *Any admissible cycle-convergence behavior is possible for restarted GMRES at its initial cycles*, Num. Lin. Algebr. Appl., 18 (2011), pp. 499–511.

[107] J. Vogel, J. Xia, S. Cauley, and V. Balakrishnan, *Superfast divide-and-conquer method and perturbation analysis for structured eigenvalue solutions*, SIAM J. Sci. Comput., 38 (2016), pp. A1358–A1382.

[108] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert, *A fast randomized algorithm for approximation of matrices*, Appl. Comput. Harmon. Anal., 25 (2008), pp. 335–366.

[109] G. Wu, W. Xu, Y. Zhang, and Y. Wei, *A preconditioned conjugate gradient algorithm for GeneRank with application to microarray data mining*, Data Min. Knowl. Disc., 26 (2013), pp. 27–56.

[110] Y. Xi and Y. Saad, *Computing partial spectra with least-squares rational filters*, SIAM J. Sci. Comput., to appear, (2016).

[111] Y. Xi and J. Xia, *On the stability of some hierarchical rank structured matrix algorithms*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 1279–1303.

[112] Y. Xi, J. Xia, S. Cauley, and V. Balakrishnan, *Superfast and stable structured solvers for Toeplitz least squares via randomized sampling*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 44–72.

[113] Y. Xi, J. Xia, and R. H. Chan, *A fast randomized eigensolver with structured LDL factorization update*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 974–996.

[114] J. Xia, *On the complexity of some hierarchical structured matrix algorithms*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 388–410.

[115] J. Xia, *Efficient structured multifrontal factorization for general large sparse matrices*, SIAM J. Sci. Comput., 35 (2013), pp. A832–A860.

[116] J. Xia, *Randomized sparse direct solvers*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 197–227.

[117] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, *Fast algorithms for hierarchically semiseparable matrices*, Numer. Linear Algebra Appl., 17 (2010), pp. 953–976.

[118] J. Xia and M. Gu, *Robust approximate Cholesky factorization of rank-structured symmetric positive definite matrices*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2899–2920.

[119] J. Xia, Z. Li, and X. Ye, *Effective matrix-free preconditioning for the augmented immersed interface method*, J. Comput. Phys., 303 (2015), pp. 295–312.

[120] J. Xia, Y. Xi, and M. Gu, *A superfast structured solver for Toeplitz linear systems via randomized sampling*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 837–858.

[121] J. Xiao, M. Gu, and J. Langou, *Fast Parallel Randomized QR with Column Pivoting Algorithms for Reliable Low-Rank Matrix Approximations*, 2017 IEEE 24th International Conference on High Performance Computing (HiPC), 2017, pp. 233–242.

[122] X. Xing and E. Chow, *An efficient method for block low-rank approximations for kernel matrix systems*, preprint, 2018.

[123] X. Ye, J. Xia, Raymond H. Chan, S. Cauley, and V. Balakrishnan, *A fast contour-integral eigensolver for non-Hermitian matrices*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 1268–1297.

[124] X. Ye, J. Xia, and L. Ying, *Analytical compression via proxy point selection and contour integration*, to be submitted, 2018.

[125] G. Yin, R. H. Chan, and M. Yeung, *A FEAST algorithm with oblique projection for generalized eigenvalue problems*, 2015, arXiv:1404.1768v4 [math.NA].

[126] L. Ying, *A kernel independent fast multipole algorithm for radial basis functions*, J. Comput. Phys., 213 (2006), pp. 451–457.

[127] L. Ying, G. Biros, and D. Zorin, *A kernel-independent adaptive fast multipole algorithm in two and three dimensions*, J. Comput. Phys., 196 (2004), pp. 591–626.

[128] W.-J. Ying and C. S. Henriquez, *A kernel-free boundary integral method for elliptic boundary value problems*, J. Comput. Phy., 227 (2007), pp. 1046–1074.

APPENDIX

A. COPYRIGHT INFORMATION

For the content used in chapter 3, attached in the next page is a photocopy of the permission letter from SIAM for using the paper [123] in this dissertation. Regarding the paper [119] used in chapter 4, the right for Personal Use is granted by the Journal Publishing Agreement of Elsevier as shown in fig. A.1, a complete version of which can be found at `https://www.elsevier.com/__data/assets/pdf_file/0004/727600/JPA_V22_2017_Clean-copy.pdf`.

Author Rights for Scholarly Purposes (see 'Definitions' clause below)
I understand that I retain or am hereby granted (without the need to obtain further permission) the Author Rights (see description below and definitions), and that no rights in patents, trademarks or other intellectual property rights are transferred to the Copyright Owner.

The Author Rights include the right to use the Preprint, Accepted Manuscript and the Published Journal Article for Personal Use, Internal Institutional Use and for Scholarly Sharing.

In the case of the Accepted Manuscript and the Published Journal Article the Author Rights exclude Commercial Use (unless expressly agreed in writing by the Copyright Owner), other than use by the author in a subsequent compilation of the author's works or to extend the Article to book length form or re-use by the author of portions or excerpts in other works (with full acknowledgment of the original publication of the Article).

(a) Author Rights

Personal Use
Use by an author in the author's classroom teaching (including distribution of copies, paper or electronic) or presentation by an author at a meeting or conference (including distribution of copies to the delegates attending such meeting), distribution of copies (including through e-mail) to known research colleagues for their personal use, use in a subsequent compilation of the author's works, inclusion in a thesis or dissertation, preparation of other derivative works such as extending the Article to book-length form, or otherwise using or re-using portions or excerpts in other works (with full acknowledgment of the original publication of the Article).

(b) Definition of Personal Use

Figure A.1. Selected parts from the Journal Publishing Agreement

siam®
SOCIETY for INDUSTRIAL and APPLIED MATHEMATICS

3600 Market Street, 6th Floor
Philadelphia, PA 19104-2688 USA
Phone +1-215-382-9800
Fax +1-215-386-7999
www.siam.org • siam@siam.org

June 5, 2018

This is to certify that Xin Ye has permission to reuse the following article in his Ph.D. thesis:

X. Ye, J. Xia, R. H. Chan, S. Cauley, and V. Balakrishnan, "A fast contour-integral eigensolver for non-Hermitian matrices," SIAM J. Matrix Anal. Appl., 38 (2017), pp. 1268-1297.

Kelly Thomas

Kelly Thomas

Managing Editor, SIAM

VITA

VITA

Xin Ye received his dual bachelor's degrees in Economics and Applied Mathematics from Wuhan University, Wuhan, China in 2013, then he started his Ph.D. study in Applied Mathematics at Purdue University, West Lafayette, IN, USA, since the fall of 2014 he has been working with Prof. Jianlin Xia on various research projects in numerical analysis and numerical linear algebra.