INTERIOR EIGENSOLVER BASED ON RATIONAL FILTER WITH COMPOSITE RULE

YUER CHEN* AND YINGZHOU LI†

Abstract. Contour-integral-based rational filter leads to interior eigensolvers for non-Hermitian generalized eigenvalue problems. Based on Zolotarev's problems, this paper proves the asymptotic optimality of the trapezoidal quadrature of the contour integral in terms of the rational function separation. A composite rule of the trapezoidal quadrature is derived, and two interior eigensolvers are proposed based on it. Both eigensolvers adopt direct factorization and multi-shift generalized minimal residual method for the inner and outer rational functions, respectively. The first eigensolver fixes the order of the outer rational function and applies the subspace iterations to achieve convergence, whereas the second eigensolver doubles the order of the outer rational function every iteration to achieve convergence without subspace iteration. The efficiency and stability of proposed eigensolvers are demonstrated on synthetic and practical sparse matrix pencils.

Key words. Generalized eigenvalue problem; non-Hermitian matrix; contour integral; trapezoidal quadrature; optimal rational approximation; Zolotarev problem.

1. Introduction.

We aim to solve the large-scale interior eigenvalue problem for non-Hermitian matrices. Such problems arise from many fields including but not limited to electronic structure calculations, dynamic system simulations, control theory, etc. Most of these applications only require part of the eigenvalues of interest, and many of which are interior eigenvalues.

The interior non-Hermitian generalized eigenvalue problem we consider is

$$Ax_i = \lambda_i B x_i, \quad \lambda_i \in \mathcal{D}, \tag{1.1}$$

where $\mathcal{D}$ is the region of interest, matrix pencil (A, B) is diagonalizable, and either or both of A and B is non-Hermitian. The goal is to find all eigenpairs (λ_i, x_i) in the region $\mathcal{D}$. The interior eigensolver for the problem can be used to compute eigenpairs in several regions in parallel to obtain the partial or full eigen-decomposition of the matrix of interest.

1.1. Related work.

Methods for non-Hermitian generalized eigenvalue problems have been developed for decades. The QZ method [9] is a popular one in practice for dense and small-to-medium scale matrices. When a sparse and large-scale matrix is considered, iterative methods [7, 17] are preferred.

Among iterative methods, many adopt the combination of a contour-based filter and the subspace iteration, e.g., Sakurai-Sugiura (SS) method [15] and variants of FEAST method [8, 13]. The original SS method suffers from numerical instability due to the ill-conditioned Hankel matrix. Then Sakurai and Sugiura propose CIRR [4], which uses Rayleigh-Ritz projection to avoid the explicit usage of the momentum and block version SS method [5]. The number of linear systems therein is reduced, and so is the order of the Hankel matrix. The FEAST method originally proposed for Hermitian matrices is extended to non-Hermitian matrices and results in many variants, dual FEAST [6], BFEAST [19], HFEAST [18], etc.

For all the contour-based filters or rational filters in the methods above, the convergence and convergence rate highly depend on the locations and weights of poles. Although the trapezoidal quadrature leads to a good convergence behavior [6], its

*School of Mathematical Sciences, Fudan University, Shanghai, China.

†School of Mathematical Sciences, Fudan University, Shanghai, China (yingzhouli@fudan.edu.cn); Shanghai Key Laboratory for Contemporary Applied Mathematics, Shanghai, China.

optimality remains unknown for non-Hermitian matrices. In this paper, we discuss the optimality of the trapezoidal quadrature and its composite rule property. Based on the composite rule, we propose two interior eigensolvers for non-Hermitian generalized eigenvalue problems.

1.2. Contribution. Our contributions in this paper are in two parts, the part of theorems, called filter design, and the part of algorithms, called filter implementation.

In filter design, we find the optimal rational filter in the sense of spectrum separation for the fast convergence of subspace iteration. By making use of the connection with Zolotarev's third problem, we prove that when the contour is a circle, the rational filter used in the inverse power method leads to an optimal separation, while the trapezoidal quadrature leads to an asymptotically optimal separation.

In filter implementation, we focus on the flexible implementations of trapezoidal quadrature. The main cost of the implementation of the rational filter $R_k(z)$ comes from matrix factorization, e.g. LU factorization, and solving phase, e.g., backward and forward substitution with given LU factorization, while the former is of higher order complexity than the latter. The composite rule of trapezoidal quadrature can help us establish a trade-off between the number of factorizations and solving phases.

Specifically, given a rational filter $R_k(z)$ from the trapezoidal quadrature, we derive a composite rule as $R_k(z) = R_{k_2}(T(R_{k_1}(z)))$ for $k = k_1k_2$ and $T(\cdot)$ being a Möbius transform. Motivated by [8], two novel algorithms are proposed based on the composite rule, both of which implement $R_k(\cdot)$ with an inner-outer structure. The inner rational function $R_{k_1}(\cdot)$ is implemented with direct matrix factorizations, whereas the outer rational function $R_{k_2}(\cdot)$ is implemented via the multi-shift generalized minimal residual method (GMRES). The first algorithm adopts the subspace iteration framework. It substitutes factorizations with solving phases, which may reduce the total runtime for large cases. The second algorithm discards the framework of subspace iteration. It achieves target precision by dynamically increasing k_2 and reusing Krylov subspace to avoid the increase of the computational cost. These two algorithms hybridize the direct method and iterative method and enable us to use computational resources more effectively. Through the experiments on synthetic and practical sparse matrix pencils, we find that the second algorithm is more efficient and practical.

1.3. Organization. The rest of this paper is organized as follows. In section 2, we introduce the basic idea of the contour-integral-based filter and discuss the computational cost and memory cost of the trapezoidal-quadrature-based rational filter. Later, we introduce our two novel algorithms based on the composite rule of trapezoidal quadrature in section 4. The optimality of rational functions is discussed in section 3 and the derivation of the composite rule of trapezoidal quadrature is shown in section 4. In section 5, there are some numerical experiments to demonstrate the efficiency of two proposed algorithms. Finally, section 6 concludes the paper.

2. Subspace iteration with rational filter. Subspace iteration with rational filter is a class of eigensolvers for interior non-Hermitian generalized eigenvalue problems (1.1). All eigensolvers in this class use the subspace iteration framework and adopt various filters, i.e., rational functions with different choices of weights and poles. Part of these rational filters are derived from various discretizations of the contour enclosing $\mathcal{D}$, which is the desired region of eigenvalues. In this section, we will first review the subspace iteration and then discuss contour-based rational filters with various discretization strategies. Some practical considerations, i.e., the number

of vectors and the number of poles, are discussed in the end.

2.1. Subspace iteration. The general framework of the subspace iteration with filter iterates between two phases: 1) refining the subspace via filter, and 2) solving a reduced eigenvalue problem in the subspace.

In the first phase, the filter is applied to an approximate basis of the eigen-subspace, and a refined representation of the eigen-subspace is obtained. For non-Hermitian eigenvalue problems, left and right eigen-subspaces are different. We can refine left and right eigen-subspaces by applying the filter twice [6], or we can only refine the right eigen-subspace and use an extra step to obtain an approximation of the left eigen-subspace [18]. In the second phase, the original large-scale eigenvalue problem is projected to the left and right eigen-subspaces and reduced to an eigen-value problem of a much smaller scale. Then the small-scale eigenvalue problem is solved by classical dense eigensolvers, which results in the approximate eigenpairs of the original problem.

Due to the potentially ill-conditioned eigenbasis of non-Hermitian matrices, the generalized Schur vectors could be extracted to represent the eigen-subspaces and lead to a more stable scheme. Such a subspace iteration idea has been combined with FEAST for non-Hermitian matrices and results in HFEAST [18]. Denote the approximate basis of the right eigen-subspace as U. The orthonormal basis of U is denoted as $V = \mathrm{orth}(U)$. As in HFEAST [18], the orthonormal basis of the left eigen-subspace could be constructed as $W = \mathrm{orth}(AV - \sigma BV)$, where σ is the shift away from the eigenvalues of (A, B). After obtaining the approximate orthonormal basis of the left and right eigen-subspaces, the reduced generalized eigenvalue problem (W^*AV, W^*BV) is solved by the QZ algorithm and yields the generalized Schur form,

$$P_L^*(W^*AV)P_R = H_A \quad \text{and} \quad P_L^*(W^*BV)P_R = H_B,$$

where P_L and P_R are orthogonal matrices, H_A and H_B are upper triangular matrices. The approximate eigenvalues are,

$$\tilde{\lambda}_i = (H_A)_{i,i}/(H_B)_{i,i}.$$

We further calculate the left and right eigenvectors of (H_A, H_B) and denote them as V_L and V_R respectively. The approximate left and right eigenvectors of (A, B) are, respectively,

$$WP_LV_L \quad \text{and} \quad VP_RV_R.$$

The overall framework of the subspace iteration in HFEAST [18] with filter $\rho(\cdot)$ is summarized in Algorithm 2.1. In the rest of the paper, we adopt the subspace iteration as in Algorithm 2.1 and focus on the design and implementation of $\rho(\cdot)$. Besides, for the sake of simplification, we abbreviate the lines 4 to 7 as $[Y, \widetilde{\Lambda}, \widetilde{X}] = \mathrm{HSRR}(A, B, U, \sigma)$, where $\widetilde{\Lambda} = \mathrm{diag}\{\widetilde{\lambda}_1, \ldots, \widetilde{\lambda}_{n_{\mathrm{col}}}\}$ and $\widetilde{X} = (\widetilde{x}_1, \ldots, \widetilde{x}_{n_{\mathrm{col}}})$.

2.2. Contour-based filter and discretization. The basic idea behind the filter is to construct a matrix function whose eigenvalues are close to zero outside the region $\mathcal{D}$ and different from zero inside $\mathcal{D}$. One good choice of matrix functions is the indicator function of $\mathcal{D}$, which could be constructed via a contour integral enclosing

Algorithm 2.1 Subspace Iteration with Filter

Input: matrix pencil (A, B), region $\mathcal{D}$, number of columns n_{col}, shift σ.
Output: All approximate eigenpairs $(\widetilde{\lambda}_i, \widetilde{x}_i)$, $\widetilde{\lambda}_i \in \mathcal{D}$.

```
Generate random Y^{N×n_col}
while not converge do
    U = ρ(B^{-1}A)Y
    V = orth(U)
    W = orth(AV − σBV)
    [H_A, H_B, P_L, P_R, V_L, V_R] = qz(W*AV, W*BV)
    λ̃_i = (H_A)_{i,i}/(H_B)_{i,i}, x̃_i = VP_R(V_R)(:, i) and Y = (x̃_1, ..., x̃_{n_col})
end while
```

the region $\mathcal{D}$. The indicator function of $\mathcal{D}$ via contour integral admits,

$$f(z) = \frac{1}{2\pi\imath} \oint_\Gamma \frac{1}{\zeta - z} \, \mathrm{d}\zeta = \begin{cases} 1, & z \in \mathcal{D} \\ 0, & z \notin \bar{\mathcal{D}} \end{cases}, \tag{2.1}$$

where Γ is the positively oriented Jordan curve encloses the region $\mathcal{D}$. [1]

For a diagonalizable matrix pencil (A, B), i.e.,

$$AX = BX\Lambda,$$

with X being the eigenvectors and Λ is a diagonal matrix with eigenvalues on its diagonal, the indicator function $f(z)$ applying to matrices admits

$$\begin{aligned} f(B^{-1}A) &= \frac{1}{2\pi\imath} \oint_\Gamma (\zeta B - A)^{-1} B \, \mathrm{d}\zeta \\ &= X \left[\frac{1}{2\pi\imath} \oint_\Gamma (\zeta I - \Lambda)^{-1} \, \mathrm{d}\zeta \right] X^{-1} = X \mathbb{1}_{\mathcal{D}}(\Lambda) X^{-1}, \end{aligned} \tag{2.2}$$

where $\mathbb{1}_{\mathcal{D}}(\cdot)$ denotes the indicator function for region $\mathcal{D}$. In [19], a result similar to (2.2) is proved, which leads to the theoretical foundation that the contour integral works even if the non-Hermitian system is defective.

Various numerical discretizations of the contour integral (2.2) lead to the various filters. In many applications, especially non-Hermitian eigenvalue problems, the contour Γ is circular. In other applications, the contour could be conformally mapped to a circle. Hence, in this paper, we will discuss the discretization of contour integrals for Γ being a circle. When the contour is a unit circle, we could reparameterize the circle by $e^{\imath\theta}$ for $\theta \in [0, 2\pi)$. The integral (2.1), then, is a one-dimensional integral and could be numerically evaluated by various quadrature rules. Generally, the contour integral (2.1) approximated by a numerical quadrature with k quadrature points could be written as

$$R_k(z) = \sum_{i=1}^{k} \frac{w_i}{p_i - z}, \tag{2.3}$$

[1] In (2.1), we implicitly assume that the eigenvalues of (A, B) do not locate on the boundary of $\mathcal{D}$.

where $\{w_i\}_{i=1}^k$ are weights, $\{p_i\}_{i=1}^k$ are poles. For example, when the trapezoidal quadrature is applied, the integral (2.1) is numerically approximated by

$$R_k^{(\mathrm{Tr})}(z) = \frac{1}{k}\sum_{i=1}^{k} \frac{e^{\imath\theta_i}}{e^{\imath\theta_i} - z}, \quad \theta_i = \frac{(2i-1)\pi}{k}.$$

When the discretized contour integral is applied to matrices, the rational matrix function yields

$$R_k(B^{-1}A) = \sum_{i=1}^{k} w_i (p_i B - A)^{-1} B. \tag{2.4}$$

The matrix function $R_k(B^{-1}A)$ in (2.4) is used as the filter in Algorithm 2.1.

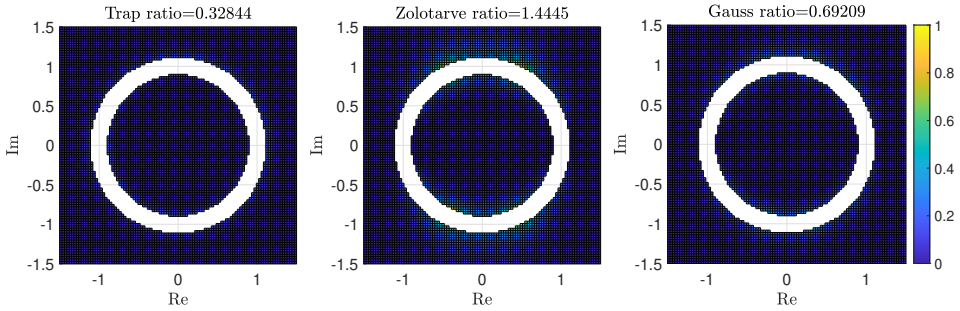

FIG. 1. *Approximation errors to the indicator function on the complex plane except an annulus* $\{x : 1/1.1 < |x| < 1.1\}$ *for trapezoidal quadrature (left), Zolotarev's function (middle) and Gauss quadrature (right).* 16 *poles are adopted for all three rules.*

Among various quadrature rules, the optimal quadrature needs to be decided based on a criterion. As we will see later, the convergence rate of subspace iteration mainly depends on the ratio (2.7). Since we do not know eigenvalues in a priori, we could assume that there is an annulus around the boundary of $\mathcal{D}$ as a generalized eigengap. The inner part and the outer part are,

$$\mathcal{I} = \{z : |z| \le a\}, \quad \text{and } \mathcal{O} = \{z : |z| \ge b\},$$

where a and b are the radii of the inner and outer parts of the annulus, I contains all the eigenvalues inside $\mathcal{D}$. Then the criterion is defined as,

$$\mathfrak{R} = \frac{\sup_{z\in\mathcal{O}} |R_k(z)|}{\inf_{z\in\mathcal{I}} |R_k(z)|}. \tag{2.5}$$

When the ratio is small, the convergence of the subspace iteration is fast. Hence, we would like to address the following optimization problem to obtain the optimal weights and poles for a given k,

$$\inf_{\{w_i\}_{i=1}^k, \{p_i\}_{i=1}^k} \frac{\sup_{z\in\mathcal{O}} |R_k(z)|}{\inf_{z\in\mathcal{I}} |R_k(z)|}. \tag{2.6}$$

From now on, we can view it as a separation problem of rational functions and no longer view it from the discretization of contour integral. One could see that as $b - a$

becomes larger, it is easier to separate the values inside and outside the annulus with rational functions. The drawback of using a larger b is that more eigenvalues may fall into the annulus $\{z : a \leq |z| \leq b\}$ and we do not explicitly know the impact of these eigenvalues on the convergence of the subspace iteration.

Figure 1 shows $|R_k(z) - f(z)|$ and the criteria ratio $\mathfrak{R}$ for three numerical discretizations of $R_k(z)$ with 16 poles, namely the trapezoidal quadrature, Zolotarev's fourth function [8] on the real axis, and the Gauss quadrature. As shown in Figure 1, Zolotarev's fourth function on the real axis is neither optimal for non-Hermitian eigenvalue problems in the L^∞ sense nor optimal in (2.6) sense. The trapezoidal quadrature outperforms the other two. In this paper, we prove in Theorem 3.4 that trapezoidal quadrature provides the asymptotically optimal weights and poles for (2.6).

2.3. Practical consideration. Given a discretization rule, the considerable computational burden in applying the filter $R_k(B^{-1}A)Y$ as in (2.4) lies in solving the shifted linear systems, $(p_iB-A)^{-1}$ for $i = 1, \ldots, k$. For the hard eigenvalue problems, the eigengap between the interior eigenvalues and outer eigenvalues is small, while the poles are on the contour. That means the linear systems will be ill-conditioned. Hence, in most contour-based filters, the shifted linear systems are solved by direct methods, e.g., LU factorization. The overall computational cost is then divided into two parts: the offline factorization part and the online solving part (backward and forward substitution). The cost could be written as

$$C_{\text{factor}} \times k + C_{\text{apply}} \times k \times n_{\text{col}} \times T + o(C_{\text{apply}}),$$

where C_{factor} is the cost of a factorization, C_{apply} is the cost of a solving part, k is the number of poles, n_{col} is the number of columns in Y, T is the number of subspace iterations, and $o(C_{\text{apply}}) = o(C_{\text{apply}}(N))$ is the rest cost lower order. Throughout the subspace iterations, the tuneable hyperparameters are k and n_{col}. The dependence of T on k and n_{col} could be reflected by the function value gap of $R_k(\lambda_i)$, since we are essentially applying power method with $R_k(B^{-1}A)$. Let σ be a permutation of $1, 2, \ldots, N$, such that

$$|R_k(\lambda_{\sigma_1})| \geq |R_k(\lambda_{\sigma_2})| \geq \cdots \geq |R_k(\lambda_{\sigma_N})|.$$

Then, the number of subspace iterations T mainly depends on the ratio,

$$\max_{i > n_{\text{col}}} |R_k(\lambda_{\sigma_i})| \Big/ \min_{\lambda_{\sigma_i} \in \mathcal{D}} |R_k(\lambda_{\sigma_i})|. \tag{2.7}$$

When the ratio is greater or equal to one, the subspace iteration would suffer from a divergence issue. When the ratio is less than one, the subspace iteration would converge and the convergence rate depends on the ratio. The smaller the ratio, the faster the convergence. In the following, we discuss the practical consideration for the number of vectors n_{col} and the number of poles k.

Number of vectors n_{col}. To extract the entire eigen-subspace we are interested, it is necessary that $n_{\text{col}} \geq s$, where s is the number of eigenvalues inside. However, s is not known in a priori. Usually, a rough estimation of s, denoted as $\tilde{s}$, is calculated and the number of vectors is set as $n_{\text{col}} = \lfloor \rho \tilde{s} \rfloor$ for ρ being a hyperparameter greater than one. Even when we have $n_{\text{col}} \geq s$, the subspace iteration may still fail to converge to all the desired eigenpairs. We provide an example of such cases in Figure 2. There are 2 eigenvalues $\lambda_1 = 0$ and $\lambda_2 = 0.75$ inside $\mathcal{D}$, and an eigenvalue $\lambda_3 = \sqrt[4]{2}e^{\imath\pi/4}$ outside. As in the Figure 2 Left, when the rational function is chosen as $R_4^{\text{Tr}}(z)$, the

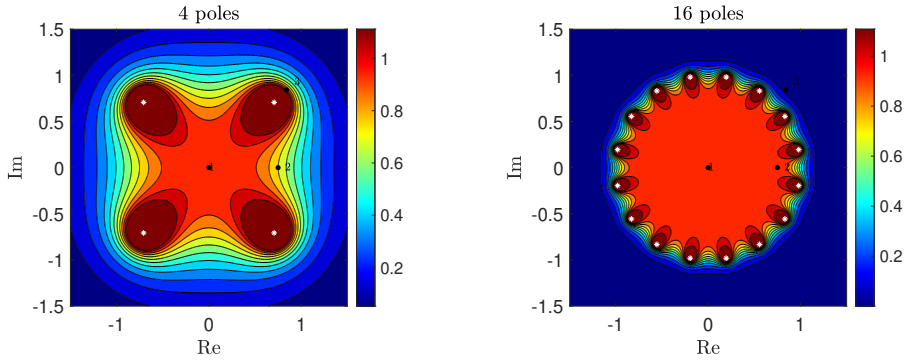

FIG. 2. *Filled contour plots for trapezoidal quadrature of the contour integral with 4 poles (Left) and 16 poles (Right). The matrix pair has two desired eigenpairs with eigenvalues being* $\lambda_1 = 0$ *and* $\lambda_2 = 0.75$*, and an unwanted eigenpair with eigenvalue being* $\lambda_3 = \sqrt[4]{2}e^{\imath\pi/4}$*. In the left figure, the function values are* $|R_4^{(\mathrm{Tr})}(\lambda_1)| = 1$*,* $|R_4^{(\mathrm{Tr})}(\lambda_2)| \approx 0.7596$*, and* $|R_4^{(\mathrm{Tr})}(\lambda_3)| = 1$*. In the right figure, the function values are* $|R_{16}^{(\mathrm{Tr})}(\lambda_1)| = 1$*,* $|R_{16}^{(\mathrm{Tr})}(\lambda_2)| \approx 0.9901$*, and* $|R_{16}^{(\mathrm{Tr})}(\lambda_3)| = 0.0667$*. The white stars indicate the locations of the poles.*

function values obey $1 = |R_4^{(\mathrm{Tr})}(\lambda_3)| > |R_4^{(\mathrm{Tr})}(\lambda_2)| \approx 0.7596$. The ratio (2.7) is greater than one, and the subspace iteration with two columns would converge to (λ_1, x_1) and (λ_3, x_3) rather than desired eigenpairs. The overall subspace iteration fails to capture all the desired eigenpairs inside $\mathcal{D}$. One way to deal with the issue is to increase n_{col} until it covers all eigenvalues whose function values are greater than $|R_4^{(\mathrm{Tr})}(\lambda_2)|$ and make the ratio (2.7) smaller than one. Even when the convergence is guaranteed, we may increase n_{col} for faster convergence. However, when there are many unwanted eigenvalues close to the contour, we need to set n_{col} extremely large for the subspace iteration to converge. In the worst case of Figure 2, all eigenvalues outside the contour cluster around λ_3. In this case, it would be more efficient to increase the number of poles.

Number of poles k. In many applications, adding poles is an inevitable choice. When more poles are added, i.e., $f(z)$ has been discretized with more points, the numerical approximation of $R_k(z)$ to $f(z)$ is improved. The ratio (2.7) is guaranteed to be smaller than one even when $n_{\mathrm{col}} = s$. For example, as in the right figure of Figure 2, the number of poles is increased from 4 to 16. Then the ratio (2.7) becomes $|R_{16}^{(\mathrm{Tr})}(\lambda_3)|/|R_{16}^{(\mathrm{Tr})}(\lambda_2)| \approx 0.0594$ away from one. The subspace iteration would converge efficiently even when $n_{\mathrm{col}} = 2$. Adding the number of poles leads to a more accurate approximation to $f(z)$, and, hence, a smaller n_{col} and T. The drawback of increasing the number of poles is the increasing number of matrix factorizations, which is computationally more expensive than that of the backward and forward substitution and limited by memory. When a massive amount of computational resources are available, all k poles can be calculated in parallel. However, when computing resources are limited, adding poles may be impossible. Motivated by [8], we propose the composite rule of trapezoidal quadrature in section 4, which establishes a trade-off between the number of factorizations and solving phases.

3. Asymptotically optimal contour discretization. In this section, we deal with our first topic, filter design, that is, finding the optimal rational filter in the sense of separation. We prove that the trapezoidal quadrature is an asymptotically

optimal discretization for a disk region $\mathcal{D}$, i.e., an asymptotically optimal solution (2.6), and the rational function behind inverse power method achieves the optimality. In section 3.1, Zolotarev's third problem is reviewed. It serves as the theoretical foundation of the asymptotic optimality of the trapezoidal quadrature. In section 3.2 we derive $R_k^{(\mathrm{Tr})}(z) = R_1^{(\mathrm{Tr})}(z^k)$, which serves as a compact form for $R_k^{(\mathrm{Tr})}(z)$. For the sake of notations, we let $R_k(z) = R_k^{(\mathrm{Tr})}(z)$ in the rest paper, which represents the trapezoidal quadrature of the unit circle contour whose center is located at the origin. When the radius of the contour is r and the center is c, we denote the rational function that comes from trapezoidal quadrature as $R_{c,r,k}(z)$. Finally, we prove that the trapezoidal quadrature is an asymptotically optimal contour discretization for a disk region $\mathcal{D}$ in section 3.3.

3.1. Zolotarev's problems. We introduce Zolotarev's third and fourth problems with their related theoretical results [12, 16]. Zolotarev's third problem is about the optimal separation of two disjoint regions. Since contour discretization yields a rational function, it is natural to bridge the contour discretization and Zolotarev's problems.

Let $\mathcal{R}_{n,m} = \{P(z)/Q(z) : \deg(P(z)) \le n, \deg(Q(z)) \le m\}$ be the set of rational functions, where $P(z)$ and $Q(z)$ are polynomials and $\deg(\cdot)$ denotes the degree of the polynomial. The Zolotarev's third is given in Definition 3.1.

DEFINITION 3.1. *Let $\mathcal{E}$ and $\mathcal{G}$ be two disjoint regions of $\mathbb{C}$, i.e., $\mathcal{E} \cap \mathcal{G} = \emptyset$. The Zolotarev's third problem is to solve the following optimal problem*

$$Z_k(\mathcal{E}, \mathcal{G}) = \inf_{r \in \mathcal{R}_{k,k}} \frac{\sup_{z\in\mathcal{E}} |r(z)|}{\inf_{z\in\mathcal{G}} |r(z)|}. \tag{3.1}$$

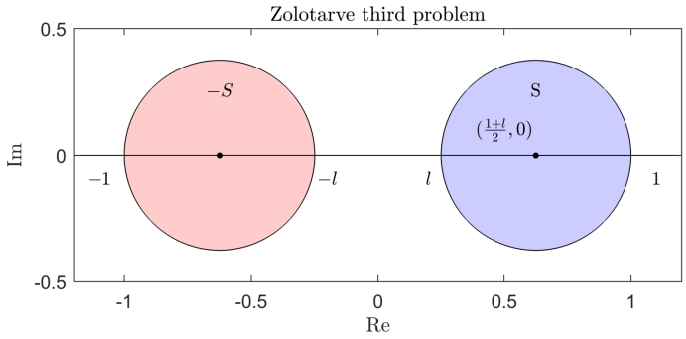

FIG. 3. *Regions in Zolotarev's third problem when E and G are symmetric disks.*

Zolotarev's third problem tends to find a rational function whose value on $\mathcal{E}$ is least from zero and the value on $\mathcal{G}$ is most away from zero. When $\mathcal{E}$ and $\mathcal{G}$ are symmetric disks as in Figure 3, the solution to Zolotarev's third problem is explicitly given in Theorem 3.2. Theorem 3.2 shown in this paper takes a different parameterized form of that in [16].

THEOREM 3.2. *Let $\mathcal{S} = \{z \in \mathbb{C} : |z - \frac{1+\ell}{2}| \le \frac{1-\ell}{2}\}, 0 < \ell < 1$. Then the rational function*

$$r_k^{(Z)}(z) = \left(\frac{z - \sqrt{\ell}}{z + \sqrt{\ell}}\right)^k,$$

attains the infimum of the Zolotarev's third problem $Z_k(\mathcal{S}, -\mathcal{S})$ and the infimum equals to $(\frac{1+\sqrt{\ell}}{1-\sqrt{\ell}})^{-2k}$.

The explicit solution to Zolotarev's third problem as in Theorem 3.2 is the key to proving the asymptotical optimality with respect to k of the trapezoidal quadrature for contour integral. The rational function in Theorem 3.2 is referred to as Zolotarev's function in the rest paper.

3.2. Compact form for $R_k(z)$. In order to connect the Zolotarev's function and the trapezoidal quadrature of the contour integral, and derive the composite formula in section 4, we establish an identity that relates $R_{k^m}(z)$ and $R_k(z^m)$. The relation heavily relies on the symmetry of the trapezoidal quadrature on the circle.

Let us start with toy cases $k = 2, 4$. The trapezoidal quadrature of the unit circular contour with two poles, R_2, could be rewritten as

$$R_2(z) = \frac{1}{2}\left(\frac{e^{\frac{\imath\pi}{2}}}{e^{\frac{\imath\pi}{2}} - z} + \frac{e^{\frac{3\imath\pi}{2}}}{e^{\frac{3\imath\pi}{2}} - z}\right) = \frac{1}{2}\frac{2e^{\imath\pi}}{e^{\imath\pi} - z^2} = \frac{1}{1+z^2} = R_1(z^2).$$

Here we use the symmetry of poles and weights with respect to the origin to derive the compact form of $R_2(z)$ and find that $R_2(z)$ is equivalent to $R_1(z^2)$. Let us further derive the compact form of $R_4(z)$,

$$\begin{aligned} R_4(z) &= \frac{1}{4}\left(\frac{e^{\frac{\imath\pi}{4}}}{e^{\frac{\imath\pi}{4}} - z} + \frac{e^{\frac{7\imath\pi}{4}}}{e^{\frac{7\imath\pi}{4}} - z} + \frac{e^{\frac{3\imath\pi}{4}}}{e^{\frac{3\imath\pi}{4}} - z} + \frac{e^{\frac{5\imath\pi}{4}}}{e^{\frac{5\imath\pi}{4}} - z}\right) \\ &= \frac{1}{2}\left(\frac{e^{\frac{\imath\pi}{2}}}{e^{\frac{\imath\pi}{2}} - z^2} + \frac{e^{\frac{3\imath\pi}{2}}}{e^{\frac{3\imath\pi}{2}} - z^2}\right) = R_2(z^2) = R_1(z^4), \end{aligned}$$

where, in the second equality, we combine the first two and last two terms, and in the last equality, we adopt the compact form of $R_2(z)$. From the derivation of the compact forms of $R_2(z)$ and $R_4(z)$, we could directly extend the derivation to obtain the compact form of $R_k(z) = R_1(z^k)$ for $k = 2^m$, $m \in \mathbb{N}_+$. Fortunately, the compact form holds for any $k \in \mathbb{N}_+$. The result is summarized in Lemma 3.3.

LEMMA 3.3. *For all $k \in \mathbb{N}_+$, let k roots of $z^k = -1$ be $\sigma_i^{(k)}$ for $i = 1, \ldots, k$. Then the compact form of $R_k(z)$ admits,*

$$R_k(z) = \frac{1}{k}\sum_{i=1}^{k}\frac{\sigma_i^{(k)}}{\sigma_i^{(k)} - z} = \frac{1}{1+z^k} = R_1(z^k). \tag{3.2}$$

Proof. We first prove two equalities, (3.3) and (3.4), and then derive the compact form of $R_k(z)$.

The k roots of the k-th degree polynomial $z^k + 1$ are denoted as $\sigma_i^{(k)}$ for $i = 1, 2, \ldots, k$. A k-th order polynomial with k roots takes form, $a_k \prod_{i=1}^{k}(z - \sigma_i^{(k)})$, where a_k is the coefficient in the leading order. Comparing with the leading order coefficient in $z^k + 1$, we know $a_k = 1$ and have,

$$z^k + 1 = \prod_{i=1}^{k}(z - \sigma_i^{(k)}). \tag{3.3}$$

Then we prove the second equality,

$$-\frac{1}{k}\sum_{i=1}^{k}\sigma_i^{(k)}\prod_{j=1, j\neq i}^{k}(z - \sigma_j) = 1. \tag{3.4}$$

The left-hand side of (3.4) is a $(k-1)$-th degree polynomial. For the equality (3.4) to hold, we only need to make sure the equality holds on k different points. Specifically, we examine that on $\sigma_i^{(k)}$ for $i = 1, \ldots, k$ and obtain,

$$-\frac{\sigma_i^{(k)}}{k} \prod_{j=1, j\neq i}^{k} (\sigma_i^{(k)} - \sigma_j^{(k)}) = -\frac{\sigma_i^{(k)}}{k} \lim_{z\to\sigma_i^{(k)}} \frac{z^k+1}{z-\sigma_i^{(k)}} = -\frac{\sigma_i^{(k)}}{k} \frac{k(\sigma_i^{(k)})^{k-1}}{1} = 1,$$

where the first equality is due to (3.3) and the continuity of $(z^k+1)/(z-\sigma_i^{(k)})$, the second equality comes from the L'Hopital rule of complex functions, and the last equality holds since $\sigma_i^{(k)}$ is a root of z^k+1.

Finally, we derive the compact form of $R_k(z)$ as in Lemma 3.3.

$$\begin{aligned} R_k(z) &= \frac{1}{k}\sum_{i=1}^{k} \frac{\sigma_i^{(k)}}{\sigma_i^{(k)} - z} = \frac{-\frac{1}{k}\sum_{i=1}^{k} \sigma_i^{(k)} \prod_{j=1, j\neq i}^{k} (z-\sigma_j)}{\prod_{i=1}^{k}(z-\sigma_i^{(k)})} \\ &= \frac{1}{\prod_{i=1}^{k}(z-\sigma_i^{(k)})} = \frac{1}{z^k+1} = R_1(z^k), \end{aligned}$$

where the second equality adopts (3.4) and the fourth equality adopts (3.3). ∎

A related compact form without detailed derivation could be found in [5]. The compact form Lemma 3.3 could be further generalized to $R_{c,r,k}(z)$ and results the compact form,

$$R_{c,r,k}(z) = \frac{1}{1+(\frac{z-c}{r})^k}.$$

3.3. Optimal solution and the asymptotic optimality of trapezoidal quadrature. In this section, we prove that, if we know the desired spectrum explicitly, the rational function used in the inverse power method achieves the optimal of (3.1) for $\mathcal{E} = \mathcal{O}$ and $\mathcal{G} = \mathcal{I}$. On the other hand, the rational function $R_k(z)$ from the trapezoidal quadrature discretization of the contour integral achieves asymptotic optimality of (3.1).

THEOREM 3.4. *The rational function z^{-k} achieves the infimum of (3.1) for $\mathcal{E} = \mathcal{O}$ and $\mathcal{G} = \mathcal{I}$. And the infimum equals to $\left(\frac{a}{b}\right)^k$.*

Proof. We address Zolotarev's third problem with region $\mathcal{I}$ and $\mathcal{O}$, i.e., $Z_k(\mathcal{O}, \mathcal{I})$. Define a Möbius transform $T(z) = \gamma\frac{z-\alpha}{z-\beta}$ such that

$$T(-b) = 1, \quad T(-a) = -1, \quad T(a) = -\ell, \quad T(b) = \ell.$$

The parameters γ, α, β, and ℓ are determined by a and b. They satisfy

$$\alpha = \sqrt{ab}, \quad \beta = -\sqrt{ab}, \quad \gamma = \frac{\sqrt{b}-\sqrt{a}}{\sqrt{b}+\sqrt{a}}, \quad \ell = \left(\frac{\sqrt{b}-\sqrt{a}}{\sqrt{b}+\sqrt{a}}\right)^2.$$

It can be verified that $T(\mathcal{I}) = -\mathcal{S}$ and $T(\mathcal{O}) = \mathcal{S}$ for $\mathcal{S}$ in Theorem 3.2. Then the composition of the Möbius transform and Zolotarev's function $r_k^{(Z)}(T(z))$ achieves the infimum of $Z_k(\mathcal{O}, \mathcal{I})$ and is denoted as,

$$R_k^{(A)}(z) = R_k^{(Z)}(T(z)) = z^{-k}. \tag{3.5}$$

The infimum of $\mathcal{I}$ is taken when $|z| = a$ and the supremum of $\mathcal{O}$ is taken when $|z| = b$. Then the infimum of the ratio is $\left(\frac{a}{b}\right)^k$. ∎

Theorem 3.4 gives the optimal rational function in solving (2.6). The rational function z^{-k} therein combined with subspace iteration corresponds to the well-known inverse power method. Further, from Theorem 3.4, the radius of $\mathcal{D}$ or the diameter of the annulus is not included in the optimal rational function. Hence, we conclude that, in the sense of convergence rate of subspace iteration, the optimal interior eigensolver is the inverse power method, assuming the center of the desired region $\mathcal{D}$ is explicitly known.

While the optimal rational function z^{-k} only has one pole and could not be written in a sum of low-order rational functions form (2.3). The inverse power method then has to be executed sequentially and could not benefit from the parallelization of distinct poles. In the following, we argue that, although the trapezoidal quadrature of contour integral is not the optimal rational function, it achieves asymptotic optimality.

We now consider that the contour is the boundary of $\mathcal{I}$ and the trapezoidal quadrature with k points is adopted. By Lemma 3.3, the discretization can be rewritten as

$$R_{0,a,k}(z) = \frac{1}{1 + (\frac{z}{a})^k}.$$

By the maximum modulus principle, the infimum of $\mathcal{I}$ and the supremum of $\mathcal{O}$ are taken when $|z| = a$ and $|z| = b$. In region $\mathcal{I}$, $|\frac{z}{a}|^k \leq 1$. The absolute value of the denominator can be viewed as the distance between -1 and $(\frac{z}{a})^k$. By simple computation, the infimum is achieved when $z = a$. Similarly, the supremum of $\mathcal{O}$ is achieved when $z = \sqrt[k]{-1}b$ from the fact that $|\frac{z}{a}|^k > 1$ in $\mathcal{O}$. The ratio (2.5) is

$$\mathfrak{R} = \frac{2}{(\frac{b}{a})^k - 1} \sim 2\left(\frac{a}{b}\right)^k,$$

which asymptotically decays with respect to k at the same rate as that in Theorem 3.4. The above discussion is summarized in the following corollary.

COROLLARY 3.5. *The trapezoidal quadrature discretization of the contour integral on the boundary of $\mathcal{G} = \mathcal{I}$ results in the rational function*

$$R_k(z) = \frac{1}{1 + (\frac{z}{a})^k}. \tag{3.6}$$

The ratio (2.5) of the trapezoidal quadrature is $2/((\frac{b}{a})^k - 1)$, which achieves the same decay rate as the infimum of (3.1) for $\mathcal{E} = \mathcal{O}$ and $\mathcal{G} = \mathcal{I}$.

Although the trapezoidal quadrature used to approximate the contour integral is not the optimal rational function for (3.1), the ratio asymptotically achieves the optimal one up to a constant prefactor 2. Hence, we call the rational function from the trapezoidal quadrature of the contour the nearly optimal rational function for (3.1). The advantage of the trapezoidal quadrature over the optimal rational function is that (3.2) could be efficiently parallelized in solving the shifted linear systems. Another advantage is as we will propose next that (3.2) admits a composite rule and benefits from the flexible trade-off between the number of matrix factorizations and the iterative linear system solves.

4. Composite rule of trapezoidal quadrature. In this section, we move to our second topic, filter implementation. A composite rule of the trapezoidal quadrature discretization of the contour integral is derived and two eigensolvers are proposed based on it from different views. In both eigensolvers, the composite rule is combined with the multi-shift GMRES to reduce the cost of outer iteration. The proposed eigensolvers can reduce cost while preserving the asymptotically optimal ratio $\mathfrak{R}$.

4.1. Composite rule. Given a positive integer k and its integer factorization $k = k_1k_2$ for $k_1 > 1$ and $k_2 > 1$, we aim to rewrite the k-th order rational function $R_k(z)$ as a composition of two k_1-th and k_2-th rational functions, $R_{k_1}(z)$ and $\hat{R}_{k_2}(z) = R_{k_2}(T(z))$, where $T(\cdot)$ is a Möbius transform function. Precisely, the composite function admits, $R_k(z) = \hat{R}_{k_2}(R_{k_1}(z)) = R_{k_2}(T(R_{k_1}(z))$.

According to Lemma 3.3, we have a natural composite expression as,

$$R_{k_1k_2}(z) = R_1(z^{k_1k_2}) = R_{k_2}(z^{k_1}).$$

For the desired composite rule holds, we should let $T(R_{k_1}(z)) = z^{k_1}$.

We determine the coefficients of $T(z) = \frac{az-b}{cz-d}$ such that $T(R_{k_1}(z)) = z^{k_1}$ holds. Substituting $R_{k_1}(z) = \frac{1}{1+z^{k_1}}$ into the expression of $T(z)$, we obtain,

$$\begin{aligned} & T(R_{k_1}(z)) = \frac{a - b(1+z^{k_1})}{c - d(1+z^{k_1})} = z^{k_1} \\ \iff \quad & dz^{2k_1} + (d - c - b)z^{k_1} + (a - b) = 0. \end{aligned}$$

The above equality holds for all z. Hence we have solutions of coefficients satisfying $d = 0$ and $a = b = -c$. These solutions of coefficients lead to the unique Möbius transform function,

$$T(z) = \frac{1-z}{z}. \tag{4.1}$$

The only concern for the above derivation is the case $z = 0$. For rational function $R_k(z)$, zero is attained if and only if $|z| = \infty$, which is not part of the spectrum of matrices. Hence $z = 0$ for $T(z)$ would not cause any trouble in practice and our composite expression holds for all z of interest. In Figure 4, the mapping of $R_{k_1}(z)$ and $T(R_{k_1}(z))$ are illustrated.

Throughout the above derivation, we conclude that $R_{k_1k_2}(z) = R_{k_2}(T(R_{k_1}(z)))$. A generalized composite rule is given in Theorem 4.1 for domains with center c and radius r. In Theorem 4.1, we compose $R_{k_2}(\cdot)$ and $T(\cdot)$ together and rewrite it as the sum of first-order rational functions. Such a summation form could later be used directly in the algorithm design.

THEOREM 4.1. *Given a positive integer k and its integer factorization $k = k_1k_2$, the rational function $R_{c,r,k}(z)$ admits the following composite rule,*

$$R_{c,r,k}(z) = R_{0,1,k_2}(T(R_{c,r,k_1}(z))),$$

where $T(\cdot)$ is the Möbius transform (4.1). When k_2 is even, the rational function $R_{c,r,k}(z)$ further admits the summation form,

$$R_{c,r,k}(z) = \sum_{i=1}^{k_2} c_i (R_{c,r,k_1}(z) - s_i)^{-1} R_{c,r,k_1}(z), \tag{4.2}$$

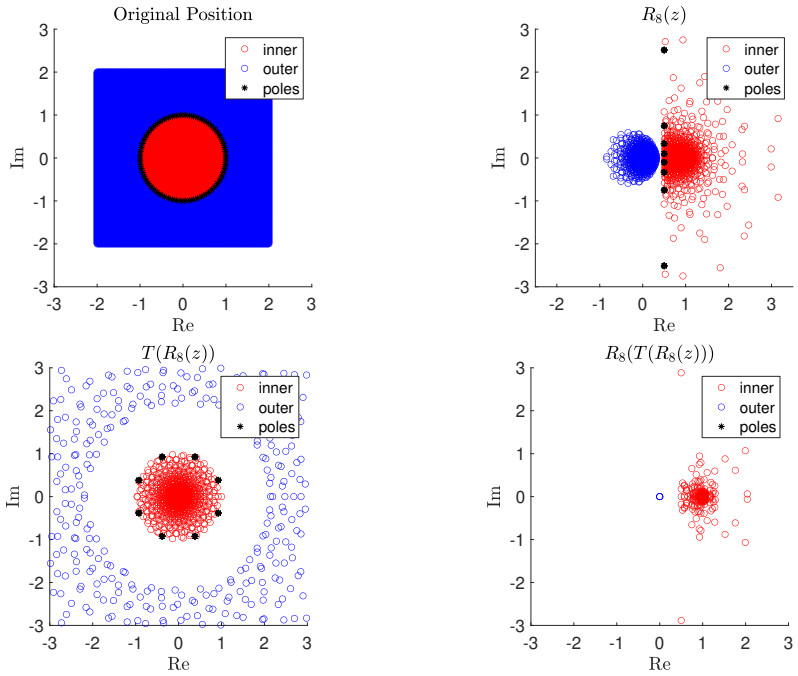

FIG. 4. *We plot the mapping on* $[-2,2]+[-2,2]*\imath$. *There are* 201 *equally spaced points in the direction of the real part and the imaginary part,* 40401 *points in total. The outer points are those* $|z|>h=1.1$ *and the inner points are those* $|z|\le 1$ *where the contour is* $|z|=1$. *We fix the figure window at* $[-3,3]+[-3,3]*\imath$ *except for top right figure which is shown at* $[-2.5,3.5]+[-3,3]*\imath$. *We let* $k_1=k_2=8$ *and the poles in all figures are the poles of* $R_{k_1k_2}(z)$. *The original eigengap is almost invisible, see the top left figure. From the top right figure,* $R_8(z)$ *maps the inner part to be close to 1 while the outer part to be close to 0 and the poles are mapped to the line* $\mathrm{Real}(z)=0.5$. *A more clear comparison of pre and post-mapping eigengaps are shown as the difference between the top left figure and bottom left figure. The composite mapping successfully maps the outer part close to 0 and the inner part close to 1 or modulus greater than 1, see bottom right figure.*

where $c_i^{(k_2)}=-\frac{1}{k_2}\frac{\sigma_i^{(k_2)}}{1+\sigma_i^{(k_2)}}$, $s_i^{(k_2)}=\frac{1}{1+\sigma_i^{(k_2)}}$, *and* $\{\sigma_i^{(k_2)}\}_{i=1}^{k_2}$ *are roots of* $x^{k_2}=-1$. *When* k_2 *is odd,*

$$R_{c,r,k}(z)=\sum_{i=1}^{k_2-1} c_i(R_{c,r,k_1}(z)-s_i)^{-1}R_{c,r,k_1}(z)+\frac{1}{k_2}R_{c,r,k_1}(z), \tag{4.3}$$

where $\sigma_{k_2}^{(k_2)}=-1$.

Theorem 4.1 could be proved through direct calculation. The detailed proof can be found in Appendix A. Besides the composite rule, there is a connection between the poles of $R_k(z)$ and the poles of the composite rule. The poles of the original rational function are transferred into the poles of the inner operator $R_{k_1}(z)$. The connection is detailed in Proposition 4.2, whose proof is in Appendix B.

PROPOSITION 4.2. *For any* $p_i^{(k)}$ *being a pole of* $R_{c,r,k}(z)$, *there exist* $s_j^{(k_2)}$ *for* $1\le j\le k_2$, *such that*

$$R_{c,r,k_1}(p_i^{(k)})=s_j^{(k_2)}, \tag{4.4}$$

where $s_{k_2}^{(k_2)}$ *could be infinite when* k_2 *is odd.*

With the composite rule, we can fix the order k or fix the order k_1 to propose two novel algorithms for the implementation of $R_k(B^{-1}A)$.

4.2. Interior eigensolver with subspace iteration. Using $R_{c,r,k}(z)$ as the filter in subspace iteration for a matrix pencil (A, B) requires the evaluation of $R_{c,r,k}(B^{-1}A)Y$ for Y being a matrix of size $N \times n_{\text{col}}$. By the composite rule for $R_{c,r,k}(z)$ in Theorem 4.1, the evaluation of $R_{c,r,k}(B^{-1}A)Y$ could be rewritten as,

$$R_{c,r,k}(B^{-1}A)Y = \left(\sum_{i=1}^{k_2} c_i (R_{c,r,k_1}(B^{-1}A) - s_i I)^{-1}\right) \left(R_{c,r,k_1}(B^{-1}A)Y\right). \tag{4.5}$$

where the operation $R_{c,r,k_1}(B^{-1}A)Y$ can be rewritten as,

$$R_{c,r,k_1}(B^{-1}A)Y = \sum_{i=1}^{k_1} w_i (p_i B - A)^{-1} BY \tag{4.6}$$

for $\{w_i\}$ and $\{p_i\}$ being the weights and poles of $R_{c,r,k_1}(\cdot)$.

In (4.5), there are inner and outer parts of rational function evaluations. For the inner part, as in (4.6), the poles are on the contour and the width of the annulus is determined by the gap between the interior eigenvalues and outer eigenvalues, which is small in many practical applications. Hence we conclude that linear systems $p_i B - A$ are generally ill-conditioned and the spectrums are not clustering. Iterative linear system solvers would often take too many iterations before convergence. Therefore, a direct solver is adopted for all these linear systems. We pre-factorize all linear systems and denote them as $K_i = p_i B - A$ for $i = 1, \ldots, k_1$. [2] Once the factorizations K_is are available, the inner part could be addressed efficiently. The inner part (4.6) essentially applies a rational filter of the matrix pencil (A, B) and multiplies it to a set of vectors Y. Without loss of generality, we treat the inner part as an operator G acting on Y.

For the outer part, we first rewrite (4.5) using the operator G,

$$R_{c,r,k}(B^{-1}A)Y = \sum_{i=1}^{k_2} c_i (G - s_i I)^{-1} \widetilde{Y} \tag{4.7}$$

for $\widetilde{Y} = G(Y)$. We notice that the spectrum of G is clustering, see Figure 4. Iterative solvers, especially GMRES, are expected to converge fast. Throughout this paper, we adopt GMRES [14] as the default iterative solver for (4.7) with G being applied as an operator. Recall that GMRES is a Krylov subspace method, by the shift-invariant property of the Krylov subspace, all k_2 shifts in (4.7) could be addressed simultaneously in the same Krylov subspace, i.e.,

$$\begin{aligned} &\mathcal{K}_n(G - s_i I, y) = \mathcal{K}_n(G, y), \\ &(G - s_i I)V_n = V_n(H_{n,n+1} - s_i I_{n,n+1}), \end{aligned}$$

for $i = 1, \ldots, k_2$ and V_n denoting the basis of $\mathcal{K}_n(G, y)$. The multi-shift GMRES [1] applies the operator G once per iteration. In all of our numerical experiments, the multi-shift GMRES converges in less than one hundred iterations, and no restarting is needed.

[2] Throughout the numerical section of this paper, dense LU factorization is used by default for dense matrices A and B. If A and B are sparse matrices, we adopt the default sparse LU factorization methods in MATLAB.

Using a direct solver and an iterative solver for the inner and outer part of (4.5), we obtain an effective algorithm for the rational matrix function filter. Combining this filter with subspace iteration leads to our first eigensolver. Algorithm 4.1 gives the overall pseudocode, where HSRR is an abbreviation for lines 4 to 7 of Algorithm 2.1.

Algorithm 4.1 Eigensolver: Composite rational function filter

Input: Pencil (A, B), center c, radius r, number of eigenvalues s, shift σ, number of poles $[k_1, k_2]$.

Output: The approximate eigenpair $(\tilde{\lambda}_i, \tilde{x}_i)$ with $\tilde{\lambda}_i \in \mathcal{D}$.

1: Compute $\{p_i, w_i\}_{i=1}^{k_1}$, $\{c_j, s_j\}_{j=1}^{k_2}$.
2: **for** $i = 1, \cdots, k_1$ **do**
3: Pre-factorize $p_i B - A$ as K_i.
4: **end for**
5: Generate algorithm for operator

$$G(V) = \sum_{i=1}^{k_1} w_i K_i^{-1} B V.$$

6: Generate an orthonormal random matrix $Y^{N \times n_{\text{col}}}$ with $n_{\text{col}} \geq s$.
7: **while** not converge **do**
8: $\widetilde{Y} = G(Y)$.
9: Solving $U_j = (G - s_j I)^{-1} \widetilde{Y}$ for $j = 1, \cdots, k_2$ via multi-shift GMRES.
10: $U = \sum_{j=1}^{k_2} c_j U_j$.
11: $[Y, \widetilde{\Lambda}, \widetilde{X}] = \text{HSRR}(A, B, U, \sigma)$.
12: **end while**

We now estimate the computational cost for Algorithm 4.1. Let C_{factor} and C_{apply} be the computational complexities of the factorization and backward and forward substitution (solving phase) of an $N \times N$ matrix. For almost all dense and sparse linear system solvers, the solving complexity is the same as its memory cost. Hence, C_{apply} is also used as the memory cost in storing a factorization.

In the preparation phase before subspace iteration, the weights and poles are computed independent of the matrix, whose computational cost is then $O(1)$. For the pre-factorization of k_1 linear systems, the computation complexity is $k_1 C_{\text{factor}}$ and the memory required is $k_1 C_{\text{apply}}$.

In the subspace iteration phase, the per-iteration computational cost is dominated by the multi-shift GMRES. If we denote $n_{\text{iter}}^{(j,t)}$ as the GMRES iteration number for j-th column in the t-th subspace iteration, the dominant computational cost in the GMRES is

$$\sum_{t=1}^{T} \sum_{j=1}^{n_{\text{col}}} n_{\text{iter}}^{(j,t)} \cdot k_1 C_{\text{apply}},$$

where T is the subspace iteration number, $k_1 C_{\text{apply}}$ is the cost in applying $G(\cdot)$ to a vector. The overall dominant computational and memory costs for Algorithm 4.1 are summarized in Table 1. In the same table, we also list the computational and memory costs for subspace iteration with $k_1 k_2$-th order rational filter without using the composite rule. Another row of ratio is added to indicate the acceleration from Algorithm 4.1. Clearly, both the computation and memory costs in the pre-factorization

phase are reduced by a factor of k_2. While the comparison for the subspace iteration part is less clear. The ratio depends on the iteration numbers of both the subspace iteration and the multi-shifted GMRES. Another interesting thing is that as the subspace iteration goes, the columns of Y become closer and closer aligned with the eigenvectors, which means the Krylov subspaces will converge faster and so will the GMRES, as we shown is Appendix C.

TABLE 1
Computational and memory complexities of the subspace iteration with the simple rational filter and the composite rational filter. The simple rational filter is of order k_1k_2 and the composite rational filter is of inner and outer order k_1 and k_2 respectively. Here C_{factor} and C_{apply} are factorization and solving cost for a matrix of size $N \times N$.

Algorithm	Computation		Memory	
	Pre-Fact	Iteration	Pre-Fact	Iteration
Simple	$k_1k_2C_{\text{factor}}$	$Tn_{\text{col}}k_1k_2C_{\text{apply}}$	$k_1k_2C_{\text{apply}}$	$n_{\text{col}}N$
Algorithm 4.1	k_1C_{factor}	$\sum_{t=1}^{T}\sum_{j=1}^{n_{\text{col}}} n_{\text{iter}}^{(j,t)}k_1C_{\text{apply}}$	k_1C_{apply}	$\max_{t=1}^{T}\sum_{j=1}^{n_{\text{col}}} n_{\text{iter}}^{(j,t)}N$
Ratio	k_2	$\frac{Tn_{\text{col}}k_2}{\sum_{t=1}^{T}\sum_{j=1}^{n_{\text{col}}} n_{\text{iter}}^{(j,t)}}$	k_2	$\frac{n_{\text{col}}}{\max_{t=1}^{T}\sum_{j=1}^{n_{\text{col}}} n_{\text{iter}}^{(j,t)}}$

4.3. Composite rule eigensolver without subspace iteration. The Algorithm 4.1 still adopts the framework of subspace iteration. It implements the same order trapezoidal quadrature as the simple rule in a different way. We can take advantage of the composite rule from another point of view, i.e., achieving higher-order approximation with the same number of factorizations as the simple rule. The better the rational approximation, the fewer subspace iteration is needed. In the limit of very accurate approximation, only one subspace iteration is enough. Making use of the shift-invariant property of Krylov subspace, we proposed Algorithm 4.2, which achieves higher-order approximation with a fixed number of factorizations and discards the framework of subspace iteration.

More specifically, the first step of Algorithm 4.2 for the initial $[k_1, k_2]$ is the same as Algorithm 4.1, which also constructs the operator G via pre-factorizations and generates Krylov subspaces of different vectors for the computation of $R_k(Y)$. When the eigenpairs do not converge, Algorithm 4.2 will double k_2 and compute new shifts and weights, $s_j^{(2k_2)}$ and $c_j^{(2k_2)}$. Importantly, Algorithm 4.2 does not regenerate new Krylov subspaces from the approximate eigenvectors. Instead, it computes $R_{2k}(Y)$ in the existing Krylov subspaces used for $R_k(Y)$ and expands them when necessary. The Algorithm 4.2 keeps enlarging k_2 until all the eigenpairs converge. As we will show in section 5, the dimension of Krylov subspace is not sensitive to k_2 and increases mildly. In addition, we find that the shift $s_j^{(k_2)}$ are parts of the shifts $s_j^{(2k_2)}$ and their weights satisfy $c_j^{(k_2)}/2 = c_j^{(2k_2)}$. This means we only need to compute $U_j = (G - s_jI)^{-1}\widetilde{Y}$ for the new shifts, then $R_{2k}(Y)$ can be computed from $R_{2k}(Y) = \frac{1}{2}R_k(Y) + \sum_{j=\hat{k}_2+1}^{k_2} c_jU_j$. It corresponds to the 9th and 10th line of Algorithm 4.2.

Compared to Algorithm 4.1, Algorithm 4.2 does not regenerate Krylov subspaces each time and enables adaptive selection of k_2, which makes Algorithm 4.2 more practical. The more interesting characteristic of Algorithm 4.2 is that it discards the framework of subspace iteration. We find that the idea of reusing Krylov subspace

Algorithm 4.2 Eigensolver: Composite rational function filter without subspace iteration

Input: Pencil (A, B), center c, radius r number of eigenvalues s, shift σ, number of poles k_1, the initial k_2 suggested to be equal to k_1, and $\hat{k}_2 = 0$.

Output: The approximate eigenpair $(\tilde{\lambda}_i, \tilde{x}_i)$ with $\tilde{\lambda}_i \in \mathcal{D}$.

1: Compute $\{p_i, w_i\}_{i=1}^{k_1}$, $\{c_j, s_j\}_{j=1}^{k_2}$.
2: **for** $i = 1, \cdots, k_1$ **do**
3: Pre-factorize $p_i B - A$ as K_i.
4: **end for**
5: Construct a function for the operation on any set of vectors V.

$$G(V) = \sum_{i=1}^{k_1} w_i K_i^{-1} B V.$$

6: Generate an orthonormal random matrix $Y^{N \times n_{\text{col}}}$ with $n_{\text{col}} \geq s$.
7: $\widetilde{Y} = G(Y)$, U be a zero matrix of the same size.
8: **while** not converge **do**
9: Solve $U_j = (G - s_j I)^{-1} \widetilde{Y}$ for $j = \hat{k}_2 + 1, \cdots, k_2$ via multi-shift GMRES in the existing Krylov subspaces and expand it when necessary.
10: $U = U/2 + \sum_{j=\hat{k}_2+1}^{k_2} c_j U_j$.
11: $[Y, \widetilde{\Lambda}, \widetilde{X}] = \text{HSRR}(A, B, U, \sigma)$.
12: $\hat{k}_2 = k_2, k_2 = 2k_2$.
13: Compute $\{c_j, s_j\}_{j=\hat{k}_2+1}^{k_2}$.
14: **end while**

for algorithm design is also shown in [3], where they use a single Cayley transform for preconditioning. Instead, we use trapezoidal quadrature with k_1 poles for preconditioning, which means the Algorithm 4.2 can enjoy the benefit of parallelization.

We can also compare Algorithm 4.2 with the simple rule with the same k_1. The separation ratio (2.5) for the simple rule is

$$\left(\frac{2}{(\frac{b}{a})^{k_1} - 1} \right)^T,$$

where T is the number of subspace iterations. We should let k_1 be big enough for the ratio to be less than 1, while k_1 is limited by memory. When the limited k_1 can not achieve a small ratio, the only way to guarantee the convergence is by adding n_{col}. When there are many unwanted eigenvalues close to the contour, extremely large n_{col} may be needed. Algorithm 4.2 gives us another way to solve the problem by dynamically increasing the order of the rational filter. In section 5.3, we find Algorithm 4.2 is more robust with respect to n_{col}.

5. Numerical experiment. In this section, we will demonstrate the efficiency and stability of the proposed algorithms through three experiments. The first experiment shows the advantage of the trapezoidal quadrature over another contour integral discretization, Gauss quadrature. The latter two experiments show the computational benefit of applying Algorithm 4.1 and Algorithm 4.2. This paper focuses on filter design rather than proposing a novel projection technique. Hence the projection techniques used in Algorithm 4.1, Algorithm 4.2, and simple rule remain identical.

Since the estimation of the number of eigenvalues is beyond the scope of this paper, we assume s is known and set the number of columns $n_{\text{col}} = \lfloor \rho s \rfloor$ in all numerical experiments.

Throughout the numerical experiments, the relative error of eigenpair is defined as

$$e_i = e(\tilde{\lambda}_i, \tilde{x}_i) = \frac{\|A\tilde{x}_i - B\tilde{x}_i\tilde{\lambda}_i\|_2}{(|c| + r)\|B\tilde{x}\|_2},$$

where c and r is the center and radius of the region $\mathcal{D}$. For the non-Hermitian interior eigenvalue problem, a phenomenon called ghost eigenvalue often appears. The ghost eigenvalue is the one that appears as a computed eigenvalue in the region $\mathcal{D}$ but will not converge to the true eigenvalue. The ghost eigenvalue would make it difficult to examine the convergence of subspace iterations.

One of the practical strategies is to set a tolerance τ_g as in [19], which is much larger than the target relative error τ. As the iteration goes, the true eigenvalues will converge to a small relative error, while the ghost eigenvalues will not converge to the same precision. After a few steps, there is a gap in the relative errors between true eigenvalues and ghost eigenvalues. When the relative error of an approximate eigenpair $(\tilde{\lambda}_i, \tilde{x}_i)$ inside $\mathcal{D}$ is smaller than τ_g, we treat it as a filtered eigenpair and denote the number of filtered eigenpairs as p. When p is not changed and all relative errors of the filtered eigenpairs are smaller than τ, we terminate the algorithm. In our experiments, we set $\tau_g = 10^{-2}$ and $\tau = 10^{-8}$.

The direct solver is the `lu` function in MATLAB with four outputs under the default setting, which leads to a sparse LU factorization for sparse matrices. The forward and background substitutions are performed by "\" in Matlab, which can handle multiple right-hand sides simultaneously. All programs are implemented and executed with MATLAB R2022b and are performed on a server with Intel(R) Xeon(R) Gold 6226R CPU at 2.90 GHz and 1 TB memory. In performance experiments, we report the single-thread wall time.

5.1. Asymptotically optimal rational filter. First we show the ratio (2.5) for the trapezoidal quadrature, Gauss quadrature and the optimal one in Theorem 3.4. The numerical results are illustrated in Figure 5. Here, we set $a = 1$ and $b = 1.1$. The infimum of I and supremum of O for Gauss quadrature is not known as a closed form, so we use the discretization of 1000 points in both directions of real and imag part on $[-1.5, 1.5] + [-1.5, 1.5] \cdot \imath$ to estimate (2.5) of Gauss quadrature. Only even k is adopted as we perform the Gauss quadrature on the upper semicircle and lower semicircle separately, but not on the full circle directly. Such a Gauss quadrature discretization that preserves the symmetry will perform better than the one that does not.

From Figure 5, we find that trapezoidal quadrature always outperforms Gauss quadrature. The figure also shows that the trapezoidal quadrature attains the same decay order and rate to k as the optimal one, as the slope of the straight line of the trapezoidal quadrature is the same as the optimal one.

We remark that the convergence behavior depends on the distribution of eigenvalues. Our analysis in section 3 views the desired spectrum and undesired spectrum as a disk and the complement of a disk. While the eigenvalues of a matrix are discrete points in these regions. There would be the case that the discrete eigenvalues avoid all bad areas in both the numerator and denominator of (2.5) with Gauss quadrature and have a small ratio $\mathfrak{R}$. In such a case, the rational filter with Gauss quadrature could

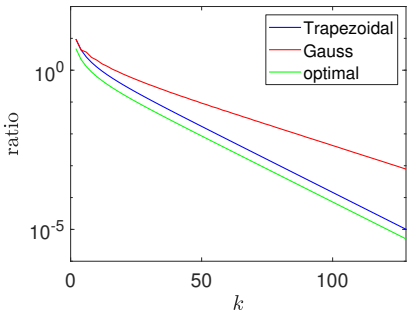

FIG. 5. *The separation ratio (2.5) for various quadrature rules and pole number ks. The pole number k ranges from 2 to 128. The trapezoidal quadrature shows the same slope as the optimal ratio, while the Gauss quadrature behaves differently.*

outperform the rational filter with trapezoidal quadrature for some matrices. Without prior knowledge of the distribution of eigenvalues, the trapezoidal-quadrature-based filter is a near-optimal choice.

5.2. Composite rule with subspace iteration. We compare Algorithm 4.1 against HFEAST with $k_1 = k_2 = 8$ and $k = k_1 \cdot k_2 = 64$.

The class of non-Hermitian generalized eigenvalue problems comes from the model order reduction tasks [2, 11] in the circuit simulation [10]. Matrices are constructed based on quasi-two-dimensional square power grids of size $n_x \times n_x \times 10$. The non-Hermitian matrix pencil is (G, C), and the pattern and distribution of eigenvalues for $n_x = 10$ are shown in Figure 6. One can find the matrix construction details in Appendix D. Table 2 lists information about matrices used in our numerical experiments as well as their target regions. The last column of Table 2 includes the runtime ratio of the matrix factorization and solving phase, where the runtime of the solving phase is the averaged cost of a backward and forward substitution. In all cases, there are 20 eigenvalues in their target regions and we adopt $n_{\text{col}} = 24$. Reference eigenvalues are calculated by `eigs` in MATLAB. The stopping criteria of GMRES is 10^{-9}. The convergence behaviors are illustrated in Figure 7. Runtime is reported in Table 3. The italic values therein are estimated numbers since the simple rule runs out of memory for those settings.

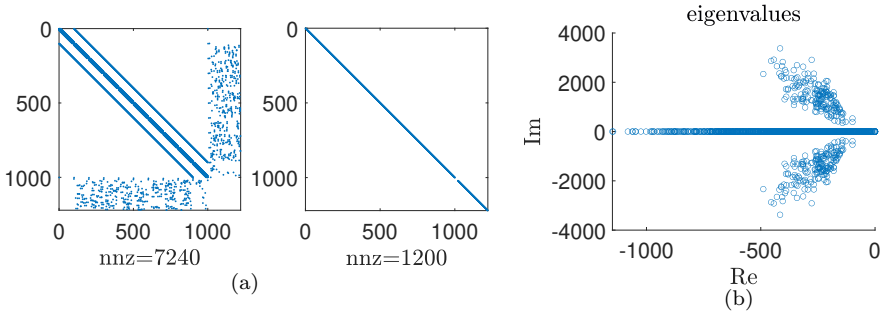

FIG. 6. *(a) Patterns of G and C when $n_x = 10$; (b) Eigenvalues distribution.*

Figure 7 shows that the composite rule converges in a similar fashion to the simple

TABLE 2

Matrix information. Columns show sizes and the number of nonzeros (nnz) of the $G + C$ matrix for various n_x. The centers and radii of target regions are included and each encloses 20 eigenvalues. The last column includes the runtime ratio of matrix factorization and solving.

n_x	Size	nnz	(c, r)	$C_{\text{factor}}/C_{\text{apply}}$
10	1,220	7,440	$(-200 + 1000\imath, 90)$	33.706
100	120,020	776,040	$(-101 + 22\imath, 3)$	47.903
200	480,020	3,112,040	$(-24 + 4.7\imath, 2.1)$	71.119
400	1,920,020	12,464,040	$(-5.3 + 1\imath, 0.9)$	118.037

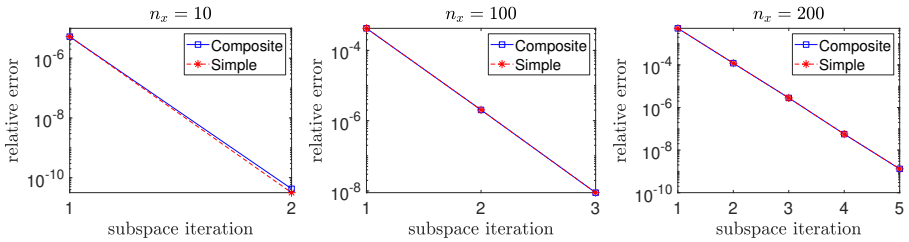

FIG. 7. *Convergence of the simple rule and the composite rule.*

rule. This indicates that both the GMRES and the direct solver achieve sufficiently good accuracy. In most cases we have tested, the subspace iteration converges effectively when many poles are used, i.e., usually in a few iterations.

The composite rule establishes a trade-off between the number of matrix factorizations and the number of solving phases in GMRES. Table 3 shows a comparison of the simple rule and the composite rule in two parts: runtime and memory. As shown in the last column of Table 2, the runtime ratio between the factorization and the solving phase grows as the matrix size increases, which is because the matrix factorization is of higher order complexity compared to that of the solving phase. Hence, reducing the number of factorizations as in the composite rule would be beneficial for large matrices.

However, as shown in all cases of Table 3, the simple rule outperforms the composite rule in total runtime, because the solving time dominates. The domination comes from the increased number of subspace iterations, which is due to the denser spectrum of the larger case. It is hard to tell when the composite rule outperforms the simple rule for a specific case. The only guidance is that when the factorization time dominates, the composite rule can help us substitute the solving time for the factorization time, which may reduce the total runtime. Regarding the memory cost, the simple rule costs about k_2 times more than that of the composite rule. In these examples, we find that the simple rule with $n_x = 400$ already exceeds our memory limit, whereas the composite rule could solve eigenvalue problems with $n_x = 400$ or even larger. Another benefit of the composite rule is that we can utilize high-order trapezoidal quadrature with limited memory.

5.3. Composite rule without subspace iteration. This experiment aims to show that with large k_2, the composite rule will converge without subspace iteration, and the GMRES iteration number does not increase dramatically when k_2 increases. Such an observation means the strategy doubling k_2 each time in Algorithm 4.2 would be affordable compared to the case with optimal k_2. Throughout this section, we reuse

TABLE 3

Runtime (second) of the simple rule and the composite rule for matrices in Table 2. Italic values are estimated due to the out-of-memory limit. `Comp` *means the composite rule.*

n_x	total		factorization		solving	
	Simple	Comp	Simple	Comp	Simple	Comp
10	1.5×10^{0}	2.8×10^{0}	6.2×10^{-1}	7×10^{-2}	5.6×10^{-1}	1.9×10^{0}
100	1.0×10^{3}	2.5×10^{3}	4.1×10^{2}	5.1×10^{1}	6.0×10^{2}	2.3×10^{3}
200	9.0×10^{3}	2.8×10^{4}	3.4×10^{3}	4.2×10^{2}	5.6×10^{3}	2.6×10^{4}
400	*5.9×10^{4}*	2.2×10^{5}	*2.9×10^{4}*	3.6×10^{3}	*3.1×10^{4}*	2.1×10^{5}

matrix pencils in section 5.2. We perform three algorithms in this section: the simple rule with $k = 8$, the composite rule with $k_1 = 8$, and various choices of fixed k_2 (Algorithm 4.1), and Algorithm 4.2 with $k_1 = 8$. Also, various choices of n_{col} are explored.

TABLE 4

Details of the simple rule and Algorithm 4.2. The column p shows the number of filtered eigenpairs, i.e., the number of approximate eigenpairs inside the region whose relative error is less than τ_g. The column e shows the relative error when the algorithm converges or the limitation of subspace iteration is attained. The column n_{iter} shows the times of applying G to a set of vectors for the simple rule. While for Algorithm 4.2, the column n_{iter} shows the maximum GMRES step of different vectors since the GMRES step will change with the vector. When not all eigenpairs are filtered, we use "-" for e and n_{iter}, since the algorithm will fail to filter the eigenpairs of interest even if we run the algorithm with infinite n_{iter}, or n_{iter} would be no less than 400 for the target precision τ, which can be derived by two equations $e^{\lfloor n_{\text{iter}}/100 \rfloor} = \tau$ and $\sqrt[4]{\tau} = \tau_g < e$.

(n_x, n_{col})	Simple			Composite		
	p	e	n_{iter}	p	e	n_{iter}
(100,21)	19	-	-	20	1.0×10^{-10}	39
(100,22)	20	7.7×10^{-9}	64	20	9.6×10^{-11}	39
(100,24)	20	7.0×10^{-9}	35	20	4.8×10^{-11}	39
(200,21)	20	2.0×10^{-4}	*217*	20	3.1×10^{-9}	51
(200,22)	20	4.2×10^{-7}	*126*	20	1.9×10^{-10}	51
(200,24)	20	8.2×10^{-9}	57	20	2.9×10^{-11}	51
(400,21)	19	-	-	20	2.3×10^{-9}	87
(400,22)	19	-	-	20	3.5×10^{-9}	87
(400,24)	20	7.0×10^{-9}	46	20	2.3×10^{-9}	87

We set the simple rule to have no more than 100 subspace iterations, while for the composite rule, the limitation is 10. We terminate Algorithm 4.1 when in the first subspace iteration all eigenpairs converge to target tolerance τ for relative error. All 20 eigenpairs inside are filtered and the relative errors are less than τ for the composite rule. Figure 8 illustrates the relative runtime of Algorithm 4.1 for various k_2, and Table 4 reports details of the simple rule and Algorithm 4.2. The relative runtime of Algorithm 4.2 could be read from Figure 8 from those first triangle marks at k_2 being a power of two. Table 4 shows more details of the simple rule and Algorithm 4.2. We can estimate the n_{iter} for the cases of the simple rule that all eigenpairs of interest are filtered but the relative error can not decrease to target τ, from the equation

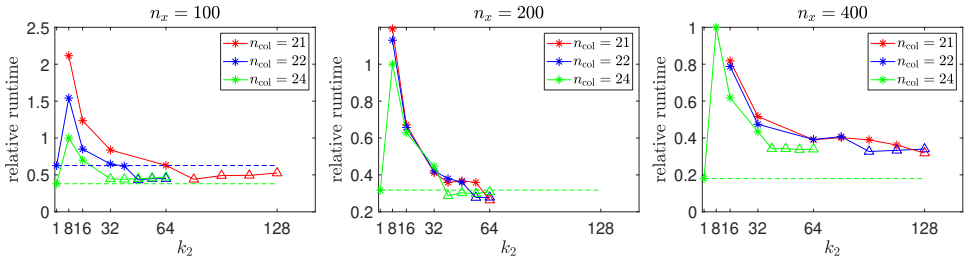

FIG. 8. *Relative runtime of Algorithm 4.1 with $k_1 = 8$ and various k_2. $k_2 = 1$ represents the simple rule. The runtime is scaled by the runtime of Algorithm 4.1 with $k_1 = k_2 = 8$ and $n_{\text{col}} = 24$. We do not plot the point that fails to converge. The star marks denote those subspace iterations converge in more than one iteration, whereas the triangle marks denote those without subspace iteration.*

$e^{\lfloor n_{\text{iter}}/100 \rfloor} = \tau$. We use italic numbers to distinguish estimated n_{iter} from the real one.

In Table 4, all three choices of n_{col} overestimate the actual number of eigenvalues in the region. The simple rule with a fixed $k = 8$ fails to converge when n_{col} is not sufficiently large, e.g., $n_{\text{col}} = 21, 22$. In contrast, Algorithm 4.2 converges in all scenarios. Based on this experiment and other experiments we tried but not listed in the current paper, the convergence of the simple rule is sensitive to the choice of two hyperparameters, k and n_{col}. While the convergence of Algorithm 4.2 is more robust. In the worst-case scenario, when the given region is enclosed by many unwanted eigenvalues, extremely large n_{col} would be needed to resolve the convergence issue in the simple rule. When the simple rule and Algorithm 4.2 converge, the latter one outperforms the former one for small n_{col}, see all the red curves and blue curves in Figure 8. From green curves, we know that when n_{col} increases, these two methods become comparable on runtime.

Figure 8 also explores the optimal choice of k_2 without subspace iteration, i.e., the first triangle marks on each curve. We find that the optimal k_2 is not necessary $2^p k_1$ as in Algorithm 4.2. Besides the factorization cost, the dominant computational cost of the composite rule is the multi-shift GMRES iteration number, i.e., the number of applying G (4.5). Increasing k_2 would add more shifts to the multi-shift GMRES but not necessarily increase iteration number, and the extra cost of orthogonalization is negligible compared to that of applying G. In all curves in Figure 8, we observe that, after their first triangle marks, the relative runtime mostly stays flat and increases extremely slowly. Hence, even if Algorithm 4.2 is not using the optimal k_2, the runtime of Algorithm 4.2 is almost the same as that with optimal k_2. We conclude that Algorithm 4.2 is an efficient and robust eigensolver and is more preferred than Algorithm 4.1.

Remark 5.1. We remark on the hyperparameter choices in Algorithm 4.2. Given a matrix pencil and a region, an overestimation n_{col} of the number of eigenvalues is required. If we perform factorizations and solving phases sequentially, we may need to choose a proper k_1 depending on whether factorizations are more expansive than that of solving phases. In the view of parallel computing, the k_1 factorizations and solving phases are ideally parallelizable. Hence, we would set k_1 as large as possible to fully use the computation resource and reduce the GMRES iterations.

6. Conclusion. This paper finds the optimal separation rational function via Zolotarev's function. The optimal rational function leads to the traditional inverse power method in numerical linear algebra. Discretizing the contour integral with the standard trapezoidal quadrature results in an asymptotically optimal separation rational function. The numerical algorithm based on the trapezoidal quadrature (the simple rule) admits natural parallel computing property, while the inverse power method is sequential. Hence, the simple rule would benefit more from modern multicore computer architecture. Further, we derive the composite rule of the trapezoidal quadrature, i.e., $R_{k_1k_2}(z) = R_{k_2}(T(R_{k_1}(z)))$ for $R_k(\cdot)$ being the simple rule of order k and $T(\cdot)$ being a simple Möbius transform.

Based on the composite rule, we propose two eigensolvers for the generalized non-Hermitian eigenvalue problems, Algorithm 4.1 and Algorithm 4.2. Both algorithms adopt direct matrix factorization for the inner rational function evaluation and multi-shift GMRES for the outer rational function. Compared to the simple rule with the same number of poles, both composite-rule-based algorithms reduce the number of factorizations and reduce the memory requirement. This is of fundamental importance when matrices are of large scale. The difference between the two composite algorithms is the subspace iteration. In Algorithm 4.1, both k_1 and k_2 are hyperparameters, and the algorithm adopts the subspace iteration to converge to desired eigenpairs. In contrast, Algorithm 4.2 is designed without subspace iteration. Algorithm 4.2 adopts k_1 as a hyperparameter and gradually increases k_2 until the rational function approximation is accurate enough and the algorithm converges to desired eigenpairs without subspace iteration. As k_2 increases in Algorithm 4.2, by the property of multi-shift GMRES, the number of GMRES iterations, i.e., the number of applying G, increases very mildly. Hence, compared to the simple rule and Algorithm 4.1, Algorithm 4.2 is a robust and efficient eigensolver.

We demonstrate the efficiency of the proposed algorithms by synthetic and practical generalized non-Hermitian eigenvalue problems. Numerical results show that Algorithm 4.1 outperforms the simple rule only if the matrix factorization is much more expensive than the solving phase. The convergence of Algorithm 4.2 is not sensitive to hyperparameter n_{col} and k_1. In terms of the runtime, Algorithm 4.2 either outperforms or is comparable to the simple rule. A suggestion for the hyperparameter choices of Algorithm 4.2 is also provided based on both the analysis and numerical results.

Acknowledgement. We thank Chao Yang for the helpful discussions. This work is supported in part by the National Natural Science Foundation of China (12271109) and Shanghai Pilot Program for Basic Research - Fudan University 21TQ1400100 (22TQ017).

Appendix A. Proof of Theorem 4.1.

Proof. We can use the equation $z = ry + c$ to transfer the contour discretization on an arbitrary circle into the case of the unit circle around the origin. The rational function then admits,

$$R_{c,r,k}(z) = R_{0,1,k}(y). \tag{A.1}$$

Combining with $R_{k_1k_2}(z) = R_{k_2}(T(R_{k_1}(z)))$, we have

$$R_{c,r,k}(z) = R_{0,1,k_1k_2}(y) = R_{0,1,k_2}(T(R_{0,1,k_1}(y))) = R_{0,1,k_2}(T(R_{c,r,k_1}(z))). \tag{A.2}$$

Now we turn to prove the summation form. We use the convention $R_k(z) = R_{0,1,k}(z)$.

TABLE 5
Number of solving phases and GMRES iteration in the simple rule and the composite rule. Italic values are estimated due to the memory required is out of limitation.

n_x	Simple	Composite	
	Solving	n_{iter}	Solving
10	[1536,1536]	[32,22]	[6128,3200]
100	[1536,1536,1536]	[39,32,31]	[7488,5504,4472]
200	[1536,1536,1536,1536,1536]	[51,43,38,37,37]	[9680,7784,6504,6296,5392]
400	[*1536*, *1536*, *1536*, *1536*, *1536*]	[86,84,75,67,58]	[16328,13968,10552,6896,3952]

When k_2 is even, $\sigma_i^{(k_2)} \neq -1$ holds. With Lemma 3.3, the summation form is,

$$\begin{aligned} R_{c,r,k_1k_2}(z) =& R_{k_2}(T(R_{c,r,k_1}(y))) = \frac{1}{k_2}\sum_{i=1}^{k_2} \frac{\sigma_i^{(k_2)}}{\sigma_i^{(k_2)} - \frac{1-R_{c,r,k_1}(y)}{R_{c,r,k_1}(y)}} \\ =& \frac{1}{k_2}\sum_{i=1}^{k_2} \frac{\sigma_i^{(k_2)} R_{c,r,k_1}(y)}{(1+\sigma_i^{(k_2)})R_{c,r,k_1}(y) - 1} \\ =& \frac{1}{k_2}\sum_{i=1}^{k_2} \frac{\sigma_i^{(k_2)}}{1+\sigma_i^{(k_2)}}(R_{c,r,k_1}(z) - \frac{1}{1+\sigma_i^{(k_2)}})^{-1} R_{c,r,k_1}(x) \\ =& \sum_{i=1}^{k_2} c_i (s_i^{(k_2)} - R_{c,r,k_1}(z))^{-1} R_{c,r,k_1}(z), \end{aligned} \tag{A.3}$$

where

$$c_i^{(k_2)} = -\frac{1}{k_2}\frac{\sigma_i^{(k_2)}}{1+\sigma_i^{(k_2)}}, \quad s_i^{(k_2)} = \frac{1}{1+\sigma_i^{(k_2)}}. \tag{A.4}$$

When k_2 is odd, the term associated with $\sigma_{k_2}^{(k_2)} = -1$ in summation form is equal to $\frac{1}{k_1}R_{k_1}(z)$. ∎

Appendix B. Proof of Proposition 4.2.

Proof. By Lemma 3.3, we know

$$R_{c,r,k_1}(p_i^{(k)}) = R_{0,1,k_1}(\sigma_i^{(k)}) = \frac{1}{1+(\sigma_i^{(k)})^{k_1}} = \frac{1}{1+\sigma_j^{(k_2)}} = s_j^{(k_2)}. \tag{B.1}$$

∎

Appendix C. GMRES iteration number.

As we mentioned in subsection 4.2, the multi-shift GMRES will converge faster as the subspace iteration converges. Table 5 reports the number of solving phases in both the simple and the composite rules and its GMRES iteration number. The normalized last column of Table 5 is visualized in Figure 9.

Table 5 shows that the number of solving phases in each subspace iteration in the simple rule stays constant, whereas that for the composite rule decreases. Notice that the n_{iter} decays much slower than the number of solving phases in the composite rule. That is because different columns converge to eigenvectors with different rates.

Appendix D. Construction of matrices.

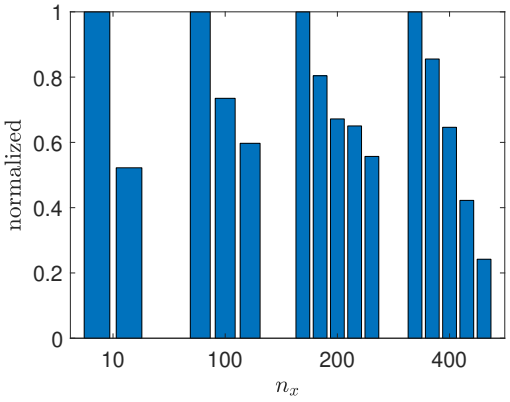

FIG. 9. *Normalized solving cost in the composite rule. In each test matrix, the bars show the number of solving phases in each subspace iteration, which are normalized by the number of solving phases in the first subspace iteration.*

Matrices are constructed based on quasi-two-dimensional square power grids of size $n_x \times n_x \times 10$. The non-Hermitian matrix pencil is (G, C) taking the block form as,

$$G = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & 0 \end{bmatrix}, \quad C = \begin{bmatrix} C_c & 0 \\ 0 & L \end{bmatrix}.$$

In particular, G_{11} represents the conductance matrix as $G_{11} = L_{n_x} \otimes I_{n_x} \otimes I_{10} + I_{n_x} \otimes L_{n_x} \otimes I_{10} + \frac{1}{10} I_{n_x} \otimes I_{n_x} \otimes L_{10}$, where L_n is a weighted one-dimensional Laplacian matrix of size $n \times n$ as

$$L_n = \frac{n}{100} \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix}_{n \times n}$$

and I_n is an identity matrix of size $n \times n$. The off-diagonal blocks of G admit $G_{12} = -G_{21}^\top \in \mathbb{R}^{10n_x^2 \times (20+2n_x^2)}$ with entries being ± 1 or zero. The first 20 columns of G_{12} correspond to 20 input ports at $(\cdot, 1, 1)$ and $(\cdot, n_x, 10)$ two edges, where the corresponding rows have a positive one. The rest $2n_x^2$ columns of G_{12} correspond to inductors. We uniformly randomly pick $2n_x^2$ interior nodes from grid nodes and add an inductor with their neighbor nodes on the same layer. The corresponding G_{12} part is the incidence matrix of the inductor graph. Matrix L is a diagonal matrix of size $20 + 2n_x^2$. The first 20×20 block of L is zero. The later $2n_x^2 \times 2n_x^2$ block has diagonal entries uniformly randomly sampled from $[0.5, 1.5] \cdot n_x \cdot 10^{-4}$ being the inductance of inductors. The submatrix C_c represents capacitors in the circuit. For each node, we add a grounded capacitor with capacitance uniformly randomly sampled from $[0.5, 1.5] \cdot 10^{-3}$, which means C_c is a diagonal matrix whose elements are equal to the capacitances.

REFERENCES

[1] T. Bakhos, P. K. Kitanidis, S. Ladenheim, A. K. Saibaba, and D. B. Szyld, *Multipreconditioned gmres for shifted systems*, SIAM Journal on Scientific Computing, 39 (2017), pp. S222–S247, https://doi.org/10.1137/16M1068694.

[2] P. Gross, R. Arunachalam, K. Rajagopal, and L. Pileggi, *Determination of worst-case aggressor alignment for delay calculation*, in 1998 IEEE/ACM International Conference on Computer-Aided Design. Digest of Technical Papers (IEEE Cat. No.98CB36287), 1998, pp. 212–219, https://doi.org/10.1145/288548.288616.

[3] R. Huang, J. Sun, J. Sun, and C. Yang, *Recursive integral method with cayley transformation*, Numerical Linear Algebra with Applications, 25 (2017).

[4] T. Ikegami and T. Sakurai, *Contour integral eigensolver for non-Hermitian systems: a rayleigh-ritz-type approach*, Taiwanese Journal of Mathematics, 14 (2010), pp. 825 – 837, https://doi.org/10.11650/twjm/1500405869.

[5] T. Ikegami, T. Sakurai, and U. Nagashima, *A filter diagonalization for generalized eigenvalue problems based on the sakurai–sugiura projection method*, Journal of Computational and Applied Mathematics, 233 (2010), pp. 1927–1936, https://doi.org/10.1016/j.cam.2009.09.029.

[6] J. Kestyn, E. Polizzi, and P. T. Peter Tang, *Feast eigensolver for non-hermitian problems*, SIAM Journal on Scientific Computing, 38 (2016), pp. S772–S799, https://doi.org/10.1137/15M1026572.

[7] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *Arpack users' guide - solution of large-scale eigenvalue problems with implicitly restarted arnoldi methods*, in Software, environments, tools, 1998.

[8] Y. Li and H. Yang, *Interior eigensolver for sparse hermitian definite matrices based on zolotarev's functions*, Communications in Mathematical Sciences, 19 (2021), pp. 1113–1135.

[9] C. B. Moler and G. W. Stewart, *An algorithm for generalized matrix eigenvalue problems*, SIAM Journal on Numerical Analysis, 10 (1973), pp. 241–256, https://doi.org/10.1137/0710024.

[10] F. N. Najm, *Circuit Simulation*, Wiley-IEEE Press, 2010.

[11] Odabasioglu, Celik, and Pileggi, *Prima: passive reduced-order interconnect macromodeling algorithm*, in 1997 Proceedings of IEEE International Conference on Computer Aided Design (ICCAD), 1997, pp. 58–65, https://doi.org/10.1109/ICCAD.1997.643366.

[12] P. P. Petrushev and V. A. Popov, *Rational Approximation of Real Functions*, Encyclopedia of Mathematics and its Applications, Cambridge University Press, 1988, https://doi.org/10.1017/CBO9781107340756.

[13] E. Polizzi, *Density-matrix-based algorithm for solving eigenvalue problems*, Phys. Rev. B, 79 (2009), p. 115112, https://doi.org/10.1103/PhysRevB.79.115112.

[14] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, second ed., 2003, https://doi.org/10.1137/1.9780898718003.

[15] T. Sakurai and H. Sugiura, *A projection method for generalized eigenvalue problems using numerical integration*, Journal of Computational and Applied Mathematics, 159 (2003), pp. 119–128, https://doi.org/10.1016/S0377-0427(03)00565-X.

[16] G. Starke, *Near-circularity for the rational zolotarev problem in the complex plane*, Journal of Approximation Theory, 70 (1992), pp. 115–130, https://doi.org/10.1016/0021-9045(92)90059-W.

[17] G. W. Stewart, *A krylov–schur algorithm for large eigenproblems*, SIAM Journal on Matrix Analysis and Applications, 23 (2002), pp. 601–614, https://doi.org/10.1137/S0895479800371529.

[18] G. Yin, *A harmonic feast algorithm for non-hermitian generalized eigenvalue problems*, Linear Algebra and its Applications, 578 (2019), pp. 75–94, https://doi.org/10.1016/j.laa.2019.04.036.

[19] G. Yin, R. H. Chan, and M.-C. Yeung, *A feast algorithm with oblique projection for generalized eigenvalue problems*, Numerical Linear Algebra with Applications, 24 (2017), p. e2092, https://doi.org/10.1002/nla.2092.