A FAST CONTOUR-INTEGRAL EIGENSOLVER FOR NON-HERMITIAN MATRICES

XIN YE*, JIANLIN XIA*, RAYMOND H. CHAN†, STEPHEN CAULEY‡, AND VENKATARAMANAN BALAKRISHNAN§

Abstract. We present a fast contour-integral eigensolver for finding selected or all the eigenpairs of a non-Hermitian matrix based on a series of analytical and computational techniques, such as the analysis of filter functions, quick and reliable eigenvalue count via low-accuracy matrix approximations, and fast shifted factorization update. The quality of some quadrature rules for approximating a relevant contour integral is analyzed. We show that a filter function based on the Trapezoidal rule has nearly optimal decay in the complex plane away from the unit circle (as the mapped contour), and is superior to the Gauss-Legendre rule. The eigensolver needs to count the eigenvalues inside a contour. We justify the feasibility of using low-accuracy matrix approximations for the quick and reliable count. Both deterministic and probabilistic studies are given. With high probabilities, the matrix approximations give counts very close to the exact one. Our eigensolver is built upon an accelerated FEAST algorithm. Both the eigenvalue count and the FEAST eigenvalue solution need to solve linear systems with multiple shifts and right-hand sides. For this purpose and also to conveniently control the approximation accuracy, we use a type of rank structured approximations and show how to update the factorization for varying shifts. The eigensolver may be used to find a large number of eigenvalues, where a search region is then partitioned into subregions. We give an optimal threshold for the number of eigenvalues inside each bottom level subregion so as to minimize the complexity, which is $\mathcal{O}(rn^2) + \mathcal{O}(r^2n)$ to find all the eigenpairs of an order-n matrix with maximum off-diagonal rank or numerical rank r. Numerical tests demonstrate the efficiency and accuracy and confirm the benefit of our acceleration techniques.

Key words. contour-integral eigensolver, quadrature rule, low-accuracy matrix approximation, eigenvalue count, rank structure, shifted factorization update.

AMS subject classifications. 15A18, 65D30, 65F05, 65F15, 65F30.

1. Introduction.

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

$$Ax = \lambda x, \quad A \in \mathbb{C}^{n\times n}, \tag{1.1}$$

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

$$A = X\Lambda X^{-1}, \tag{1.2}$$

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

Classical methods for solving (1.1) include power iterations, inverse iterations, and QR iterations. The main operations involve matrix-vector multiplications, linear

*Department of Mathematics, Purdue University, West Lafayette, IN 47907, USA (ye83@purdue.edu, xiaj@math.purdue.edu). The research of Jianlin Xia was supported in part by an NSF CAREER Award DMS-1255416.

†Department of Mathematics, the Chinese University of Hong Kong, Shatin, NT, Hong Kong (rchan@math.cuhk.edu.hk).

‡Athinoula A. Martinos Center for Biomedical Imaging, Department of Radiology, Massachusetts General Hospital, Harvard University, Charlestown, MA 02129, USA (stcauley@nmr.mgh.harvard.edu).

§School Of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA (ragu@ecn.purdue.edu).

system solutions, or QR factorizations. In QR iterations, A is usually first reduced to an upper Hessenberg form.

Recently, a class of contour-integral based eigensolvers have been developed to find a partial spectrum. They have some very appealing features, such as the robustness in terms of convergence rates, the natural accommodation of eigenvalue multiplicity, and the nice scalability. In [43, 44], the Sakurai-Sugiura (SS) method is proposed to reduce a generalized eigenvalue problem to a smaller one with Hankel matrices, and later in [45] a stable version called CIRR is introduced by combining the contour-integral technique with the Rayleigh-Ritz procedure. The FEAST algorithm is first proposed in [41] for Hermitian matrices, where a spectral projector is constructed via the integration of the resolvent of a matrix, followed by projected subspace iterations. Some non-Hermitian FEAST methods can be found in [33, 36, 48, 62]. Contour-integral eigensolvers utilize a filter function, whose quality is a key factor of the effectiveness of the eigenvalue solutions. Rational filter functions are discussed in [25, 49]. Other types of filter functions can be obtained via the solution of optimization or least-squares problems [50, 53].

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

$$\phi(z) = \frac{1}{2\pi \mathbf{i}} \int_\Gamma \frac{1}{\mu - z} \mathrm{d}\mu, \quad z \notin \Gamma, \quad \mathbf{i} = \sqrt{-1}. \tag{1.3}$$

A spectral projector to the desired eigenspace $\operatorname{span}\{x_1, x_2, \ldots, x_s\}$ is constructed based on Cauchy's residue theorem [47] in complex analysis:

$$\begin{aligned}\Phi \equiv \phi(A) &= \frac{1}{2\pi \mathbf{i}} \int_\Gamma (\mu I - A)^{-1} \mathrm{d}\mu = \frac{1}{2\pi \mathbf{i}} \int_\Gamma (\mu I - X\Lambda X^{-1})^{-1} \mathrm{d}\mu \\ &= X \left(\frac{1}{2\pi \mathbf{i}} \int_\Gamma (\mu I - \Lambda)^{-1} \mathrm{d}\nu \right) X^{-1} = X \begin{pmatrix} I_s & \\ & 0 \end{pmatrix} X^{-1}.\end{aligned} \tag{1.4}$$

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

$$Z = \Phi Y = \frac{1}{2\pi \mathbf{i}} \int_\Gamma (\mu I - A)^{-1} Y \mathrm{d}\mu. \tag{1.5}$$

This needs to evaluate the contour integral, which is done by numerical quadratures. In the process, linear systems are solved for $(\mu I - A)^{-1} Y$. After Z is evaluated, it is used as starting vectors in projected subspace iterations to compute the desired eigenpairs. The accuracy of the quadrature approximation is essential to the convergence rate of the subspace iterations.

In the FEAST algorithm and other contour-integral eigensolvers, it needs an estimate of the number of eigenvalues of A inside Γ, denoted $\#_\Lambda(A, \Gamma)$, which is sometimes assumed to be known in advance. Some estimation methods have been proposed in [40, 43, 62] based on stochastic strategies.

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

Here, we seek to design a fast contour-integral eigensolver. There are three major tasks. (1) One task is to analyze some numerical quadrature rules for the design of filter functions and understand their quality. This helps us choose an appropriate quadrature with justified optimality for the contour integration. (2) The next task is to show why some low-accuracy approximations can be used to quickly and reliably count the eigenvalues inside Γ. Both deterministic and probabilistic justifications are included. (3) The third task is to present a fast algorithm to find selected or all eigenpairs of A based on the analysis and a type of fast shifted factorizations. Some tools to use include structured matrices and shifted structured factorization update. The matrices we consider include some rank structured ones and more general cases. Previously, for non-Hermitian rank-structured eigenvalue problems, fast QR iterations are designed for special cases such as companion matrices [5, 10, 14, 51]. Here, we consider more general cases.

Our first task is to perform some analysis on the quality of some commonly used quadrature rules for approximating (1.3). The quadrature approximation is expected to be not too far away from 1 for z inside Γ and not too close to Γ, and to decay quickly for z outside and away from Γ. Existing FEAST algorithms usually use the Gauss-Legendre rule [36, 41, 62], though recent numerical observations find that the Trapezoidal rule may be preferable [48, 50]. Here, we analytically show that the Trapezoidal rule is much superior in the sense that it yields quadrature approximations with nearly optimal decay outside the unit circle (as the mapped contour Γ) in the complex plane. Thus, the Trapezoidal rule will be used in our eigensolver. We would like to mention that interesting analysis has been performed for approximating the operator exponent by contour integration of the matrix resolvent [19], where an exponentially convergent sinc quadrature rule is proposed and is also applicable to other common kernel functions such as $1/|x-y|$ and $\log|x-y|$ [29].

The next task is to show the feasibility of using low-accuracy matrix approximations for the quick and reliable estimate of $\#_\Lambda(A,\Gamma)$. The eigenvalue count involves quadrature approximations similar to (1.5) and needs linear solutions with multiple shifts and multiple right-hand sides. Certain low-accuracy matrix approximations with fast solutions enable us to quickly estimate $\#_\Lambda(A,\Gamma)$, as long as the count is unchanged or remains close. We show that, when Γ is not too close to the eigenvalues inside it, A can be approximated by a matrix $\tilde{A}$ with a low accuracy so that the eigenvalue count remains the same ($\#_\Lambda(A,\Gamma) = \#_\Lambda(\tilde{A},\Gamma)$). The farther away Γ is from the eigenvalues, the lower the approximation accuracy of $\tilde{A}$ is allowed to be. On the other hand, if there are eigenvalues close to Γ, we use probabilistic methods to justify the reliability of $\#_\Lambda(\tilde{A},\Gamma)$. We show that, for some situations, with high probabilities, the eigenvalue count is off by only a very small number α. Roughly speaking, the probability of miscounting the eigenvalues by α decays exponentially with α. This is sufficient for us since we do not need the count to be exact.

Our choice of $\tilde{A}$ is based on rank structured forms, since it is convenient to control the approximation accuracy and also quick to perform structured direct solutions with multiple right-hand sides and even multiple shifts. (Note that the approximation analysis for the eigenvalue count is not restricted to rank structured forms.) The rank structured forms involve low-rank approximations of some off-diagonal blocks. Examples of such forms include $\mathcal{H}$ [26], $\mathcal{H}^2$ [27, 6], and hierarchically semiseparable (HSS)

[8, 60] representations. For matrices with small off-diagonal ranks or numerical ranks, fast direct solvers exist. Such matrices widely appear in numerical computations, such as polynomial root finding, Toeplitz problems, and some discretized problems. Here, even if A itself is not rank structured, we may still use a rank structured approximation $\tilde{A}$ to quickly count the eigenvalues.

Our third task is then to design a fast contour-integral eigensolver for rank structured A and even more general cases. This is based on fast factorizations of rank structured approximations, as well as fast factorization update with varying shifts for the quadrature evaluations. We will adaptively choose the approximation accuracy to balance the efficiency and the accuracy. Previously, for Hermitian HSS matrices, a shifted structured LDL factorization update is designed [4, 56]. Here, we further show that, even for non-Hermitian HSS matrices, we can still update the factorization for varying shifts so as to save nearly 40% of the factorization cost for each shift.

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

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

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

The outline of the remaining presentation is as follows. In Section 2, we show our analysis of the quadrature rules for the filter function design. The idea of low-accuracy matrix approximations for the eigenvalue count is given in Section 3. Our fast contour-integral eigensolver is presented in Section 4. Section 5 gives the numerical experiments to illustrate the efficiency and accuracy.

The following notation is used throughout the paper:

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

2. Analysis of quadrature rules for filter function design.

In contour-integral eigensolvers, the quality of the quadrature approximation is critical for the accuracy of the eigenvalue computation. Here, we first perform analysis on some quadrature rules so as to choose an appropriate one for (1.3) and (1.5).

If the contour Γ has a parametrization $\Gamma = \{h(t) : t \in [a, b)\}$, then (1.3) becomes

$$\phi(z) = \frac{1}{2\pi\mathbf{i}} \int_a^b \frac{h'(t)}{h(t) - z} \mathrm{d}t. \tag{2.1}$$

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

$$\tilde{\phi}(z) = \frac{1}{2\pi\mathbf{i}} \sum_{j=1}^{q} w_j \frac{h'(t_j)}{h(t_j) - z}, \tag{2.2}$$

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

We focus on the case when Γ is a circle $\mathcal{C}_\gamma(z_0)$. Specifically when $\mathcal{C}_\gamma(z_0)$ is the unit circle $\mathcal{C}_1(0)$, write $\phi(z)$ in (1.3) and (2.1) as $\phi_0(z)$ and write $\tilde{\phi}(z)$ in (2.2) as $\tilde{\phi}_0(z)$. Then $\phi(z)$ can be transformed directly into $\phi_0(z)$ by

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

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

$$\phi_0(z) = \frac{1}{2} \int_{-1}^{1} \frac{e^{\mathbf{i}\pi t}}{e^{\mathbf{i}\pi t} - z} \mathrm{d}t, \tag{2.3}$$

$$\tilde{\phi}_0(z) = \frac{1}{2} \sum_{j=1}^{q} w_j \frac{e^{\mathbf{i}\pi t_j}}{e^{\mathbf{i}\pi t_j} - z} \equiv \frac{1}{2} \sum_{j=1}^{q} \frac{w_j z_j}{z_j - z}, \tag{2.4}$$

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

$$z_j = e^{\mathbf{i}\pi t_j}, \quad j = 1, 2, \ldots, q. \tag{2.5}$$

Rewrite (2.4) as a rational form

$$\tilde{\phi}_0(z) \equiv \frac{f(z)}{g(z)} \quad \text{with} \quad g(z) = \prod_{j=1}^{q} (z - z_j), \tag{2.6}$$

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

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

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

PROPOSITION 2.1. *For $z \in \mathbb{C}$ and $\tilde{\phi}_0(z)$ in (2.4) resulting from any interpolatory quadrature rule applied to (2.3),*

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

Proof. For any interpolatory quadrature rule, the weights $\{w_j\}_{j=1}^q$ satisfy $\sum_{j=1}^q w_j = 2$. Then directly from (2.4), we get

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

When $|z| < 1$, we have

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

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

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

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

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

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

□

The proposition means, if z is inside $\mathcal{D}_1(0)$, then $|\tilde{\phi}_0(z)|$ has a lower bound $1/2$. It also has an upper bound $1/\delta$ for z not within a distance δ of any mapped quadrature point z_j. If z is too close to any z_j, then it is possible for $|\tilde{\phi}_0(z)|$ to be large. This can also be observed from Figure 2.1 below.

We then study the decay property of $|\tilde{\phi}_0(z)|$. From the rational form (2.6), we can see that $|\tilde{\phi}_0(z)|$ decays as $\mathcal{O}(|z|^{d_f - d_g})$ for $|z| > 1$. This means, the smaller the degree d_f is, the faster $|\tilde{\phi}_0(z)|$ decays outside $\mathcal{C}_1(0)$ and thus the better the quadrature approximation is. The next theorem compares two popular quadrature rules: the Trapezoidal rule and the Gauss-Legendre quadrature, in terms of the degree d_f.

THEOREM 2.2. *For $\tilde{\phi}_0(z)$ in (2.4)–(2.6), the degree d_f of $f(z)$ satisfies:*

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

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

That is, the Trapezoidal rule gives the optimal d_f.

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

$$d_f \ge 1.$$

Proof. Comparing (2.4) and (2.6) yields

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

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

$$C_{q-k} = \frac{(-1)^k}{2} \sum_{1\le i_1<i_2<\cdots<i_k\le q} (w_{i_1} + w_{i_2} + \cdots + w_{i_k}) z_{i_1} z_{i_2} \cdots z_{i_k}. \tag{2.7}$$

For the Trapezoidal rule, the mapped quadrature nodes z_j in (2.5) satisfy

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

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

$$g(z) = z^q - (-1)^q. \tag{2.8}$$

Since all the weights w_j are equal, (2.7) can be simplified as

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

Note that the part in parenthesis in the above equation is the coefficient of the term z^{q-k} in the polynomial $g(z)$ in (2.6) and also in (2.8). Thus,

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

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

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

$$\sum_{j=1}^q t_j = 0, \quad t_j + t_{q+1-j} = 0, \quad w_j = w_{q+1-j}, \quad 1 \le j \le q, \tag{2.9}$$

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

$$\prod_{j=1}^q z_j = 1, \quad z_j z_{q+1-j} = 1, \quad 1 \le j \le q. \tag{2.10}$$

Define $\mathcal{S}_k = \{(i_1, i_2, \ldots, i_k) : 1 \le i_1 < i_2 < \cdots < i_k \le q\}$ to be the set of index sequences of the summation in (2.7). Then for any $1 \le k \le q-1$, the two sets $\mathcal{S}_k$ and $\mathcal{S}_{q-k}$ have a one-to-one correspondence in the sense that, for any sequence $\sigma \in \mathcal{S}_k$, there is a unique sequence $\beta \in \mathcal{S}_{q-k}$ such that $\sigma \cup \beta = \{1, 2, \ldots, q\}$ and $\sigma \cap \beta = \varnothing$. Therefore, for any $1 \le k \le q-1$, similar to (2.7),

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

We can then use (2.9) and (2.10) to get

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

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

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

The above equation together with $\prod_{j=1}^{q} z_j = 1$ indicate that z_j in (2.5) must be roots of the polynomial $z^q + (-1)^q$. Thus, the roots of Legendre polynomial must be

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

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

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

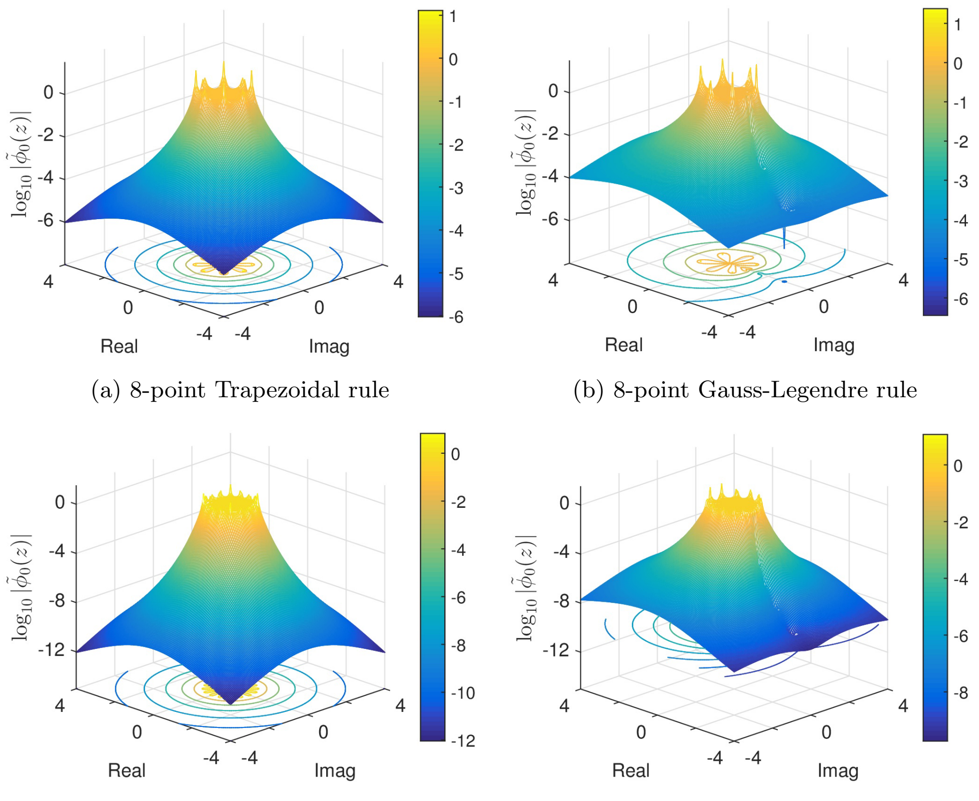
$$|\tilde{\phi}_0(z)| \sim \mathcal{O}(|z|^{-q}) \quad \text{for large } |z|,$$

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

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

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

To illustrate the decay, we plot $|\tilde{\phi}_0(z)|$ from the two quadrature rules with $q = 8$ and 16 in Figure 2.1. Outside $\mathcal{C}_1(0)$, $|\tilde{\phi}_0(z)|$ decays quickly when $|z|$ increases, and moreover, the Trapezoidal rule yields much faster decay than the Gauss-Legendre quadrature. For $q = 8$, $|\tilde{\phi}_0(z)|$ from the Trapezoidal rule is about two orders of magnitude smaller at the corners of the mesh (with $|z|$ not even very large). For $q = 16$, this difference increases to over four orders of magnitude.



(a) 8-point Trapezoidal rule (b) 8-point Gauss-Legendre rule

(c) 16-point Trapezoidal rule (d) 16-point Gauss-Legendre rule

Fig. 2.1. $\log_{10}|\tilde{\phi}_0(z)|$ *on the* $[-4,4]\times[-4,4]$ *mesh obtained with the Trapezoidal rule and the Gauss-Legendre rule.*

Theorem 2.2 and Figure 2.1 also align with the numerical observations in [48, 50]. In [50], an optimization method is used to design filter functions, and in the unit circle case, the best filter function is observed to be precisely the one obtained by applying the Trapezoidal rule. Our analysis provides a theoretical justification.

Therefore, unlike in [36, 41, 62], our eigensolver below uses the Trapezoidal rule to evaluate (1.5) in both the eigenvalue counts and the subspace iterations.

3. Low-accuracy matrix approximation for fast eigenvalue counts.

3.1. Motivations. In contour-integral eigensolvers, it usually requires to know the eigenvalue count $\#_\Lambda(A,\Gamma)$ inside a contour Γ in advance. In our eigensolver in the next section, we may need to estimate eigenvalue counts for many subregions, so it is essential to quickly perform the estimation. Some methods to estimate eigenvalue counts have been proposed in [18, 40, 43, 62] based on stochastic evaluations of the rank or trace [30] of Φ in (1.4). The basic idea is as follows.

According to (1.4), the trace and also the rank of Φ give the exact eigenvalue count $\#_\Lambda(A,\Gamma)$. To estimate the trace, we can pick a small number of random vectors to form an $n\times m$ matrix Y, and compute $Y^T\Phi Y = Y^T Z$, where Z looks like (1.5). Then

$$\#_\Lambda(A,\Gamma)\equiv \mathrm{trace}(\Phi)\approx \frac{1}{m}\,\mathrm{trace}(Y^TZ). \tag{3.1}$$

Theoretically, a small number m can lead to a high probability of accurate estimation. However, since Z in (1.5) is approximated by numerical quadratures, m may not be too small. We can start from a very small m and gradually include more random vectors in Y until a reliable estimate is reached.

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

1. Since we are just interested in the eigenvalue count (at this stage) instead of the precise eigenvalues, as long as the eigenvalues are not too close to Γ, a small perturbation to A does not alter the eigenvalue count.
2. Moreover, in our eigensolver, we will quadsect a search region containing the eigenvalues and only need to know whether the eigenvalue count inside each subregion is much larger than a threshold k or not. Thus, the eigenvalue count does not even have to be very accurate.

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

1. $\#_\Lambda(\tilde{A},\Gamma) \approx \#_\Lambda(A,\Gamma)$ and it is convenient to control how accurately $\tilde{A}$ approximates A;
2. $\#_\Lambda(\tilde{A},\Gamma)$ can be quickly estimated, i.e., the linear systems with multiple shifts and right-hand sides in the quadrature approximation of (1.5) (with A replaced by $\tilde{A}$) can be quickly solved.

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

Before justifying the reliability of our low-accuracy matrix approximation for fast eigenvalue counts, we briefly review HSS representations. The reader is referred to [8, 60] for more details. An HSS matrix $\tilde{A}$ can be recursively bipartitioned following a postordered binary tree T (called HSS tree) with nodes $i = 1, 2, \ldots, t$, where t is the root. Initially, let $D_t = \tilde{A}$. For any nonleaf node i of T, the partition of D_i looks like $D_i = \begin{pmatrix} D_{c_1} & U_{c_1}B_{c_1}V_{c_2}^T \\ U_{c_2}B_{c_2}V_{c_1}^T & D_{c_2} \end{pmatrix}$, where c_1 and c_2 are the children of i. Here, the off-diagonal basis matrices U, V also satisfy a nested property: $U_i = \begin{pmatrix} U_{c_1}R_{c_1} \\ U_{c_2}R_{c_2} \end{pmatrix}$, $V_i = \begin{pmatrix} V_{c_1}W_{c_1} \\ V_{c_2}W_{c_2} \end{pmatrix}$. All such matrices D, U, V, R, W, B are called HSS generators that define $\tilde{A}$. The block row or column corresponding to D_i but excluding D_i is called an HSS block. The HSS matrix has l levels of partition if the HSS tree T has l levels, where the root is at level 0 and the leaves are at level l. The maximum rank (or numerical rank) of all the HSS blocks at all the levels is called the *HSS rank*.

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

3.2. Reliability of eigenvalue count with low-accuracy matrix approximation. In our eigensolver, we will use an HSS form $\tilde{A}$ to approximate A. To see how such an approximation perturbs the eigenvalues, we give following lemma that extends a Hermitian version in [52].

LEMMA 3.1. *Suppose A has simple eigenvalues, and $\tilde{A}$ is an l-level HSS approximation to A in (1.2), so that each off-diagonal block $U_i B_i V_j^T$ of $\tilde{A}$ approximates the corresponding block in A to an accuracy τ which is sufficiently small. Let λ be a simple eigenvalue of A, then there exists an eigenvalue $\tilde{\lambda}$ of $\tilde{A}$ such that*

$$|\lambda - \tilde{\lambda}| \leq \kappa(\lambda) l\tau + \mathcal{O}((l\tau)^2), \tag{3.2}$$

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

The lemma follows directly from the HSS approximation error $\|A - \tilde{A}\|_2 \leq l\tau$ [52] and standard eigenvalue perturbation analysis [3, 13].

Throughout this section, we will assume that all eigenvalues λ_i of A are simple and the perturbation to the matrix is sufficiently small, so as to identify a one-to-one correspondence between the eigenvalues of A and those of its approximation $\tilde{A}$. More specifically, Lemma 3.1 indicates that for any eigenvalue λ_i, there must be a perturbed eigenvalue $\tilde{\lambda}$ within a disk centered at λ_i and with radius $\kappa(\lambda_i) l\tau + \mathcal{O}((l\tau)^2)$. $\tilde{\lambda}$ is unique if this disk is isolated from all the other such disks which yields the desired one-to-one correspondence, we can guarantee this by enforcing the following sufficient condition:

$$\tilde{\kappa} l\tau + \mathcal{O}((l\tau)^2) \leq \frac{1}{2} \min_{1\leq i,j\leq n, i\neq j} |\lambda_i - \lambda_j|, \tag{3.3}$$

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

$$\tilde{\kappa}\|A - \tilde{A}\| + \mathcal{O}(\|A - \tilde{A}\|^2) \leq \frac{1}{2} \min_{1\leq i,j\leq n, i\neq j} |\lambda_i - \lambda_j|. \tag{3.4}$$

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

THEOREM 3.2. *Suppose A has simple eigenvalues λ with $|\lambda| < \rho$, $\tilde{A}$ is an approximation to A satisfying (3.4), and any eigenvalue λ of A and the corresponding eigenvalue $\tilde{\lambda}$ of $\tilde{A}$ satisfy*

$$|\lambda - \tilde{\lambda}| < \delta < \gamma < \rho. \tag{3.5}$$

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

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

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

Proof. Figure 3.1 can be used to assist in the understanding of the results and proof.

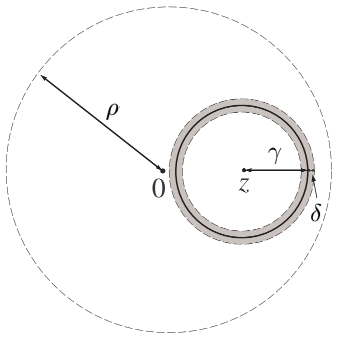


FIG. 3.1. *The annulus region $\mathcal{A}_{\gamma,\delta}(z)$ (shaded area) related to a circle $\mathcal{C}_\gamma(z)$, where the outer disk $\mathcal{D}_\rho(0)$ is where all the eigenvalues are located.*

The first statement can be shown as follows. Since no eigenvalue of A lies inside $\mathcal{A}_{\gamma,\delta}(z)$, any eigenvalue λ satisfies $|\lambda - z| \geq \gamma + \delta$ or $|\lambda - z| \leq \gamma - \delta$. If $|\lambda - z| \geq \gamma + \delta$, according to (3.5),

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

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

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

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

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

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

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

Theorem 3.2 means, if the contour is not too close to the eigenvalues, then $\tilde{A}$ can be used to obtain the exact eigenvalue count. The farther away the contour is from the eigenvalues, the lower accuracy of $\tilde{A}$ can be used. This is especially effective if the eigenvalues are scattered. On the other hand, if the eigenvalue count with $\tilde{A}$ is off by α or more, then there must be at least α eigenvalues within a distance δ of the contour. We then use probabilistic methods to study the error in the count based on the relation between the eigenvalues and $\mathcal{A}_{\gamma,\delta}(z)$.

LEMMA 3.3. *Suppose the eigenvalues λ of A are uniformly i.i.d. in $\mathcal{D}_\rho(0)$. Then for any fixed $z \in \mathbb{C}$ and $\gamma, \delta \in (0, \rho)$, the probability for any λ to lie inside $\mathcal{A}_{\gamma,\delta}(z)$ satisfies*

$$\Pr\{\lambda \in \mathcal{A}_{\gamma,\delta}(z)\} \leq \mathcal{P} \equiv \frac{4\delta \max(\gamma, \delta)}{\rho^2}. \tag{3.6}$$

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

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

If $\gamma < \delta$,

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

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

Lemma 3.3 gives a probability bound for λ to fall inside $\mathcal{A}_{\gamma,\delta}(z)$ when the eigenvalues are random and uniformly distributed in $\mathcal{D}_\rho(0)$. Thus, if A is approximated by $\tilde{A}$ as in Theorem 3.2, then the probability of incorrectly counting λ for $\#_\Lambda(A, \mathcal{C}_\gamma(z))$ is at most $\mathcal{P}$. If $\tilde{A}$ is an HSS approximation as in Lemma 3.1, δ can be chosen to be a strict upper bound for the error in (3.2).

In addition, Lemma 3.3 means that, the larger ρ is or the smaller δ and γ are, the less likely λ falls inside $\mathcal{A}_{\gamma,\delta}(z)$. In particular, later in our eigensolver, since the search region is recursively partitioned into smaller ones, γ gets smaller along the partition and so does the probability $\mathcal{P}$. This combined with Theorem 3.2 means that it is more likely to get reliable eigenvalue counts based on $\tilde{A}$.

Lemma 3.3 assumes the circle $\mathcal{C}_\gamma(z)$ is fixed and the eigenvalues are random. We can also assume an eigenvalue λ is fixed and $\mathcal{C}_\gamma(z)$ is random, and study the probability of $\mathcal{A}_{\gamma,\delta}(z)$ to include λ.

LEMMA 3.4. *Suppose λ is a fixed point in the complex plane, z is uniformly i.i.d. in $\mathcal{D}_\rho(0)$, γ is random and uniformly distributed on $(0, \rho)$, and z and γ are independent. Then for any $\delta \in (0, \rho)$,*

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

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

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

Similarly to the proof of Lemma 3.3, we can get

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

Thus,

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

□

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

THEOREM 3.5. *Suppose the eigenvalues of A are uniformly i.i.d. in $\mathcal{D}_\rho(0)$, and $\tilde{A}$ is an l-level HSS approximation to A as in Lemma 3.1 and satisfies (3.3). Also, suppose $\delta < \rho$ is a strict upper bound for the right-hand side in (3.2) for all the eigenvalues. Let $\mathcal{P}$ be given in (3.6). Then for any integer $\alpha \geq n\mathcal{P}$ and any fixed $z \in \mathbb{C}$ and $\gamma \in (0, \rho)$,*

(3.7)
$$\Pr\{|\#_\Lambda(A, \mathcal{C}_\gamma(z)) - \#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z))| \geq \alpha\} \leq \frac{(\alpha+1)}{\alpha+1-(n+1)\mathcal{P}} \binom{n}{\alpha} \mathcal{P}^\alpha (1-\mathcal{P})^{n-\alpha+1}.$$

Proof. According to Theorem 3.2,

$$\begin{aligned} &\Pr\{|\#_\Lambda(A, \mathcal{C}_\gamma(z)) - \#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z))| \geq \alpha\} \\ &\leq \Pr\{\text{there are at least } \alpha \text{ eigenvalues of } A \text{ in } \mathcal{A}_{\gamma,\delta}(z)\}. \end{aligned} \tag{3.8}$$

Now from Lemmas 3.1 and 3.3, the eigenvalues satisfy

$$\Pr\{\lambda \in \mathcal{A}_{\gamma,\delta}(z)\} \equiv \widehat{\mathcal{P}} \leq \mathcal{P}. \tag{3.9}$$

Let $\hat{y}$ be the number of eigenvalues inside $\mathcal{A}_{\gamma,\delta}(z)$. Since the eigenvalues are i.i.d., $\hat{y}$ has a binomial distribution with parameters $\widehat{\mathcal{P}}$ and n. Also, let y be a binomial random variable with parameters $\mathcal{P}$ and n. Thus, (3.8) and (3.9) yield

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

Since $\alpha \geq n\mathcal{P}$, by [34, Proposition 1], the tail probability of the binomial random variable y is bounded by

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

The result then follows. □

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

To give an idea of this probability bound in (3.7), we show it with different eigenvalue perturbation errors δ. See Table 3.1, where the parameters correspond to a matrix in Example 1 below. Clearly, even though δ is not very small, the probability of miscounting the number of eigenvalues by $\alpha > 2$ is extremely low. When α slightly increases and/or δ decreases, the probability decreases rapidly.

We would also like to mention that Theorem 3.5 is still a very conservative estimate. For example, consider A to be the matrix with size $n = 1600$ in Example 1 below. Let $\tilde{A}$ be an HSS approximation obtained with a relative tolerance $\tau = 10^{-1}, 10^{-2}, \ldots, 10^{-5}$. We run the eigenvalue counts for 100 randomly selected circles. For 57 of the cases, we get the exact counts for all these τ's. For the other cases, $\#_\Lambda(A, \mathcal{C}_\gamma(z_0))$ and $\#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z_0))$ differ by a very small number with $\tau = 10^{-1}$ or 10^{-2}. With smaller τ, exact counts are obtained for almost all the cases. Table 3.2 shows some of the results.

Table 3.1

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

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

Table 3.2

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

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

Table 3.2 also shows the HSS ranks r. Note that when τ reduces from 10^{-1} to 10^{-4} (and all the counts then become exact), the HSS rank increases from $r = 4$ to $r = 11$. Since HSS factorization and solution have asymptotic complexities $\mathcal{O}(r^2 n)$ and $\mathcal{O}(rn)$ [60], respectively, using $\tau = 10^{-1}$ makes the factorization about 7.6 times faster and the solution about 2.8 times faster than using $\tau = 10^{-4}$. For examples where the HSS ranks are higher, the difference is even bigger. See Example 2 below. This clearly demonstrates the benefit of low-accuracy matrix approximations for the eigenvalue count.

4. Our fast contour-integral eigensolver.

In this section, we show the design of our fast contour-integral eigensolver for finding a partial spectrum or the full spectrum of A. We will start from an initial contour that encloses the desired eigenvalues, and then repeatedly *quadsect* the search region into smaller subregions. When the total number of desired eigenvalues is large, a significant amount of efforts is to make sure each subregion includes no more than a certain number of eigenvalues. Before a contour gets very close to the eigenvalues, the discussions in the previous section indicate that we can use low-accuracy approximations to A to obtain a reliable count of the eigenvalues enclosed by the contour.

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

4.1. Review of the non-Hermitian FEAST algorithm. The basic procedure of the FEAST algorithm for non-Hermitian matrices is as follows [33, 36, 41, 48, 62]. Consider the case when Γ in (1.4) is a circle $\mathcal{C}_\gamma(z_0)$. The matrix Z in (1.5) is used to extract an approximation to the eigenspace span$\{x_1, x_2, \ldots, x_s\}$. Φ in (1.4) can be approximated by numerical integration:

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

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

$$\widetilde{Z} = \widetilde{\Phi}Y = \frac{1}{2}\sum_{j=1}^{q} w_j(z_j - z_0)(z_jI - A)^{-1}Y \equiv \frac{1}{2}\sum_{j=1}^{q} c_jS_j, \tag{4.1}$$

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

$$(z_jI - A)S_j = Y, \quad j = 1, 2, \ldots, q. \tag{4.2}$$

Solve the linear systems and evaluate $\widetilde{Z}$, which is used to compute the desired eigenpairs in Rayleigh-Ritz iterations. This is summarized in Algorithm 1.

Algorithm 1 *Basic FEAST algorithm with projected subspace iteration* [41, 62]

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

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

```
 1: procedure FEAST
 2:     $\hat{s} \leftarrow$ upper bound of $\#_\Lambda(A, \mathcal{C}_\gamma(z_0))$                 ▷ Initial subspace size
 3:     $Y \leftarrow n \times \hat{s}$ Gaussian random matrix
 4:     $c_j = w_j(z_j - z_0),\ j = 1, \ldots, q$   ▷ Weights $w_j$ & nodes $z_j$ in a quadrature rule
 5:     repeat
 6:         $S_j \leftarrow (z_jI - A)^{-1}Y, \quad j = 1, \ldots, q$                   ▷ Solving (4.2)
 7:         $\widetilde{Z} \leftarrow \frac{1}{2}\sum_{j=1}^{q} c_jS_j$                  ▷ Evaluating $\widetilde{Z} = \widetilde{\Phi}Y$ by (4.1)
 8:         $\widetilde{Q} \leftarrow$ orthonormal basis of $\widetilde{Z}$      ▷ This is important and is added in [62]
 9:         $\hat{A} \leftarrow \widetilde{Q}^T A \widetilde{Q}$                              ▷ Reduced problem
10:         $\hat{A} = \widehat{X}\widehat{\Lambda}\widehat{X}^{-1}$              ▷ Solving the reduced eigenvalue problem
11:         $Y \leftarrow \widetilde{Q}\widehat{X}$                ▷ Recovery of approximate eigenvectors of $A$
12:     until convergence
13:     $\widehat{X} \leftarrow Y$                        ▷ Convergent approximate eigenvectors of $A$
14: end procedure
```

In steps 2–3 of Algorithm 1, it is sufficient for convergence when the initial subspace size $\hat{s}$ is not smaller than the actual eigenvalue count. To ensure a good overall convergence rate, it is preferable to make $\hat{s}$ a little larger than the actual eigenvalue count [25, 41]. In the iterations, after step 11, $(\widehat{\Lambda}, \widehat{X})$ gives the Ritz pairs of A. It is easy to identify spurious eigenvalues by either checking whether they are inside Γ or computing the relative residuals. Discussions on the convergence criteria can be found in [25, 33, 62].

4.2. Fast contour-integral eigensolver. Our fast contour-integral eigensolver includes two major stages.

1. *Quadsection stage.* Start from an initial search region, estimate the number of eigenvalues inside. If the number is much larger than a given threshold, quadsect the region into subregions. Then repeat the procedure. This stage involves eigenvalue counts with low-accuracy structured matrix approximations as discussed in Section 3. Fast structured matrix factorization, factorization update for varying shifts, and fast structured solution will be used.
2. *Subspace iteration stage.* In the subregions generated by the previous stage, apply projected subspace iteration as in the FEAST algorithm, where structured accelerations for the matrix factorizations and linear solutions also apply if A is rank structured.

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

4.2.1. Structured factorization update for varying shfits. Both the quadsection stage and the subspace iteration stage involve solutions of linear systems of the following form for multiple shifts μI:

$$(\mu I - \tilde{A})\tilde{S} = Y. \tag{4.3}$$

We precompute a ULV factorization for the HSS matrix $\tilde{A}$ with the algorithms in [8, 60, 61] and it costs $\mathcal{O}(r^2 n)$ flops, where r is the HSS rank of $\tilde{A}$. Then for each shifted matrix $\mu I - \tilde{A}$, we can update the ULV factorization, and the ULV factors are used to solve (4.3). If μ is set to be z_j in (4.2), we can get an approximation to S_j.

This shifted ULV factorization is an extension of the Hermitian version in [56]. Suppose the HSS generators of $\tilde{A}$ are $D_i, U_i, V_i, R_i, W_i, B_i$ as defined in Section 3.1. We briefly outline the ULV factorization procedure for $\tilde{A}$ in [8, 60] without justification, and then show which steps can be updated to quickly get the factors of $\mu I - \tilde{A}$. For notational convenience, we present the update for $\tilde{A} - \mu I$.

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

$$U_i = Q_i \begin{pmatrix} 0 \\ \tilde{U}_i \end{pmatrix}, \tag{4.4}$$

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

$$\tilde{D}_i = Q_i^T D_i \equiv \begin{pmatrix} \tilde{D}_{i;1,1} & \tilde{D}_{i;1,2} \\ \tilde{D}_{i;2,1} & \tilde{D}_{i;2,2} \end{pmatrix}, \tag{4.5}$$

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

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

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

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

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

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

$$\begin{pmatrix} L_{i;2,2} & \tilde{U}_i B_i \tilde{V}_j^T \\ \tilde{U}_j B_j \tilde{V}_i^T & L_{j;2,2} \end{pmatrix}, \quad \begin{pmatrix} \tilde{U}_i R_i \\ \tilde{U}_j R_j \end{pmatrix}, \quad \begin{pmatrix} \tilde{V}_i W_i \\ \tilde{V}_j W_j \end{pmatrix}, \tag{4.6}$$

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

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

- No HSS construction is need for $\tilde{A} - \mu I$, since all the generators remain the same except the D_i generators which just need to be shifted as:
$$D_i \leftarrow D_i - \mu I.$$
- In the ULV factorization, (4.4) remains unchanged.
- (4.5) can be quickly updated as
$$\tilde{D}_i \leftarrow \tilde{D}_i - \mu Q_i^T.$$
This avoids a dense block multiplication.
- In (4.6), the following multiplications remain unchanged:
$$\tilde{U}_i B_i, \quad \tilde{U}_i R_i, \quad \tilde{U}_j B_j, \quad \tilde{U}_j R_j. \tag{4.7}$$

Thus, the entire HSS construction cost and part of the ULV factorization cost are saved. The steps (4.4), (4.5), and (4.7) can be precomputed. For convenience, we call these operations the *pre-shift factorization.* The remain operations are to be done for each shift μI in a *post-shift factorization.* Assuming the leaf level diagonal block size is $2r$ as often used [60], then the costs for the precomputations and the update are given in Table 4.1. Clearly, for each shift μI, we save about 40% of the HSS factorization cost (which is $\frac{116}{3} r^2 n$ [57, Section 4.2]).

TABLE 4.1
Costs of the precomputations for $\tilde{A}$ and the factorization update for $\tilde{A} - \mu I$.

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

A similar precomputation strategy can also be applied when a type of structure-preserving HSS construction in [61] is used. The corresponding pre-shift factorization cost is $6r^2 n$, which is about 30% of the total factorization cost $\frac{58}{3} r^2 n$ [61, Section 3.6]. The algorithm is similar to that of the one mentioned above is thus omitted.

4.2.2. Fast eigenvalue count. To count the eigenvalues inside a circle $\mathcal{C}_\gamma(z_0)$, we choose a random matrix Y with a small column size m and evaluate $\widetilde{Z}$ just like in (4.1). Then (3.1) becomes

$$\#_\Lambda(A, \mathcal{C}_\gamma(z_0)) \approx \frac{1}{m} \operatorname{trace}(Y^T \widetilde{Z}). \tag{4.8}$$

As in [62], we can start from m that is very small and gradually increase it. The algorithm stops if the estimate converges to a number s smaller than a prespecified threshold k or if the estimate is much larger than k. The selection of k will be discussed in Section 4.2.4 based on an optimality condition. In addition, we may even use a power method similar to [23] to improve the quality of this estimator.

As discussed in Section 3, we use a low-accuracy HSS approximation $\tilde{A} \approx A$ to evaluate $\widetilde{Z}$ in (4.8). $\tilde{A}$ may be constructed directly with an algorithm in [60] or via randomization. The randomized HSS construction in [61] is used here. It is especially attractive when A can be quickly applied to vectors. In the construction, we first compute the product of A and a skinny random matrix (with column size equal to r plus a small constant). This product is adaptively modified to yield the product of each off-diagonal block of A and a certain subblock of the random matrix, so as to apply randomized compression to produce the relevant basis matrices. The reader is referred to [61, Section 3.3 and Algorithm 1] for the details. The cost of this construction is $\mathcal{O}(r^2 n)$ plus the cost for matrix-vector multiplications. The matrix-vector multiplication cost typically ranges from $O(rn)$ to $\mathcal{O}(rn^2)$. The cost of $\mathcal{O}(rn^2)$ is the most general case when the construction is performed directly on a dense matrix A. Sometimes when A results from discretizations of certain kernels, then an analytical construction can be done quickly [7].

The shifted factorization update in the previous subsection is then applied to $\tilde{A}$. This leads to the fast eigenvalue count method in Algorithm 2. Following the discussions in Section 2, the Trapezoidal rule is used for the numerical integration. In addition, Section 3 also means that we can use a smaller number of quadrature points in the eigenvalue counts than in the later subspace iterations.

Algorithm 2 *Fast eigenvalue count*

```
 1: procedure $s = \mathsf{EigCount}(\tilde{A}, \mathcal{C}_\gamma(z_0), k)$
      Input: HSS factors of $\tilde{A}$ (from precomputations); $\mathcal{C}_\gamma(z_0)$ (contour); $k$ (threshold
          for eigenvalue count)
      Output: $s \approx \#_\Lambda(A, \mathcal{C}_\gamma(z_0))$ if $\#_\Lambda(A, \mathcal{C}_\gamma(z_0))$ is not much larger than $k$
 2:     $m \leftarrow$ a small integer                         ▷ Initial number of random vectors
 3:     $Y \leftarrow n \times m$ random matrix
 4:     $t \leftarrow 0$                                             ▷ Total trace
 5:     $c_j = w_j(z_j - z_0),\ j = 1, \ldots, q$    ▷ Weights $w_j$ & nodes $z_j$ in Trapezoidal rule
 6:     Update the HSS factors of $\tilde{A}$ to get those of $z_j I - \tilde{A}$,   $j = 1, \ldots, q$
 7:     repeat                               ▷ Adaptive estimate of the eigenvalue count
 8:         $S_j \leftarrow (z_j I - \tilde{A})^{-1} Y,\ j = 1, \ldots, q$            ▷ HSS ULV solution
 9:         $\widetilde{Z} \leftarrow \frac{1}{2}\sum_{j=1}^{q} c_j S_j$,    $t \leftarrow t + \mathrm{trace}(Y^T \widetilde{Z})$
10:         $s \leftarrow \frac{t}{m}$                     ▷ Current-step estimate of the eigenvalue count
11:         if $s$ remains the same for some consecutive steps then
12:             Return                                    ▷ Estimate count is identified
13:         else        ▷ Attaching one extra vector a time; multiple may be attached
14:             $Y \leftarrow$ random vector
15:             $m \leftarrow m + 1$
16:         end if
17:     until $s$ is much larger than $k$     ▷ Further partitioning of the region is needed
18: end procedure
```

4.2.3. Structured FEAST eigenvalue solution with deflation. For a subregion, if the approximate eigenvalue count s is smaller than or near the threshold k, we then solve for the eigenpairs with a structured FEAST algorithm. The FEAST Algorithm 1 can be accelerated with a high-accuracy HSS approximation $\tilde{A}$ to A. Similarly to Algorithm 2, the factorizations and solutions can be performed in HSS forms. In particular, the structured factorization update for varying shifts can greatly save the cost. Moreover, the matrix-vector multiplications needed to form the reduced problem (step 10 of Algorithm 3) can also be performed quickly in HSS forms.

In practice, due to different convergence rates of the eigenpairs in the subspace iteration, a deflation technique called locking [31, 42] is often used to save some computation costs. Those eigenpairs that have already converged to a desired accuracy can be locked and excluded from later iterations. This structured FEAST algorithm with deflation is summarized in Algorithm 3.

Algorithm 3 *Structured FEAST eigenvalue solution with subspace iteration and deflation*

```
1: procedure $[\widehat{\Lambda}, \widehat{X}] = \mathsf{SFEAST}(\tilde{A}, \mathcal{C}_\gamma(z_0), \tilde{k})$
       Input: HSS factors of $\tilde{A}$ (high-accuracy approximation of $A$); $\mathcal{C}_\gamma(z_0)$ (contour);
           $s$ (eigenvalue count)
       Output: $(\widehat{\Lambda}, \widehat{X})$ (eigenvalues inside $\mathcal{C}_\gamma(z_0)$ and the corresponding eigenvectors)
2:     $c_j = w_j(z_j - z_0),\ j = 1, \ldots, q$    ▷ Weights $w_j$ & nodes $z_j$ in Trapezoidal rule
3:     Update the HSS factors of $\tilde{A}$ to get those of $z_jI - \tilde{A}$,  $j = 1, \ldots, q$
4:     $\widehat{\Lambda} \leftarrow \varnothing$,  $\widehat{X} \leftarrow \varnothing$,  $\widehat{Q} \leftarrow \varnothing$    ▷ $\widehat{Q}$: convergent eigenspace
5:     $Y \leftarrow n \times (\frac{3}{2}s)$ random matrix
                                ▷ More than $s$ columns used for faster convergence
6:     repeat
7:         $S_j \leftarrow (z_jI - \tilde{A})^{-1}Y$,  $j = 1, \ldots, q$    ▷ HSS ULV solution
8:         $\widetilde{Z} \leftarrow \frac{1}{2}\sum_{j=1}^{q} c_jS_j$    ▷ Approximating $\widetilde{Z} = \widetilde{\Phi}Y$ in (4.1) based on $\tilde{A}$
9:         $Q \leftarrow$ basis of $\widetilde{Z}$ orthonormalized with respect to $\widehat{Q}$
10:        $\hat{A} \leftarrow Q^TAQ$    ▷ Reduced problem via HSS matrix-vector multiplication
11:        $\hat{A} = \widetilde{X}\widetilde{\Lambda}\widetilde{X}^{-1}$    ▷ Solving the reduced eigenvalue problem
12:        $Y \leftarrow Q\widetilde{X}$    ▷ Recovery of approximate eigenvectors of $A$
13:        $\begin{pmatrix} \widehat{\Lambda}_1 & \widehat{\Lambda}_2 \end{pmatrix} \leftarrow \widehat{\Lambda}$    ▷ Partition with convergent eigenvalues in $\widehat{\Lambda}_1$
14:        $\begin{pmatrix} Y_1 & Y_2 \end{pmatrix} \leftarrow Y$    ▷ Partition with convergent eigenvectors in $Y_1$
15:        $\begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \leftarrow Q$    ▷ Partition with convergent eigenspace in $Q_1$
16:        $\widehat{\Lambda} \leftarrow \mathrm{diag}(\widehat{\Lambda}, \widehat{\Lambda}_1)$,  $\widehat{X} \leftarrow \begin{pmatrix} \widehat{X} & X_1 \end{pmatrix}$,  $\widehat{Q} \leftarrow \begin{pmatrix} \widehat{Q} & Q_1 \end{pmatrix}$
17:        $Y \leftarrow Y_2$
18:    until convergence
19: end procedure
```

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

Discussion on the initial search region will be given in Section 4.3. For convenience, we assume all the intermediate search regions are squares. (In practice, depending on the actual problem, the regions may be made more flexible and more precise.) For each square, we estimate the number of eigenvalues based on $\#_\Lambda(\tilde{A}, \mathcal{C}_\gamma(z_0))$ in Algorithm 2, where $\mathcal{C}_\gamma(z_0)$ is the smallest circle that encloses the square. Since the area of $\mathcal{D}_\gamma(z_0)$ is about 1.57 times the area of the square, this gives an intuitive way of choosing the column size in step 5 of Algorithm 3, which is suggested in [25, 41] to be around 1.5 times the actual eigenvalue count. In practice, Algorithm 3 may then find eigenvalues belonging to neighbor subregions (squares). In this case, we can deflate those eigenvalues when the neighbor subregions are visited.

The complete algorithm of our fast eigensolver is summarized in Algorithm 4, where we assume A can be approximated accurately by an HSS form in step 14. Then the low-accuracy HSS approximation in step 3 can be simply obtained by appropriate truncations.

Algorithm 4 *Fast structured non-Hermitian contour-integral eigensolver*

```
 1: procedure $[\Lambda, X] = \mathsf{FastEig}(\tilde{A}, \Gamma, k)$
        Input: $A$ (explicit or implicit via matrix-vector multiplications); $\Gamma$ (contour that
            encloses desired eigenvalues); $k$ (threshold for subregion eigenvalue count)
        Output: $(\Lambda, X)$ (partial or full spectrum of $A$)
                                                              ▷ 2D Quadsection stage
 2:     Push the initial search region enclosed by $\Gamma$ onto a stack $\mathcal{S}$
 3:     $\tilde{A} \approx A$          ▷ Low-accuracy HSS construction for $A$ and ULV factorization
 4:     while $\mathcal{S} \neq \varnothing$ do
 5:         Pop a subregion $\mathcal{R}_i$ from $\mathcal{S}$
 6:         Find the smallest circle $\mathcal{C}_\gamma(z_0)$ that encloses $\mathcal{R}_i$
 7:         $s_i = \mathsf{EigCount}(\tilde{A}, \mathcal{C}_\gamma(z_0), k)$                    ▷ Algorithm 2
 8:         if $\tilde{k}_i \leq k$ then                        ▷ No further quadsection is needed
 9:             Mark $\mathcal{R}_i$ as a target subregion
10:         else
11:             Quadsect $\mathcal{R}_i$ into 4 subregions and push the subregions onto $\mathcal{S}$
12:         end if
13:     end while
                                                              ▷ Eigenpair solution stage
14:     $\tilde{A} \approx A$          ▷ High-accuracy HSS construction for $A$ and ULV factorization
15:     $\Lambda \leftarrow \varnothing, \quad X \leftarrow \varnothing$
16:     for each target subregion $\mathcal{R}_i$ do
17:         $[\widehat{\Lambda}, \widehat{X}] = \mathsf{SFEAST}(\tilde{A}, \mathcal{R}_i, s_i)$       ▷ Algorithm 3 (with minor modifications)
18:         $\Lambda \leftarrow \mathrm{diag}(\Lambda, \widehat{\Lambda}), \quad X \leftarrow \begin{pmatrix} X & \widehat{X} \end{pmatrix}$
19:     end for
20: end procedure
```

We now analyze the asymptotic complexity of Algorithm 4 for finding all the eigenpairs of a matrix A with maximum off-diagonal (numerical) rank r, and also decide the optimal threshold k. Due to the nature of quadsection, a quadtree can be used to organize the process. Each node of the tree represents a subregion, and the leaf nodes represent the target subregions with roughly k eigenvalues or less. Note that this tree may be unbalanced.

Due to the independence of the computations for non-overlapping subregions, the complexity is directly related to the number of nodes in the quadtree. Without loss of generality, suppose each leaf of the tree corresponds to a subregion with about k eigenvalues, so that the tree has $\mathcal{O}(\frac{n}{k})$ leaves and also $\mathcal{O}(\frac{n}{k})$ nodes. (This modest assumption just eliminates extreme cases where the eigenvalues are highly clustered so that the tree has too many empty nodes. In fact, as long as each node is nonempty, the quadtree has at most n leaves and the asymptotic complexity count would remain about the same for small r.) The computation costs of some basic operations are listed in Table 4.2.

TABLE 4.2
Computation costs of some basic operations, where r is the HSS rank of A.

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

In the quadsection stage, the eigenvalue count Algorithm 2 is performed for every node of the quadtree. The HSS construction cost will be counted in the eigenvalue solution stage since the low-accuracy HSS approximation can be obtained from truncation. We count the costs associated with each node. A smaller HSS rank ($\tilde{r} \le r$) is used in the low-accuracy HSS approximation, and the pre-shift ULV factorization costs $\xi_{1,0} = \mathcal{O}(\tilde{r}^2n)$. The post-shift factorization update costs

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

where q is the number of quadrature nodes and is small (see Section 2). Approximating (4.1) needs to solve m systems and to add q solution matrices, where m is in (4.8). The cost is

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

All the trace computations for (4.8) cost $\xi_{1,3} = \mathcal{O}(m^2n)$. Thus, the total cost for the quadsection stage is

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

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

In the second stage, we use Algorithm 3 to solve for the eigenpairs in the subregions associated with all the leaves of the quadtree. A high-accuracy HSS approximation and the pre-shift ULV factorization cost no more than $\xi_{2,0} = \mathcal{O}(rn^2) + \mathcal{O}(r^2n)$ in the precomputation. We then count the costs associated with each leaf. Similar to the above, the post-shift factorization update costs

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

The linear system solutions for the quadrature approximation costs

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

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

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

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

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

The cost for this stage is then

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

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

$$\xi = \xi_1 + \xi_2 = \mathcal{O}(rn^2) + [\mathcal{O}(k^2n) + \mathcal{O}(kn^2)] + \mathcal{O}\left(\frac{r^2n^2}{k}\right). \tag{4.9}$$

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

THEOREM 4.1. *If A has HSS rank r, then the optimal eigenvalue count threshold for the subregions in Algorithm 4 is $k = O(r)$, and the optimal cost of the algorithm to find all eigenpairs of A is*

$$\xi = \mathcal{O}(rn^2) + \mathcal{O}(r^2n). \tag{4.10}$$

Proof. In (4.9), the term $\mathcal{O}(k^2n)+\mathcal{O}(kn^2)$ increases with k, and the term $\mathcal{O}\left(\frac{r^2n^2}{k}\right)$ decreases with k. Clearly, the minimum of ξ is achieved when $k = \mathcal{O}(r)$. □

In addition, the backward stability of relevant HSS construction and factorization algorithms has been studied in [54, 55].

4.3. Applications and initial search region.

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

- Banded matrices, where the HSS rank r is the bandwidth and the HSS form can be obtained on the fly. If the bandwidth is finite, the cost (4.10) to find all the eigenpairs is $\xi = \mathcal{O}(n^2)$. Banded eigenvalue problems arise in many computations and applications. For example, tridiagonal eigenvalue solution is needed in one type of non-Hermitian eigensolvers that reduce more general matrices (such as complex symmetric ones) to tridiagonal forms. Banded non-Hermitian eigenvalue problems also appear in the study of some 1D PDEs and in sparse neural networks [1].
- Companion matrices, where the HSS rank is $r = 2$ and the HSS form can be directly written out. Companion eigenvalue solution is usually used to find the roots of univariate polynomials. Our algorithm can achieve the same asymptotic complexity $\mathcal{O}(n^2)$ as other fast QR-type companion eigensolvers (e.g., [10]). However, since the companion matrix has more delicate structures that are not fully utilized here, the actual cost is likely higher than that in [10]. On the other hand, the scalability is likely better due to the partitioning of the search region into independent subregions.

- Toeplitz matrices, which in Fourier space have HSS ranks $r = \mathcal{O}(\log n)$ and the HSS construction costs $\mathcal{O}(n \log^2 n)$ [61]. The cost (4.10) is $\xi = \mathcal{O}(n^2 \log n)$. Toeplitz eigenvalue problems are often involved in the studies of time series, wave phenomena in periodic lattices, quantum spin chains, and many other physics and engineering problems [11, 12, 16, 32, 39].
- Some kernel functions (e.g., $1/|x-y|$ and $\log|x-y|$) discretized on 1D curves, where $r = \mathcal{O}(\log n)$ and the HSS construction costs $\mathcal{O}(n \log n)$ [7]. The cost (4.10) is $\xi = \mathcal{O}(n^2 \log n)$. Related eigenvalue problems appear in the studies of radial basis functions and integral kernels, in data science areas such as spectral clustering and kernel principal component analysis [46], and in some physics areas such as entanglement theory [35].

For the last two examples, a much smaller HSS rank $\tilde{r}$ may be used for the eigenvalue counts. In addition, the matrix-vector multiplication needed in forming the reduced eigenvalue problem can also be quickly conducted using FFTs or the fast multipole method (FMM) [22].

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

For some cases, extensions and modifications can be made to accommodate additional matrix properties. For some structured sparse problems [58, 59], we may extend our eigensolver by replacing the HSS methods by structured sparse factorizations, where low-accuracy HSS approximations are used for the intermediate fill-in. This is particularly effective for discretized elliptic PDEs. For cases such as those with tensor product structures, the structured approximation and factorization costs may be significantly reduced. See [20, 21] for some examples, where the structured approximation cost is sublinear in n. Tensor structured methods for the eigenvalue solution of these problems can be found in [28]. For such problems, when n is large, it may be more practical to use our method to extract selected eigenvalues.

We can also adopt the eigensolver to extract certain specific types of eigenvalues, such as the real ones. This will be useful in applications such as control. The search for eigenvalues is then restricted to the real line. More effective filter functions can be designed by setting the contour close to the interval, e.g., with a flat ellipse [25].

4.3.2. Determining the initial search region.

When Algorithm 4 is used to find the entire spectrum, it requires an initial search region. There are many strategies to obtain the region, such as the estimation of the spectral radius and the study of inclusion regions for the field of values. Depending on specific applications, efficient and effective estimations may be available. Here, we just briefly mention the most basic and general method based on the spectral radius. To estimate the spectral radius, we may use the Gershgorin theorem, an estimate of certain matrix norms, or the following well-known result.

LEMMA 4.2. *Let ρ be the spectral radius of $A \in \mathbb{C}^{n\times n}$ and $\|\cdot\|$ be a consistent matrix norm. Then $\rho = \lim_{j\to\infty} \|A^j\|^{1/j}$.*

For A with fast matrix-vector multiplications, we may choose an appropriate j and estimate $\|A^j\|_1$ using Hager's method or a randomized Hager's method [23].

For some matrices, it may be quick to find $||A||_1$ exactly. For example, if A is a

Toeplitz matrix, let u be its first column and v^T be its first row. We can compute

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

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

If A is a companion matrix, other than the bound from $||A||_1$, we can find a nearly optimal bound on the eigenvalues based on a result for the roots of a polynomial $p(\lambda) = \sum_{i=0}^{n} a_i \lambda^i$ ($a_n \neq 0$) [17]:

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

5. Numerical experiments. Now, we show the performance of our fast eigensolver (FastEig Algorithm 4) for some test examples. In order to observe how the complexity depends on the matrix size n, we use quadsection to find all the eigenpairs and report the total clock time. The structure-preserving HSS construction and the corresponding shifted factorization schemes mentioned at the end of Section 4.2.1 are used. Since our eigensolver uses structured direct linear solutions in the intermediate computations, some comparisons are performed with structured direct solutions without our acceleration techniques for one example. (Standard dense direct solvers are obviously much slower and are thus not compared.) It will demonstrate the benefits of the shifted structured factorization update and the eigenvalue count with low-accuracy HSS approximations.

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

- $\mathbf{e}_i = \frac{|\lambda_i - \tilde{\lambda}_i|}{|\lambda_i|}$: relative error, where $\tilde{\lambda}_i$ is the computed eigenvalue and the eigenvalue returned by the Intel MKL subroutine ZGEEV is treated as the exact eigenvalue λ_i;
- $\hat{\mathbf{e}} = \frac{\sqrt{\sum_{i=1}^{n} |\lambda_i - \tilde{\lambda}_i|^2}}{n\sqrt{\sum_{i=1}^{n} |\lambda_i|^2}}$: relative error as used in [52];
- $\mathbf{r}_i = \frac{\|A\tilde{x}_i - \tilde{\lambda}_i \tilde{x}_i\|_2}{\|A\tilde{x}_i\|_2 + \|\tilde{\lambda}_i \tilde{x}_i\|_2}$: relative residual, where $\tilde{x}_i$ is the computed eigenvector;
- $\hat{\mathbf{r}}_i = \frac{\|A\tilde{x}_i - \tilde{\lambda}_i \tilde{x}_i\|_2}{n\|A\|_2}$: relative residual as used in [24].
- mean$(\cdot)$: geometric mean.

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

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

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

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

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

According to FMM, the maximum off-diagonal numerical rank is $\mathcal{O}(\log n)$. The complexity of the eigensolver is then expected to be $\mathcal{O}(n^2 \log n)$. In the test, we let the matrix size n range from $1,600$ to $25,600$. We use relative tolerance $\tau_1 = 10^{-1}$ for the HSS compression in the quadsection stage and $\tau_2 = 10^{-8}$ in the eigenvalue solution stage. The clock times are reported in Figure 5.1 for reaching modest accuracies in Table 5.1, and are compared with the runtimes of the Intel MKL subroutine ZGEEV. Two reference lines for $\mathcal{O}(n^2 \log n)$ and $\mathcal{O}(n^3)$ are also included. We can see that the CPU times are roughly consistent with the complexity analysis. In fact, the slope for the plot of FastEig is significantly lower. The crossover point between these two algorithms for this particular test can also be observed.

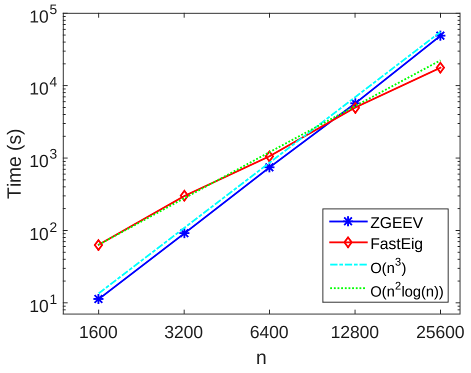


Fig. 5.1. *Example 1. Clock times of FastEig for finding all the eigenvalues.*

Table 5.1
Example 1. Accuracies of the eigenvalue solution.

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

Example 2. Next, consider A to be a discretized matrix from the Foldy-Lax formulation for studying scattering effects due to multiple point scatters [15, 37]. Let

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

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

$$G(s,t) = \frac{e^{\mathbf{i}\omega|s-t|}}{4\pi|s-t|}, \quad s \neq t. \tag{5.1}$$

Here, $\omega = 4\pi$ and σ_j is random in $(0, 1)$, as used in [2].

If the problem is discretized on one dimensional meshes, we observe performance similar to that in the previous example. Thus, we only consider A resulting from the discretization of (5.1) on $M \times N$ regular meshes with equidistance $h = 0.1$ in each direction. The matrix has order $n = MN$. Here, we fix $M = 20$ and let N increase from 80 to 1,280. We use a rank bound 40 in the quadsection stage and a relative tolerance $\tau = 10^{-8}$ in the eigenvalue solution stage. In this case, the off-diagonal ranks are much higher than in the pervious example, so that our acceleration strategies make a significant difference.

Since our eigensolver involves direct linear solutions, we give some comparisons with different structured direct solution methods, depending on whether to use shifted factorization update in the linear solutions and/or low-accuracy approximation for the eigenvalue count. The timings are given in Figure 5.2. We can observe the overall complexity of $\mathcal{O}(n^2 \log n)$. The scaling of the complexity is much better than that of ZGEEV (though it needs larger n to see a significant advantage in timing). We can also see how the acceleration strategies help to improve the performance. In particular, we show in Table 5.2 the detailed time for one of the matrices. The shifted factorization update accelerates both the quadsection stage and the subspace iteration stage. By using low-accuracy HSS approximations for the eigenvalue count, the cost of the quadsection stage becomes significantly lower.

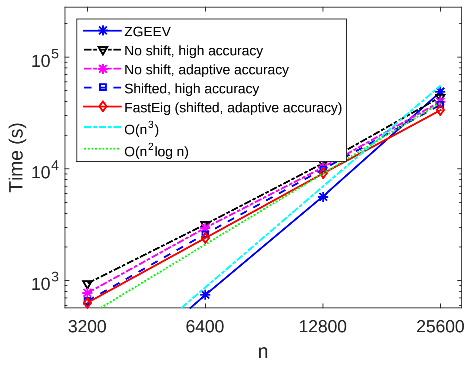


Fig. 5.2. *Example 2. Clock times of* **FastEig** *for finding all the eigenvalues, where "shifted" means structured linear solution with shifted factorization update, and "adaptive accuracy" means using low-accuracy HSS approximation for the eigenvalue count and high accuracy approximation for the later eigenvalue solution.*

Table 5.2

Example 2. Detailed times for the matrix with $n = 6,400$ in Figure 5.2, depending on whether the acceleration strategies are used or not.

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

The benefit of the low-accuracy HSS approximation can also be seen from another aspect. Table 5.3 lists the HSS ranks of $\tilde{A}$ used in the two stages of FastEig. A small rank in the eigenvalue counts leads to significant savings.

TABLE 5.3
Example 2. HSS ranks of $\tilde{A}$ in the two stages of FastEig.

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

The accuracies of the eigenpairs are given in Table 5.4. In addition, Figure 5.3 illustrates how the quadsection of the search region is performed.

TABLE 5.4
Example 2. Accuracies of the eigenvalue solution.

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

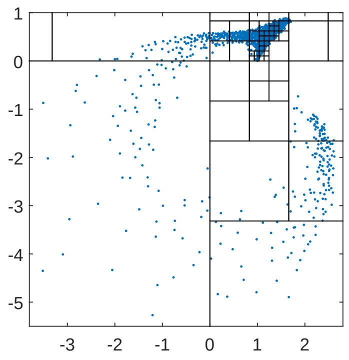


FIG. 5.3. *Example 2. Eigenvalue distribution and quadsection process for finding the eigenvalues of the matrix with $n = 1600$.*

6. Conclusions.

In this paper, we have designed a fast contour-integral eigensolver based on a series of analytical and computational techniques. We show that the Trapezoidal rule is an ideal quadrature for constructing filter functions in contour-integral eigenvalue solutions. This is based on the study of the decay away from the unit circle. We then provide a strategy to use low-accuracy matrix approximations to achieve reliable eigenvalue counts. Such counts are either exact or only off by a small number with low probabilities under some assumptions. Probability estimates are given. In the eigenvalue count algorithm and the FEAST algorithm, rank structured methods are used to accelerate the computations, especially the factorization update for varying shifts. The eigensolver may be used to find a large number of eigenvalues or the full spectrum in a quadsection framework, where we derive an optimal threshold for the number of eigenvalues within each subregion. The eigensolver has nearly $\mathcal{O}(n^2)$ complexity for rank structured matrices, and some strategies can also benefit more general matrices. Due to the nice scalability of both contour-integral eigensolvers and HSS methods, our algorithms have a great potential to be parallelized. We plan to produce a scalable implementation. We are also in the process of extending the methods to more general matrix classes and matrices with clustered eigenvalues.

Acknowledgment. The authors are grateful to the anonymous referees for the valuable suggestions, and would also like to thank Yuanzhe Xi for some discussions and thank Guojian Yin for sharing the GFEAST code. The research of Jianlin Xia was supported in part by an NSF CAREER Award DMS-1255416.

REFERENCES

[1] A. Amir, N. Hatano, and D. R. Nelson, *Non-Hermitian localization in biological networks*, Phys. Rev. E 93 (2016), 042310.

[2] G. Bao, K. Huang, P. Li, and H. Zhao, *A direct imaging method for inverse scattering using the generalized Foldy-Lax formulation*, Contemp. Math., 615 (2014), pp. 49–70.

[3] F. Bauer and C. Fike, *Norms and exclusion theorems*, Numer. Math., 2 (1960), pp. 137–141.

[4] P. Benner and T. Mach, *Computing all or some eigenvalues of symmetric $\mathcal{H}_l$-matrices*, SIAM J. Sci. Comput., 34-1 (2012), pp. A485–A496.

[5] D. A. Bini, L. Gemignani, and V. Y. Pan, *Fast and stable QR eigenvalue algorithms for generalized companion matrices and secular equations*, Numer. Math. 100 (2005), pp. 373–408.

[6] S. Börm and W. Hackbusch, *Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices*, Computing, 69 (2002), pp. 1–35.

[7] D. Cai and J. Xia, *A stable and efficient matrix version of the fast multipole method*, preprint to be submitted, 2016.

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

[9] S. Chandrasekaran, M. Gu, X. Sun, J. Xia, and J. Zhu, *A superfast algorithm for Toeplitz systems of linear equations*, SIAM J. Matrix Anal. Appl., 29 (2008), pp. 1247–1266.

[10] S. Chandrasekaran, M. Gu, J. Xia, and J. Zhu, *A fast QR algorithm for companion matrices*, Oper. Theory Adv. Appl., Birkhauser Basel, 179 (2007), pp. 111–143.

[11] I. Chremmos and G. Fikioris, *Spectral asymptotics in one-dimensional periodic lattices with geometric interaction*, SIAM J. Appl. Math., 76 (2016), pp. 950–975.

[12] H. Dai, Z. Geary, and L. P. Kadanoff, *Asymptotics of eigenvalues and eigenvectors of Toeplitz matrices*, J. Stat. Mech., (2009) P05012.

[13] J. Demmel, *Applied Numerical Linear Algebra*, SIAM, 1997.

[14] Y. Eidelman, I. Gohberg and, V. Olshevsky, *The QR iteration method for Hermitian quasiseparable matrices of an arbitrary order*, Linear Alg. Appl., 404 (2005), pp. 305–324.

[15] L. Foldy, *The multiple scattering of waves*, Phys. Rev., 67 (1945), pp. 107–119.

[16] P. J. Forrester and N. E. Frankel, *Applications and generalizations of Fisher–Hartwig asymptotics*, J. Math. Phys., 45 (2004), pp. 2003–2028.

[17] M. Fujiwara, *Über die obere Schranke des absoluten Betrages der Wurzeln einer algebraischen Gleichung*, Tôhoku Math. J., 10 (1916), pp. 167–171.

[18] Y. Futamura, H. Tadano, and T. Sakurai, *Parallel stochastic estimation method of eigenvalue distribution*, JSIAM Letters, 2 (2010), pp. 127–130.

[19] I. P. Gavrilyuk, W. Hackbusch, and B. N. Khoromskij, *$\mathcal{H}$-matrix approximation for the operator exponential with applications*, Numer. Math., 92 (2002), pp. 83–111.

[20] I. P. Gavrilyuk, W. Hackbusch, and B. N. Khoromskij, *Hierarchical tensor-product ap-*

proximation to the inverse and related operators for high-dimensional elliptic problems, Computing, 94 (2005), pp. 131–157.

[21] L. Grasedyck, *Existence and computation of low kronecker-rank approximations for large linear systems of tensor product structure*, Computing, 72 (2004), pp. 247–265.

[22] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comp. Phys., 73 (1987), pp. 325–348.

[23] M. Gu, *Subspace iteration randomization and singular value problems*, SIAM J. Sci. Comput., 37 (2015), pp. A1139–A1173.

[24] M. Gu and S. C. Eisenstat, *A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 79–92.

[25] S. Güttel, E. Polizzi, P. Tang, and G. Viaud, *Zolotarev quadrature rules and load balancing for the FEAST eigensolver*, SIAM J. Sci. Comput., 37 (2015), pp. 2100–2122.

[26] W. Hackbusch, *A Sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: introduction to $\mathcal{H}$-matrices*, Computing, 62 (1999), pp. 89–108.

[27] W. Hackbusch, B. Khoromskij, and S. Sauter, *On $\mathcal{H}^2$ matrices*, Lectures on Applied Mathematics, Springer-Verlag, Berlin, 2000, pp. 9–29.

[28] W. Hackbusch, B. Khoromskij, S. Sauter, and E. Tyrtyshnikov, *Use of tensor formats in elliptic eigenvalue problems*, Numer. Linear Algebra Appl., 19 (2012), pp. 133–151.

[29] W. Hackbusch, B. Khoromskij, and E. Tyrtyshnikov, *Hierarchical kronecker tensor-product approximations*, Numer. Math., 13 (2005), pp. 119–156.

[30] M. F. Hutchinson, *A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines*, J. Commun. Statist. Simula., 19 (1990), pp. 433–450.

[31] A. Jennings and W. J. Stewart, *A simultaneous iteration algorithm for real matrices*, ACM Trans. of Math. Software, 7 (1981), pp. 184–198.

[32] B.-Q. Jin and V. E. Korepin, *Quantum spin chain, Toeplitz determinants and the Fisher-Hartwig conjecture*, J. Stat. Phys., 116 (2004), pp 79–95.

[33] J. Kestyn, E. Polizzi, and P. Tang, *FEAST eigensolver for non-Hermitian problems*, 2015, arXiv:1506.04463 [math.NA].

[34] B. Klar, *Bounds on tail probabilities of discrete distributions*, Prob. Eng. Inform. Sci., 14 (2000), pp. 161–171.

[35] J. I. Latorre and A. Riera, *A short review on entanglement in quantum spin systems*, J. Phys. A: Math. Theor., 42 (2009), 504002.

[36] S. E. Laux, *Solving complex band structure problems with the feast eigenvalue algorithm*, Physical Review B, 86 (2012), p. 075103.

[37] M. Lax, *Multiple scattering of waves*, Rev. Mod. Phys., 23 (1951), pp. 287–310.

[38] X. Liu, J. Xia, and, M. V. de Hoop, *Parallel randomized and matrix-free direct solvers for large structured dense linear systems*, SIAM J. Sci. Comput., to appear, (2016).

[39] R. Movassagh and L. P. Kadanoff, *Eigenpairs of Toeplitz and disordered Toeplitz matrices with a Fisher-Hartwig symbol*, J Stat Phys (2017) 167, pp 959–996.

[40] E. Napoli, E. Polizzi, and Y. Saad, *Efficient estimation of eigenvalue counts in an interval*, Numer. Linear Algebra Appl., (2016), DOI: 10.1002/nla.2048.

[41] E. Polizzi, *A density matrix-based algorithm for solving eigenvalue problems*, Phys. Rev. B, 79 (2009), pp. 115112.

[42] Y. Saad, *Numerical methods for large eigenvalue problems*, Second Edition, SIAM, 2011.

[43] T. Sakurai, Y. Futamura, and H. Tadano, *Efficient parameter estimation and implementation of a contour integral-based eigensolver*, J. Algorithms Comput. Technol., 7 (2013), pp. 249–270.

[44] T. Sakurai and H. Sugiura, *A projection method for generalized eigenvalue problems using numerical integration*, J. Comput. Appl. Math., 159 (2003), pp. 119–128.

[45] T. Sakurai and H. Tadano, *CIRR: a Rayleigh-Ritz type method with contour integral for generalized eigenvalue problems*, Hokkaido Math. J., 36 (2007), pp. 745–757.

[46] T. Shi, M. Belkin, and B. Yu, *Data spectroscopy: Eigenspaces of convolution operators and clustering*, Ann. Statist., 37 (2009), pp. 3960–3984.

[47] E. M. Stein and R. Shakarchi, *Complex analysis. Princeton Lectures in Analysis, II*, Princeton University Press, NJ, 2003.

[48] P. Tang, J. Kestyn, and E. Polizzi, *A new highly parallel non-hermitian eigensolver*, in Proceedings of the High Performance Computing Symposium, HPC '14, San Diego, CA, USA, 2014, Society for Computer Simulation International, pp. 1:1–1:9.

[49] P. Tang and E. Polizzi, *FEAST as a subspace iteration eigensolver accelerated be approximate spectral projection*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 354–390.

[50] M. Van Barel, *Designing rational filter functions for solving eigenvalue problems by contour integration*, Linear Algebra Appl., 502 (2016), pp. 346–365.

[51] M. Van Barel, R. Vandebril, P. Van Dooren, and K. Frederix, *Implicit double shift QR-algorithm for companion matrices*, Numer. Math., 116 (2010), pp. 177–212.
[52] J. Vogel, J. Xia, S. Cauley, and V. Balakrishnan, *Superfast divide-and-conquer method and perturbation analysis for structured eigenvalue solutions*, SIAM J. Sci. Comput., 38 (2016), pp. A1358–A1382.
[53] Y. Xi and Y. Saad, *Computing partial spectra with least-squares rational filters*, SIAM J. Sci. Comput., to appear, (2016).
[54] Y. Xi and J. Xia, *On the stability of some hierarchical rank structured matrix algorithms*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 1279–1303.
[55] Y. Xi, J. Xia, S. Cauley, and V. Balakrishnan, *Superfast and stable structured solvers for Toeplitz least squares via randomized sampling*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 44–72.
[56] Y. Xi, J. Xia, and R. H. Chan, *A fast randomized eigensolver with structured LDL factorization update*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 974–996.
[57] J. Xia, *On the complexity of some hierarchical structured matrix algorithms*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 388–410.
[58] J. Xia, *Efficient structured multifrontal factorization for general large sparse matrices*, SIAM J. Sci. Comput., 35 (2013), pp. A832–A860.
[59] J. Xia, *Randomized sparse direct solvers*, SIAM J. Matrix Anal. Appl., 74 (2013), pp. 17–34.
[60] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, *Fast algorithms for hierarchically semiseparable matrices*, Numer. Linear Algebra Appl., 17 (2010), pp. 953–976.
[61] J. Xia, Y. Xi, and M. Gu, *A superfast structured solver for Toeplitz linear systems via randomized sampling*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 837–858.
[62] G. Yin, R. H. Chan, and M. Yeung, *A FEAST algorithm with oblique projection for generalized eigenvalue problems*, 2015, arXiv:1404.1768v4 [math.NA].