Fast Hermitian Diagonalization with Nearly Optimal Precision

Rikhav Shah

June 2024

Abstract

Algorithms for numerical tasks in finite precision simultaneously seek to minimize the number of floating point operations performed, and also the number of bits of precision required by each floating point operation. This paper presents an algorithm for Hermitian diagonalization requiring only $\lg(1/\varepsilon) + O(\log(n) + \log\log(1/\varepsilon))$ bits of precision where n is the size of the input matrix and ε is the target error. Furthermore, it runs in near matrix multiplication time.

In the general setting, the first complete analysis of the stability of a near matrix multiplication time algorithm for diagonalization is that of Banks et al [BGVKS20]. They exhibit an algorithm for diagonalizing an arbitrary matrix up to ε backward error using only $O(\log^4(n/\varepsilon)\log(n))$ bits of precision. This work focuses on the Hermitian setting, where we determine a dramatically improved bound on the number of bits needed. In particular, the result is close to providing a practical bound. The exact bit count depends on the specific implementation of matrix multiplication and QR decomposition one wishes to use, but if one uses suitable $O(n^3)$-time implementations, then for $\varepsilon = 10^{-15}, n = 4000$, we show 92 bits of precision suffice (and 59 are necessary). By comparison, the same parameters in [BGVKS20] does not even show that $682,916,525,000$ bits suffice.

1 Introduction

This paper considers Hermitian diagonalization in finite arithmetic. Given a Hermitian matrix A and target accuracy ε, our goal is to compute nearly unitary U and exactly diagonal D such that $\|A - UDU^*\| \leq \varepsilon \|A\|$ with high probability. An algorithm for such a task can be evaluated on two primary metrics: the number of floating point operations performed (which we call the runtime) and the number of bits of precision required for each floating point operation.

Despite the widespread, highly successful use of procedures for this task in practice, the literature long lacked precise guarantees of their performance in finite arithmetic. It's common to treat n and ε as formal variables satisfying the relations $\text{poly}(n) \cdot \varepsilon = \varepsilon$ and $\varepsilon^2 = 0$. This dramatically reduces the complexity of proofs as one only needs to keep track of the existence of a single error term, but introduces the possibility of subtle mistakes; namely one must be careful that the coefficients appearing in the $\text{poly}(n)$ factor are absolute constants, and one also cannot apply either of those relations more than $\text{poly}(n)$ times in the proof. This approach also precludes the determination of precise, quantitative bounds on stability. In general, the bound numerical analysts typically aim for is showing that $\lg(1/\varepsilon) + O(\log n)$ bits suffice, or stated another way, that backward error is $\text{poly}(n)\mathbf{u}$ where $\mathbf{u}^{-\#\text{ bits of precision}}$ is machine precision. For Hermitian diagonalization in particular, Proposition 1.1 shows that no fewer than $\lg(1/\varepsilon) + 0.5\lg(n) - 2$ bits suffice.

Most algorithms used and analyzed fall into two categories: QL/QR-algorithms first introduced by Francis [Fra61, Fra62] and spectral divide-and-conquer algorithms first introduced by Beavers and Denman [BD73, BD74, DB76]. These algorithms, as is thematic in this field, are frequently built on top of three essential primitives: matrix multiplication, QR decomposition, and matrix inversion. And so, the stability and runtime of algorithms depend on the stability and runtime of the implementations of these primitives. There is a trade-off among the best algorithms between runtime and stability. On the slow side, implementations of each of these primitives exist using $O(n^3)$ time requiring only $\lg(1/\varepsilon) + \lg(n) + O(1)$ bits to achieve ε backward error. Faster implementations use the innovations of [DDHK07, DDH07] which show for each

$\eta > 0$ an implementation of matrix multiplication and QR decomposition using $O(n^{\omega+\eta})$ time requiring only $\lg(1/\varepsilon)+O(\log(n))$ bits, where $O(n^\omega)$ is the speed of matrix multiplication in exact arithmetic. It also shows an implementation of inversion with the same runtime using $\lg(1/\varepsilon) + O(\log(n)\log(\kappa(A)))$ bits to achieve a *forward* error of ε.

Unfortunately, all published QL/QR-algorithms use $O(n^3)$ time as they require reduction of a matrix to tridiagonal or Hessenberg form. In exact arithmetic, the first globally convergent QR-algorithm for Hermitian matrices was proposed by [Wil68]. Both [DT71, HP78] bound the rate of convergence of that algorithm, resulting in an $O(n^3 + n^2\log(1/\varepsilon))$ time algorithm. [BGVS23] proposed a more expensive version that globally converges in the non-Hermitian setting as well. The follow-up work [BGVS22] analyzed it in finite precision, finding that it uses $\widetilde{O}(n^3)$ operations performed with $O(\log(n/\varepsilon)^2\log\log(n/\varepsilon))$ bits of precision plus $\widetilde{O}(n)$ operations performed with $O(\log(n/\varepsilon)^4\log\log(n/\varepsilon)^2)$ bits of precision. They also conjecture that $O(\log(n/\varepsilon))$ bits suffice in the Hermitian setting.

Spectral divide-and-conquer algorithms better exploit fast primitives. Many works can be adapted to form part of the "divide" step. In particular, computing the matrix sign or polar decomposition can both be used in the Hermitian case. A history up to 1995 of algorithms computing the matrix sign can be found in Section 1 of [KL95]. Since then, multiple works have appeared, typically assuming the input matrix is reasonably well conditioned[1]. We highlight three algorithms for matrix sign / polar decomposition that have been explicitly incorporated into diagonalization procedures.

Newton iteration: this method involves matrix inversion, and so the overall stability guarantee is substantially worse if one wants a near matrix multiplication time algorithm. Assuming access to stable inversion, [KZ03] shows a scaled version of Newton Iteration converges stably. [BX08] gives a much shorter argument to the same effect, but [KZ09] points out a flaw in the proof arising from the "$\text{poly}(n)\cdot\varepsilon = \varepsilon$" framework. [NH12] provides a much more compact proof in a way that generalizes to several other algorithms. Finally, [BGVKS20] provides a complete, unconditional analysis of Newton iteration using fast inversion, and furthermore incorporates that method into a full end-to-end analysis of a divide-and-conquer algorithm. They show that their diagonalization algorithm runs in near matrix multiplication time and succeeds when using $O(\log^4(n/\varepsilon)\log(n))$ bits of precision. This was the first concrete bound appearing for any algorithm for diagonalization, and remains the best known bound among algorithms running in near-matrix multiplication time.

QR-based dynamically weighted Halley (QDWH): [NBG10] finds a quickly-converging iterative scheme, QDWH, for computing the matrix sign built on top of QR decomposition. Unfortunately, the proof of stability of QDWH appearing in [NH12] (indeed the only known proof) requires a stronger notion of stability of QR decomposition than what [DDH07] shows can be achieved quickly[2]. As a consequence, we only know how to stably perform QDWH in $O(n^3)$ time. [NH13] shows how to build a Hermitian eigen-decomposition on top of QDWH, following the usual divide-and-conquer setup. However, the proof of its stability is incomplete, with at least a couple limitations. The first limitation is stated explicitly as condition number 2. A crucial step of divide-and-conquer is computing a basis for the range of a projection matrix. [NH13] assumes this can be done stably; this paper finally provides that required analysis in Section 3. The second limitation is choice of the "split points". [NH13] recommends a couple different techniques for picking the split points, which determine the size of the sub-problems. But the worst case guarantee is that a problem of size n gets split into problems of size 1 and $n-1$. This means the overall method may take $O(n^4)$ time.

Implicit repeated squaring (IRS): [BDD11] improves upon the IRS method of [BDG97] using [DDH07] thereby giving a stable algorithm for computing an analog[3] of the matrix sign in near matrix multiplication time. [BDD11] also provides finite-arithmetic analysis of the full "divide" step; namely it tightens the analysis of rank-revealing URV appearing in [DDH07] to show how to convert the approximate spectral projectors into approximate bases of the invariant subspaces. However, they do not explicitly bound the error of the

[1] This assumption is satisfied with high probability by adding a random shift to the matrix.

[2] [NH12] requires a "per-row" backward error guarantee. That is, A, A' appearing in the leftmost inequality of (3) must be replaced by e_j^*A, e_j^*A' for every $j \in [n]$. The difference in these guarantees is the most stark when the lengths of the rows of A are quite different.

[3] Let $P_{(*)}$ denote the projection onto the span of the eigenvectors whose eigenvalues satisfy $(*)$. Then the matrix sign is $P_{(\cdot)>0} - P_{(\cdot)<0}$. [BDD11, BDG97] compute $P_{|\cdot|>1} - P_{|\cdot|<1}$

computed bases nor the accumulation of error throughout the entire divide-and-conquer algorithm.

The work [SML24] extends a key subroutine of [BGVKS20] to the Hermitian pencil case, but does not reduce the precision required. Another algorithm of note is that of [ABB+18]. They consider the general eigenpair problem and find an algorithm using homotopy continuation and running in $O(n^{10}/\varepsilon^2)$ in general [4]. They provide only an informal argument that their method is stable.

1.1 Contributions

The main result of this paper is Theorem 4.2. It presents an algorithm for Hermitian diagonalization running in near matrix multiplication time and shows it requires only $\lg(1/\varepsilon) + O(\log(n) + \log\log(1/\varepsilon))$ bits of precision—near the optimal dependence on ε and linear in the optimal dependence on n. This paper offers several contributions.

Proposition 1.1 states a concrete lower bound on required bits of precision; in particular, it shows the existence of two matrices with far apart true diagonalizations that would get rounded to the same matrix without at least $\lg(1/\varepsilon) + 0.5\lg(n) - 2$ bits of precision.

Section 2 provides a rigorous quantitative analysis of the stability of Newton-Schulz iteration for computing the matrix sign function in the Hermitian setting. This iteration has been considered [BD93, NH12] but quantitative bounds on it's stability have not appeared. We show it to be significantly more stable than Newton iteration for matrix sign, which is used by [BGVKS20], though it only succeeds in the Hermitian setting. Section 2.1 discusses this difference in stability in depth—the improvements are owed both to the orthgonality of eigenvectors in this setting and to the fact Newton-Schulz is inverse-free. This analysis enables the algorithmic improvement of replacing Newton with Newton-Schulz, resulting in a significant reduction in the precision's dependence on ε—namely $2 \cdot \lg(1/\varepsilon)$ compared to $O(\lg(1/\varepsilon)^4)$ of [BGVKS20].

The next contribution, in Section 3, is a streamlined version of the deflation algorithm appearing in [BGVKS20], along with a much tighter analysis. This strengthened analysis reduces the bit dependence on ε from $2 \cdot \lg(1/\varepsilon)$ to $1 \cdot \lg(1/\varepsilon)$ and saves several $\lg(n)$ terms, and therefore brings us to near-optimal precision.

Finally, in Section 4 we highlight two additional differences between our spectral bisection method and that of [BGVKS20]. The first difference—discussed in Remark 8—is an adaptive setting of parameters used by spectral bisection. The most straightforward approach, used by [BGVKS20], geometrically decreases ε in recursive calls. But smaller ε leads to both a longer runtime and a higher bit requirement. We manage to decrease ε more slowly. The key to enabling this is tracking the depth of recursion and decreasing ε depending on the current depth. This again significantly reduces our bit requirement. The second difference—discussed in Remark 9—is the elimination of spectral shattering, which played a central role in the algorithm of [BGVKS20]. Instead of shattering, we implement the suggestions in [BDD11]: picking shifts randomly (as opposed to using binary search to find a good split point) and adding an additional base case (rather than just the $n = 1$ case).

The rest of the introduction is dedicated to background and preliminaties. Section 2 addresses the primary bottleneck of [BGVKS20], which is the computation of the matrix sign function. Section 3 provides stronger analysis of deflate which allows us to achieve near-optimal dependence on ε. Section 4 puts these pieces together into a spectral bisection method for Hermitian diagonalization.

1.2 Model of computation

We adopt the standard model of floating point arithmetic. Numbers which are stored exactly are called floating point numbers. For each $z \in \mathbb{C}$, there exists a floating point number $\texttt{fl}(z)$ satisfying

$$|\texttt{fl}(z) - z| \leq \mathbf{u}|z|.$$

For each operation $\circ \in \{+, -, \times\}$ and pair of floating point numbers x, y, the result of computing $x \circ y$ in floating point arithmetic yields

$$|\texttt{fl}(x \circ y) - x \circ y| \leq \mathbf{u}|x \circ y|. \tag{1}$$

[4]The result is stated as n^9/σ^2 where σ is the standard deviation of an normalized Gaussian perturbation, which must be order $\varepsilon/\sqrt{n}$ for the desired error.

Additionally, division by two can be done exactly. That is, $\texttt{fl}(x/2) = \texttt{fl}(x)/2$. All nonzero values used in our algorithm are polynomial in n, ε^{-1}, so we can avoid underflow and overflow errors by using $\lg\lg(n/\varepsilon) + O(1)$ bits to store the exponent.

Remark 1 (Double-double precision)**.** One may simulate precision $\mathbf{u}^2$ by approximating a real (or complex) x by the sum $y + z$ where $y = \texttt{fl}(x)$ and $z = \texttt{fl}(x - y)$. This suggests a natural trade-off between precision and runtime: by increasing the number of floating point operations by a constant factor, one can get away with a constant factor reduction in the number of bits used. Though theoretically sound, this technique has serious drawbacks in practice. Namely, hardware implementations of various matrix operations assume that each entry of an input fits into a single machine word. That is, each entry of an input matrix is a single floating point number and isn't abstractly expressed as the sum of two values. Because of this, we build our algorithm on top of subroutines that don't allow inputs to be formatted like this. Throughout this paper, every real or complex number is approximated by a single floating point number. With this constraint, it is possible to obtain a lower bound on the precision required; see Proposition 1.1.

Proposition 1.1 (Lower bound)**.** *Any method that computes U, D for a given symmetric A satisfying $\|A - UDU^*\| \le \varepsilon \|A\|$ requires $\lg(1/\mathbf{u}) \ge \lg(1/\varepsilon) + 0.5\lg(n) - 2$ bits of precision.*

Proof. Let A be an $n \times n$ Hadamard matrix, i.e. $|a_{ij}| = 1$ and $\|A\| = \sqrt{n}$. Let (U, D) be the result of diagonalizing A. Consider the number of positive versus negative entries of the residual $A - UDU^*$. If there are more positive entries, set B to be the all ones matrix, otherwise set it to be the all minus one matrix. Note $A' := A + (\mathbf{u}/2)B$ will be stored exactly as A in floating point arithmetic, so the result of diagonalizing A' will again be (U, D). Thus the residual of diagonizing A' is $A' - UDU^* = A - UDU^* + (\mathbf{u}/2)B$. But by construction of B, at least half the entries of this residual have absolute value at least $\mathbf{u}/2$. In particular, it's norm is at least $\mathbf{u}n/4$, which we desire to be at most $\sqrt{n}\varepsilon$. So we need $\mathbf{u} \le 4\varepsilon/\sqrt{n}$. □

1.3 Subroutines

The algorithm of this paper is built on top of several essential primitives. We assume black-box access to the following four methods.

Definition 1.1 (From [DDHK07])**.** $\texttt{MM}$ is an algorithm for matrix multiplication using $T_{\texttt{MM}}(n)$ floating point operations satisfying

$$\|\texttt{MM}(A,\, B) - AB\| \le \mu_{\texttt{MM}}(n)\mathbf{u} \cdot \|A\| \cdot \|B\| \,. \tag{2}$$

$\mu_{\texttt{MM}}(n)$ is a low-degree polynomial in n. If $A = B^T$, then $\texttt{MM}(A, B)$ will be exactly Hermitian. For simplicity of many bounds we assume $\mu_{\texttt{MM}}(n) \ge 10$. We also assume $T_{\texttt{MM}}(n)$ is convex.

Definition 1.2 (From [DDH07])**.** $[Q, R] = \texttt{QR}(A)$ is an algorithm for matrix multiplication using $T_{\texttt{QR}}(n)$ floating point operations satisfying for some matrix A' and unitary matrix Q',

$$(Q')^* A' = R \text{ is upper triangular} \quad \& \quad \|Q - Q'\| \le \mu_{\texttt{QR}}(n)\mathbf{u} \quad \& \quad \|A - A'\| \le \mu_{\texttt{QR}}(n)\mathbf{u} \cdot \|A\| \,. \tag{3}$$

$\mu_{\texttt{QR}}(n)$ is a low-degree polynomial in n. We assume $T_{\texttt{QR}}(n)$ is convex.

Definition 1.3. $\texttt{Unif}$ is a method for computing approximately uniform random samples from symmetric intervals. It should use only a constant $T_{\texttt{Unif}}$ number of operations and satisfy the following. If c' is a random variable distributed uniformly on the real interval $[-s, s]$, there exists a coupling of $\texttt{Unif}(s)$ and c' such that $\texttt{Unif}(s) \in [-s, s]$ & $|\texttt{Unif}(s) - c'| \le s\mathbf{u}$ with probability 1.

Definition 1.4. $\texttt{Normal}$ is a method for computing approximately normal random samples. It should use only a constant $T_{\texttt{N}}$ number of operations. Let z be a complex Gaussian where $\mathrm{Re}(z)$ and $\mathrm{Im}(z)$ are i.i.d. Gaussian samples with mean 0 and variance 1/2. Then for a constant $c_{\texttt{N}}$, there exists a coupling of $\texttt{Normal}()$ and z such that $|\texttt{Normal}() - z| \le |z| c_{\texttt{N}}\mathbf{u}$ with probability 1.

2 Matrix sign function

2.1 Insufficiency of Newton iteration

The bottleneck in the algorithm of [BGVKS20] is in the estimation of the matrix sign function. The iterative scheme used is the same one initially proposed by [BD73]:

$$A_0 = A \quad \& \quad A_{k+1} = \frac{A_k + A_k^{-1}}{2}. \tag{4}$$

In exact arithmetic, this is equivalent to running Newton iteration for the system $\lambda^2 - 1 = 0$, which enjoys quadratic convergence to ± 1 everywhere except for $\mathrm{Re}(\lambda) = 0$. So for matrices with no eigenvalues on the imaginary axis, we have $A_k \to \mathrm{sign}(A)$ as $k \to \infty$. In finite arithmetic, a crucial step in the proof of convergence is showing that the pseudospectrum of A_k does not grow too quickly with k. This becomes more and more difficult the closer the eigenvalues of A_k get to each other, so represents an obstacle for any iterative scheme, not just (4). In the bounds of [BGVKS20], the bit complexity required to handle this growth depends *exponentially* in the number of iterations used. They show at most

$$\lg(1/(1-\alpha_0)) + 3\lg\lg(1/(1-\alpha_0)) + \lg\lg((1/\beta\varepsilon_0)) + O(1)$$

iterations are needed, where for the purposes of this discussion you can think of $1/(1-\alpha), \beta, \varepsilon_0$ as all being polynomials in n, ε^{-1}. Each $1 \cdot \lg\lg(x)$ term contributes another *factor* of $\lg(x)$ to the bit complexity. But since these are not the dominate terms in the expression, shaving off the logs from the bit complexity of this algorithm, if possible, requires much care. The Hermitian setting completely sidesteps this issue since the pseudospectra of Hermitian matrices are always well-behaved.

The secondary source of instability relates to this particular iterative scheme. Specifically, to the computation of A_k^{-1}. If one had a "backward-stable" algorithm `INV` for computing inverses such that for every A there existed a small E satisfying $\texttt{INV}(A) = (A + \|A\| E)^{-1}$, some algebra reveals that one would also obtain the "forward error" bound

$$\left\|A^{-1} - \texttt{INV}(A)\right\| \leq O(\|E\|) \cdot \kappa(A).$$

Unfortunately, no such algorithm running in near matrix multiplication time is known. The closest we have is the work of [DDH07], which finds a *logarithmically stable* algorithm, i.e. one satisfying the weaker guarantee that

$$\left\|A^{-1} - \texttt{INV}(A)\right\| \leq O(\|E\|) \cdot \kappa(A)^{O(\log n)}.$$

So [BGVKS20] must perform everything with enough precision to promise that $\|E\| \ll \kappa(A)^{-O(\log n)}$. This necessitates another factor of $O(\log \kappa(A) \cdot \log(n))$ bits of precision. For this reason, an 'inverse-free' method that does not use inversion (even well-conditioned inversion) is desirable. This motivates the iterative scheme

$$A_0 = A \quad \& \quad A_{k+1} = \frac{3A_k - A_k^3}{2} = \frac{1}{2} \cdot A_k \cdot (3I - A_k^2) \tag{5}$$

corresponding to Newton-Schultz iteration, which in this instance is equivalent to Newton iteration for the system $\lambda^{-2} - 1 = 0$. This system has been considered [BD93, NH12] but quantitative bounds on it's stability have not appeared. Each iteration uses only two calls to a method for multiplying matrices and no other expensive primitives. This method is insufficient for the non-Hermitian problem as it does not converge for many complex starting points z; for instance it does not converge when $|\mathrm{Im}(z)| \geq 2|\mathrm{Re}(z)|$. Figure 2.1 shows the regions of convergence for the two methods. Fortunately, it converges on the interval $(-\sqrt{5}, \sqrt{5})$ in the real line, which is enough for Hermitian diagonalization.

Remark 2 (Polar decomposition)**.** For Hermitian matrices, computing the polar decomposition recovers the matrix sign since $A = \mathrm{sign}(A)(A^*A)^{1/2}$ is the unique factorization of A into the product of a unitary matrix $\mathrm{sign}(A)$ and a positive definite matrix $(A^*A)^{1/2}$.

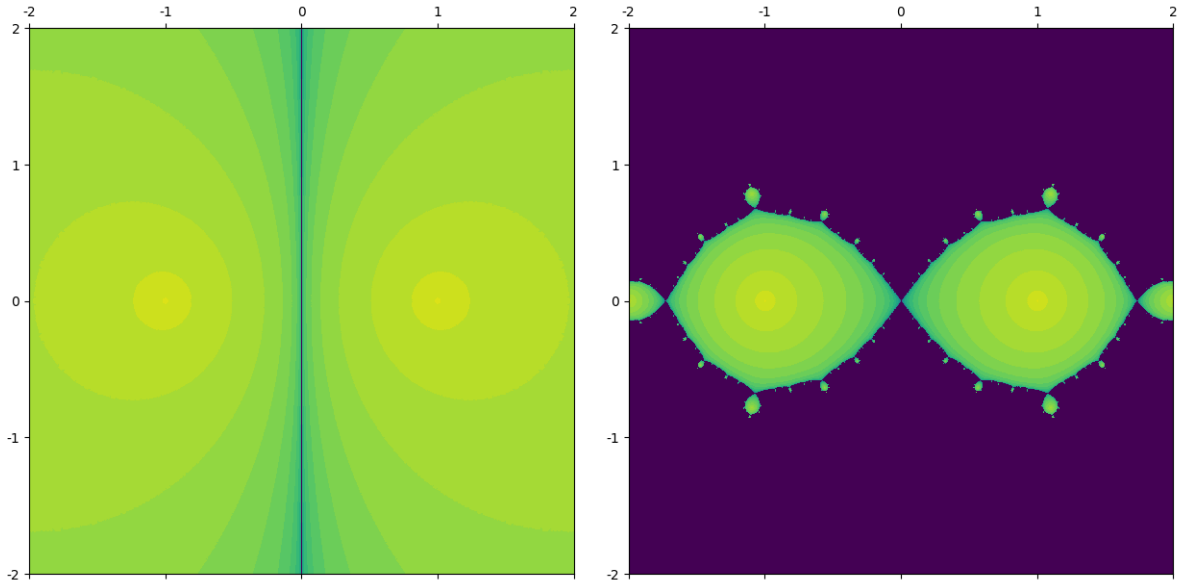

Figure 1: Convergence plots for Newton iteration (left) and Newton-Schulz iteration (right). The color denotes the number of iterations k until $\left|z_k^2 - 1\right| < 10^{-15}$. The lightest shade of yellow is one iteration and each ring denotes one additional iteration. Purple denotes "does not converge".

Remark 3 (Faster iterative algorithms)**.** As we will see later, the number of iterations required for this method to converge depends on the condition number of the input matrix. This paper is mostly concerned with the precision required, which in the Hermitian setting depends only mildly on the number of iterations required. Additionally, random shifts are employed to ensure the condition number is never too large. So analysis of faster iterative schemes is left to future work. Nevertheless it's worth discussing some candidates. QDWH of Nakatsukasa and Higham [NBG10] for polar decomposition was shown to be stable in [NH12], and experimentally was found to converge in just 6 iterations for $\varepsilon = 10^{-15}$ regardless of n and κ. However, the proof of stability requires QR-decomposition with a *per-row* backward error. That is, each row of the input matrix may receive a multiplicative norm-wise perturbation, irrespective of the norms of the other rows. This can be accomplished via Householder reflections with pivoting in $O(n^3)$ time, but no reduction to matrix multiplication is known. Their scheme is a 'weighted' version of $A_{k+1} = (3A_k + A_k^3)(1 + 3A_k^2)^{-1}$ which adaptively changes the coefficients of the iterates as the method is run. An even higher degree rational function is used by the method Zolo-pd of Nakatsukasa and Freund, which experimentally converges in just 2 iterations [NF16], though no proof of stability is known. Several more morally similar variants of that and (4), (5) can be found in [KL95, CCNS14, NBG10, Gan90]. These methods frequently require some additional knowledge about the matrix (e.g. QDWH) requires a crude upper estimate on the condition number) which can typically be computed in matrix multiplication time. These methods present good candidates for replacing (5), but require complete analysis of stability and runtime in finite precision.

2.2 Analysis of Newton-Schulz iteration

As motivated in the previous subsection, we use Newton-Schulz iteration to approximate the matrix sign function. Fix

$$g(x) = \frac{3x - x^3}{2} = \frac{1}{2} \cdot x \cdot (3 - x^2)$$

and implement it using MM by

$$\mathsf{g}(A) = \frac{1}{2}\mathtt{MM}(A, (3I - \mathtt{MM}(A, A))). \tag{6}$$

In exact arithmetic, by the functional calculus applying (5) to a Hermitian matrix is equivalent to applying (5) to it's eigenvalues. We need to argue that this equivalence does not break too much in the presence of round-off error. We also need to analyze the convergence of (5) in the presence of error when applied to scalars. The next two lemmas accomplish the first task. We start by bounding the forward error of computing a single iteration.

Lemma 2.1 (One step error bound)**.** *Assume* $\mathbf{u} \leq \min(1/3, \mu_{\mathtt{MM}}(n)^{-1})$*. Then* g *specified in (6) satisfies*

$$\|\mathsf{g}(A) - g(A)\| \leq \mu_{\mathsf{g}}(n, \|A\|)\mathbf{u}$$

for

$$\mu_{\mathsf{g}}(n, a) := \frac{1}{2}\big(7 + (6 + \mu_{\mathtt{MM}}(n))a^2\big)a.$$

Furthermore, such an implementation requires only

$$T_{\mathsf{g}}(n) = 2T_{\mathtt{MM}}(n) + n^2 + n$$

floating point operations.

Proof. We can numerically compute g as In this proof, E_k denote matrices with $\|E_k\| \leq \mathbf{u}$. Let $\mathtt{MM}(A, A) = A^2 + \|A\|^2 \mu_{\mathtt{MM}}(n)E_1$. Let B be the result of numerically subtracting $\mathtt{MM}(A, A)$ from $3I$. This subtraction incurs an entry-wise multiplicative $(1 + \mathbf{u})$ error along the diagonal only. In particular, because the error is diagonal the entry-wise absolute value bound is upgraded to a matrix norm bound for free. In particular,

$$\begin{aligned} B &= 3I - \mathtt{MM}(A, A) + \|3I - \mathtt{MM}(A, A)\| E_2 \\ &= 3I - A^2 - \|A\|^2 \mu_{\mathtt{MM}}(n)E_1 + \left\|\left(3I - A^2 - \|A\|^2 \mu_{\mathtt{MM}}(n)E_1\right)\right\| E_2 \end{aligned}$$

so the forward error of computing B is at most

$$\|A\|^2 \mu_{\mathtt{MM}}(n)\mathbf{u} + \left(3 + \|A\|^2 + \|A\|^2 \mu_{\mathtt{MM}}(n)\mathbf{u}\right)\mathbf{u} \leq \left(3 + (2 + \mu_{\mathtt{MM}}(n)) \|A\|^2\right)\mathbf{u}$$

Let $\mu_B = 3 + (2 + \mu_{\mathtt{MM}}(n)) \|A\|^2$ so $B = 3I - A^2 + \mu_B E_3$. Then

$$\begin{aligned} \mathtt{MM}(A, B) - 2g(A) &= AB - 2g(A) + \|A\| \|B\| E_4 \\ &= \mu_B A E_3 + \|A\| \|B\| E_4. \end{aligned}$$

Finally, division by 2 can be done exactly by decrementing the exponent. So the forward error of computing $g(A)$ is at most

$$\begin{aligned} \frac{1}{2}\left(\mu_B \|A\| \mathbf{u} + \|A\| \left(3 + \|A\|^2 + \mu_B \mathbf{u}\right)\mathbf{u}\right) &= \frac{1}{2}\left(\mu_B + 3 + \|A\|^2 + \mu_B \mathbf{u}\right) \|A\| \mathbf{u} \\ &= \frac{1}{2}\left(7 + (3 + \mu_{\mathtt{MM}}(n)) \|A\|^2 + (2 + \mu_{\mathtt{MM}}(n)) \|A\|^2 \mathbf{u}\right) \|A\| \mathbf{u} \\ &= \frac{1}{2}\left(7 + (6 + \mu_{\mathtt{MM}}(n)) \|A\|^2\right) \|A\| \mathbf{u} \end{aligned}$$

as required.

□

Remark 4. The work [NH12] considers an alternate implementation of $\mathtt{g}$, namely

$$\mathtt{g}(A) = \frac{3A - \mathtt{MM}(A, \mathtt{MM}(A, A))}{2}.$$

It shows that this implementation is stable, but only gives a qualitative analysis. In particular, as far as this author can tell, this implementation suffers from an extra factor of n in the error bound. This comes since the addition is dense and therefore incurs $\mathbf{u}$ error in every entry, as opposed to merely the diagonal as in the implementation (6).

The next lemma shows adding noise does not change the value of sign by too much.

Lemma 2.2. *Let A, B, ε be such that 0 is not in the interior of $\Lambda_\varepsilon(A)$ and $\|A - B\| < \varepsilon$. Then*

$$\|\mathrm{sign}(A) - \mathrm{sign}(B)\| \le n \cdot \frac{\|A - B\|}{\varepsilon - \|A - B\|}.$$

Proof. Let γ be the boundary of a connected component of $\Lambda_\varepsilon(A)$. Say it encloses k eigenvalues. Then

$$\begin{aligned}
\left\| \frac{1}{2\pi} \oint_\gamma \left[(z - A)^{-1} - (z - B)^{-1} \right] \right\| &= \left\| \frac{1}{2\pi} \oint_\gamma \left[(z - B)^{-1} (A - B)(z - A)^{-1} \right] \right\| \\
&\le \frac{1}{2\pi} \cdot \mathrm{length}(\gamma) \cdot \|A - B\| \cdot \frac{1}{\varepsilon - \|A - B\|} \cdot \frac{1}{\varepsilon} \\
&\le k \cdot \|A - B\| \cdot \frac{1}{\varepsilon - \|A - B\|}.
\end{aligned}$$

By the assumption, none of these components cross the imaginary axis. So we may sum over the appropriate components to form the spectral projector onto all the positive or negative eigenvalues. This gives the desired bound. □

We now tackle converge for scalars. One notices that $x = -1, 0, 1$ are fixed points of g and that $g'(-1) = g'(1) = 0$, which is needed for quadratic convergence. Unfortunately, the iterations do not converge to ± 1 for $x = 0$ (where it stays at 0) and for $x \ge \sqrt{5}$ (where it does not converge at all). Additionally, which fixed point it converges to for $x \in \pm[\sqrt{3}, \sqrt{5})$ is difficult to control. We start by showing monotone convergence, even in the presence of error.

Lemma 2.3 (Monotone convergence)**.** *Fix $u \in (0, 3/16]$ and $|\xi| \le u$. Then*

$$\begin{aligned}
\mathrm{sign}(x) &= \mathrm{sign}(g(x) + \xi) && \forall x \in \pm\Big(u,\ \sqrt{3} - (\sqrt{3} - 1)u\Big), \\
|x| &\le |g(x) + \xi| \le 1 + u && \forall x \in [(8/3)u,\ 1 - (8/3)u].
\end{aligned}$$

Proof. We prove the statements for $x \ge 0$ and the results for $x \le 0$ follow symmetrically by noting that g is odd. $g(x)$ is concave on $[0,\ \sqrt{3}]$ so is lower bounded by its linear spline with nodes $[0,\ 1,\ \sqrt{3}]$. Namely,

$$g(x) \ge \begin{cases} x & 0 \le x \le 1 \\ \sqrt{3}/(\sqrt{3} - 1) - x/(\sqrt{3} - 1) & 1 \le x \le \sqrt{3} \end{cases}$$

which is larger than u for $x \in (u,\ \sqrt{3} - (\sqrt{3} - 1)u)$. This implies $\mathrm{sign}(g(x) + \xi) = 1$ establishing the first claim. Now note the polynomial $g(x) - x = (x - x^3)/2$ is concave on the region $[0, 1]$ and so is lower bounded by its linear spline with nodes $[0,\ 1/2,\ 1]$. Namely,

$$g(x) \ge \begin{cases} (3/8)x & 0 \le x \le 1/2 \\ (3/8)(1 - x) & 1/2 \le x \le 1 \end{cases}$$

which is at least u for $x \in [(8/3)u,\ 1 - (8/3)u]$. This implies $g(x) - u \ge x$ establishing the second claim. Finally by the triangle inequality $|g(x) + \xi| \le |g(x)| + |\xi| \le 1 + u$. □

In exact arithmetic, this iteration enjoys quadratic convergence. In finite precision, we have quadratic convergence for a large range of values. We will analyze the convergence of the iterations using the potential function $m(x) = \left|1 - x^2\right|$.

Lemma 2.4 (Quadratic convergence)**.** *When* $20|\xi| \le |x| \le 1 - \sqrt{10|\xi|}$*, one has*

$$m(x)^2 \ge m(g(x) + \xi).$$

When $|x| \le \sqrt{2}$ *and* $|\xi| \le 1$*, one has*

$$m(x)^2 + 4|\xi| \ge m(g(x) + \xi).$$

Proof. Note g is odd and m is even, and that the condition on ξ is symmetric in x. So it suffices to prove the statement for $x \ge 0$. Fix any $s \in [-1, 1]$. Simple algebraic manipulations reveal that

$$\begin{aligned}
|x| \le 0.5 &\implies |x| \le \sqrt{\frac{180 - \sqrt{180^2 - 4 \cdot 100 \cdot 41}}{200}} \\
&\implies 41 - 180x^2 + 100x^4 \ge 0 \\
&\implies (30 - 10x^2 - 1)^2 \ge 20^2 \cdot (2 - x^2) \\
&\implies 30 - 10x^2 - 1 \ge 20\sqrt{2 - x^2} \\
&\implies \frac{3 - x^2}{2} - \sqrt{2 - x^2} \ge s/20 \\
&\implies g(x) - \sqrt{2x^2 - x^4} \ge s/20 \\
&\implies g(x) - sx/20 \ge \sqrt{2x^2 - x^4} \\
&\implies (g(x) - sx/20)^2 - 1 \ge 2x^2 - x^4 - 1 \\
&\implies \left|(g(x) - sx/20)^2 - 1\right| \le \left|1 - x^2\right|^2 \\
&\implies m(g(x) - sx/20) \le m(x)^2.
\end{aligned}$$

By the Descartes rule of signs, the polynomial $25x^4 + 60x^3 + 26x^2 - 32x + 1$ has at most two positive roots, and changes sign at the points $0, 1/4, 1/2$. So the polynomial is positive for $x > 1/2$. In the below, assume $x \le \sqrt{2}$.

$$\begin{aligned}
x \ge 0.5 &\implies (x - 1)^2\left(25x^4 + 60x^3 + 26x^2 - 32x + 1\right) \ge 0 \\
&\implies \left(15x - 5x^3 - (x - 1)^2\right)^2 \ge 100x^2\left(2 - x^2\right) \\
&\implies 15x - 5x^3 - (x - 1)^2 \ge 10x\sqrt{2 - x^2} \\
&\implies g(x) - \sqrt{2x^2 - x^4} \ge (x - 1)^2/10 \\
&\implies (g(x) - s(x - 1)^2/10)^2 \ge 2x^2 - x^4 \\
&\implies (g(x) - s(x - 1)^2/10)^2 - 1 \ge -1 + 2x^2 - x^4 \\
&\implies m(g(x) - s(x - 1)^2/10) \le m(x)^2.
\end{aligned}$$

Note

$$\min\left(x/20,\, (x - 1)^2/10\right) = \begin{cases} x/20 & x \le 0.5 \\ (x - 1)^2/10 & x \ge 0.5 \end{cases},$$

so whenever $|\xi| \le \min\left(x/20,\, (x - 1)^2/10\right)$ for $x \in [0, \sqrt{2}]$ we have $m(g(x) + \xi) \le m(x)^2$. Solving for x in terms of $|\xi|$ yields the desired result. One may take $\xi = 0$ for any x so $m(g(x)) \le m(x)^2$ for all $|x| \le \sqrt{2}$. The derivative of m is bounded by 4 on the interval $[-2, 2]$, and $g(x) + \xi \in [-1, 1]$ for $|x| \le \sqrt{2}, |\xi| \le 1$. So $m(g(x) + \xi) \le m(g(x)) + 4|\xi| \le m(x)^2 + 4|\xi|$ establishing the second claim. □

Lemma 2.5 (Overall scalar convergence)**.** *Fix a precision u and tolerance ε so that $10u \le \varepsilon \le 3/80$. Let $x_0 \in \pm[20u, 1.5]$ and $x_{k+1} = g(x_k) + \xi_k$ for adversarial $|\xi_k| \le u$. Then for*

$$N \ge N_{\texttt{SCALAR}}(x_0, \varepsilon) := 2.5 + 2\lg\min(|x_0|, 0.5)^{-1} + \lg\lg(1/\varepsilon)$$

one has

$$\left|1 - x_N^2\right| \le \varepsilon.$$

Proof. Note

$$m(x_k) \le \varepsilon \implies m(x_{k+1}) \le m(x_k)^2 + 4u \le \varepsilon^2 + 4u \le (\varepsilon + 2/5)\varepsilon \le \varepsilon$$

so it suffices to argue $m(x_k) \le \varepsilon$ for some $k \le N := N_{\texttt{SCALAR}}(x_0, \varepsilon)$. We first claim $|x_k| \le 1 + u$ for all $k > 0$. The range of g on the domain $[-1.5, 1.5]$ is $[-1, 1]$, so $x_k \in [-1.5, 1.5] \implies x_{k+1} \in [-1-u, 1+u] \subset [-1.5, 1.5]$. Then note $x_0 \in [-1.5, 1.5]$ by assumption so the claim follows by induction. Thus if $|x_k| \ge 1$, we'd have $m(x_k) \le (1+u)^2 - 1 = 2u + u^2 \le \varepsilon$, so we may assume $|x_k| \le 1$ for $k = 1, \cdots, N$.

Let $S = [20u, 1 - \sqrt{\varepsilon}]$. Let M be the lowest positive index such that $|x_M| \notin S$. If $M \le N - 2$, then $|x_M| \in (1 - \sqrt{\varepsilon}, 1]$ and $m(x_M) \le 1 - (1 - \sqrt{\varepsilon})^2 = 2\sqrt{\varepsilon}$ so

$$\begin{aligned} m(x_{M+1}) &\le m(x_M)^2 + 4u \le (2\sqrt{\varepsilon})^2 + 4u = 4\varepsilon + 4u \le 4.4\varepsilon, \\ m(x_{M+2}) &\le m(x_{M+1})^2 + 4u \le (4.4\varepsilon)^2 + 4u = 4.4^2\varepsilon^2 + 0.4\varepsilon \le \varepsilon, \end{aligned}$$

which would establish the claim. Let us now bound M. If $x_1 \notin S$, then $M = 1$. Otherwise, $x_1 \in S \subset [(8/3)u, 1 - (8/3)u]$, Lemma 2.3 implies x_k is monotonically increasing in k while in S so in fact $x_1, \cdots, x_{M-1} \in S$. Since $1 - \sqrt{\varepsilon} \le 1 - \sqrt{10u}$, Lemma 2.4 implies quadratic convergence for $k \le M$,

$$m(x_k) \le m(x_{k-1})^2 \le m(x_{k-2})^{2^2} \le \cdots \le m(x_1)^{2^{k-1}}.$$

Since $x_{M-1} \in S$, we have $m(x_{M-1}) \ge 1 - (1 - \sqrt{\varepsilon})^2 = 2\sqrt{\varepsilon} - \varepsilon \ge \sqrt{\varepsilon}$. Taking logs gives

$$\begin{aligned} \sqrt{\varepsilon} \le (1 - x_1^2)^{2^{M-2}} &\implies \frac{1}{2}\lg\varepsilon \le 2^{M-2}\lg(1 - x_1^2) \le -2^{M-2} \cdot \frac{1}{\log 2} \cdot x_0^2 \\ &\implies \frac{\log 2}{2x_1^2}\lg(1/\varepsilon) \ge 2^{M-2} \\ &\implies \lg\left(\frac{\log 2}{2x_1^2}\right) + \lg\lg(1/\varepsilon) \ge M - 2 \\ &\implies \lg(2\log 2) + 2\lg(1/|x_1|) + \lg\lg(1/\varepsilon) \ge M. \end{aligned}$$

Note $|x_1| \ge |x_0|$ or else $|x_0| \in [1 - (8/3)u, 1.5]$ implying $|x_1| \ge 0.5$. So $\lg(1/|x_1|) \le \lg\min(|x_0|, 0.5)^{-1}$. Then $\lg(2\log 2) < 0.5$. Combining those together yields $M \le N - 2$ as required. □

We're now ready to state and analyze the algorithm for approximating $\operatorname{sign}(A)$. In the below, g is defined by (6).

Algorithm 1 $\texttt{SIGN}(A, \varepsilon, b)$

Require: Nonsingular Hermitian matrix A with $\|A\| \le b$, desired accuracy ε.
Ensure: $\|\texttt{SIGN}(A, \varepsilon, b) - \operatorname{sign}(A)\| \le \varepsilon$.

```
A_0 ← A/b
k ← 0
repeat
    A_{k+1} ← g(A_k)
    k ← k + 1
until ‖I − MM(A_k, A_k)‖_max ≤ ε/(4n)
return A_k
```

Theorem 2.6 (Main guarantee for `SIGN`)**.** *Let*

$$N := N_{\texttt{SIGN}}(A, \varepsilon, n) := N_{\texttt{SCALAR}}\left(\frac{1}{\|A^{-1}\| \cdot b}, \frac{\varepsilon}{8n}\right).$$

Algorithm 1 has the advertised properties when run with

$$\mathbf{u} \le \mathbf{u}_{\texttt{SIGN}}(\varepsilon, b, \left\|A^{-1}\right\|) := \frac{1}{\|A^{-1}\| \cdot b} \cdot \frac{1}{4\max\left(N \cdot n\mu_{\mathtt{g}}(n, 1.1), n^2\right)} \cdot \varepsilon$$

using

$$3N \cdot T_{\texttt{MM}}(n) + O(Nn^2)$$

floating point operations.

Proof. Let $a = \left\|A^{-1}\right\|^{-1}$ so that the spectrum of A/b is contained in $[-1, -a/b] \cup [a/b, 1]$. Then by Lemma 2.2,

$$\|\operatorname{sign}(A_0) - \operatorname{sign}(A/b)\| \le n \cdot \frac{\|A_0 - A/b\|}{a/b - \|A_0 - A/b\|} \le n \cdot \frac{n\mathbf{u}}{a/b - n\mathbf{u}} \le \frac{2n^2}{a/b}\mathbf{u}.$$

Lemma 2.1 implies one can express

$$A_{k+1} = g(A_k) + E_k, \qquad \|E_k\| \le \mu_{\mathtt{g}}(n, \|A_k\|)\mathbf{u}.$$

We claim that $\|A_k\| \le 1.1$ for all k. It's clearly true $k = 0$. Then inductively,

$$\|A_{k+1}\| \le 1 + \mu_{\mathtt{g}}(n, \|A_k\|)\mathbf{u} \le 1 + \mu_{\mathtt{g}}(n, 1.1)\mathbf{u} \le 1.1$$

for our selection of $\mathbf{u} \le 0.1 \cdot \mu_{\mathtt{g}}(n, 1.1)^{-1}$. As a consequence, $\|E_k\| \le \mu_{\mathtt{g}}(n, 1.1)\mathbf{u}$. In particular, each eigenvalue of A_{k+1} is contained in the $\mu_{\mathtt{g}}(n, 1.1)\mathbf{u}$ pseudospectrum of $g(A_k)$, i.e. can be expressed as $g(\lambda)+\xi$ where λ is an eigenvalue of A_k and $|\xi| \le \mu_{\mathtt{g}}(n, 1.1)\mathbf{u}$. Therefore, Lemma 2.5 means for our selection of N that that each eigenvalue of A_N satisfies $\left|1 - \lambda_j(A_N)^2\right| \le \varepsilon/(8n)$. In particular,

$$\|I - \texttt{MM}(A_N, A_N)\|_{\max} \le \|I - \texttt{MM}(A_N, A_N)\| \le 2\left\|I - A_N^2\right\| = 2\sup_j\left|1 - \lambda_j(A_N)^2\right| \le \varepsilon/(4n)$$

so the algorithm terminates no later than iteration N. On the other hand, say M is the iteration on which the algorithm terminates. Then

$$\frac{\varepsilon}{4n} \ge \|I - \texttt{MM}(A_N, A_N)\|_{\max} \ge \frac{1}{2}\left\|I - A_N^2\right\|_{\max} \ge \frac{1}{2n}\left\|I - A_N^2\right\| \ge \frac{1}{2n}\sup_j\left|1 - \lambda_j(A_N)^2\right|$$

Therefore since A_M and $\operatorname{sign}(A_M)$ are simultaneously unitarily diagonalizable, we have

$$\|A_M - \operatorname{sign}(A_M)\| \le \max_j|\lambda_j(A_M) - \operatorname{sign}(\lambda_j(A_M))| \le \max_j\left|\lambda_j(A_M)^2 - 1\right| \le \varepsilon/2.$$

Note $\operatorname{sign} \circ g = \operatorname{sign}$ on the domain $[-\sqrt{3}, \sqrt{3}]$ so

$$\begin{aligned}
\|\operatorname{sign}(A_{k+1}) - \operatorname{sign}(A_k)\| &= \|\operatorname{sign}(A_{k+1}) - \operatorname{sign}(g(A_k))\| \\
&= \|\operatorname{sign}(A_{k+1}) - \operatorname{sign}(A_{k+1} - E_k)\| \\
&\le n \cdot \frac{\|E_k\|}{a/b - \|E_k\|} \\
&\le n \cdot \frac{\mu_{\mathtt{g}}(n, 1.1)\mathbf{u}}{a/b - \mu_{\mathtt{g}}(n, 1.1)\mathbf{u}} \\
&\le \frac{2n\mu_{\mathtt{g}}(n, 1.1)}{a/b}\mathbf{u}
\end{aligned}$$

where the third step is by Lemma 2.2 and last is a rearrangement of the requirement on $\mathbf{u}$. By the triangle inequality,

$$\begin{aligned}
\|A_M - \operatorname{sign}(A)\| &= \|A_M - \operatorname{sign}(A_0)\| + \|\operatorname{sign}(A_0) - \operatorname{sign}(A)\| \\
&\leq \|A_M - \operatorname{sign}(A_M)\| + \sum_{j=1}^{M} \|\operatorname{sign}(A_k) - \operatorname{sign}(A_{k-1})\| + \|\operatorname{sign}(A_0) - \operatorname{sign}(A/b)\|. \\
&\leq \frac{\varepsilon}{2} + \left(N \cdot \frac{2n\mu_{\mathtt{g}}(n, 1.1)}{a/b} + \frac{2n^2}{a/b} \right) \mathbf{u} \\
&\leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} \\
&= \varepsilon
\end{aligned}$$

as required. The number of floating point operations used is $N \cdot (T_{\mathtt{g}}(n) + T_{\mathtt{MM}}(n) + n^2) + n^2$. Using $T_{\mathtt{g}}(n) \leq 2T_{\mathtt{MM}}(n) + O(n^2)$ from Lemma 2.1 gives the final result. □

Remark 5 (b parameter)**.** If one is not supplied with the value b in the call $\mathtt{SIGN}(A, \varepsilon, b)$, an acceptable value can easily be computed in $O(n^2)$ or $O(T_{\mathtt{MM}}(n))$ time by taking $b = \|A\|_F$ or $b = \operatorname{tr}(A^{2p})^{1/(2p)}$. In that case, $\left\|A^{-1}\right\| \cdot b$ would be somewhere in between the condition number and weighted condition number of A.

3 Analysis of deflate

The secondary bottleneck in [BGVKS20] and a main limitation of [NH13] is the computation of `DEFLATE`. Deflation is a procedure for recovering from a low rank matrix P (often a projection), an orthonormal basis for its range. To the author's knowledge, up to minor variations, the algorithm we reproduce here is the only one running in near matrix multiplication time[5]. [BGVKS20] conceives of this algorithm as a slight modification of the rank-revealing QR-decomposition analyzed by [DDH07]. It turns out to be equivalent to the proposal by [NH13], which conceives of the algorithm as running a single iteration of subspace iteration. However, [NH13] glosses over the requirements of the starting matrix, which is a non-trivial part of the analyses of [BGVKS20, DDH07]. The algorithm is exceedingly simple and is essentially the following: output the QR-decomposition of the first $\operatorname{rank}(P)$ columns of PG where G is the random "starting" matrix. Notice that the method completely fails if $\operatorname{rank}(PG) < \operatorname{rank}(P)$, and struggles when the $\operatorname{rank}(P)$th singular value of PG is small. In order to ensure this value is sufficiently large often enough, one must include an additional factor of $\operatorname{poly}(n)$ in the precision. This factor is unavoidable. However, a much larger issue appearing in [BGVKS20] can be avoided. Their argument for the correctness of `DEFLATE` goes through the following argument: if the input is the projection UU^* and output is Q, they first show that U^*Q is close to a unitary matrix W. Then they convert this into a bound on $\|U - QW^*\|$, which is needed by the spectral bisection method. But this conversion introduces a square-root in the error. To see this, consider $Q = \begin{bmatrix} \cos\sqrt{2\varepsilon} & \sin\sqrt{2\varepsilon} \end{bmatrix}^*$ and $U = \begin{bmatrix} 1 & 0 \end{bmatrix}^*$. In this example, $U^*Q \approx 1 - \varepsilon$ whereas $\|U - Q\| \approx \sqrt{2\varepsilon}$. According to this analysis, in order to get the desired accuracy out of `DEFLATE`, one must double the number of bits used to overcome the square-root. We strengthen this analysis by removing the square-root, thereby reducing the bit requirement by a factor of two. The additional insight allowing this is that not only is U^*Q close to unitary, we can show that $(U^\perp)^*Q$ is close to 0 where $U^\perp$ is a basis for an orthogonal complement of the range of U. This turns out to be sufficient to remove the square-root, giving a tight analysis (up to constants).

This analysis also removes the restriction appearing in [BGVKS20] that the input matrix is close to a matrix satisfying $\operatorname{rank}(A^2) = \operatorname{rank}(A)$. This restriction wasn't an issue for our setting since A is an always an orthogonal projection matrix, but the removal of the restriction may be of independent interest. Our implementation of `DEFLATE` is the following.

[5] One candidate alternative is QR-decomposition with pivoting, which is recommended by [BDG97], but no efficient reduction to matrix multiplication is known.

Algorithm 2 $\texttt{DEFLATE}(\widetilde{A}, r)$

Require: There exists $A \in \mathbb{C}^{n\times n}$ and parameters $t, x, \beta > 0$ such that

$$\left\|\widetilde{A} - A\right\| \leq \beta \leq \frac{1}{5} \cdot \frac{\sigma_r(A)}{\sigma_1(A) + 2} \cdot \frac{x}{(2\sqrt{2} + t)\sqrt{n}} \leq 1$$

and $\operatorname{rank}(A) = r$.

Ensure: For output $\widetilde{U}$, there exists semi-unitary $U \in \mathbb{C}^{n\times r}$ such that $\operatorname{range}(U) = \operatorname{range}(A)$ and

$$\left\|\widetilde{U} - U\right\| \leq 6 \cdot \frac{\sigma_1(A) + 2}{\sigma_r(A)} \cdot \frac{(2\sqrt{2} + t)\sqrt{n}}{x} \cdot \beta$$

with probability $1 - 2e^{-nt^2} - (r/2)x^2$.

1: $\widetilde{G}_{ij} = \texttt{Normal}() \quad \forall i, j \in [n]$
2: $M = \texttt{MM}(\widetilde{A}, \widetilde{G})$
3: $(Q, R) = \texttt{QR}(M)$
4: **return** First r columns of Q

A convenient choice of parameters is $x = \sqrt{\rho/r}$ and $t = \sqrt{\log(4/\rho)/n}$ for some $\rho \in (0, 1)$. Then for each $0 < \eta \leq 6/5$, if

$$\beta \leq \frac{\rho^{1/2}\eta}{\sqrt{nr}} \cdot \frac{\sigma_r(A)}{\sigma_1(A) + 2} \cdot \frac{1}{12\sqrt{2} + 6\sqrt{\log(4/\rho)/n}}$$

then

$$\left\|\widetilde{U} - U\right\| \leq \eta$$

with probability $1 - \rho$.

Remark 6. The only difference between our method 2 and the one appearing in [BGVKS20] is that we take G to be a matrix of Gaussians rather than a Haar unitary matrix. In particular, [BGVKS20] replaces G with first output of $\texttt{QR}(G)$. However, this requires an additional call to $\texttt{QR}$ that is ultimately not necessary and in fact introduces some additional error.

In the following two lemmas, G is an $n \times n$ matrix with i.i.d. complex Gaussian entries.

Lemma 3.1 (Theorem 3.2 from [Ede88])**.**

$$\Pr(\sigma_n(G) \leq x) \leq \frac{n}{2}x^2.$$

Lemma 3.2 (Lemma 2.2 from [BKMS21])**.**

$$\Pr\Big(\|G\| > (2\sqrt{2} + t)\sqrt{n}\Big) \leq 2e^{-nt^2}.$$

Theorem 3.3. $\texttt{DEFLATE}$ *has the advertised guarantee when run with precision*

$$\mathbf{u} \leq \mathbf{u}_{\texttt{DEFLATE}}(\beta, n) = \frac{\beta}{4\mu_{\texttt{QR}}(n) + 2\sqrt{n}c_{\texttt{N}} + 2\mu_{\texttt{MM}}(n)}.$$

Furthermore, it uses only

$$T_{\texttt{MM}}(n) + T_{\texttt{QR}}(n) + T_{\texttt{N}} \cdot n^2.$$

floating point operations.

Proof. The runtime is clear the algorithm consists of n^2 calls to `Normal`, one call to `MM`, and one call to `QR` and no additional work. Throughout the proof we use $(\cdot)_1$ to denote the first r columns of a matrix and use $(\cdot)_{11}$ to denote the upper-left $r \times r$ submatrix. In particular, the output of the algorithm is Q_1. By assumption, we may express $A = U\Sigma V^*$ and $\widetilde{A} = A + E$ for $U, V \in \mathbb{C}^{n\times r}$, $\Sigma \in \mathbb{C}^{r\times r}$, and $\|E\| \le \beta$. Our goal is to bound $\inf_{\text{unitary } W} \|U - Q_1 W\|$. Let G_{ij} be the Gaussian random variable coupled with $\widetilde{G}_{ij}$ such that $|\widetilde{G}_{ij} - G_{ij}| \le |G_{ij}| c_{\mathtt{N}}\mathbf{u}$. Then $\|\widetilde{G} - G\| \le \|G\| \sqrt{n} c_{\mathtt{N}}\mathbf{u}$. Let $\widetilde{G} = G + E_G$. Then for some $\|F\| \le \|\widetilde{A}\|\|\widetilde{G}\|\mu_{\mathtt{MM}}(n)\mathbf{u}$,

$$\begin{aligned} M &= \mathtt{MM}(\widetilde{A}, \widetilde{G}) \\ &= \widetilde{A} \cdot \widetilde{G} + F \\ &= (U\Sigma V^* + E) \cdot (G + E_G) + F \\ &= U\Sigma V^* G + U\Sigma V^* E_G + E \cdot (G + E_G) + F. \end{aligned}$$

Set $X = U\Sigma V^* E_G + E \cdot (G + E_G) + F$. Definition 1.2 guarantees that for some $\|E_M\| \le \|M\| \, \mu_{\mathtt{QR}}(n)\mathbf{u}$ and $\|E_Q\| \le \mu_{\mathtt{QR}}(n)\mathbf{u}$ that

$$\begin{aligned} & (M + E_M) = (Q + E_Q)R \\ \implies & M = QR + E_Q R - E_M \\ \implies & U\Sigma V^* G = QR + E_Q R - E_M - X \\ \implies & U\Sigma V^* G_1 = Q \begin{bmatrix} R_{11} \\ 0 \end{bmatrix} + E_Q \begin{bmatrix} R_{11} \\ 0 \end{bmatrix} - (E_M)_1 - X_1 \\ \implies & U\Sigma V^* G_1 = Q_1 R_{11} + (E_Q)_1 R_{11} - (E_M)_1 - X_1 \\ \implies & U = Q_1 R_{11} (V^* G_1)^{-1} \Sigma^{-1} + [(E_Q)_1 R_{11} - (E_M)_1 - X_1](V^* G_1)^{-1}\Sigma^{-1}. \end{aligned}$$

We bound the first factor of the second term by

$$\begin{aligned} \|(E_Q)_1 R_{11}\| &\le \|E_Q\| \, \|R\| \\ &\le \mu_{\mathtt{QR}}(n)\mathbf{u} \cdot \|M + E_M\| \\ &\le \mu_{\mathtt{QR}}(n)\mathbf{u} \cdot (1 + \mu_{\mathtt{QR}}(n)\mathbf{u}) \, \|M\| \\ &\le \mu_{\mathtt{QR}}(n)\mathbf{u} \cdot (1 + \mu_{\mathtt{QR}}(n)\mathbf{u})(1 + \mu_{\mathtt{MM}}(n)\mathbf{u})\|\widetilde{A}\|\|\widetilde{G}\| \\ &\le \mu_{\mathtt{QR}}(n)\mathbf{u} \cdot (1 + \mu_{\mathtt{QR}}(n)\mathbf{u})(1 + \mu_{\mathtt{MM}}(n)\mathbf{u})(\|\Sigma\| + \beta)(1 + \sqrt{n} c_{\mathtt{N}}\mathbf{u}) \, \|G\| \\ &\le 2 \, \|G\| \, (\|\Sigma\| + \beta)\mu_{\mathtt{QR}}(n)\mathbf{u} \\ \|(E_M)_1\| &\le \|E_M\| \\ &\le \mu_{\mathtt{QR}}(n)\mathbf{u} \cdot (1 + \mu_{\mathtt{MM}}(n)\mathbf{u})\|\widetilde{A}\|\|\widetilde{G}\| \\ &\le 2 \, \|G\| \, (\|\Sigma\| + \beta)\mu_{\mathtt{QR}}(n)\mathbf{u} \\ \|X_1\| &\le \|X\| \\ &\le \|\Sigma\| \sqrt{n} c_{\mathtt{N}}\mathbf{u} \cdot \|G\| + \beta \cdot \|\widetilde{G}\| + \|\widetilde{A}\|\|\widetilde{G}\|\mu_{\mathtt{MM}}(n)\mathbf{u} \\ &\le \|\Sigma\| \sqrt{n} c_{\mathtt{N}}\mathbf{u} \cdot \|G\| + \beta \cdot (1 + \sqrt{n} c_{\mathtt{N}}\mathbf{u})\|G\| + (\|\Sigma\| + \beta)(1 + \sqrt{n} c_{\mathtt{N}}\mathbf{u})\|G\|\mu_{\mathtt{MM}}(n)\mathbf{u} \\ &= \|G\| \, (\|\Sigma\| + \beta)\big[(\sqrt{n} c_{\mathtt{N}} + \mu_{\mathtt{MM}}(n)) + \sqrt{n} c_{\mathtt{N}}\mu_{\mathtt{MM}}(n)\mathbf{u}\big]\mathbf{u} + \beta\|G\| \\ &\le 2(\sqrt{n} c_{\mathtt{N}} + \mu_{\mathtt{MM}}(n)) \, \|G\| \, (\|\Sigma\| + \beta)\mathbf{u} + \beta\|G\| \end{aligned}$$

Altogether, for $C = R_{11}(V^*G_1)^{-1}\Sigma^{-1}$ this gives

$$\begin{aligned}
\|U - Q_1 C\| &\le \|(E_Q)_1 R_{11} - (E_M)_1 - X_1\| \cdot \left\|(V^*G_1)^{-1}\Sigma^{-1}\right\| \\
&\le \left[\left(4\mu_{\mathtt{QR}}(n) + 2\sqrt{n}c_{\mathtt{N}} + 2\mu_{\mathtt{MM}}(n)\right)(\|\Sigma\| + \beta)\mathbf{u} + \beta\right] \|G\| \cdot \left\|(V^*G_1)^{-1}\Sigma^{-1}\right\| \\
&\le \left[\left(4\mu_{\mathtt{QR}}(n) + 2\sqrt{n}c_{\mathtt{N}} + 2\mu_{\mathtt{MM}}(n)\right)\left(\frac{\|A\| + \beta}{\sigma_r(A)}\right)\mathbf{u} + \frac{\beta}{\sigma_r(A)}\right] \cdot \|G\| \left\|(V^*G_1)^{-1}\right\| \qquad (7) \\
&\le \left[\frac{\|A\| + \beta + 1}{\sigma_r(A)}\right] \cdot \|G\| \left\|(V^*G_1)^{-1}\right\| \cdot \beta \\
&=: m \cdot \beta
\end{aligned}$$

If C was unitary we'd be nearly done. Unfortunately we have no such guarantee. Instead we derive three different inequalities from (7).

$$\begin{aligned}
(7) &\implies \|U - Q_1 C\| \le m\beta \\
&\implies \big|\, \|Ux\| - \|Q_1 Cx\| \,\big| \le m\beta \|x\| \\
&\implies (1 - m\beta)\|x\| \le \|Q_1 Cx\| \le (1 + m\beta)\|x\| \\
&\implies \frac{1 - m\beta}{1 + \mu_{\mathtt{QR}}(n)\mathbf{u}} \le \frac{\|Cx\|}{\|x\|} \le \frac{1 + m\beta}{1 - \mu_{\mathtt{QR}}(n)\mathbf{u}}.
\end{aligned}$$

In in particular, that implies bounds on the singular values of C, which we now apply.

$$\begin{aligned}
(7) &\implies \|U^*U - U^*Q_1 C\| \le m\beta \\
&\implies \big|\, \|x\| - \|U^*Q_1 Cx\| \,\big| \le m\beta \|x\| \\
&\implies (1 - m\beta)\|x\| \le \|U^*Q_1 Cx\| \le (1 + m\beta)\|x\| \\
&\implies \frac{1 - m\beta}{\sigma_1(C)}\|x\| \le \|U^*Q_1 x\| \le \frac{1 + m\beta}{\sigma_r(C)}\|x\| \\
&\implies (1 - \mu_{\mathtt{QR}}(n)\mathbf{u})\frac{1 - m\beta}{1 + m\beta}\|x\| \le \|U^*Q_1 x\| \le \frac{1 + m\beta}{1 - m\beta}(1 + \mu_{\mathtt{QR}}(n)\mathbf{u})\|x\|. \\
(7) &\implies \left\|(U^\perp)^*U - (U^\perp)^*Q_1 C\right\| \le m\beta \\
&\implies \left\|(U^\perp)^*Q_1 Cx\right\| \le m\beta\|x\| \\
&\implies \left\|(U^\perp)^*Q_1 x\right\| \le \frac{1}{\sigma_r(C)} m\beta\|x\| \\
&\implies \left\|(U^\perp)^*Q_1 x\right\| \le \frac{1 + \mu_{\mathtt{QR}}(n)\mathbf{u}}{1 - m\beta} m\beta\|x\|
\end{aligned}$$

Now consider the quantity we need to bound and apply the two above inequalities

$$\begin{aligned}
\inf_{\text{unitary } W} \|U - Q_1 W\| &\le \inf_{\text{unitary } W} \left(\|I - U^*Q_1 W\| + \left\|(U^\perp)^*Q_1 W\right\|\right) \\
&= \inf_{\text{unitary } W} \|W^* - U^*Q_1\| + \left\|(U^\perp)^*Q_1\right\| \\
&= \max(\sigma_1(U^*Q_1) - 1, 1 - \sigma_r(U^*Q_1)) + \left\|(U^\perp)^*Q_1\right\| \\
&= \max\left(\frac{1 + m\beta}{1 - m\beta}(1 + \mu_{\mathtt{QR}}(n)\mathbf{u}) - 1, 1 - (1 - \mu_{\mathtt{QR}}(n)\mathbf{u})\frac{1 - m\beta}{1 + m\beta}\right) + \frac{1 + \mu_{\mathtt{QR}}(n)\mathbf{u}}{1 - m\beta} m\beta \\
&\le 4m\beta + 2m\beta \\
&= 6m\beta.
\end{aligned}$$

For the last inequality, we assumed that $m\beta \le 0.2$. We end with a tail estimate on m. Note by rotational invariance that V^*G_1 is an $r \times r$ matrix of complex Gaussian entries. So applying Lemmas 3.1 and 3.2 we

have by union bound that

$$\Pr\left(m \geq \frac{\sigma_1(A)+2}{\sigma_r(A)} \cdot \frac{(2\sqrt{2}+t)\sqrt{n}}{x}\right) \leq e^{-nt^2} + \frac{r}{2}x^2.$$

Note the complementary event implies $m\beta \leq 0.2$ by the requirement of β, so the desired bound holds. □

Remark 7. One may notice all we really need for spectral bisection is a basis of an invariant subspace of A and wonder why we're bothering with matrix sign and deflate. After all, we have ready-made methods for computing bases of invariant subspaces: Lanczos and subspace iteration. For instance, consider

$$X_0 = \text{random} \in \mathbb{C}^{n\times(n/2)} \qquad [X_{k+1}, R_k] = \texttt{QR}(AX_k).$$

Then X_k converges to an invariant subspace of dimension $n/2$. The issue is the rate of convergence. If $\lambda_1 > \lambda_2 > \cdots > \lambda_n \geq 0$ are the eigenvalues of A, then we only converge after

$$\frac{\log(1/\varepsilon)}{\log\left(\lambda_{n/2+1}/\lambda_{n/2}\right)} \approx \frac{\log(1/\varepsilon)}{1-\lambda_{n/2+1}/\lambda_{n/2}}$$

iterations. This is a *polynomial* dependence on the relative eigenvalue gap, which is totally unacceptable. Lanczos offers only a square root improvement over this. Since spectral projectors are square with a functional calculus, we have 'repeated squaring' flavor algorithms which simulate computing something akin to X_{2^k} or even X_{3^k} in just k iterations.

4 Spectral bisection

We're now ready to describe our spectral bisection algorithm, Algorithm 3 `EIGH-INTERNAL`. As mentioned in the introduction, this work makes three main changes to the version of spectral bisection appearing in [BGVKS20]. One is to use Newton-Schulz for sign estimation. The other two are highlighted in Remarks 8 and 9. In the pseudocode, the symbol $\leftarrow$ is used to denote floating-point assignment. That is, $x \leftarrow r$ means $x = \texttt{fl}(r)$ is the floating point number closest to r. The "root call" to `EIGH-INTERNAL` is

$$[U, D] = \texttt{EIGH}(A, \varepsilon, \theta) = \texttt{EIGH-INTERNAL}(A, \|A\|, \|A\|, \varepsilon, \lceil \lg(1/\varepsilon)\rceil + 5, \theta/4n).$$

Remark 8. (Recursive parameters) [BGVKS20] passes $0.8\cdot\varepsilon$ in for ε in the recursive calls. For a computation tree of $O(\log(n))$ depth, this makes the ε seen by a deep node $\varepsilon/\operatorname{poly}(n)$. This contributes to the precision $\mathbf{u}$ required by that node, which itself is is a polynomial in ε. Algorithm 3 passes in $(1-1/\ell)\varepsilon$ and increments ℓ. Instead of 0.8^k, one ends up with a telescoping product resulting in only a constant factor smaller ε seen at the leaves.

Remark 9. (Shattering/Split points/Base case) [BGVKS20] uses binary search to find a good value of the split point c' that ensures $k_\pm$ are both at least $n/5$. This ensures that the sub-problems are constant factors smaller than the original, guaranteeing that recursion halts at depth $\log_{5/4}(n)$. [NH13] uses the median of the diagonal entries to find a split point that ensures $k_\pm \geq 1$. In the worst case, this leads to a depth of n. We pick our split points c' randomly, as suggested by [BDD11]. In order to avoid having to perturb the input matrix, the avoidance of binary search and adoption of random split points are both necessary. If we picked c' deterministically, an adversarial input may place an eigenvalue exactly at our selection preventing `SIGN` from converging. We also cannot assume that our eigenvalues are spaced out. In particular, there may not even exist a split point with $k_\pm \geq n/5$ so binary searching for one would fail. The random split points are picked close to the midpoint, so they reduce the range of the spectrum by almost a factor of two each time.

Algorithm 3 $[U, D] = \texttt{EIGH-INTERNAL}(A, R_0, R, \varepsilon, \ell, \rho)$

Require: Hermitian matrix A with floating point entries. Initial size $R_0 \geq \|A\|$. Window size $R \geq \|A\|$. Target accuracy $\varepsilon > 0$. Integer $\ell \geq 20$. Failure parameter $\rho > 0$.

Ensure: With some probability, all singular values of U lie in $[1-\varepsilon/3, 1+\varepsilon/3]$ and $\|UDU^* - A\| \leq \varepsilon \cdot (R_0 + R)$.

if $n = 1$ **then**
  return $(\begin{bmatrix}1\end{bmatrix}, a_{11})$
end if
if $R \leq \varepsilon R_0$ **then**
  return $(I_{n\times n}, 0)$
end if
$\varepsilon' \leftarrow (1 - 1/\ell)\varepsilon$
$R' \leftarrow (1/2 + 2/\ell)R$

$$\delta \leftarrow \frac{3}{4} \cdot \frac{\rho^{1/2}\eta}{n} \cdot \frac{1}{3} \cdot \frac{1}{12\sqrt{2} + 6\sqrt{\log(4/\rho)/n}} \text{ where } \eta = \frac{\varepsilon'}{5\ell}$$

$c = \texttt{Unif}(R/\ell)$
$A_{\texttt{shift}} \leftarrow A - cI$
$B = \texttt{SIGN}(A_{\texttt{shift}}, \delta, 2R)$
$P_\pm \leftarrow (I \pm B)/2$
$k_\pm = \texttt{round}(\operatorname{tr} P_\pm)$
if $k_\pm = n$ **then** ▷ Note $k_+ + k_- = n$ so this condition is met for at most one of k_+ and k_-.
  $A_\pm \leftarrow A \mp (R/2)I$
  $(U, D) = \texttt{EIGH-INTERNAL}(A_\pm, R_0, R', \varepsilon', \ell + 1, \rho)$
  return $(U, D \pm (R/2)I)$
end if
$Q_\pm = \texttt{DEFLATE}(P_\pm, k_\pm)$
$C_\pm = \texttt{MM}(\texttt{MM}(Q_\pm^*, A), Q_\pm)$
$A_\pm \leftarrow C_\pm \mp (R/2)I$
$(U_\pm, D_\pm) = \texttt{EIGH-INTERNAL}(A_\pm, R_0, R', \varepsilon', \ell + 1, \rho)$
$W_\pm = \texttt{MM}(Q_\pm, U_\pm)$
return $\left(\begin{bmatrix} W_+ & W_- \end{bmatrix}, \begin{bmatrix} D_+ + (R/2)I & \\ & D_- - (R/2)I \end{bmatrix} \right)$

There are four quantities we need to bound for `EIGH`. 1. Residual error 2. Success probability 3. Precision requirement 4. Runtime. We do so in two parts. First, we analyze those for quantities within each call to `EIGH-INTERNAL` treating the recursive calls as oracle queries; we call this the "local" guarantee. In particular, applying the local guarantee to the root call ensures the residual error of `EIG` satisfies desired bound. For the other three quantities, we need a global analysis. In particular, we apply union bound to obtain the probability every call to `EIGH-INTERNAL` succeeds, sum the relevant runtimes, and take the minimum over all the precisions required.

Lemma 4.1. *(Local guarantee) Consider a call to* `EIGH-INTERNAL` *with inputs* $(A, R, \varepsilon, \ell, \rho)$. *Fix any* $w > 0$ *and set*

$$t = 12\sqrt{2} + 6\sqrt{\log(4/\rho)/n} \qquad \& \qquad \beta = \frac{\rho^{1/2}\eta}{3nt}.$$

Let c' *be the sample from the real interval* $[-R/\ell, R/\ell]$ *coupled with* $c = \texttt{UNIF}(R/\ell)$ *in Step 10 so that* $\|c - c'\| \le (R/\ell)\mathbf{u}$. *When using precision,*

$$\mathbf{u} \le \min\left(\mathbf{u}_{\texttt{DEFLATE}}(\beta, n), \mathbf{u}_{\texttt{SIGN}}(\beta/2, 2R, 2/w), \frac{1}{4} \cdot \frac{\beta}{n} \cdot \frac{w}{R}\right)$$

if one conditions on

Event I: $c' \notin \Lambda_w(A)$,

Event II: Step 20 succeeds with error at most η, *and*

Event III: The recursive calls in Step 23 succeed (i.e. produce $U_\pm, D_\pm$ *with small residual),*

then the values returned in Step 25 have the advertised guarantee. Furthermore Event I occurs with probability at least $1 - wn/R$ *and Event II occurs with probability at least* $1 - 2\rho$.

Proof. Our first task is to quantify $\|A_{\texttt{shift}} - (A - c'I)\|$.

$$\begin{aligned}
\|A_{\texttt{shift}} - (A - c'I)\| &\le \|A_{\texttt{shift}} - (A - cI)\| + \|(A - cI) - (A - c'I)\| \\
&\le \|(A - cI)\|\,\mathbf{u} + |c - c'| \\
&\le (R + R/\ell)\mathbf{u} + (R/\ell)\mathbf{u} \\
&\le (1 + 2/\ell)R\mathbf{u}.
\end{aligned} \tag{8}$$

Our next task is to quantify $\left\|P_\pm - \frac{\text{sign}(A-c'I)\pm I}{2}\right\|$

$$\left\|P_\pm - \frac{\text{sign}(A - c'I) \pm I}{2}\right\| = \left\|P_\pm - \frac{B \pm I}{2}\right\| \tag{9}$$

$$+ \left\|\frac{B \pm I}{2} - \frac{\text{sign}(A_{\texttt{shift}}) \pm I}{2}\right\| \tag{10}$$

$$+ \left\|\frac{\text{sign}(A_{\texttt{shift}}) \pm I}{2} - \frac{\text{sign}(A - c'I) \pm I}{2}\right\| \tag{11}$$

We bound each of the three terms starting with (11). Note we may pull out $\frac{1}{2}$ and the identity terms cancel. Then Event I implies that we may apply Lemma 2.2 to bound it by

$$(11) = \frac{1}{2}\|\text{sign}(A_{\texttt{shift}}) - \text{sign}(A - c'I)\| \le \frac{1}{2} \cdot n \cdot \frac{(1 + 2/\ell)R\mathbf{u}}{w - (1 + 2/\ell)R\mathbf{u}}.$$

Note incidentally that $\Lambda_w(A)$ is a set of measure at most $2nw$ and c' is sampled from a density bounded by $\ell/2R$ so the probability Event I fails is at most $2nw \cdot \ell/2R = nw/R$. Next we consider (10). Note again we may pull out $\frac{1}{2}$ and the identity terms cancel, so

$$(10) = \frac{1}{2}\left\|\texttt{SIGN}(A_{\texttt{shift}}, \delta, 2R) - \text{sign}(A_{\texttt{shift}})\right\|.$$

In order to apply Theorem 2.6, we must verify $\mathbf{u}$ is small enough given the inputs. First, (8) implies

$$\|A_{\texttt{shift}}\| \le \|A_{\texttt{shift}} - (A - c'I)\| + \|A - c'I\| \le [(1 + 2/\ell)R\mathbf{u}] + [(1 + 1/\ell)R] \le 2R.$$

Second, Event I means $c' \notin \Lambda_w(A)$ so

$$\left\|A_{\texttt{shift}}^{-1}\right\| \le \frac{1}{\frac{1}{\|(A-c'I)^{-1}\|} - \|A_{\texttt{shift}} - (A - c'I)\|} \le \frac{1}{w - (1 + 2/\ell)R\mathbf{u}} \le \frac{2}{w}.$$

Our selection of $\mathbf{u}$ is bounded by $\mathbf{u}_{\texttt{SIGN}}(\delta, 2R, 2/w)$, so Theorem 2.6 bounds (10) $\le \delta/2$. Finally

$$(9) \le \frac{\|B\| + 1}{2}\mathbf{u} \le \frac{(1 + \delta) + 1}{2}\mathbf{u} \le (1 + \delta/2)\mathbf{u}.$$

Summing (9)+(10)+(11) gives

$$\left\|P_\pm - \frac{\operatorname{sign}(A - c'I) \pm I}{2}\right\| \le \left[\frac{n}{2}\frac{(1 + 2/\ell)\frac{R}{w}}{1 - (1 + 2/\ell)\frac{R}{w}\mathbf{u}} + 1 + \frac{\delta}{2}\right]\mathbf{u} + \frac{\delta}{2} \le \delta$$

since $\mathbf{u} \le \frac{1}{4} \cdot \frac{\beta}{n} \cdot \frac{w}{R}$. Note that $\frac{\operatorname{sign}(A-c'I)\pm I}{2}$ are the true spectral projectors into the the parts of the spectrum of A above and below c'. Since $\beta + n\mathbf{u} \ll 1/2$, the estimates $k_\pm$ are the exact ranks of $\frac{\operatorname{sign}(A-cI)\pm I}{2}$. Additionally, we have $\mathbf{u} \le \mathbf{u}_{\texttt{DEFLATE}}(n, \beta)$. These are exactly the conditions for `DEFLATE` to succeed with probability $1 - \rho$. Applying union bound gives the probability both calls to `DEFLATE` succeed. Since we are conditioning on Event II, we have that

$$\|Q_\pm - V_\pm\| \le \eta \;\;\&\;\; \|Q_\pm\| \le 1 + \mu_{\texttt{QR}}(n)\mathbf{u} \tag{12}$$

for some choice of $V_\pm$ having orthonormal columns with $\operatorname{sign}(A - c'I) = V_+V_+^* - V_-V_-^*$. We now show the recursive calls to `EIGH-INTERNAL` in Step 23 satisfy the input requirements—namely, we must ensure $R' \ge \|A_\pm\|$.

$$\|A_\pm\| \le \left\|V_\pm^*\left(A \mp \frac{R}{2}I\right)V_\pm\right\| \tag{13}$$

$$+ \left\|A_\pm - V_\pm^*\left(A \mp \frac{R}{2}I\right)V_\pm\right\|. \tag{14}$$

Since $V_\pm$ are the bases for the upper and lower parts of the spectrum of A, the spectrums of $V_\pm AV_\pm$ are contained in $[c', R]$ and $[-R, c']$ respectively, and so the spectrums of $V_\pm\big(A \mp \frac{R}{2}I\big)V_\pm$ are contained in $[c - R/2, R/2]$ and $[-R/2, c + R/2]$ respectively, both of which are contained in $[-(1/2+1/\ell)R, (1/2+1/\ell)R]$ since $|c| \le R/\ell$. This results in (13)$\le (1/2 + 1/\ell)R$. Bounding (14) is more involved.

$$\left\|A_\pm - V_\pm^*\left(A \mp \frac{R}{2}I\right)V_\pm\right\| = \left\|A_\pm - \left(C_\pm \mp \frac{R}{2}I\right)\right\| + \left\|\left(C_\pm \mp \frac{R}{2}I\right) - V_\pm^*\left(A \mp \frac{R}{2}I\right)V_\pm\right\| \tag{15}$$

$$\le \left(\|C_\pm\| + \frac{R}{2}\right)\mathbf{u} + \left\|C_\pm - V_\pm^*AV_\pm\right\| \tag{16}$$

$$\le \left(\|C_\pm\| + \frac{R}{2}\right)\mathbf{u} + \left\|C_\pm - Q_\pm^*AQ_\pm\right\| + \left\|Q_\pm^*AQ_\pm - V_\pm^*AV_\pm\right\|. \tag{17}$$

The first term of (17) can be bounded using

$$\|C_\pm\| \le \left\|C_\pm - Q_\pm^*AQ_\pm\right\| + \left\|Q_\pm^*AQ_\pm - V_\pm^*AV_\pm\right\| + \left\|V_\pm^*AV_\pm\right\|. \tag{18}$$

Then both the first term of (18) and second term of (17) can be bounded by using the guarantee for MM twice,

$$\begin{aligned}\left\|C_\pm - Q_\pm^* A Q_\pm\right\| &\le \mu_{\texttt{MM}}(n)\,\|A\|\,\|Q_\pm\|^2\,(2+\mu_{\texttt{MM}}(n)\mathbf{u})\cdot\mathbf{u}\\ &\le \mu_{\texttt{MM}}(n)\cdot R\cdot(1+\mu_{\texttt{QR}}(n)\mathbf{u})^2\cdot(2+\mu_{\texttt{MM}}(n)\mathbf{u})\cdot\mathbf{u}\\ &\le 3R\mu_{\texttt{MM}}(n)\mathbf{u}.\end{aligned} \tag{19}$$

Then both the second term of (18) and third term of (17) can be bounded using (12)

$$\begin{aligned}\left\|Q_\pm^* A Q_\pm - V_\pm A V_\pm\right\| &= \left\|(V_\pm+\eta E'_\pm)^* A (V_\pm+\eta E'_\pm) - V_\pm A V_\pm\right\|\\ &\le \left\|V_\pm^* A E'_\pm\eta + E'^*_\pm A V_\pm \eta + E'^*_\pm A E'_\pm \eta^2\right\|\\ &\le (2+\eta)R\eta.\end{aligned} \tag{20}$$

Summing these estimates and rearranging gives

$$\begin{aligned}(14)\le(17) &\le [(1+\mathbf{u})\cdot 3\mu_{\texttt{MM}}(n)+(3/2)]R\mathbf{u} + (1+\mathbf{u})\cdot(2+\eta)R\eta\\ &\le (2+2\eta)R\eta\end{aligned} \tag{21}$$

and consequently

$$\|A_\pm\| \le (13)+(14) \le \left(\frac{1}{2}+\frac{1}{\ell}+(2+\eta)\eta\right)R \le \left(\frac{1}{2}+\frac{2}{\ell}\right)(1-\mathbf{u})R \le R'.$$

Let's now show that $\begin{bmatrix}W_+ & W_-\end{bmatrix}$ is nearly unitary. By (12), we may express $Q_\pm = V_\pm + \eta E'_\pm$ for some $\left\|E'_\pm\right\| \le 1$. By Definition 1.1, we may express

$$\begin{aligned}W_\pm &= Q_\pm U_\pm + \|Q_\pm\|\,\|U_\pm\|\,\mu_{\texttt{MM}}(n)\mathbf{u}E\\ &= V_\pm U_\pm + \eta E'_\pm U_\pm + \|Q_\pm\|\,\|U_\pm\|\,\mu_{\texttt{MM}}(n)\mathbf{u}E\end{aligned} \tag{22}$$

for some $\|E\| \le 1$. Set $X_\pm = W_\pm - V_\pm U_\pm$ and note we have the estimate

$$\begin{aligned}\|X_\pm\| &\le \eta\,\|U_\pm\| + \|Q_\pm\|\,\|U_\pm\|\,\mu_{\texttt{MM}}(n)\mathbf{u}\\ &\le \eta(1+\varepsilon') + (1+\mu_{\texttt{QR}}(n)\mathbf{u})(1+\varepsilon')\mu_{\texttt{MM}}(n)\mathbf{u}\\ &\le \eta(1+2\varepsilon')\end{aligned} \tag{23}$$

By construction, we have

$$\begin{bmatrix}W_+ & W_-\end{bmatrix} = \begin{bmatrix}V_+ & V_-\end{bmatrix}\begin{bmatrix}U_+ & \\ & U_-\end{bmatrix} + \begin{bmatrix}X_+ & X_-\end{bmatrix}.$$

Note that $\begin{bmatrix}V_+ & V_-\end{bmatrix}$ is a unitary matrix and that Event III guarantees the singular values of $U_\pm$ lie in $[1-\varepsilon'/3, 1+\varepsilon'/3]$. So the singular values of $\begin{bmatrix}W_+ & W_-\end{bmatrix}$ satisfy

$$(1-\varepsilon'/3) - \sqrt{2}\max_\pm\|X_\pm\| \le \sigma_j\left(\begin{bmatrix}W_+ & W_-\end{bmatrix}\right) \le (1+\varepsilon'/3) + \sqrt{2}\max_\pm\|X_\pm\|.$$

Finally, applying (23) gives

$$\begin{aligned}\frac{\varepsilon'}{3} + \sqrt{2}\max_\pm\|X_\pm\| &= \frac{\varepsilon'}{3} + \sqrt{2}\eta(1+2\varepsilon')\\ &= \left(1+3\sqrt{2}\cdot\frac{\eta}{\varepsilon'}\cdot(1+2\varepsilon')\right)\frac{\varepsilon'}{3}\\ &\le \left(1+\frac{1}{\ell}\right)\frac{\varepsilon'}{3}\\ &\le \left(1+\frac{1}{\ell}\right)(1+\mathbf{u})\left(1-\frac{1}{\ell}\right)\frac{\varepsilon}{3}\\ &\le \varepsilon/3\end{aligned}$$

so each singular value of $\begin{bmatrix} W_+ & W_- \end{bmatrix}$ is in the interval $[1-\varepsilon/3, 1+\varepsilon/3]$ as required. Our final task is showing our output has sufficiently small residual. Let $G_\pm = D_\pm \pm \frac{R}{2} I$ and $\widetilde{G}_\pm = \mathtt{fl}(G_\pm)$. The approximate factorization of A returned in Step 25 is

$$\begin{bmatrix} W_+ & W_- \end{bmatrix} \begin{bmatrix} \widetilde{G}_+ & \\ & \widetilde{G}_- \end{bmatrix} \begin{bmatrix} W_+ & W_- \end{bmatrix}^* = W_+ \widetilde{G}_+ W_+^* + W_- \widetilde{G}_- W_-^*.$$

The analogous exact factorization is

$$A = \begin{bmatrix} V_+ & V_- \end{bmatrix} \begin{bmatrix} V_+^* A V_+ & \\ & V_-^* A V_- \end{bmatrix} \begin{bmatrix} V_+ & V_- \end{bmatrix}^* = V_+ V_+^* A V_+ V_+^* + V_- V_-^* A V_- V_-^*.$$

We define two additional factorizations, namely the sums $\widetilde{F}_+ + \widetilde{F}_-$ and $F_+ + F_-$ for

$$\widetilde{F}_\pm := (V_\pm U_\pm) \widetilde{G}_\pm (V_\pm U_\pm)^* \quad \& \quad F_\pm := (V_\pm U_\pm) G_\pm (V_\pm U_\pm)^*$$

Then the final residual is bounded by

$$\left\| A - \begin{bmatrix} W_+ & W_- \end{bmatrix} \begin{bmatrix} \widetilde{G}_+ & \\ & \widetilde{G}_- \end{bmatrix} \begin{bmatrix} W_+ & W_- \end{bmatrix}^* \right\| \le \|A - (F_+ + F_-)\| \tag{24}$$

$$+ \left\| (F_+ + F_-) - (\widetilde{F}_+ + \widetilde{F}_-) \right\| \tag{25}$$

$$+ \left\| \widetilde{F}_+ - W_+ \widetilde{G}_+ W_+^* \right\| + \left\| \widetilde{F}_- - W_- \widetilde{G}_- W_-^* \right\| \tag{26}$$

We begin by bounding each term in (26). Event III guarantees $\left\| A_\pm - U_\pm D_\pm U_\pm^* \right\| \le \varepsilon'$ which implies $\|D_\pm\| \le \frac{\|A_\pm\| + \varepsilon'}{(1-\varepsilon')^2} \le \frac{R' + \varepsilon'}{(1-\varepsilon')^2}$. Then

$$\begin{aligned}
\left\| \widetilde{F}_\pm - W_\pm \widetilde{G}_\pm W_\pm^* \right\| &= \left\| (V_\pm U_\pm) \widetilde{G}_\pm (V_\pm U_\pm)^* - W_\pm \widetilde{G}_\pm W_\pm^* \right\| \\
&= \left\| (V_\pm U_\pm) \widetilde{G}_\pm (V_\pm U_\pm)^* - (V_\pm U_\pm + X_\pm) \widetilde{G}_\pm (V_\pm U_\pm + X_\pm)^* \right\| \\
&= \left\| -X_\pm \widetilde{G}_\pm (V_\pm U_\pm)^* - (V_\pm U_\pm) \widetilde{G}_\pm X_\pm^* - X_\pm \widetilde{G}_\pm X_\pm^* \right\| \qquad (27) \\
&\le (2 + 2\varepsilon' + \|X_\pm\|) \|\widetilde{G}_\pm\| \, \|X_\pm\| \\
&\le (2 + 2\varepsilon' + (1+2\varepsilon')\eta) \left(\frac{R' + \varepsilon'}{(1-\varepsilon')^2} + \frac{R}{2} \right) (1 + \mathbf{u}) \cdot (1 + 2\varepsilon')\eta
\end{aligned}$$

Next we bound (25).

$$\begin{aligned}
\left\| (F_+ + F_-) - (\widetilde{F}_+ + \widetilde{F}_-) \right\| &= \left\| \begin{bmatrix} V_+ & V_- \end{bmatrix}^* \left(F_+ + F_- - \widetilde{F}_+ - \widetilde{F}_- \right) \begin{bmatrix} V_+ & V_- \end{bmatrix} \right\| \\
&= \left\| \begin{bmatrix} U_+ (G_+ - \widetilde{G}_+) U_+^* & \\ & U_- (G_- - \widetilde{G}_-) U_-^* \end{bmatrix} \right\| \\
&\le \max_\pm \|U_\pm\|^2 \, \|G_\pm\| \, \mathbf{u} \\
&\le (1 + \varepsilon'/3)^2 \left(\frac{R' + \varepsilon'}{(1-\varepsilon')^2} + \frac{R}{2} \right) \mathbf{u} \\
&\le 2R\mathbf{u}
\end{aligned}$$

Finally, we bound (24). Since $\begin{bmatrix} V_+ & V_- \end{bmatrix}$ is unitary, conjugation by it does not change norms.

$$\begin{aligned}
\|A - (F_+ + F_-)\| &= \left\| \begin{bmatrix} V_+ & V_- \end{bmatrix}^* (A - (F_+ + F_-)) \begin{bmatrix} V_+ & V_- \end{bmatrix} \right\| \\
&= \left\| \begin{bmatrix} V_+^* A V_+ & \\ & V_-^* A V_- \end{bmatrix} - \begin{bmatrix} U_+ G_+ U_+^* & \\ & U_- G_- U_-^* \end{bmatrix} \right\| \\
&= \max_{\pm} \left\| V_\pm^* A V_\pm - U_\pm G_\pm U_\pm^* \right\| \\
&= \max_{\pm} \left\| V_\pm^* A V_\pm - U_\pm \left(D_\pm \pm \frac{R}{2} I \right) U_\pm^* \right\| \\
&= \max_{\pm} \left\| V_\pm^* A V_\pm - U_\pm D_\pm U_\pm^* \mp \frac{R}{2} U_\pm U_\pm^* \right\|
\end{aligned}$$

We bound that final expression as the sum of three terms

$$\left\| V_\pm^* A V_\pm - U_\pm D_\pm U_\pm^* \mp \frac{R}{2} U_\pm U_\pm^* \right\| \le \left\| V_\pm^* A V_\pm \mp \frac{R}{2} I - A_\pm \right\| \tag{28}$$

$$+ \left\| A_\pm - U_\pm D_\pm U_\pm^* \right\| \tag{29}$$

$$+ \left\| \frac{R}{2} U_\pm U_\pm^* - \frac{R}{2} I \right\| \tag{30}$$

We obtained a bound for (28) in (21), which is $(2 + 2\eta)R\eta$. Event III is exactly that (29) is bounded by $(R' + R_0)\varepsilon'$ and that (30) is bounded by $\frac{R}{2}[(1 + \varepsilon'/3)^2 - 1] \le 0.34\varepsilon' R$. The final residual then is bounded by the sum

$$\begin{aligned}
(28) + (29) + (30) + (25) + (26) &\le (4 + 10\varepsilon)R\eta + (29) + (30) \\
&\le (4 + 10\varepsilon)R\eta + (R' + R_0)\varepsilon' + 0.34R\varepsilon' \\
&\le \left(\frac{4 + 10\varepsilon}{5\ell} + (1 + \mathbf{u}) \left(\frac{1}{2} + \frac{2}{\ell} \right) + 0.34 \right) R\varepsilon' + R_0\varepsilon' \\
&\le (R + R_0)\varepsilon' \\
&\le (R + R_0)\varepsilon
\end{aligned}$$

as required. □

Theorem 4.2 (Main guarantee)**.** *Let A be an $n \times n$ Hermitian matrix, $\theta \in (16ne^{-7.4n}, 1)$, and $\varepsilon \in (0, 2^{-15})$. Using*

$$\begin{aligned}
\lg(1/\mathbf{u}) \ge \lg(1/\varepsilon) &+ \lg \max\left(n^{1.5} \mu_{\texttt{QR}}(n), n^{1.5} \sqrt{n} c_{\texttt{N}}, n^{4.5} \mu_{\texttt{MM}}(n) \right) \\
&+ 2 \lg \lg(1/\varepsilon) + 1.5 \lg(1/\theta) + \lg \lg(n \lg(1/\varepsilon)/\theta) + 23
\end{aligned}$$

bits of precision and

$$O\big(\log(1/\varepsilon)(\log(n) + \log(1/\theta) + \log\log(1/\varepsilon)) T_{\texttt{MM}}(n) + \log(1/\varepsilon) T_{\texttt{QR}}(n) \big)$$

floating point operations, the function call

$$[U, D] = \texttt{EIGH}(A, \varepsilon, \theta) = \texttt{EIGH-INTERNAL}(A, \|A\|, \|A\|, \varepsilon, \lceil \lg(1/\varepsilon) \rceil + 5, \theta/4n)$$

achieves the following guarantee with probability at least $1 - \theta$. First, $\|A - UDU^*\| \le 2\varepsilon \|A\|$ and second, all the singular values of U lie in the interval $[1 - \varepsilon/3, 1 + \varepsilon/3]$.

Proof. Consider the computation tree associated with the recursive structure of this algorithm. It is a binary tree where each node denotes a call to `EIGH-INTERNAL`. The number of children of each node is the number of recursive calls to `EIGH-INTERNAL` it makes. Let d be the depth of the tree. Let X be the set of all nodes and X_D the set of nodes in which deflate is run. The values of $R_0 = \|A\|$ and $\rho = \theta/4n$ used throughout do not change, so we may use those variables without ambiguity. For the other parameters, let $R_x, \varepsilon_x, \ell_x$ denote the values of R, ε, ℓ input to node $x \in X$. We omit the subscript when x is the root node. In particular, $R = \|A\|$ and $\ell = \lceil \lg(1/\varepsilon) \rceil + 5$. Set $w_x = \theta \cdot R_x/(2 \cdot n_x \cdot nd)$. Let β_x, t_x, η_x be the values as specified by Lemma 4.1 and `EIGH-INTERNAL`. Note that for each node, Event III is implied by Events I-III for its children. Further note Event III is trivially satisfied for leaves. Consequently, it suffices to bound the probabilities of Events I and II for each node. For $x \in X_D$, Lemma 4.1 implies this probability is

$$1 - \frac{n_x w_x}{R_x} - 2\rho_x = 1 - \frac{\theta}{2nd} - 2\rho.$$

For $x \in X \backslash X_D$, only Event I is necessary, so the relevant probability is just

$$1 - \frac{n_x w_x}{R_x} = 1 - \frac{\theta}{2nd}.$$

Note that X_D are the internal nodes of the computational tree with two children. Since this tree has at most n leaves, this means $|X_D| = n - 1$. The number of nodes at each depth is at most n, so $|X| \leq nd$. So by union bound the failure probability is at most

$$\begin{aligned}\left(\sum_{x \in X} \frac{\theta}{2nd}\right) + \left(\sum_{x \in X_D} 2\rho\right) &\leq |X| \cdot \frac{\theta}{2nd} + |X_D| \cdot 2\rho \\ &\leq \frac{\theta}{2} + (n-1) \cdot 2 \cdot \frac{\theta}{4n} \\ &\leq \theta.\end{aligned}$$

We now turn our attention to the number of bits used. If y is a child of x, then $\ell_y = \ell_x + 1$ and

$$R_y \leq (1+\mathbf{u})\left(\frac{1}{2} + \frac{2}{\ell_x}\right) R_x = (1+\mathbf{u}) \cdot \frac{1}{2} \cdot \frac{\ell_x + 4}{\ell_x} \cdot R_x, \tag{31}$$

$$\varepsilon_y \geq (1-\mathbf{u})\left(1 - \frac{1}{\ell_x}\right)\varepsilon_x = (1-\mathbf{u}) \cdot \frac{\ell_x - 1}{\ell_y - 1} \cdot \varepsilon_x. \tag{32}$$

In both cases, we obtain a telescoping product for a node y at depth k giving

$$R_y \leq (1+\mathbf{u})^k \cdot 2^{-k} \cdot \frac{\ell + k}{\ell} \cdot \frac{\ell + k + 1}{\ell + 1} \cdot \frac{\ell + k + 2}{\ell + 2} \cdot \frac{\ell + k + 3}{\ell + 3} \cdot R, \tag{33}$$

$$\leq (1+\mathbf{u})^k \cdot 2^{-k} \cdot (1 + k/\ell)^4 \cdot R \tag{34}$$

$$\varepsilon_y \geq (1-\mathbf{u})^k \cdot \frac{\ell - 1}{\ell + k - 1} \cdot \varepsilon. \tag{35}$$

We claim that the depth of the tree d is no more than ℓ. To see this, note that at depth $k = \ell = \lceil \lg(1/\varepsilon) \rceil + 5$, we have $R_y \leq \varepsilon R = \varepsilon R_0$ which guarantees the algorithm terminates. We therefore obtain $\varepsilon_y \geq (1-\mathbf{u})^\ell \cdot \frac{\ell-1}{2\ell-1}\varepsilon \geq \varepsilon/3$ and $\ell_y \leq 2\ell$ for all $y \in X$. We need to bound the constraints on $\mathbf{u}$ from Lemma 4.1 at each node. We first compute bounds in terms of β_x.

$$\begin{aligned}\min_{x \in X} \mathbf{u}_{\texttt{DEFLATE}}(\beta_x, n) &= \min_{x \in X} \frac{1}{4\mu_{\texttt{QR}}(n)(n_x) + 2\sqrt{n}c_{\texttt{N}}(n_x) + 2\mu_{\texttt{MM}}(n)(n_x)} \cdot \beta_x \\ &\geq \frac{1}{4\mu_{\texttt{QR}}(n)(n) + 2\sqrt{n}c_{\texttt{N}}(n) + 2\mu_{\texttt{MM}}(n)(n)} \cdot \min_{x \in X} \beta_x\end{aligned} \tag{36}$$

$$\begin{aligned}
\min_{x\in X} \mathbf{u}_{\mathtt{SIGN}}(\beta_x/2, 2R_x, 2/w_x) &= \min_{x\in X} \frac{w_x}{4R_x} \cdot \frac{1}{4\max\Big(N_{\mathtt{SCALAR}}\Big(\frac{w_x}{4R_x}, \frac{\varepsilon_x}{8n_x}\Big)\cdot n_x\mu_{\mathtt{g}}(n_x, 1.1), n_x^2\Big)} \cdot \frac{\beta_x}{2} \\
&= \frac{1}{64}\min_{x\in X} \frac{\theta}{n_x n d} \cdot \frac{1}{\max\Big(N_{\mathtt{SCALAR}}\Big(\frac{\theta}{8n_x nd}, \frac{\varepsilon_x}{8n_x}\Big)\cdot n_x\mu_{\mathtt{g}}(n_x, 1.1), n_x^2\Big)} \cdot \beta_x \\
&\geq \frac{\theta}{64n^2 d} \cdot \frac{1}{\max\Big(N_{\mathtt{SCALAR}}\Big(\frac{\theta}{8n^2 d}, \frac{\varepsilon/3}{8n}\Big)\cdot n\mu_{\mathtt{g}}(n, 1.1), n^2\Big)} \cdot \min_{x\in X}\beta_x \\
&\geq \frac{\theta}{64n^3\mu_{\mathtt{g}}(n,1.1)} \cdot \frac{1}{\ell \cdot N_{\mathtt{SCALAR}}\big(\frac{\theta}{8n^2\ell}, \frac{\varepsilon}{24n}\big)} \cdot \min_{x\in X}\beta_x \\
&\geq \frac{\theta}{64n^3\mu_{\mathtt{g}}(n,1.1)} \cdot \frac{1}{\ell\cdot(2.5 + 2\lg(8n^2\ell/\theta) + \lg\lg(24n/\varepsilon))} \cdot \min_{x\in X}\beta_x \\
&\geq \frac{\theta}{64n^3\mu_{\mathtt{g}}(n,1.1)} \cdot \frac{1}{\ell\cdot\lg(363n^4\ell^2\theta^{-2}\lg(24n/\varepsilon))} \cdot \min_{x\in X}\beta_x
\end{aligned} \tag{37}$$

We now bound $\min_{x\in X}\beta_x$. First note

$$\min_{x\in X}\eta_x = \min_{x\in X}\frac{\varepsilon'_x}{5\ell_x} \geq \frac{\varepsilon/3}{10\ell}. \tag{38}$$

Then note that

$$n_x\cdot t_x = n_x\cdot(12\sqrt{2} + 6\sqrt{\log(4/\rho)/n_x}) = 12\sqrt{2}n_x + 6\sqrt{n_x\log(16n/\theta)}$$

is monotonically increasing in n_x, so the maximum is obtained for $n_x = n$. Apply that observation with (38) and $\theta \geq 16ne^{-7.4n}$ to obtain

$$\begin{aligned}
\min_{x\in X}\beta_x = \min_{x\in X}\frac{\rho^{1/2}\eta_x}{3n_x t_x} &= \frac{\theta^{1/2}}{6n^{1/2}}\cdot\min_{x\in X}\frac{\eta_x}{n_x t_x} \\
&\geq \frac{\theta^{1/2}}{6n^{1/2}}\cdot\frac{1}{12\sqrt{2}n + 6\sqrt{n\log(16n/\theta)}}\min_{x\in X}\eta_x \\
&\geq \frac{\theta^{1/2}}{6n^{1/2}}\cdot\frac{1}{12\sqrt{2}n + 6\sqrt{n\log(16n/\theta)}}\cdot\frac{\varepsilon}{30\ell} \\
&\geq \frac{1}{6000}\frac{\theta^{1/2}\varepsilon}{n^{3/2}\ell}.
\end{aligned} \tag{39}$$

Combining (39) with (36) gives

$$\min_{x\in X}\mathbf{u}_{\mathtt{DEFLATE}}(n_x, \eta_x, \rho_x) \geq 2\times 10^{-5}\cdot\frac{\theta^{1/2}\varepsilon}{\ell n^{3/2}\max(\mu_{\mathtt{MM}}(n), \sqrt{n}c_{\mathtt{N}}, \mu_{\mathtt{QR}}(n))}.$$

Combining (39) with (37) and $\mathbf{u}_{\mathtt{g}}(n, 1.1) \leq 2\mu_{\mathtt{MM}}(n)$ for $\mu_{\mathtt{MM}}(n) \geq 10$ from Lemma 2.1 gives

$$\begin{aligned}
\min_{x\in X}\mathbf{u}_{\mathtt{SIGN}}(\beta_x/2, 2R_x, 2/w_x) &\geq 2.6\times 10^{-6}\cdot\frac{\theta^{3/2}\varepsilon}{\ell^2 n^{9/2}\mu_{\mathtt{g}}(n,1.1)}\cdot\frac{1}{\lg(363n^4\ell^2\theta^{-2}\lg(24n/\varepsilon))} \\
&\geq 1.3\times 10^{-6}\cdot\frac{\theta^{3/2}\varepsilon}{\ell^2 n^{9/2}\mu_{\mathtt{MM}}(n)}\cdot\frac{1}{\lg(363n^4\ell^2\theta^{-2}\lg(24n/\varepsilon))}
\end{aligned} \tag{40}$$

By taking logs, one sees that it suffices to take $\lg(1/\mathbf{u})$ to be at least the larger of the following two quantities.

$$\begin{aligned}
&\lg(1/\mathbf{u}) \geq \lg(1/\varepsilon) + \lg\Big(n^{3/2}\max\big(\sqrt{n}c_{\mathtt{N}}, \mu_{\mathtt{QR}}(n)\big)\Big) + \lg(\ell) + 0.5\lg(1/\theta) + 15.7, \\
&\lg(1/\mathbf{u}) \geq \lg(1/\varepsilon) + \lg(n^{9/2}\mu_{\mathtt{MM}}(n)) + 2\lg(\ell) + 1.5\lg(1/\theta) + \lg\lg\big(363n^4\ell^2\lg(24n/\varepsilon)/\theta^2\big) + 19.6.
\end{aligned} \tag{41}$$

Plugging in $\ell = \lceil \lg(1/\varepsilon) \rceil + 5$ and simplifying gives the final precision result. We conclude with analysis of the runtime. The number of floating point operations used by `SIGN` in node x is

$$\begin{aligned} O\left(\left(\lg\left(\frac{R_x}{w_x}\right) + \lg\lg\left(\frac{n_x}{\varepsilon_x}\right)\right) T_{\texttt{MM}}(n_x)\right) &= O\left(\left(\lg\left(\frac{n_x n d}{\theta}\right) + \lg\lg\left(\frac{n}{\varepsilon}\right)\right) T_{\texttt{MM}}(n_x)\right) \\ &= O\left(\left(\lg\left(\frac{n}{\theta}\right) + \lg\lg\left(\frac{1}{\varepsilon}\right)\right) T_{\texttt{MM}}(n_x)\right) \\ &=: f_{\texttt{SIGN}}(n_x) \end{aligned} \tag{42}$$

The number of floating point operations used by `DEFLATE` in node x is

$$T_{\texttt{QR}}(n_x) + T_{\texttt{MM}}(n_x) + O(n^2) =: f_{\texttt{DEFLATE}}(n_x) \tag{43}$$

The number of floating point operations used by all other steps in the algorithm are dominated by these terms. Let X_k be the set of nodes at depth k. Then $\sum_{x \in X_k} n_x \leq n$. Note $f_{\texttt{SIGN}}$ and $f_{\texttt{DEFLATE}}$ are convex, so

$$\sum_{x \in X_k} \left(f_{\texttt{SIGN}}(n_x) + f_{\texttt{DEFLATE}}(n_x)\right) \leq f_{\texttt{SIGN}}(n) + f_{\texttt{DEFLATE}}(n).$$

Summing over $k = 1, \cdots, d = \lg\lceil 1/\varepsilon \rceil + 5$ gives the final result. □

We conclude with a final remark about removing the $\lg(1/\theta)$ terms from the bit requirement.

Remark 10 (Boosting success probability)**.** One can estimate the residual in $O(n^2)$ time by computing

$$\|(UDU^* - A)x\|$$

for randomly selected x. If the residual is too high, one can rerun `EIG` with fresh randomness. This allows one to boost the probability of success. If the desired failure probability is θ', one can take $\theta = 1/2$ and repeat the call to `EIGH` $\lg(1/\theta')$ times. So at the expense of a longer runtime, one can remove the $\lg(1/\theta)$ terms from the precision bound.

References

[ABB+18] 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.

[BD73] A. N. Beavers and E. D. Denman. A computational method for eigenvalues and eigenvectors of a matrix with real eigenvalues. *Numerische Mathematik*, 21:389–396, 1973.

[BD74] A.N. Beavers and E.D. Denman. A new similarity transformation method for eigenvalues and eigenvectors. *Mathematical Biosciences*, 21(1):143–169, 1974.

[BD93] Zhaojun Bai and James W. Demmel. Design of a parallel nonsymmetric eigenroutine toolbox, part i. Technical Report UCB/CSD-92-718, EECS Department, University of California, Berkeley, Feb 1993.

[BDD11] Grey Ballard, James Demmel, and Ioana Dumitriu. Minimizing communication for eigenproblems and the singular value decomposition. Technical Report UCB/EECS-2011-14, Feb 2011.

[BDG97] Zhaojun Bai, James Demmel, and Ming Gu. An inverse free parallel spectral divide and conquer algorithm for nonsymmetric eigenproblems. *Numerische Mathematik*, 1997.

[BGVKS20] Jess Banks, Jorge Garza-Vargas, Archit Kulkarni, and Nikhil Srivastava. Pseudospectral shattering, the sign function, and diagonalization in nearly matrix multiplication time. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 529–540, 2020.

[BGVS22] Jess Banks, Jorge Garza-Vargas, and Nikhil Srivastava. Global convergence of hessenberg shifted qr ii: Numerical stability. *arXiv preprent arXiv:2205.06810*, 2022.

[BGVS23] Jess Banks, Jorge Garza-Vargas, and Nikhil Srivastava. Global Convergence of Hessenberg Shifted QR I: Exact Arithmetic. *arXiv preprint arXiv:2111.07976*, 2023.

[BKMS21] Jess Banks, Archit Kulkarni, Satyaki Mukherjee, and Nikhil Srivastava. Gaussian regularization of the pseudospectrum and davies' conjecture. *Communications on Pure and Applied Mathematics*, 74(10):2114–2131, 2021.

[BX08] Ralph Byers and Hongguo Xu. A new scaling for Newton's iteration for the polar decomposition and its backward stability. *SIAM Journal on Matrix Analysis and Applications*, 30(2):822–843, 2008.

[CCNS14] Jie Chen, Edmond Chow, and The Newton-Schulz. A stable scaling of newton-schulz for improving the sign function computation of a hermitian matrix. 2014.

[DB76] Eugene D. Denman and Alex N. Beavers. The matrix sign function and computations in systems. *Applied Mathematics and Computation*, 2(1):63–94, 1976.

[DDH07] James Demmel, Ioana Dumitriu, and Olga Holtz. Fast linear algebra is stable. *Numerische Mathematik*, 108(1):59–91, October 2007.

[DDHK07] James Demmel, Ioana Dumitriu, Olga Holtz, and Robert Kleinberg. Fast matrix multiplication is stable. *Numerische Mathematik*, 106(2):199–224, February 2007.

[DT71] TJ Dekker and JF Traub. The shifted qr algorithm for hermitian matrices. *Linear Algebra Appl*, 4:137–154, 1971.

[Ede88] Alan Edelman. Eigenvalues and condition numbers of random matrices. *SIAM Journal on Matrix Analysis and Applications*, 9(4):543–560, 1988.

[Fra61] John GF Francis. The QR transformation a unitary analogue to the LR transformation—Part 1. *The Computer Journal*, 4(3):265–271, 1961.

[Fra62] John GF Francis. The QR transformation—Part 2. *The Computer Journal*, 4(4):332–345, 1962.

[Gan90] Walter Gander. Algorithms for the polar decomposition. *SIAM J. Sci. Comput.*, 11:1102–1115, 1990.

[HP78] W. Hoffmann and B. N. Parlett. A new proof of global convergence for the tridiagonal ql algorithm. *SIAM Journal on Numerical Analysis*, 15(5):929–937, 1978.

[KL95] Charles S. Kenney and Alan J Laub. The matrix sign function. *IEEE Transactions on Automatic Control*, 40(8):1330–1348, 1995.

[KZ03] Andrzej Kiełbasiński and Krystyna Zietak. Numerical behaviour of higham's scaled method for polar decomposition. *Numerical Algorithms*, 32:105–140, 2003.

[KZ09] Andrzej Kielbasinski and Krystyna Zietak. Note on "a new scaling for newton's iteration for the polar decomposition and its backward stability" by r. byers and h. xu*. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1538–1539, 2009. Copyright - Copyright Society for Industrial and Applied Mathematics 2009; Document feature - Equations; ; Last updated - 2023-12-04.

[NBG10] 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.

[NF16] Yuji Nakatsukasa and Roland W. Freund. Computing fundamental matrix decompositions accurately via the matrix sign function in two iterations: The power of Zolotarev's functions. *SIAM Review*, 58(3):461–493, 2016.

[NH12] Yuji Nakatsukasa and Nicholas J. Higham. Backward stability of iterations for computing the polar decomposition. *SIAM Journal on Matrix Analysis and Applications*, 33(2):460–479, 2012.

[NH13] 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.

[SML24] Aleksandros Sobczyk, Marko Mladenović, and Mathieu Luisier. Invariant subspaces and pca in nearly matrix multiplication time, 2024.

[Wil68] J.H. Wilkinson. Global convergene of tridiagonal qr algorithm with origin shifts. *Linear Algebra and its Applications*, 1(3):409–420, 1968.