Deterministic Complexity Analysis of Hermitian Eigenproblems

Aleksandros Sobczyk[1]
IBM Research Europe, Rüschlikon, Switzerland
ETH Zurich, Zurich, Switzerland

Abstract

In this work we revisit the arithmetic and bit complexity of Hermitian eigenproblems. Recently, [BGVKS, FOCS 2020] proved that a (non-Hermitian) matrix $\mathbf{A}$ can be diagonalized with a randomized algorithm in $O(n^\omega \log^2(\frac{n}{\epsilon}))$ arithmetic operations, where $\omega \lesssim 2.371$ is the square matrix multiplication exponent, and [Shah, SODA 2025] significantly improved the bit complexity for the Hermitian case. Our main goal is to obtain similar deterministic complexity bounds for various Hermitian eigenproblems. In the Real RAM model, we show that a Hermitian matrix can be diagonalized deterministically in $O(n^\omega \log(n) + n^2 \text{polylog}(\frac{n}{\epsilon}))$ arithmetic operations, improving the classic deterministic $\widetilde{O}(n^3)$ algorithms, and derandomizing the aforementioned state-of-the-art. The main technical step is a complete, detailed analysis of a well-known divide-and-conquer tridiagonal eigensolver of Gu and Eisenstat [GE95], when accelerated with the Fast Multipole Method, asserting that it can accurately diagonalize a symmetric tridiagonal matrix in nearly-$O(n^2)$ operations. In finite precision, we show that an algorithm by Schönhage [Sch72] to reduce a Hermitian matrix to tridiagonal form is stable in the floating point model, using $O(\log(\frac{n}{\epsilon}))$ bits of precision. This leads to a deterministic algorithm to compute all the eigenvalues of a Hermitian matrix in $O\left(n^\omega \mathcal{F}\left(\log(\frac{n}{\epsilon})\right) + n^2 \text{polylog}(\frac{n}{\epsilon})\right)$ bit operations, where $\mathcal{F}(b) \in \widetilde{O}(b)$ is the bit complexity of a single floating point operation on b bits. This improves the best known $\widetilde{O}(n^3)$ deterministic and $O\left(n^\omega \log^2(\frac{n}{\epsilon})\mathcal{F}\left(\log(\frac{n}{\epsilon})\right)\right)$ randomized complexities. We conclude with some other useful subroutines such as computing spectral gaps, condition numbers, and spectral projectors, and with some open problems.

2012 ACM Subject Classification Theory of computation → Divide and conquer; Theory of computation → Numeric approximation algorithms

Keywords and phrases Hermitian eigenproblem, eigenvalues, SVD, tridiagonal reduction, matrix multiplication time, diagonalization, bit complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2025.131

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2410.21550 [79]

Acknowledgements I am grateful to Efstratios Gallopoulos, Daniel Kressner, and David Woodruff for helpful discussions.

1 Introduction

Eigenproblems arise naturally in many applications. Given a matrix $\mathbf{A}$, the goal is to compute (a subset of) the eigenvalues λ and/or the eigenvectors $\mathbf{v}$, which satisfy

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}.$$

The properties of the given matrix, as well as the quantities that need to be computed can vary depending on the application, giving rise to different variants of the eigenproblem.

[1] Work performed while at IBM Research and ETH – Currently at Huawei Zurich Research Center

These include (*i*) *eigenvalue problems*, such as the approximation of eigenvalues, singular values, spectral gaps, and condition numbers, (*ii*) *eigenspace problems*, which refer to the approximation of eigenvectors, spectral projectors, and invariant subspaces, and (*iii*) *diagonalization problems*, which involve the (approximate) computation of all the eigenvalues and eigenvectors of a matrix (or pencil), i.e., a full spectral factorization and/or the Singular Value Decomposition (SVD).

In this work we focus on algorithms for *Hermitian eigenproblems*, i.e., the special case where the input matrix is Hermitian. Our motivation to dedicate the analysis to this special class is twofold. First, they arise in many fundamental applications in Machine Learning, Spectral Graph Theory, and Scientific Computing. For example, the SVD, which is ubiquitous for many applications such as low rank approximations [69, 39, 35, 22], directly reduces to a Hermitian eigenproblem. Second, the spectral theorem states that a Hermitian matrix $\mathbf{A}$ can always be written in a factored form $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^*$, where $\mathbf{Q}$ is unitary and $\mathbf{\Lambda}$ is diagonal with real entries. This alleviates several difficulties of the non-Hermitian case, which can lead to efficient dedicated algorithms.

Algorithms for eigenproblems have been studied for decades, some of the earliest being attributed to Jacobi [53, 43]. We refer to standard textbooks for an overview of the rich literature [31, 44, 70, 12, 73]. Some landmark works include the power method [60], the Lanczos algorithm [57], and the paramount QR algorithm [37, 38, 56], which has been recognized as one of the "top ten algorithms of the twentieth century" [33], signifying the importance of the eigenproblem in science and engineering. Given a Hermitian tridiagonal matrix $\mathbf{T}$ with size $n \times n$, the algorithm can compute a set of approximate eigenvalues in $O(n^2 \log(\frac{n}{\epsilon}))$ arithmetic operations, based on the seminal analyses of Wilkinson [84] and Dekker and Traub [25]. A set of approximate eigenvectors can be also returned in $O(n^3 \log(\frac{n}{\epsilon}))$ operations. In conjunction with the classic unitary similarity transformation algorithm of Householder [52], the shifted-QR algorithm has heavylifted the computational burden of solving eigenvalue problems for decades, both in theory and in practice. A detailed bit complexity analysis was provided recently in [8, 7, 9].

Despite the daunting literature, several details regarding the computational complexity of many algorithms remain unclear. It is well-known, for example, that the cubic arithmetic complexity to compute the eigenvalues of a dense matrix is not optimal: a classic work by Pan and Chen [68] showed that the eigenvalues can be approximated in $O(n^\omega)$ arithmetic operations, albeit without detailing how to also compute eigenvectors, or to fully diagonalize a matrix. Here $\omega \geq 2$ is the *matrix multiplication exponent*, i.e. the smallest number such that two $n \times n$ matrices can be multiplied in $O(n^{\omega+\eta})$ arithmetic operations, for any $\eta > 0$. The current best known upper bound is $\omega < 2.371339$ [1], and we will write n^ω instead of $n^{\omega+\eta}$ hereafter for simplicity. Recently, Banks, Garza-Vargas, Kulkarni, and Srivastava [6] described a numerically stable randomized algorithm to compute a provably accurate diagonalization, in $\widetilde{O}(n^\omega)$ operations, improving the previous best-known bounds, specifically, $\widetilde{O}(n^3)$ (Hermitian) and $O(n^{10})$ (non-Hermitian) [3]. [78] further improved the analysis for the Hermitian case, and several works have studied extensions and related applications [28, 29, 74, 80]. To date, we are not aware of any *deterministic* algorithm that achieves the same arithmetic (or bit) complexity with provable approximation guarantees, even for the Hermitian case. In this work, we proceed step-by-step and analyze several variants of the aforementioned eigenvalue, eigenspace, and diagonalization problems, in different *models of computation*, and report complexity upper bounds with provable approximation guarantees.

1.1 Models of computation and complexity

From the Abel-Ruffini theorem it is known that the eigenvalues and/or the eigenvectors of matrices with size larger than $n = 5$ can not be computed exactly. Even in exact arithmetic, they can only be approximated. Before analyzing algorithms, we first need to clarify what are the quantities of interest, to define how accuracy is measured, and what is the underlying model of computation. The main two models that we use to analyze algorithms are the Real RAM and the Floating Point model, described below.

Exact real arithmetic (Real RAM): For the Real RAM model we follow the definition of [36]. The machine has a memory (RAM) and registers that store real numbers in infinite precision. Moving real numbers between the memory and the registers takes constant time. A processor can execute arithmetic operations $\{+, -, \times, /, \sqrt{\cdot}, >\}$ on the real numbers stored in the registers exactly, without any errors, in constant time. Other functions such as $\log(x), \exp(x)$, and trigonometric functions, are not explicitly available, but they can be efficiently approximated up to some additive error, e.g. with a truncated polynomial expansion, using a polylogarithmic number of basic arithmetic operations. Bit-wise operations on real numbers are forbidden, since this can give the machine unreasonable computing power [48, 76, 36].

Floating point: In this model, a real number $\alpha \in \mathbb{R}$ is rounded to a floating point number $\mathbf{fl}(\alpha) = s \times 2^{e-t} \times m$, where $s = \pm 1$, e is the exponent, t is the bias, and m is the significand (or "mantissa"). Floating point operations also introduce rounding errors, i.e., for two floating point numbers α and β, each operation $\odot \in \{+, -, \times, /\}$ satisfies:

$$\mathbf{fl}(\alpha \odot \beta) = (1+\theta)(\alpha \odot \beta), \qquad \text{and also} \qquad \mathbf{fl}(\sqrt{a}) = (1+\theta)\sqrt{a}, \tag{1}$$

where $|\theta| \leq \mathbf{u}$, and $\mathbf{u}$ is the machine precision. Assuming a total of b bits for each number, every floating point operation costs $\mathcal{F}(b)$ bit operations, where typically $\mathcal{F}(b) \in O(b^2)$, or even $\widetilde{O}(b)$, with more advanced algorithms [77, 40, 49]. More details are provided in Appendix D of the full version [79].

In Section 3.4 we will also use as a subroutine an algorithm from [13, 15], which was originally analyzed in Boolean RAM model. We describe in detail how to use it in the corresponding section.

Arithmetic and boolean complexity

Given a model of computation, the *arithmetic complexity* of an algorithm is quantified in terms of the arithmetic operations executed. The *boolean* (or *bit*) *complexity*, accounts for the total number of boolean operations (i.e. boolean gates with maximum fan-in two).

1.2 Notation

Matrices and vectors are denoted with bold capital and small letters, respectively. $\|\mathbf{A}\|$ is the spectral norm of $\mathbf{A}$, $\|\mathbf{A}\|_F$ its Frobenius norm, $\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^\dagger\|$ is the condition number, and $\Lambda(\mathbf{A})$ its spectrum. For the complexity analysis we denote the geometric series $S_x(m) = \sum_{l=1}^{m} (2^{x-2})^l$. As already mentioned, for simplicity, the complexity of multiplying two $n \times n$ matrices will be denoted as $O(n^\omega)$, instead of $O(n^{\omega+\eta})$, for arbitrarily small $\eta > 0$. We also use the standard notation $\omega(a, b, c)$ for the complexity of multiplying two rectangular matrices with sizes $n^a \times n^b$ and $n^b \times n^c$ in time $O(n^{\omega(a,b,c)})$, and therefore $\omega := \omega(1, 1, 1)$. For example, for $a = c = 1$ and $b = 2$, the best known bound for $\omega(1, 2, 1) \approx 3.250035$ [1], which is slightly better than naively performing n square multiplications in $O(n \cdot n^\omega) \lesssim O(n^{3.371339})$. $\mathcal{F}(b)$ denotes the bit complexity of a single floating point operation on b bits.

1.3 Methods and Contributions

1.3.1 Real RAM

We start with the analysis in exact arithmetic, which is the simplest model to analyze numerical algorithms. The goal is to obtain end-to-end upper bounds for the arithmetic complexity of approximately diagonalizing symmetric tridiagonal matrices and, ultimately, dense Hermitian matrices, as well as to approximate the SVD. To measure the accuracy, we follow the notion of *backward-stability* (or *backward-approximation*) for Hermitian diagonalization from [63]. Formally, the following problems are considered.

▶ **Problem 1.1.** *Backward-approximate diagonalization problems in exact arithmetic.*

(i) ***Symmetric arrowhead/tridiagonal diagonalization:*** *Given a symmetric arrowhead or tridiagonal matrix* $\mathbf{A}$ *with size* $n \times n$, $\|\mathbf{A}\| \leq 1$, *and accuracy* $\epsilon \in (0,1)$, *compute a diagonal matrix* $\widetilde{\mathbf{\Lambda}}$ *and a matrix* $\widetilde{\mathbf{U}}$, *such that* $\widetilde{\mathbf{U}} = \mathbf{U} + \mathbf{E}_{\mathbf{U}}$, *where* $\mathbf{U}$ *is orthogonal and* $\|\mathbf{E}_{\mathbf{U}}\| \leq \epsilon$, *and* $\left\|\mathbf{A} - \mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\top}\right\| \leq \epsilon$.

(ii) ***Hermitian diagonalization:*** *Given a Hermitian matrix* $\mathbf{A}$ *with size* $n \times n$, $\|\mathbf{A}\| \leq 1$, *and accuracy* $\epsilon \in (0,1)$, *compute a diagonal matrix* $\widetilde{\mathbf{\Lambda}}$ *and a matrix* $\widetilde{\mathbf{U}}$, *such that* $\widetilde{\mathbf{U}} = \mathbf{U} + \mathbf{E}_{\mathbf{U}}$, *where* $\mathbf{U}$ *is unitary and* $\|\mathbf{E}_{\mathbf{U}}\| \leq \epsilon$, *and* $\left\|\mathbf{A} - \mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{*}\right\| \leq \epsilon$.

(iii) ***SVD:*** *Given a matrix* $\mathbf{A} \in \mathbb{C}^{m \times n}$, $m = n^k$, $k \geq 1$, $\|\mathbf{A}\| \leq 1$, *and accuracy* $\epsilon \in (0,1)$ *compute a diagonal matrix* $\widetilde{\mathbf{\Sigma}}$, *an* $m \times n$ *matrix* $\widetilde{\mathbf{U}} = \mathbf{U} + \mathbf{E}_{\mathbf{U}}$, *and an* $n \times n$ *matrix* $\widetilde{\mathbf{V}} = \mathbf{V} + \mathbf{E}_{\mathbf{V}}$, *such that* $\mathbf{U}^{*}\mathbf{U} = \mathbf{V}^{*}\mathbf{V} = \mathbf{I}$, $\left\|\mathbf{E}_{\{\mathbf{U},\mathbf{V}\}}\right\| \leq \epsilon$, *and* $\left\|\mathbf{A} - \mathbf{U}\widetilde{\mathbf{\Sigma}}\mathbf{V}^{*}\right\| \leq \epsilon$.

To obtain rigorous complexity guarantees for these problems, with respect to all the involved parameters, we start from Problem 1.1-(i), namely, diagonalization of symmetric tridiagonal matrices. To that end, we revisit the so-called fast tridiagonal eigensolvers, which aim to reduce the complexity from cubic to $\widetilde{O}(n^2)$ operations. Many such algorithms have been studied in the literature [34, 32, 14, 13, 41, 83, 66, 81, 10], most of which are based on the divide-and-conquer (DC) strategy of Cuppen [24]. The algorithms in all of the aforementioned works have been rigorously analyzed, however, explicit complexity bounds in terms of strictly solving Problem 1.1-(i) are not detailed. We resolve this by providing an end-to-end complexity analysis of the algorithm of Gu and Eisenstat [46]. In their original work, the authors outlined how to accelerate several parts of the algorithm with the Fast Multipole Method (FMM) [71], which could eventually lead to a final complexity of $\widetilde{O}(n^2)$. However, the actual analysis of this approach and the FMM details were not provided. [61] further extended the analysis in floating point, but it also relies on a numerically stable FMM implementation, which is not detailed. In this work, we use the elegant FMM analysis of [45, 58, 21], which is particularly suited for the problems considered. It is summarized in the following Proposition 1.1.

▶ **Proposition 1.1** (FMM)**.** *There exists an algorithm, which we refer to as* (ϵ, n)*-approximate FMM (or* (ϵ, n)*-FMM, for short), which takes as input*

- *a kernel function* $k(x) \in \left\{\log(|x|), \frac{1}{x}, \frac{1}{x^2}\right\}$,
- $2n + m$ *real numbers:* $\{x_1, \ldots, x_m\} \cup \{c_1, \ldots, c_n\} \cup \{y_1, \ldots, y_n\}$, *and a constant* C, *such that* $m \leq n$ *and for all* $i \in [m], j \in [n]$ *it holds that*

$$|x_i|, |c_j|, |y_j| < C \qquad \text{and} \qquad |x_i - y_j| \geq \Omega(\text{poly}(\tfrac{\epsilon}{n})).$$

It returns m *values* $\widetilde{f}(x_1), \ldots, \widetilde{f}(x_m)$ *such that* $\left|\widetilde{f}(x_i) - f(x_i)\right| \leq \epsilon$, *for all* $i \in [m]$, *where* $f(x) = \sum_{j=1}^{n} c_j k(x_i - y_j)$, *in a total of* $O\left(n \log^{\xi}(\frac{n}{\epsilon})\right)$ *arithmetic operations, where* $\xi \geq 1$ *is a small constant that is independent of* ϵ, n.

Proof. The result follows from the analysis of [45, 58, 21]. A detailed description and a short proof is provided for completeness in the full version [79], specifically in Appendix B, Proposition B.1. ◀

By taking advantage of the FMM, the analysis from Section 2 leads to the following Theorem 1.2, whose proof can be found in Section 2.2.1.

▶ **Theorem 1.2.** *Given an unreduced symmetric tridiagonal matrix* $\mathbf{T}$ *with size* $n \times n$, $\|\mathbf{T}\| \leq 1$, *an accuracy* $\epsilon \in (0, 1/2)$, *and an* (ϵ, n)*-FMM implementation as in Proposition 1.1, the recursive algorithm of [46] (Algorithm 1), returns an approximately orthogonal matrix* $\widetilde{\mathbf{U}}$ *and a diagonal matrix* $\widetilde{\mathbf{\Lambda}}$ *such that*

$$\left\|\mathbf{T} - \widetilde{\mathbf{U}}\widetilde{\mathbf{\Lambda}}\widetilde{\mathbf{U}}^{\top}\right\| \leq \epsilon, \quad \left\|\widetilde{\mathbf{U}}^{\top}\widetilde{\mathbf{U}} - \mathbf{I}\right\| \leq \epsilon/n^2,$$

or, stated alternatively,

$$\widetilde{\mathbf{U}} = \mathbf{U} + \mathbf{E}_{\mathbf{U}}, \quad \mathbf{U}^{\top}\mathbf{U} = \mathbf{I}, \quad \|\mathbf{E}_{\mathbf{U}}\| \leq \epsilon/n^2, \quad \left\|\mathbf{T} - \mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^{\top}\right\| \leq \epsilon,$$

using a total of $O\left(n^2 \log^{\xi+1}(\frac{n}{\epsilon})\right)$ *arithmetic operations and comparisons, where* $\xi \geq 1$ *is a small constant that depends on the specific FMM implementation and it is independent of* ϵ, n.

Table 1 Comparison of algorithms for the Problems 1.1 (i)-(iii) in the Real RAM model. Here $\xi \geq 1$ is a small constant which depends on the FMM implementation (see Prop. 1.1). The algorithms marked with (**R**) are randomized and succeed with high probability (at least $1 - 1/\operatorname{poly}(n)$).

	Arithmetic Complexity	Comment
Prob. 1.1-(i) Arrowhead/Trid. diagonalization		
Refs. [24, 65, 46]	$O(n^3) + \widetilde{O}(n^2)$	Conjectured $\widetilde{O}(n^2)$
Theorem 1.2	$O\left(n^2 \log^{\xi+1}(\frac{n}{\epsilon})\right)$	Req. FMM (Prop. 1.1)
Prob. 1.1-(ii) Hermitian diagonalization		
Refs. [6, 78] (**R**)	$O\left(n^{\omega} \log^2(\frac{n}{\epsilon})\right)$	-
Ref. [11] (**R**)	$\widetilde{O}\left(n^{\omega+1}\right)$	Conjectured $\widetilde{O}(n^{\omega})$
Refs. [25, 84, 51]	$O\left(n^3 \log(\frac{n}{\epsilon})\right)$	Shifted-QR
Ref. [63]	$\widetilde{O}\left(n^3\right)$	Req. separated spectrum
Ref. [68]	$O\left(n^{\omega} + n \operatorname{polylog}(\frac{n}{\epsilon})\right)$	Only eigenvalues
Corollary 2.4	$O\left(n^{\omega} \log(n) + n^2 \log^{\xi+1}(\frac{n}{\epsilon})\right)$	Req. FMM (Prop. 1.1)
Prob. 1.1-(iii), SVD		
Shifted-QR on $\mathbf{A}^*\mathbf{A}$	$O\left(n^{\omega(1,k,1)} + n^3 \log(\frac{n}{\epsilon})\right)$	Partial error analysis
Refs. [6, 78, 55] (**R**)	$O\left(n^{\omega(1,k,1)} + n^{\omega} \log^2(\frac{n}{\epsilon})\right)$	Partial error analysis
Theorem 1.3	$O\left(n^{\omega(1,k,1)} + n^{\omega} \log(n) + n^2 \operatorname{polylog}(\frac{n\kappa(\mathbf{A})}{\epsilon})\right)$	Req. FMM (Prop. 1.1)

This result has a direct application to the dense Hermitian diagonalization problem. The best-known complexities of deterministic algorithms, with end-to-end analysis, are at least cubic. For example, the aforementioned shifted-QR algorithm requires $O(n^3 \log(n/\epsilon))$ arithmetic operations, even for tridiagonal matrices. The cubic barrier originates from the accumulation of the elementary rotation matrices to form the final eigenvector matrix. The

so-called *spectral divide-and-conquer* methods (see e.g. [31, 63]) have also at least cubic complexities in the deterministic case. The two main difficulties in their analysis are the basis computation of spectral projectors, for which the best-known deterministic complexity bounds are $\Omega(n^3)$, e.g. via Strong Rank-Revealing QR (RRQR) factorization [47], and the choice of suitable splitting points, which relies on the existence of spectral gaps.

Randomness can help to overcome certain difficulties in the analysis. [26] analyzed a numerically stable Randomized URV decomposition, which can be used to replace RRQR for basis computations in spectral DC algorithms. A highly parallelizable Hermitian diagonalization algorithm with end-to-end analysis was proposed in [11]. While the reported sequential arithmetic complexity is $O(n^{\omega+1})$, the authors conjectured that it can be further reduced to $\widetilde{O}(n^\omega)$. The first end-to-end $\widetilde{O}(n^\omega)$ complexity upper bound was achieved in [6]. One of the main techniques was to use random perturbations to ensure that the pseudospectrum is well-separated, which helps to find splitting points in spectral DC. [78] further improved the analysis for Hermitian matrices.

Theorem 1.2 can be directly used to obtain a simple and deterministic solution for the Hermitian diagonalization problem. Specifically, Corollary 2.4 states that the problem can be solved in $O(n^\omega \log(n) + n^2 \operatorname{polylog}(n/\epsilon))$ arithmetic operations, which is both faster and fully deterministic. This is achieved by combining Theorem 1.2 with the (rather overlooked) algorithm of Schönhage [75], who proved that a Hermitian matrix can be reduced to tridiagonal form with unitary similarity transformations in $O(n^\omega \log(n) c(\omega))$ arithmetic operations, where $c(\omega) = O(1)$ if ω is treated as a fixed constant larger than 2, while $c(\omega) = O(\log(n))$ if it turns out that $\omega = 2$.

As a consequence, Theorem 1.3 reports similar deterministic complexity results for the SVD. The SVD is a fundamental computational kernel in many applications, such as Principal Component Analysis [54], and it is also widely used as a subroutine for many other advanced algorithms, e.g. [69, 39, 35, 18, 20, 23, 19, 22], to name a few. A straightforward algorithm to compute it is to first form the Gramian matrix $\mathbf{A}^*\mathbf{A}$, and then diagonalize it. Other classic SVD algorithms, such as the widely adopted Golub-Kahan bidiagonalization and its variants [42], or polar decomposition-based methods [63, 62], avoid the computation of $\mathbf{A}^*\mathbf{A}$ for numerical stability reasons, but they also rely on a diagonalization algorithm as a subroutine. [55] elaborated on a matrix multiplication time SVD algorithm, by using [6] to diagonalize $\mathbf{A}^\top\mathbf{A}$, albeit without fully completing the backward stability analysis. The following Theorem 1.3 states our main result. The proof can be found in the full version [79], in Appendix C.4.

▶ **Theorem 1.3** (SVD). *Let $\mathbf{A} \in \mathbb{C}^{m\times n}$, $m = n^k$, $k \geq 1$. Assume that $1/n^c \leq \|\mathbf{A}\| \leq 1$, for some constant c. Given accuracy $\epsilon \in (0, 1/2)$ and an (ϵ, n)-FMM (see Prop. 1.1), we can compute three matrices $\widetilde{\mathbf{U}} \in \mathbb{C}^{m\times n}, \widetilde{\mathbf{\Sigma}} \in \mathbb{R}^{n\times n}, \widetilde{\mathbf{V}} \in \mathbb{C}^{n\times n}$ such that $\widetilde{\mathbf{\Sigma}}$ is diagonal with positive diagonal elements and*

$$\mathbf{A} = \widetilde{\mathbf{U}}\widetilde{\mathbf{\Sigma}}\widetilde{\mathbf{V}}^*, \quad \left\|\widetilde{\mathbf{U}}^*\widetilde{\mathbf{U}} - \mathbf{I}\right\| \leq \epsilon, \quad \left\|\widetilde{\mathbf{V}}^*\widetilde{\mathbf{V}} - \mathbf{I}\right\| \leq \epsilon/(\kappa(\mathbf{A})n^2) \ll \epsilon,$$

or, stated alternatively,

$$\left\|\mathbf{A} - \mathbf{U}\widetilde{\mathbf{\Sigma}}\mathbf{V}^*\right\| \leq \epsilon, \quad \widetilde{\mathbf{U}} = \mathbf{U} + \mathbf{E}_\mathbf{U}, \quad \|\mathbf{E}_\mathbf{U}\| \leq \epsilon, \quad \mathbf{U}^*\mathbf{U} = \mathbf{I},$$
$$\widetilde{\mathbf{V}} = \mathbf{V} + \mathbf{E}_\mathbf{V}, \quad \|\mathbf{E}_\mathbf{V}\| \leq \epsilon/n^2, \quad \mathbf{V}^*\mathbf{V} = \mathbf{I}.$$

The algorithm requires a total of at most $O\left(n^{\omega(1,k,1)} + n^\omega \log(n) + n^2 \operatorname{polylog}(\frac{n\kappa(\mathbf{A})}{\epsilon})\right)$ arithmetic operations, where $\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^\dagger\|$.

To summarize this section, in Table 2 we list all the deterministic arithmetic complexities achieved in the Real RAM model, for all the aforementioned problems, and compare with some important existing algorithms.

1.3.2 Finite precision

Similar deterministic complexity upper bounds are obtained for several problems in finite precision. In particular, we study the following problems, for which we seek to bound the boolean complexity, i.e., the total number of bit operations.

▶ **Problem 1.2.** *Main problems in finite precision.*

(i) ***Tridiagonal reduction:*** *Given a Hermitian matrix* $\mathbf{A}$ *with floating point elements, reduce* $\mathbf{A}$ *to tridiagonal form using (approximate) unitary similarity transformations. In particular, return a tridiagonal matrix* $\widetilde{\mathbf{T}}$*, and (optionally) an approximately unitary matrix* $\widetilde{\mathbf{Q}}$*, such that*

$$\left\|\widetilde{\mathbf{Q}}\widetilde{\mathbf{Q}}^* - \mathbf{I}\right\| \leq \epsilon, \quad \textit{and} \quad \left\|\mathbf{A} - \widetilde{\mathbf{Q}}\widetilde{\mathbf{T}}\widetilde{\mathbf{Q}}^*\right\| \leq \epsilon \left\|\mathbf{A}\right\|,$$

(ii) ***Hermitian eigenvalues:*** *Given a Hermitian matrix* $\mathbf{A}$*,* $\|\mathbf{A}\| \leq 1$*, and accuracy* $\epsilon \in (0,1)$*, compute a set of approximate eigenvalues* $\widetilde{\lambda}_i$ *such that* $\left|\widetilde{\lambda}_i - \lambda_i(\mathbf{A})\right| \leq \epsilon$*.*

Table 2 Boolean complexity for Problems 1.2, for matrix size $n \times n$ and accuracy $\epsilon \in (0,1)$.

	Boolean Complexity	Comment
Prob. 1.2-(i) Tridiag. Reduction		
Refs. [52, 50]	$O\left(n^3 \mathcal{F}\left(\log\left(\frac{n}{\epsilon}\right)\right)\right)$	Standard Householder reduction
Theorem 3.2	$O\left(n^\omega \log(n) \mathcal{F}\left(\log(\frac{n}{\epsilon})\right)\right)$	[75] with stable fast QR [26]
Prob. 1.2-(ii) Herm. Eigenvalues		
Ref. [78]	$O\left(n^\omega \log^2(\frac{n}{\epsilon}) \mathcal{F}\left(\log(\frac{n}{\epsilon})\right)\right)$	Randomized, $\Pr[\text{success}] \geq 1 - \frac{1}{n}$
Theorem 3.4	$O\left(n^\omega \mathcal{F}\left(\log(\frac{n}{\epsilon})\right) + n^2 \operatorname{polylog}(\frac{n}{\epsilon})\right)$	Deterministic, Thm. 3.2 + [15]

Regarding deterministic algorithms, with end-to-end-analysis, the standard approach is to first reduce the Hermitian matrix to tridiagonal form with Householder transformations [52], which can be done stably in $O(n^3)$ arithmetic operations using $O(\log(n/\epsilon))$ bits of precision; see e.g. [50]. Thereafter, there is a plethora of algorithms (e.g. the ones mentioned in the previous section) for the eigenvalues of the tridiagonal matrix, with varying complexities and stability properties. However, the total boolean complexity cannot be lower than $\Omega(n^3 \mathcal{F}(\log(\frac{n}{\epsilon})))$ due to the Householder reduction step. Other well-known deterministic and numerically stable algorithms in the literature also require at least n^3 arithmetic operations to compute all the eigenvalues [67, 30, 63], and at least $\operatorname{polylog}(n, 1/\epsilon)$ bits of precision in a floating point machine. The arithmetic complexity of the algorithm of [68] scales as $O(n^\omega)$ with respect to the matrix size n, but the boolean complexity can increase up to $O(n^{\omega+1})$ in rational arithmetic. [59] described a randomized algorithm to compute only the largest eigenvalue in nearly $O(n^\omega)$ bit complexity. The fastest algorithm to compute all the eigenvalues of a Hermitian matrix is [78], which requires $O(n^\omega \operatorname{polylog}(n/\epsilon))$ boolean operations and succeeds with high probability.

Randomized eigenvalue algorithms have also been studied in the sketching/streaming setting [2, 64, 82]. The (optimal) algorithm of [82] has not been analyzed in finite-precision, but, due to its simplicity, it should be straightforward to achieve. The algorithm approximates

all the eigenvalues of a Hermitian matrix $\mathbf{A}$ up to additive error $\epsilon\|\mathbf{A}\|_F$. However, to reduce the error to a spectral-norm bound $\epsilon\|\mathbf{A}\|$, the algorithm internally needs to diagonalize a matrix with size $\Omega(n)$, and therefore it does not provide any improvement against any other Hermitian eigenvalue solver. Nevertheless, our main results can be directly applied as the main eigenvalue subroutine of this algorithm, and to help analyze its bit complexity.

To improve the aforementioned Hermitian eigenvalue algorithms, we first prove in Theorem 3.2 it is proved that Schönhage's algorithm is numerically stable in the floating point model of computation. Thereafter, we carefully combine it with the algorithms of [13, 14, 15] which provably and deterministically approximate all the eigenvalues of a symmetric tridiagonal matrix in $\widetilde{O}(n^2)$ boolean operations. The latter algorithm is analyzed in the Boolean RAM model, therefore, in order to use it we need to convert the floating point elements of the tridiagonal matrix that is returned by Theorem 3.2 to integers, which can be done efficiently under reasonable assumptions on the initial number of bits used to represent the floating point numbers. This is described in detail in the proof of our main Theorem 3.4. Our result derandomizes and slightly improves the final bit complexity of the algorithm of [78], which has the currently best known bit complexity for this problem. Table 2 summarizes this discussion.

As a direct consequence of Theorem 3.4, in Section 4 of the full version [79] we also provide the analysis for several other useful subroutines related to eigenvalue/eigenvector computations, including:

- **(i) Singular values and condition number**: In [79, Proposition 4.1] we describe how to obtain relative error approximations of singular values. In [79, Corollary 4.1] we show how to compute the condition number.
- **(ii) Definite pencil eigenvalues**: In [79, Corollary 4.2] we demonstrate how to extend Theorem 3.4 to compute the eigenvalues of Hermitian-definite pencils.
- **(iii) Spectral gaps**: In [79, Corollary 4.4] we show how to compute the spectral gap and the midpoint between any two eigenvalues of a Hermitian-definite pencil. Our algorithm is deterministic and it requires significantly less bits of precision than the algorithm of [80], who described a randomized algorithm for this problem that is slightly faster than applying [6] as a black-box, but it only computes a single spectral gap.
- **(iv) Spectral projector**: [79, Corollary 4.5] details how to compute spectral projectors on invariant subspaces of Hermitian-definite pencils, which are important for many applications.

1.4 Outline

The paper is organized as follows. In Section 2 we analyze the algorithm of [46] when implemented with the FMM, and its application to Hermitian diagonalization and the SVD. In Section 3 it is proved that Schönhage's algorithm is numerically stable in floating point, and it is used as a preprocessing step to compute the eigenvalues of Hermitian matrices. We finally conclude and state some open problems in Section 4.

2 Diagonalization of symmetric tridiagonal and arrowhead matrices in Real RAM

Our main target is to compute the eigenvalues and the eigenvectors of tridiagonal symmetric matrices in nearly linear time. To derive the desired result, we provide an analysis the divide-and-conquer algorithm of Gu and Eisenstat [46], when implemented with the FMM from Proposition 1.1.

The algorithm of [46] first "divides" the problem by partitioning the (unreduced) tridiagonal matrix $\mathbf{T}$ as follows:

$$\mathbf{T} = \begin{pmatrix} \mathbf{T}_1 & \beta_{k+1}\mathbf{e}_k & \\ \beta_{k+1}\mathbf{e}_k^\top & \alpha_{k+1} & \beta_{k+2}\mathbf{e}_1^\top \\ & \beta_{k+2}\mathbf{e}_1 & \mathbf{T}_2 \end{pmatrix}.$$

If one has access to the spectral decomposition of $\mathbf{T}_1$ and $\mathbf{T}_2$, i.e. $\mathbf{T}_1 = \mathbf{Q}_1\mathbf{D}_1\mathbf{Q}_1^\top$ and $\mathbf{T}_2 = \mathbf{Q}_2\mathbf{D}_2\mathbf{Q}_2^\top$, then $\mathbf{T}$ can be factorized as

$$\begin{pmatrix} & \mathbf{Q}_1 & \\ 1 & & \\ & & \mathbf{Q}_2 \end{pmatrix} \begin{pmatrix} \alpha_{k+1} & \beta_{k+1}\mathbf{l}_1^\top & \beta_{k+2}\mathbf{f}_2^\top \\ \beta_{k+1}\mathbf{l}_1 & \mathbf{D}_1 & \\ \beta_{k+2}\mathbf{f}_2 & & \mathbf{D}_2 \end{pmatrix} \begin{pmatrix} & 1 & \\ \mathbf{Q}_1^\top & & \\ & & \mathbf{Q}_2^\top \end{pmatrix} = \mathbf{Q}\mathbf{H}\mathbf{Q}^\top, \tag{2}$$

where $\mathbf{l}_1^\top$ is the last row of $\mathbf{Q}_1$ and $\mathbf{f}_2^\top$ is the first row of $\mathbf{Q}_2$, and $\mathbf{H}$ has the so-called *arrowhead* structure. Thus, given this form, to compute the spectral decomposition of $\mathbf{T}$, it suffices to diagonalize $\mathbf{H}$. One can then apply recursively the algorithm to compute the spectral decompositions of $\mathbf{T}_1$ and $\mathbf{T}_2$, and, finally, at the "conquering stage," combine the solutions with the eigendecomposition of $\mathbf{H}$. For the individual steps to be computed efficiently, we will need to appropriately use the FMM from Proposition 1.1.

2.1 Symmetric arrowhead diagonalization

The first step is to provide an end-to-end complexity analysis for the arrowhead diagonalization algorithm of [46], when implemented with the FMM. We start with an $n \times n$ arrowhead matrix of the form

$$\mathbf{H} = \begin{pmatrix} \alpha & \mathbf{z}^\top \\ \mathbf{z} & \mathbf{D} \end{pmatrix}, \tag{3}$$

where $\mathbf{D}$ is a diagonal matrix, $\mathbf{z}$ is a vector of size $n-1$ and α is a scalar. Without loss of generality we assume that $\|\mathbf{H}\| \leq 1$. The main result is stated in Theorem 2.1. The full proof of Theorem 2.1 relies on several technical lemmas that can be found in the full version [79, Appendices A and B]. Here we briefly describe the methodology, which was originally described in [46], but it was not analyzed in detail.

1. Deflation: In the first step, the matrix $\mathbf{H}$ is preprocessed to ensure that it satisfies the following: $d_{i+1} - d_i \geq \tau$, and $|\mathbf{z}_i| \geq \tau$, where $\tau \in (0,1)$ is an appropriately chosen parameter that will be specified later. If this holds, it implies several useful properties, in particular, the eigenvalues λ_i of $\mathbf{H}$ are the roots of the secular equation

$$f(\lambda) = \lambda - \alpha + \sum_{j=2}^{n} \frac{\mathbf{z}_j^2}{d_j - \lambda}, \tag{4}$$

and they satisfy the following:

$$\lambda_1 < d_2 < \lambda_2 < \ldots < d_n < \lambda_n, \tag{5}$$

$$\min\left\{|d_i - \lambda_i|, |d_i - \lambda_{i-1}|\right\} \geq \frac{\tau^3}{n+1}, \quad \text{for all } i = 2, \ldots, n. \tag{6}$$

We refer to [79, Lemma A.1] for more details. This allows us to efficiently utilize the FMM in the subsequent steps. The full deflation procedure takes $O(n\log(n))$ steps, and it is described in [79, Proposition A.1].

2. Eigenvalues: The eigenvalues of the deflated matrix can be conveniently approximated as the roots of Eq. (4). FMM-based root finder is described in [79, Lemma B.1], which requires a total of at most $O(n \operatorname{polylog}(n, \epsilon^{-1}, \tau^{-1}))$ arithmetic operations.
3. From [17, 46], it is known that the approximate eigenvalues returned by the root finder are the exact eigenvalues of another arrowhead matrix $\widehat{\mathbf{H}} = \begin{pmatrix} \widehat{\alpha} & \widehat{\mathbf{z}}^\top \\ \widehat{\mathbf{z}} & \mathbf{D} \end{pmatrix}$. There is an analytical expression for the elements of $\widehat{\mathbf{H}}$. [79, Lemma B.2] describes how to approximate those elements up to error ϵ with the FMM in $O(n \operatorname{polylog}(n, \epsilon^{-1}, \tau^{-1}))$ arithmetic operations.
4. Next, given that we know the exact eigenvalues and the approximate elements of the matrix $\widehat{\mathbf{H}}$, we can focus on computing its eigenvectors. In particular, in [79, Lemma B.3] it is shown how to use the FMM to approximate the inner products between the eigenvectors of $\widehat{\mathbf{H}}$ and some arbitrary unit vector $\mathbf{b}$. Computing such inner products with all the columns of the identity, we obtain the final approximate eigenvector matrix of $\mathbf{H}$ and, ultimately, an approximate diagonalization of $\mathbf{H}$.

▶ **Theorem 2.1.** *Given a symmetric arrowhead matrix $\mathbf{H} \in \mathbb{R}^{n \times n}$ as in Eq. (3), with $\|\mathbf{H}\| \leq 1$, an accuracy parameter $\epsilon \in (0, 1)$, a matrix $\mathbf{B}$ with r columns $\mathbf{B}_i, i \in [r]$, where $\|\mathbf{B}_i\| \leq 1$, and an (ϵ, n)-FMM implementation (see Prop. 1.1), we can compute a diagonal matrix $\widetilde{\mathbf{\Lambda}}$, and a matrix $\widetilde{\mathbf{Q}}_{\mathbf{B}}$, such that*

$$\left\| \mathbf{H} - \mathbf{Q}\widetilde{\mathbf{\Lambda}}\mathbf{Q}^\top \right\| \leq \epsilon, \quad \left| \left(\mathbf{Q}^\top \mathbf{B} - \widetilde{\mathbf{Q}}_{\mathbf{B}} \right)_{i,j} \right| \leq \epsilon / n^2,$$

where $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is (exactly) orthogonal, in $O\left(nr \log^{\xi+1}(\frac{n}{\epsilon})\right)$ arithmetic operations and comparisons, where $\xi \geq 1$ is a small constant that depends on the specific FMM implementation and it is independent of ϵ, n.

Alternatively, if only want to compute a set of approximate values $\widetilde{\lambda}_1, \ldots, \widetilde{\lambda}_n$, such that $|\lambda_i(\mathbf{H}) - \widetilde{\lambda}_i| \leq \epsilon$, the complexity reduces to $O\left(n \log(\frac{1}{\epsilon}) \log^{\xi}(\frac{n}{\epsilon})\right)$ arithmetic operations.

Proof. The full proof follows the methodology described above, and it is provided [79, Theorem B.1] in the full version. ◀

▶ Remark 2.2. We note that, if one is only interested in a full diagonalization of the arrowhead matrix, the use of the FMM for Step 3 is redundant, i.e., we can naively compute the elements of $\widehat{\mathbf{H}}$ exactly without the FMM in $O(n^2)$ operations, without affecting the final complexity. However, it is beneficial if we need to compute only a few matrix-vector products with the eigenvector matrix.

2.2 Tridiagonal diagonalization

Given the analysis for arrowhead diagonalization, we can now proceed to tridiagonal matrices. The next lemma bounds the error of the reduction to arrowhead form when the spectral factorizations of the matrices $\mathbf{T}_1$ and $\mathbf{T}_2$ in Equation (2) are approximate rather than exact. This will be used as an inductive step for the final algorithm.

▶ **Lemma 2.3.** *Let $\epsilon \in (0, 1/2)$ be a given accuracy parameter and $\mathbf{T} = \begin{pmatrix} \mathbf{T}_1 & \beta_{k+1}\mathbf{e}_k & \\ \beta_{k+1}\mathbf{e}_k^\top & \alpha_{k+1} & \beta_{k+2}\mathbf{e}_1^\top \\ & \beta_{k+2}\mathbf{e}_1 & \mathbf{T}_2 \end{pmatrix}$ be a tridiagonal matrix with size $n \geq 3$ and $\|\mathbf{T}\| \leq 1$, where*

$\mathbf{T}_1 = \mathbf{U}_1\mathbf{D}_1\mathbf{U}_1^\top$ *and* $\mathbf{T}_2 = \mathbf{U}_2\mathbf{D}_2\mathbf{U}_2^\top$ *be the exact spectral factorizations of* $\mathbf{T}_1$ *and* $\mathbf{T}_2$*. Let* $\widetilde{\mathbf{U}}_1, \widetilde{\mathbf{D}}_1, \widetilde{\mathbf{U}}_2, \widetilde{\mathbf{D}}_2$ *be approximate spectral factorizations of* $\mathbf{T}_1, \mathbf{T}_2$*. If these factors satisfy*

$$\left\|\mathbf{T}_{\{1,2\}} - \widetilde{\mathbf{U}}_{\{1,2\}}\widetilde{\mathbf{D}}_{\{1,2\}}\widetilde{\mathbf{U}}_{\{1,2\}}^\top\right\| \leq \epsilon_1, \quad \left\|\widetilde{\mathbf{U}}_{\{1,2\}}\widetilde{\mathbf{U}}_{\{1,2\}}^\top - \mathbf{I}\right\| \leq \epsilon_1/n,$$

for some $\epsilon_1 \in (0, 1/2)$*, where* $\widetilde{\mathbf{D}}_{\{1,2\}}$ *are both diagonal, then, assuming an* (ϵ, n)*-FMM implementation as in Prop. 1.1, we can compute a diagonal matrix* $\widetilde{\mathbf{D}}$ *and an approximately orthogonal matrix* $\widetilde{\mathbf{U}}$ *such that*

$$\left\|\widetilde{\mathbf{U}}^\top\widetilde{\mathbf{U}} - \mathbf{I}\right\| \leq 3(\epsilon_1 + \epsilon)/n, \quad \textit{and} \quad \left\|\mathbf{T} - \widetilde{\mathbf{U}}\widetilde{\mathbf{D}}\widetilde{\mathbf{U}}^\top\right\| \leq 2\epsilon_1 + 7\epsilon,$$

in a total of $O\left(n^2 \log^{\xi+1}(\frac{n}{\epsilon})\right)$ *arithmetic operations and comparisons, where* $\xi \geq 1$ *is a small constant that depends on the specific FMM implementation and it is independent of* ϵ, n*.*

Proof. The proof can be found in the full version, in [79, Appendix C, Lemma C.1]. ◀

Lemma 2.3 gives rise to the following recursive algorithm. We can finally proceed with the proof of Theorem 1.2, which gives the complexity of Algorithm 1.

Algorithm 1 Recursive algorithm based on [46] to diagonalize a symmetric tridiagonal matrix.

Algorithm: $[\widetilde{\mathbf{U}}, \widetilde{\mathbf{\Lambda}}] \leftarrow \mathsf{DIAGONALIZE}(\mathbf{T}, \epsilon)$
if $n \leq 2$ **:**
  Compute $\widetilde{\mathbf{U}}, \widetilde{\mathbf{\Lambda}}$ to be the exact diagonalization of $\mathbf{T}$.
else:
  Partition $\mathbf{T} = \begin{pmatrix} \mathbf{T}_1 & \beta_{k+1}\mathbf{e}_k & \\ \beta_{k+1}\mathbf{e}_k^\top & \alpha_{k+1} & \beta_{k+2}\mathbf{e}_1^\top \\ & \beta_{k+2}\mathbf{e}_1 & \mathbf{T}_2 \end{pmatrix}$.
  $[\widetilde{\mathbf{U}}_1, \widetilde{\mathbf{D}}_1] \leftarrow \mathsf{DIAGONALIZE}(\mathbf{T}_1, \epsilon)$.
  $[\widetilde{\mathbf{U}}_2, \widetilde{\mathbf{D}}_2] \leftarrow \mathsf{DIAGONALIZE}(\mathbf{T}_2, \epsilon)$.
  Assemble $\widetilde{\mathbf{U}}, \widetilde{\mathbf{\Lambda}}$ from $\mathbf{T}, \widetilde{\mathbf{U}}_1, \widetilde{\mathbf{D}}_1, \widetilde{\mathbf{U}}_2, \widetilde{\mathbf{D}}_2$ using Lemma 2.3 with parameter ϵ.
return $\widetilde{\mathbf{U}}, \widetilde{\mathbf{\Lambda}}$.

2.2.1 Proof of Theorem 1.2

Proof. Let $\mathbf{T}^{(p)}$ be a matrix at recursion depth p. We can always divide $\mathbf{T}^{(p)}$ such that the sizes of $\mathbf{T}^{(p)}_{\{1,2\}}$ differ by at most 1. If we keep dividing the matrix $\mathbf{T}^{(p)}$ in this manner, then the recursion tree has depth at most $d = \lceil \log(n) \rceil$.

The correctness follows by induction. Consider the base case d where $\mathbf{T}^{(d)}$ has size at most 5×5, which means that $\mathbf{T}^{(d)}_{1,2}$ have size 1×1 or 2×2. We can compute the exact diagonalization of $\mathbf{T}^{(d)}_{1,2}$, and therefore they satisfy the requirements of Lemma 2.3 for $\epsilon_1 = 0$. Thus, if we apply Lemma 2.3 with parameter ϵ' to compute the matrices $\widetilde{\mathbf{U}}^{(d)}$ and $\widetilde{\mathbf{D}}^{(d)}$, they will satisfy the following bounds:

$$\mathsf{err}_{\widetilde{\mathbf{U}}}(d) := \left\|\widetilde{\mathbf{U}}^{(d)\top}\widetilde{\mathbf{U}}^{(d)} - \mathbf{I}\right\| \leq 3(\epsilon_1 + \epsilon')/n = 3\epsilon'/n,$$

$$\mathsf{err}_{\mathbf{T}}(d) := \left\|\mathbf{T} - \widetilde{\mathbf{U}}^{(d)}\widetilde{\mathbf{D}}^{(d)}\widetilde{\mathbf{U}}^{(d)\top}\right\| \leq 7\epsilon'.$$

We need to bound the error at the root (depth $p = 0$). As long as the error at depth $p-1$ is smaller than $1/2$, the the error at depth p is bounded by

$$\begin{aligned}&\mathsf{err}_{\widetilde{\mathbf{U}}}(p) \leq 3\mathsf{err}_{\widetilde{\mathbf{U}}}(p+1) + 3\epsilon'/n < 4\mathsf{err}_{\widetilde{\mathbf{U}}}(p+1) + 4\epsilon'/n,\\ &\mathsf{err}_{\mathbf{T}}(p) \leq 2\mathsf{err}_{\mathbf{T}}(p+1) + 7\epsilon' < 8\mathsf{err}_{\mathbf{T}}(p+1) + 8\epsilon'.\end{aligned}$$

Solving the recursion we have

$$\mathsf{err}_{\widetilde{\mathbf{U}}}(0) < \sum_{p=1}^{\lceil \log(n) \rceil} 4^p\epsilon'/n = O(n^2)\frac{\epsilon'}{n} = O(n)\epsilon', \quad \mathsf{err}_{\mathbf{T}}(0) < \sum_{p=1}^{\lceil \log(n) \rceil} 8^p\epsilon' = O(n^3)\epsilon'.$$

It thus suffices to run the algorithm with initial error $\epsilon' = c\epsilon/n^3$ for some small constant c.

The complexity analysis for ϵ' follows. For size n, the complexity is given by

$$T(n) \leq 2T(\tfrac{n}{2}) + Cn^2 \log^{\xi+1}(\tfrac{n}{\epsilon'}).$$

Solving the recursion yields $T(n) = O\left(n^2 \log^{\xi+1}(\frac{n}{\epsilon'})\right) = O\left(n^2 \log^{\xi+1}(\frac{n}{\epsilon})\right)$.

The final matrices $\widetilde{\mathbf{U}} = \widetilde{\mathbf{U}}^{(0)}$ and $\widetilde{\mathbf{\Lambda}} = \widetilde{\mathbf{D}}^{(0)}$ satisfy $\left\|\widetilde{\mathbf{U}}^{(d)\top}\widetilde{\mathbf{U}}^{(d)} - \mathbf{I}\right\| \leq \epsilon/n^2$ and $\left\|\mathbf{T} - \widetilde{\mathbf{U}}\widetilde{\mathbf{\Lambda}}\widetilde{\mathbf{U}}^\top\right\| \leq \epsilon$, which means that:

$$\left\|\widetilde{\mathbf{\Lambda}}\right\| \leq \left\|\widetilde{\mathbf{U}}^{-1}\right\|^2 \left(\|\mathbf{T}\| + \left\|\mathbf{T} - \widetilde{\mathbf{U}}\widetilde{\mathbf{\Lambda}}\widetilde{\mathbf{U}}^\top\right\|\right) \leq \frac{1}{1-\epsilon/n^2}(1+\epsilon).$$

We can then write $\widetilde{\mathbf{U}} = \mathbf{U} + \mathbf{E}_{\mathbf{U}}$, which gives

$$\begin{aligned}\left\|\mathbf{T} - \mathbf{U}\widetilde{\mathbf{\Lambda}}\mathbf{U}^\top\right\| &\leq \left\|\mathbf{T} - \widetilde{\mathbf{U}}\widetilde{\mathbf{\Lambda}}\widetilde{\mathbf{U}}^\top\right\| + 2\left\|\widetilde{\mathbf{\Lambda}}\right\|\|\mathbf{E}_{\mathbf{U}}\| + \left\|\widetilde{\mathbf{\Lambda}}\right\|\|\mathbf{E}_{\mathbf{U}}\|^2\\ &\leq \epsilon + \frac{(1+\epsilon)}{1-\epsilon/n^2}\frac{\epsilon}{n^2} + \frac{1+\epsilon}{1-\epsilon/n^2}\left(\frac{\epsilon}{n^2}\right)^2 = \epsilon\left(1 + \frac{(1+\epsilon)(1+\epsilon/n^2)}{n^2-\epsilon}\right).\end{aligned}$$

Rescaling ϵ by a small constant slightly larger than one gives the alternative statement. ◀

2.3 Hermitian diagonalization

Given an algorithm to diagonalize tridiagonal matrices, the following corollary is immediate.

▶ **Corollary 2.4.** *Let $\mathbf{A}$ be a Hermitian matrix of size n with $\|\mathbf{A}\| \leq 1$. Given accuracy $\epsilon \in (0, 1/2)$, and an (ϵ, n)-FMM implementation of Prop. 1.1, we can compute a matrix $\widetilde{\mathbf{Q}}$ and a diagonal matrix $\widetilde{\mathbf{\Lambda}}$ such that*

$$\left\|\mathbf{A} - \widetilde{\mathbf{Q}}\widetilde{\mathbf{\Lambda}}\widetilde{\mathbf{Q}}^*\right\| \leq \epsilon, \quad \left\|\widetilde{\mathbf{Q}}^*\widetilde{\mathbf{Q}} - \mathbf{I}\right\| \leq \epsilon/n^2.$$

The algorithm requires a total of $O\left(n^\omega \log(n) + n^2 \log^{\xi+1}(\frac{n}{\epsilon})\right)$ arithmetic operations and comparisons, where $\xi \geq 1$ is a small constant that depends on the specific FMM implementation and it is independent of ϵ, n.

Proof. The first step is to use the algorithm of Schönhage [75] to reduce the matrix to tridiagonal form in $O(n^\omega \log(n))$ arithmetic operations using unitary similarity transformations. Thereafter we diagonalize the tridiagonal matrix using Theorem 1.2. The full proof, as well as an alternative statement of the result, can be found in [79, Corollary C.2, Appendix C.3]. ◀

3 Stability of tridiagonal reduction

In this section we analyze the numerical stability and the boolean complexity of Schönhage's algorithm the floating point model. For this we will use the following stable matrix multiplication and backward-stable QR factorization algorithms as subroutines from [26, 27].

3.1 Matrix nomenclature

Schönhage [75] used a block variant of Rutishauser's algorithm [72] to reduce a matrix to tridiagonal form, where elementary rotations are replaced with block factorizations; see also [16, 4, 5] for similar methodologies. We start with a $n \times n$ block-pentadiagonal matrix $\mathbf{A}^{(k,s,t)}$, where $k \in \{0, \ldots, \log(n) - 2\}$ (we assume without loss of generality that n is a power of two). The matrix $\mathbf{A}^{(k,s,t)}$ is partitioned in $b_k \times b_k$ blocks of size $n_k \times n_k$ each, where $b_k = \frac{n}{n_k}$ and $n_k = 2^k$. The integer $s \in 2, \ldots, b_k$ t denotes that all the blocks $\mathbf{A}_{i,i-2}$ and $\mathbf{A}_{i-2,i}$, for all $i = 2, \ldots, s$ are equal to zero. $s = 2$ is a special case to denote a full block pentadiagonal matrix. The integer $t \in \{s+2, \ldots, b_k\}$ denotes that the matrix has two additional nonzero blocks in the third off-diagonals, specifically at positions $\mathbf{A}_{t,t-3}$ and $\mathbf{A}_{t-3,t}$. These blocks are often called the "bulge" in the literature. When $t = 0$, there is no bulge. As a consequence, the matrix $\mathbf{A}^{(k,2,0)}$ is full block-pentadiagonal, while the matrix $\mathbf{A}^{(k,b_k,0)}$ is block-tridiagonal. An illustration of these definitions is shown in Equation (7). A box is placed around the bulge on the second matrix.

$$\overbrace{\begin{pmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \mathbf{A}_{1,3} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \mathbf{A}_{2,3} & \mathbf{A}_{2,4} & 0 & 0 & 0 & 0 \\ \mathbf{A}_{3,1} & \mathbf{A}_{3,2} & \mathbf{A}_{3,3} & \mathbf{A}_{3,4} & \mathbf{A}_{3,5} & 0 & 0 & 0 \\ 0 & \mathbf{A}_{4,2} & \mathbf{A}_{4,3} & \mathbf{A}_{4,4} & \mathbf{A}_{4,5} & \mathbf{A}_{4,6} & 0 & 0 \\ 0 & 0 & \mathbf{A}_{5,3} & \mathbf{A}_{5,4} & \mathbf{A}_{5,5} & \mathbf{A}_{5,6} & \mathbf{A}_{5,7} & 0 \\ 0 & 0 & 0 & \mathbf{A}_{6,4} & \mathbf{A}_{6,5} & \mathbf{A}_{6,6} & \mathbf{A}_{6,7} & \mathbf{A}_{6,8} \\ 0 & 0 & 0 & 0 & \mathbf{A}_{7,5} & \mathbf{A}_{7,6} & \mathbf{A}_{7,7} & \mathbf{A}_{7,8} \\ 0 & 0 & 0 & 0 & 0 & \mathbf{A}_{8,6} & \mathbf{A}_{8,7} & \mathbf{A}_{8,8} \end{pmatrix}}^{\mathbf{A}^{(k,2,0)}}, \quad \overbrace{\begin{pmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \mathbf{A}_{2,3} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{A}_{3,2} & \mathbf{A}_{3,3} & \mathbf{A}_{3,4} & \mathbf{A}_{3,5} & \boxed{\mathbf{A}_{3,6}} & 0 & 0 \\ 0 & 0 & \mathbf{A}_{4,3} & \mathbf{A}_{4,4} & \mathbf{A}_{4,5} & \mathbf{A}_{4,6} & 0 & 0 \\ 0 & 0 & \mathbf{A}_{5,3} & \mathbf{A}_{5,4} & \mathbf{A}_{5,5} & \mathbf{A}_{5,6} & \mathbf{A}_{5,7} & 0 \\ 0 & 0 & \boxed{\mathbf{A}_{6,3}} & \mathbf{A}_{6,4} & \mathbf{A}_{6,5} & \mathbf{A}_{6,6} & \mathbf{A}_{6,7} & \mathbf{A}_{6,8} \\ 0 & 0 & 0 & 0 & \mathbf{A}_{7,5} & \mathbf{A}_{7,6} & \mathbf{A}_{7,7} & \mathbf{A}_{7,8} \\ 0 & 0 & 0 & 0 & 0 & \mathbf{A}_{8,6} & \mathbf{A}_{8,7} & \mathbf{A}_{8,8} \end{pmatrix}}^{\mathbf{A}^{(k,4,6)}}. \tag{7}$$

3.2 Rotations

The algorithm defines two types of block rotations R_i and R'_i, which are unitary similarity transformations, with the following properties.

▶ **Definition 3.1** (Rotations). *The algorithm of [75] uses the following two types of rotations:*

1. *$R_i(\mathbf{A}^{(k,i,0)})$, for $i = 2, \ldots, b_k - 1$, operates on a block-pentadiagonal matrix without a bulge. It transforms the matrix $\mathbf{A}^{(k,i,0)}$ to a matrix $\mathbf{A}^{(k,i+1,i+3)}$. In paricular, the block $\mathbf{A}_{i,i-1}$ becomes upper triangular, the block $\mathbf{A}_{i+1,i-1}$ becomes zero, and a new bulge block arises at $\mathbf{A}_{i,i+3}$. Due to symmetry, $\mathbf{A}_{i-1,i}$ becomes lower triangular, $\mathbf{A}_{i-1,i+1}$ is eliminated, and $\mathbf{A}_{i+3,i}$ becomes non-zero.*
2. *$R'_j(\mathbf{A}^{(k,s,j+1)})$, for some $j = s+1, \ldots, b_k - 1$, operates on a block-pentadiagonal matrix with a bulge at positions $\mathbf{A}_{j+1,j-2}$, $\mathbf{A}_{j-2,j+1}$. It transforms the matrix $\mathbf{A}^{(k,s,j+1)}$ to a matrix $\mathbf{A}^{(k,s,j+3)}$, such that the bulge is moved two positions "down-and-right", i.e. the blocks $\mathbf{A}_{j-2,j+1}$ and $\mathbf{A}_{j+1,j-2}$ become zero and the blocks $\mathbf{A}_{j,i+3}$ and $\mathbf{A}_{j,j+3}$ become the new bulge. In addition, the matrices $\mathbf{A}_{j,j-2}$ and $\mathbf{A}_{j+1,j-1}$ become upper triangular, and, by symmetry, the matrices $\mathbf{A}_{j-2,j}$ and $\mathbf{A}_{j-1,j+1}$ become lower triangular.*

In the full version, in [79, Lemmas E.1 and E.2] it is proved that both rotation types can be stably implemented in floating point using fast QR factorizations.

3.3 Recursive bandwidth halving

The following Algorithm 2 halves the bandwidth of a symmetric matrix using rotations. Its complexity and stability properties are stated in the full version [79, Lemma E.3]. Applying this algorithm recursively gives the main Theorem 3.2.

Algorithm 2 Halves the bandwidth of a Hermitian matrix with unitary rotations.

Algorithm: $[\widehat{\mathbf{Q}}^{(k)}, \mathbf{A}^{(k-1,2,0)}] \leftarrow \mathsf{HALVE}(\mathbf{A}^{(k,2,0)}, k, n)$
1: Set $n_k = 2^k$, $b_k = n/n_k$.
2: **for** $i = 2, \ldots, b_k$ **:**
3: Compute $\mathbf{Q}_{i,i}, \mathbf{A}^{(k,i+1,i+3)} \leftarrow R_i(\mathbf{A}^{(k,i,0)})$.
4: **for** $j = i+2, \ldots, b_k$ with step 2 **:**
5: Compute $\mathbf{Q}_{i,j}, \mathbf{A}^{(k,i+1,j+3)} \leftarrow R'_j(\mathbf{A}^{(k,i+1,j+1)})$.
6: Stack together all the matrices $\mathbf{Q}_{i,j}$ to form $\mathbf{Q}_i$.
7: Assemble the matrix $\widehat{\mathbf{Q}}^{(k)}$ by multiplying the matrices $\mathbf{Q}_i$.
8: **return** $\widehat{\mathbf{Q}}^{(k)}, \mathbf{A}^{(k-1,2,0)}$.

▶ **Theorem 3.2.** *There exists a floating point implementation of the tridiagonal reduction algorithm of [75], which takes as input a Hermitian matrix* $\mathbf{A}$*, and returns a tridiagonal matrix* $\widetilde{\mathbf{T}}$*, and (optionally) an approximately unitary matrix* $\widetilde{\mathbf{Q}}$*. If the machine precision* $\mathbf{u}$ *satisfies* $\mathbf{u} \leq \epsilon \frac{1}{cn^{\beta+4}}$*, where* $\epsilon \in (0,1)$*, and* c, β *are constants that are independent of* n*, which translates to* $O(\log(n) + \log(1/\epsilon))$ *bits of precision, then the following hold:*

$$\left\| \widetilde{\mathbf{Q}}\widetilde{\mathbf{Q}}^* - \mathbf{I} \right\| \leq \epsilon, \quad \textit{and} \quad \left\| \mathbf{A} - \widetilde{\mathbf{Q}}\widetilde{\mathbf{T}}\widetilde{\mathbf{Q}}^* \right\| \leq \epsilon \left\| \mathbf{A} \right\|.$$

The algorithm executes at most:

- $O\left(n^2 S_\omega(\log(n))\right)$ *floating point operations to return only* $\widetilde{\mathbf{T}}$*, where* $S_x(m) = \sum_{l=1}^{m} (2^{x-2})^l$*.*
- *If* $\mathbf{A}$ *is banded with* $1 \leq d \leq n$ *bands, the floating point operations reduce to* $O(n^2 S_\omega(\log(d))$*.*
- *If* $\widetilde{\mathbf{Q}}$ *is also returned, the complexity increases to* $O(n^2 C_\omega(\log(n)))$*, where* $C_x(n) := \sum_{k=2}^{\log(n)-2} (S_x(\log(n)-1) - S_x(k))$*.*

If ω *is treated as a constant* $\omega \approx 2.371$*, the corresponding complexities are* $O(n^\omega), O(n^2 d^{\omega-2})$*, and* $O(n^\omega \log(n))$*, respectively.*

Proof. The proof can be found in the full version [79, Theorem E.3, Appendix E]. ◀

3.4 Eigenvalues of Hermitian matrices

We now have all the prerequisites to compute the eigenvalues Hermitian matrices in nearly matrix multiplication time in finite precision. For this we can use the eigenvalue solver of [13, 15], which has $\widetilde{O}(n^2)$ boolean complexity, albeit in the Boolean RAM model, instead of floating point. The algorithm accepts as input symmetric tridiagonal matrices with bounded integer entries.

▶ **Theorem 3.3** (Imported from [13, 15]). *Let $\mathbf{T}$ be a symmetric tridiagonal matrix with integer elements bounded in magnitude by 2^m for some m. Let $\epsilon = 2^{-u} \in (0,1)$ be a desired accuracy. Algorithm 4.1 of [13] computes a set of approximate eigenvalues $\widetilde{\lambda}_i \in \mathbb{R}$ (which are returned as rationals) such that $\left|\widetilde{\lambda}_i - \lambda_i(\mathbf{T})\right| < \epsilon$. The algorithm requires $O\left(n^2 b \log^2(n) \log(nb)(\log^2(b) + \log(n)) \log(\log(nb))\right)$ boolean operations, where $b = m + u$.*

▶ **Theorem 3.4.** *Let $\mathbf{A}$ be a (banded) Hermitian matrix, with $\|\mathbf{A}\| \leq 1$, $1 \leq d \leq n-1$ off-diagonals, and let $\epsilon \in (0,1)$ be an accuracy parameter. Assume that the elements of $\mathbf{A}$ are floating point numbers on a machine with precision $\mathbf{u}$, $t = \log(1/\mathbf{u})$ bits for the significand, and $p = O(\log(\log(n)))$ bits for the exponent. There exists an algorithm that returns a set of n approximate eigenvalues $\widetilde{\lambda}_1, \ldots, \widetilde{\lambda}_n$ such that*

$$\left|\widetilde{\lambda}_i - \lambda_i(\mathbf{A})\right| \leq \epsilon,$$

using at most $O\left(n^2 S_\omega(\log(d)) \cdot \mathcal{F}(\log(\frac{n}{\epsilon})) + n^2 \operatorname{polylog}(\frac{n}{\epsilon})\right)$ boolean operations, where $\mathcal{F}(b)$ is the bit complexity of a floating point operation on b bits, and $n^2 S_\omega(\log(d)) = O(n^2 d^{\omega-2})$ if ω is treated as a constant greater than two.

Proof. First, recall that each element of a floating point matrix $\mathbf{A}$ is represented by a floating point number $\mathbf{A}_{i,j} = \pm 1 \times 2^{e_{i,j}} \times m_{i,j}$, where $e_{i,j} \in [-2^p, 2^p]$ is its exponent, p is the number of exponent bits, and $m_{i,j}$ is the significand, which is an integer with $t = \log(1/\mathbf{u})$ bits. To represent all the elements of $\mathbf{A}$ with normalized floating point numbers it is sufficient to use $p = O\left(\log(\log(A))\right)$, where $A = \max\left\{\|\mathbf{A}\|_{\max}, \frac{1}{\|\mathbf{A}\|_{\min}}\right\}$. By assumption, $\|\mathbf{A}\|_{\max} \leq \|\mathbf{A}\| \leq 1$. It is common to assume that $\frac{1}{\|\mathbf{A}\|_{\min}} \in \operatorname{poly}(n)$, which means that $O(\log(\log(n)))$ bits for the exponent are sufficient.

To compute the eigenvalues of $\mathbf{A}$ we work as follows. We first run the algorithm of Theorem 3.2 to reduce $\mathbf{A}$ to tridiagonal form $\widetilde{\mathbf{T}}$, with parameter ϵ and $O\left(\log(\frac{n}{\epsilon})\right)$ bits of precision, and a total of $O(n^2 S_\omega(\log(d)))$ floating point operations, which is equal to $O(n^2 d^{\omega-2})$ if ω is treated as a constant. It holds that

$$\left|\lambda_i(\widetilde{\mathbf{T}}) - \lambda_i(\mathbf{A})\right| \leq \epsilon' \|\mathbf{A}\| \leq \epsilon \frac{\|\mathbf{A}\|}{n 2^{2^{p+1}}} \leq \epsilon,$$

where in the last inequality we used that $\|\mathbf{A}\| \leq n\|\mathbf{A}_{\max}\| \leq n 2^{2^{p+1}}$.

Therefore, it suffices to approximate the eigenvalues of $\widetilde{\mathbf{T}}$. For this we first transform $\widetilde{\mathbf{T}}$ to a tridiagonal matrix with integer values and use the algorithm of [13] to compute its eigenvalues. A floating point matrix can be transformed to one with integer entries as follows. Assuming that we have p bits for the floating point exponent, and $t = \log(1/\mathbf{u})$ bits for the significand, the floating point numbers with $O(p+t)$ bits can be transformed to integers of $O(2^{2p}+t)$ bits by multiplying all the elements with $2^{2^{p}+t}$. This is achieved by first allocating a tridiagonal matrix with $3n-2$ integers with $O(2^{2p}+t) = O(\log^2(n) + \log(n/\epsilon))$ bits each, that are initially set to zero. We then visit each floating point element $\widetilde{\mathbf{T}}_{i,j}$ of the original matrix, which is represented as a number $\pm 1 \times 2^{e_{i,j}} \times m_{i,j}$, where $e_{i,j} \in [-2^p, 2^p]$ is its exponent and $m_{i,j}$ is the significand, which is an integer with t bits. We copy the t bits of $m_{i,j}$ at the positions $e_{i,j} + 2^p, e_{i,j} + 2^p + 1, \ldots, e_{i,j} + 2^p + t$ of the corresponding element of the new matrix. This takes $O(n(2^{2p}+t)) = O(n(\log^2(n) + \log(n/\epsilon)))$ boolean operations in total to allocate the new matrix and copy the elements.

Now that we have an integer valued symmetric tridiagonal matrix $\mathbf{T}'$, with $O(2^{2p}+t) = O(\log^2(n) + \log(n/\epsilon))$ bits per element, we can compute its eigenvalues up to additive accuracy ϵ with Theorem 3.3. Specifically, we set $\epsilon'' = \epsilon \cdot 2^{2^{p}+t}$, and run the algorithm with the matrix

$\mathbf{T}'$ as input and $u = \log(\frac{1}{\epsilon''})$. Let

$$b = u + m = \log\left(\frac{1}{\epsilon''}\right) + 2^{p+1} + t = \log\left(\frac{1}{\epsilon 2^{2^p+t}}\right) + \log\left(2^{2^{p+1}+t}\right) = O(\log(\tfrac{n}{\epsilon})),$$

where the last inequality follows from the assumption that $p = O(\log(\log(n)))$. The returned eigenvalues λ'_i satisfy

$$\left|\lambda'_i - \lambda_i(\mathbf{T}')\right| \leq \epsilon'' \Rightarrow \left|\frac{\lambda'_i}{2^{2^p+t}} - \lambda_i(\widetilde{\mathbf{T}})\right| \leq \epsilon.$$

Therefore, if we set $\widetilde{\lambda}_i = \frac{\lambda'_i}{2^{2^p+t}}$, we finally obtain that $\left|\widetilde{\lambda}_i - \lambda_i(\mathbf{A})\right| \leq 2\epsilon$. Rescaling ϵ gives the final bound. The algorithm runs in

$$\begin{aligned} B &= O\Big(n^2 b \log^2(n) \log(nb)(\log^2(b) + \log(n)) \log(\log(nb))\Big) \\ &= O\Big(n^2 \log(\tfrac{n}{\epsilon}) \log^2(n) \log(n \log(\tfrac{n}{\epsilon})) \left[\log^2(\log(\tfrac{n}{\epsilon})) + \log(n)\right] \log(\log(n \log(\tfrac{n}{\epsilon})))\Big) \end{aligned}$$

boolean operations. ◀

4 Conclusion

In this work we provided a deterministic complexity analysis for Hermitian eigenproblems. In the Real RAM model, we reported nearly-linear complexity upper bounds for the full diagonalization of arrowhead and tridiagonal matrices, and nearly matrix multiplication-type complexities for diagonalizing Hermitian matrices and for the SVD. This was achieved by analyzing the divide-and-conquer algorithm of [46], when implemented with the Fast Multipole Method. We also showed that the tridiagonal reduction algorithm of [75] is numerically stable in the floating point model. This paved the way to obtain improved deterministic boolean complexities for computing the eigenvalues, singular values, spectral gaps, and spectral projectors, of Hermitian matrices and Hermitian-definite pencils. Some interesting questions for future research are the following:

1. **Stability of arrowhead diagonalization:** The FMM-accelerated arrowhead diagonalization algorithm was only analyzed in the Real RAM model. Several works [46, 83, 66, 21] have provided stabilization techniques of related algorithms in floating point, albeit, without an end-to-end complexity analysis. Such an analysis will be insightful to better understand the boolean complexity of (Hermitian) diagonalization.
2. **Condition number in the SVD complexity:** The complexity of the SVD in Theorem 1.3 has a polylogarithmic dependence on the condition number. Frustratingly, we were not able to remove it at the time of this writing.
3. **Non-Hermitian diagonalization:** Schönhage also proved that a non-Hermitian matrix can be reduced to upper Hessenberg form in matrix multiplication time [75]. In this work we only provided the stability analysis for the Hermitian case. It would be interesting to investigate whether the Hessenberg reduction algorithm can be used to diagonalize non-Hermitian matrices in matrix multiplication time deterministically (e.g. in conjunction with [8, 7, 9]).
4. **Deterministic pseudospectral shattering:** One of the main techniques of the seminal work of [6] is called "pseudospectral shattering." The main idea is to add a tiny random perturbation to the original matrix to ensure that the minimum eigenvalue gap between any

pair of eigenvalues will be at least polynomial in $1/n$. We highlight that the arrowhead deflation preprocessing step has this precise effect: the pseudospectrum is shattered with respect to a deterministic grid. Can this be generalized to obtain *deterministic* pseudospectral shattering techniques, for Hermitian or even non-Hermitian matrices?

References

1 Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication. In *Proc. 2025 ACM-SIAM Symposium on Discrete Algorithms*, pages 2005–2039. SIAM, 2025. doi:10.1137/1.9781611978322.63.

2 Alexandr Andoni and Huy L Nguyen. Eigenvalues of a matrix in the streaming model. In *Proc. 24th ACM-SIAM Symposium on Discrete Algorithms*, pages 1729–1737. SIAM, 2013. doi:10.1137/1.9781611973105.124.

3 Diego Armentano, Carlos Beltrán, Peter Bürgisser, Felipe Cucker, and Michael Shub. A stable, polynomial-time algorithm for the eigenpair problem. *Journal of the European Mathematical Society*, 20(6):1375–1437, 2018.

4 Grey Ballard, James Demmel, and Nicholas Knight. Communication avoiding successive band reduction. *ACM SIGPLAN Notices*, 47(8):35–44, 2012. doi:10.1145/2145816.2145822.

5 Grey Ballard, James Demmel, and Nicholas Knight. Avoiding communication in successive band reduction. *ACM Transactions on Parallel Computing*, 1(2):1–37, 2015. doi:10.1145/2686877.

6 Jess Banks, Jorge Garza-Vargas, Archit Kulkarni, and Nikhil Srivastava. Pseudospectral Shattering, the Sign Function, and Diagonalization in Nearly Matrix Multiplication Time. *Foundations of Computational Mathematics*, pages 1–89, 2022.

7 Jess Banks, Jorge Garza-Vargas, and Nikhil Srivastava. Global Convergence of Hessenberg Shifted QR II: Numerical Stability. *arXiv*, 2022. doi:10.48550/arXiv.2205.06810.

8 Jess Banks, Jorge Garza-Vargas, and Nikhil Srivastava. Global convergence of Hessenberg Shifted QR I: Exact Arithmetic. *Foundations of Computational Mathematics*, pages 1–34, 2024.

9 Jess Banks, Jorge Garza-Vargas, and Nikhil Srivastava. Global Convergence of Hessenberg Shifted QR III: Approximate Ritz Values via Shifted Inverse Iteration. *SIAM Journal on Matrix Analysis and Applications*, 46(2):1212–1246, 2025.

10 Jesse L Barlow. Error analysis of update methods for the symmetric eigenvalue problem. *SIAM Journal on Matrix Analysis and Applications*, 14(2):598–618, 1993. doi:10.1137/0614042.

11 Michael Ben-Or and Lior Eldar. A Quasi-Random Approach to Matrix Spectral Analysis. In *Proc. 9th Innovations in Theoretical Computer Science Conference*, pages 6:1–6:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICS.ITCS.2018.6.

12 Rajendra Bhatia. *Perturbation bounds for matrix eigenvalues*. SIAM, 2007.

13 Dario Bini and Victor Pan. Parallel complexity of tridiagonal symmetric eigenvalue problem. In *Proc. 2nd ACM-SIAM Symposium on Discrete Algorithms*, pages 384–393, 1991. URL: http://dl.acm.org/citation.cfm?id=127787.127855.

14 Dario Bini and Victor Pan. Practical improvement of the divide-and-conquer eigenvalue algorithms. *Computing*, 48(1):109–123, 1992. doi:10.1007/BF02241709.

15 Dario Bini and Victor Y Pan. Computing matrix eigenvalues and polynomial zeros where the output is real. *SIAM Journal on Computing*, 27(4):1099–1115, 1998. doi:10.1137/S0097539790182482.

16 Christian H Bischof, Bruno Lang, and Xiaobai Sun. A framework for symmetric band reduction. *ACM Transactions on Mathematical Software*, 26(4):581–601, 2000. doi:10.1145/365723.365735.

17 D Boley and Gene Howard Golub. Inverse eigenvalue problems for band matrices. In *Numerical Analysis: Proceedings of the Biennial Conference Held at Dundee, June 28–July 1, 1977*, pages 23–31. Springer, 1977.

18 Christos Boutsidis and David P Woodruff. Optimal CUR matrix decompositions. In *Proc. 46th ACM Symposium on Theory of Computing*, pages 353–362, 2014. doi:10.1145/2591796.2591819.

19 Christos Boutsidis, David P Woodruff, and Peilin Zhong. Optimal principal component analysis in distributed and streaming models. In *Proc. 48th ACM Symposium on Theory of Computing*, pages 236–249, 2016. doi:10.1145/2897518.2897646.

20 Christos Boutsidis, Anastasios Zouzias, Michael W Mahoney, and Petros Drineas. Randomized dimensionality reduction for k-means clustering. *IEEE Transactions on Information Theory*, 61(2):1045–1062, 2014. doi:10.1109/TIT.2014.2375327.

21 Difeng Cai and Jianlin Xia. A stable matrix version of the fast multipole method: stabilization strategies and examples. *ETNA-Electronic Transactions on Numerical Analysis*, 54, 2020.

22 Kenneth L Clarkson and David P Woodruff. Low-rank approximation and regression in input sparsity time. *Journal of the ACM*, 63(6):1–45, 2017.

23 Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proc. 47th ACM Symposium on Theory of Computing*, pages 163–172, 2015. doi:10.1145/2746539.2746569.

24 Jan JM Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numerische Mathematik*, 36:177–195, 1980.

25 TJ Dekker and JF Traub. The shifted QR algorithm for Hermitian matrices. *Linear Algebra and its Applications*, 4(3):137–154, 1971.

26 James Demmel, Ioana Dumitriu, and Olga Holtz. Fast linear algebra is stable. *Numerische Mathematik*, 108(1):59–91, 2007. doi:10.1007/S00211-007-0114-X.

27 James Demmel, Ioana Dumitriu, Olga Holtz, and Robert Kleinberg. Fast matrix multiplication is stable. *Numerische Mathematik*, 106(2):199–224, 2007. doi:10.1007/S00211-007-0061-6.

28 James Demmel, Ioana Dumitriu, and Ryan Schneider. Fast and inverse-free algorithms for deflating subspaces. *arXiv*, 2024. arXiv:2310.00193.

29 James Demmel, Ioana Dumitriu, and Ryan Schneider. Generalized Pseudospectral Shattering and Inverse-Free Matrix Pencil Diagonalization. *Foundations of Computational Mathematics*, pages 1–77, 2024.

30 James Demmel and Krešimir Veselić. Jacobi's method is more accurate than QR. *SIAM Journal on Matrix Analysis and Applications*, 13(4):1204–1245, 1992. doi:10.1137/0613074.

31 James W Demmel. *Applied numerical linear algebra*. SIAM, 1997. doi:10.1137/1.9781611971446.

32 Inderjit Singh Dhillon. *A new $O(N^2)$ algorithm for the symmetric tridiagonal eigenvalue/eigenvector problem*. University of California, Berkeley, 1997.

33 Jack Dongarra and Francis Sullivan. Guest Editors Introduction to the top 10 algorithms. *Computing in Science & Engineering*, 2(01):22–23, 2000. doi:10.1109/MCISE.2000.814652.

34 Jack J Dongarra and Danny C Sorensen. A fully parallel algorithm for the symmetric eigenvalue problem. *SIAM Journal on Scientific and Statistical Computing*, 8(2):s139–s154, 1987.

35 Petros Drineas, Michael W Mahoney, and Shan Muthukrishnan. Relative-error CUR matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30(2):844–881, 2008. doi:10.1137/07070471X.

36 Jeff Erickson, Ivor Van Der Hoog, and Tillmann Miltzow. Smoothing the gap between NP and ER. *SIAM Journal on Computing*, 0(0):FOCS20–102–FOCS20–138, 2022.

37 John GF Francis. The QR transformation a unitary analogue to the LR transformation—Part 1. *The Computer Journal*, 4(3):265–271, 1961. doi:10.1093/COMJNL/4.3.265.

38 John GF Francis. The QR transformation—Part 2. *The Computer Journal*, 4(4):332–345, 1962.

39 Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. *Journal of the ACM*, 51(6):1025–1041, 2004. doi:10.1145/1039488.1039494.

40 Martin Fürer. Faster integer multiplication. In *Proc. 39th ACM Symposium on Theory of Computing*, pages 57–66, 2007. doi:10.1145/1250790.1250800.

41 Doron Gill and Eitan Tadmor. An $O(N^2)$ Method for Computing the Eigensystem of $N \times N$ Symmetric Tridiagonal Matrices by the Divide and Conquer Approach. *SIAM Journal on Scientific and Statistical Computing*, 11(1):161–173, 1990. doi:10.1137/0911010.

42 Gene Golub and William Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224, 1965.

43 Gene H Golub and Henk A Van der Vorst. Eigenvalue computation in the 20th century. *Journal of Computational and Applied Mathematics*, 123(1-2):35–65, 2000.

44 Gene H Golub and Charles F Van Loan. *Matrix Computations*. Johns Hopkins University Press, 2013.

45 Ming Gu and Stanley C Eisenstat. A stable and fast algorithm for updating the singular value decomposition, 1993.

46 Ming Gu and Stanley C Eisenstat. A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. *SIAM Journal on Matrix Analysis and Applications*, 16(1):172–191, 1995. doi:10.1137/S0895479892241287.

47 Ming Gu and Stanley C Eisenstat. Efficient algorithms for computing a strong rank-revealing qr factorization. *SIAM Journal on Scientific Computing*, 17(4):848–869, 1996. doi:10.1137/0917055.

48 Juris Hartmanis and Janos Simon. On the power of multiplication in random access machines. In *15th Annual Symposium on Switching and Automata Theory*, pages 13–23. IEEE, 1974. doi:10.1109/SWAT.1974.20.

49 David Harvey and Joris Van Der Hoeven. Integer multiplication in time $O(n \log n)$. *Annals of Mathematics*, 193(2):563–617, 2021.

50 Nicholas J Higham. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

51 Walter Hoffmann and Beresford N Parlett. A new proof of global convergence for the tridiagonal QL algorithm. *SIAM Journal on Numerical Analysis*, 15(5):929–937, 1978.

52 Alston S Householder. Unitary triangularization of a nonsymmetric matrix. *Journal of the ACM*, 5(4):339–342, 1958. doi:10.1145/320941.320947.

53 C.G.J. Jacobi. Über ein leichtes Verfahren die in der Theorie der Säcularstörungen vorkommenden Gleichungen numerisch aufzulösen. *Journal für die reine und angewandte Mathematik*, 30:51–94, 1846. URL: http://eudml.org/doc/147275.

54 Ian T Jolliffe. *Principal component analysis for special types of data*. Springer, 2002.

55 Praneeth Kacham and David P Woodruff. Faster Algorithms for Schatten-p Low Rank Approximation. In *Proc. Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.

56 Vera N Kublanovskaya. On some algorithms for the solution of the complete eigenvalue problem. *USSR Computational Mathematics and Mathematical Physics*, 1(3):637–657, 1962.

57 Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4), 1950.

58 Oren E Livne and Achi Brandt. N roots of the secular equation in $O(N)$ operations. *SIAM Journal on Matrix Analysis and Applications*, 24(2):439–453, 2002. doi:10.1137/S0895479801383695.

59 Anand Louis and Santosh S Vempala. Accelerated newton iteration for roots of black box polynomials. In *Proc. 57th IEEE Symposium on Foundations of Computer Science*, pages 732–740. IEEE, 2016. doi:10.1109/FOCS.2016.83.

60 RV Mises and Hilda Pollaczek-Geiringer. Praktische Verfahren der Gleichungsauflösung. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 9(1):58–77, 1929.

61 Cameron Musco, Christopher Musco, and Aaron Sidford. Stability of the Lanczos method for matrix function approximation. In *Proc. 29th ACM-SIAM Symposium on Discrete Algorithms*, pages 1605–1624. SIAM, 2018. doi:10.1137/1.9781611975031.105.

62 Yuji Nakatsukasa, Zhaojun Bai, and François Gygi. Optimizing Halley's iteration for computing the matrix polar decomposition. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2700–2720, 2010. doi:10.1137/090774999.

63 Yuji Nakatsukasa and Nicholas J Higham. Stable and efficient spectral divide and conquer algorithms for the symmetric eigenvalue decomposition and the SVD. *SIAM Journal on Scientific Computing*, 35(3):A1325–A1349, 2013. doi:10.1137/120876605.

64 Deanna Needell, William Swartworth, and David P Woodruff. Testing positive semidefiniteness using linear measurements. In *Proc. 2022 IEEE Symposium on Foundations of Computer Science*, pages 87–97. IEEE, 2022. doi:10.1109/FOCS54457.2022.00016.

65 Dianne P O'Leary and Gilbert W Stewart. Computing the eigenvalues and eigenvectors of symmetric arrowhead matrices. *Journal of Computational Physics*, 90(2):497–505, 1990.

66 Xiaofeng Ou and Jianlin Xia. Superdc: superfast divide-and-conquer eigenvalue decomposition with improved stability for rank-structured matrices. *SIAM Journal on Scientific Computing*, 44(5):A3041–A3066, 2022.

67 Chris C Paige. Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem. *Linear algebra and its Applications*, 34:235–258, 1980.

68 Victor Y Pan and Zhao Q Chen. The complexity of the matrix eigenproblem. In *Proc. 31st ACM Symposium on Theory of Computing*, pages 507–516, 1999. doi:10.1145/301250.301389.

69 Christos H Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. In *Proc. 17th ACM Symposium on Principles of Database Systems*, pages 159–168, 1998. doi:10.1145/275487.275505.

70 Beresford N Parlett. *The Symmetric Eigenvalue Problem*. SIAM, 1998.

71 Vladimir Rokhlin. Rapid solution of integral equations of classical potential theory. *Journal of computational physics*, 60(2):187–207, 1985.

72 H Rutishauser. On Jacobi rotation patterns. In *Proceedings of Symposia in Applied Mathematics*, volume 15, pages 219–239, 1963.

73 Yousef Saad. *Numerical methods for large eigenvalue problems: revised edition*. SIAM, 2011.

74 Ryan Schneider. *Pseudospectral divide-and-conquer for the generalized eigenvalue problem*. University of California, San Diego, 2024.

75 Arnold Schönhage. Unitäre transformationen großer matrizen. *Numerische Mathematik*, 20:409–417, 1972.

76 Arnold Schönhage. On the power of random access machines. In *Proc. International Colloquium on Automata, Languages, and Programming*, pages 520–529. Springer, 1979. doi:10.1007/3-540-09510-1_42.

77 Arnold Schönhage and Volker Strassen. Fast multiplication of large numbers. *Computing*, 7:281–292, 1971.

78 Rikhav Shah. Hermitian Diagonalization in Linear Precision. In *Proc. 2025 ACM-SIAM Symposium on Discrete Algorithms*, pages 5599–5615. SIAM, 2025. doi:10.1137/1.9781611978322.192.

79 Aleksandros Sobczyk. Deterministic complexity analysis of Hermitian eigenproblems. *arXiv*, 2025. doi:10.48550/arXiv.2410.21550.

80 Aleksandros Sobczyk, Marko Mladenovic, and Mathieu Luisier. Invariant subspaces and PCA in nearly matrix multiplication time. *Advances in Neural Information Processing Systems*, 37:19013–19086, 2024.

81 Nevena Jakovčević Stor, Ivan Slapničar, and Jesse L Barlow. Accurate eigenvalue decomposition of real symmetric arrowhead matrices and applications. *Linear Algebra and its Applications*, 464:62–89, 2015.

82 William Swartworth and David P Woodruff. Optimal eigenvalue approximation via sketching. In *Proc. 55th ACM Symposium on Theory of Computing*, pages 145–155, 2023. doi:10.1145/3564246.3585102.

83 James Vogel, Jianlin Xia, Stephen Cauley, and Venkataramanan Balakrishnan. Superfast divide-and-conquer method and perturbation analysis for structured eigenvalue solutions. *SIAM Journal on Scientific Computing*, 38(3):A1358–A1382, 2016. doi:10.1137/15M1018812.

84 James Hardy Wilkinson. Global convergene of tridiagonal QR algorithm with origin shifts. *Linear Algebra and its Applications*, 1(3):409–420, 1968.