Fast Randomized non-Hermitian Eigensolvers Based on Rational Filtering and Matrix Partitioning

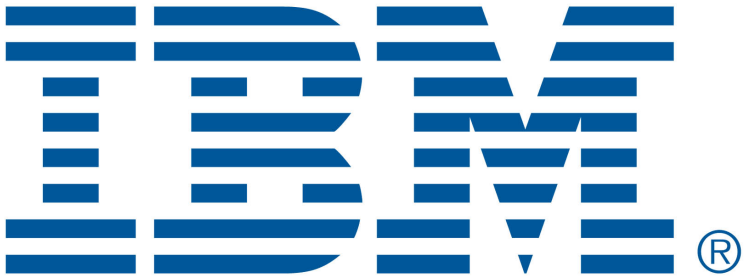
Vassilis Kalantzis, Yuanzhe Xi, and Lior Horesh

September 2021

EPrint ID: 2021.3

IBM Research
Thomas J. Watson Research Center

Preprints available from:
`https://researcher.watson.ibm.com/researcher/view.php?person=ibm-vkal`



FAST RANDOMIZED NON-HERMITIAN EIGENSOLVERS BASED ON RATIONAL FILTERING AND MATRIX PARTITIONING*

VASSILIS KALANTZIS†, YUANZHE XI‡, AND LIOR HORESH†

Abstract. This paper describes a set of rational filtering algorithms to compute a few eigenvalues (and associated eigenvectors) of non-Hermitian matrix pencils. Our interest lies in computing eigenvalues located inside a given disk, and the proposed algorithms approximate these eigenvalues and associated eigenvectors by harmonic Rayleigh-Ritz projections on subspaces built by computing range spaces of rational matrix functions through randomized range finders. These rational matrix functions are designed so that directions associated with non-sought eigenvalues are dampened to (approximately) zero. Variants based on matrix partitionings are introduced to further reduce the overall complexity of the proposed framework. Compared with existing eigenvalue solvers based on rational matrix functions, the proposed technique requires no estimation of the number of eigenvalues located inside the disk. Several theoretical and practical issues are discussed, and the competitiveness of the proposed framework is demonstrated via numerical experiments.

Key words. Rational filtering, matrix partitioning, contour integral eigensolvers, non-Hermitian eigenvalue problems, randomized algorithms

AMS subject classifications. 65F15, 15A18, 65F50

1. Introduction.

This paper describes a rational filtering framework to compute a few eigenvalues and associated eigenvectors of non-Hermitian eigenvalue problems of the form

$$Ax = \lambda Mx, \tag{1.1}$$

where the matrices $A \in \mathbb{C}^{n\times n}$ and $M \in \mathbb{C}^{n\times n}$ are assumed large and sparse, and the pencil (A, M) is assumed regular and diagonalizable. The focus of this paper lies in computing all eigenvalues located in the interior of a disk $\mathcal{D}$ prescribed in the complex domain. An illustrative example is shown in Figure 1.1.

Rational filtering eigenvalue solvers can be seen as (harmonic) Rayleigh-Ritz procedures in which the projection subspace is built by exploiting a (complex) rational transformation of the matrix pencil (A, M). These transformations are constructed so that the gap between eigenvalues located inside the disk $\mathcal{D}$ versus those located outside the latter is as big as possible after the transformation. Applying a projection scheme to the transformed pencil can then significantly enhance the convergence towards the sought invariant subspace. The most popular approaches to construct efficient rational transformations is either via shift-and-invert or via a discretization of the Cauchy integral representation of the eigenprojector along the boundary of the disk $\mathcal{D}$ [6, 39, 44, 45, 51, 54]. Compared to shift-and-invert, contour integral eigensolvers are generally oblivious to the location of the sought eigenvalues inside the disk $\mathcal{D}$, and enjoy enhanced scalability when implemented in distributed memory computing environments [1, 18, 21, 24, 50]. Other rational filters, though not necessarily based on contour integration, can be found in [4, 5, 15, 25, 28, 31, 41, 47, 48, 49].

In this paper we consider algorithms in which the projection subspace is set equal to the column space of matrices formed after applying a rational transformation to the matrix pencil (A, M). These rational transformations are constructed so

*Submitted to the editors January 8, 2022.

†IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA (vkal@ibm.com, lhoresh@us.ibm.com).

‡Department of Mathematics, Emory University, Atlanta, GA 30322, USA (yxi26@emory.edu).

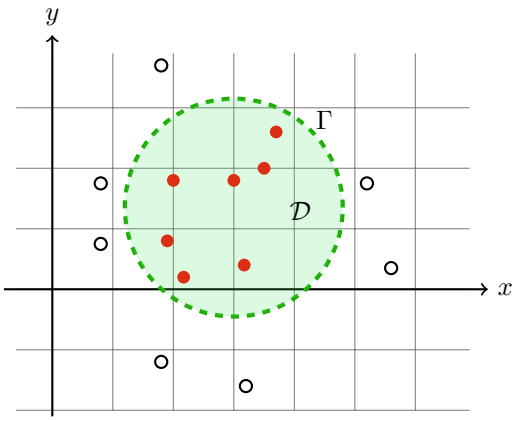


FIG. 1.1. *Sought eigenvalues are denoted by red filled dots. Unwanted eigenvalues located outside the circumference* Γ *(denoted by a green dashed curve) of the disk* $\mathcal{D}$ *are denoted by black solid circles.*

that eigenvector directions associated with eigenvalues located outside the disk $\mathcal{D}$ are approximately mapped to zero, and the corresponding column spaces are captured through randomized range finders [33, 34]. The algorithms proposed in this paper are also combined with matrix partitionings to reduce the computational complexity of the construction of the projection subspace. So far matrix partitioning approaches have been featured within the context of rational filtering only for symmetric eigenvalue problems [21, 23]. One of the main motivations of this paper is to extend this class of techniques to non-Hermitian eigenvalue problems.

Overall, the proposed framework possesses the following advantages:

Improved robustness. Classical rational filtering approaches such as FEAST [39] or the SS algorithm [44, 45] require an estimation of the number of eigenvalues located inside the disk $\mathcal{D}$. However, such an estimation is not always readily available or easy to compute for generalized eigenvalue problems. On the other hand, an inaccurate estimation can lead to slow convergence or failure to capture all required eigenpairs. The proposed algorithms bypass this issue by dynamically increasing the dimension of the projection subspace.

Reduced complexity. By combining rational filtering with substructuring, the projection subspace is formed as the direct sum of two separate subspaces approximated independently. This is done by applying a 2×2 block partitioning to the pencil (A, M). Two specialized algorithms are proposed to further reduce the computational costs associated with classical rational filtering eigensolvers. The 2×2 block partitioning can be created either in an ad hoc way or by applying a graph partitioner to the adjacency graph of the pencil (A, M).

Enhanced parallelism. In addition to the ample opportunities for parallelism offered by rational filtering eigensolvers, the proposed algorithms can take advantage of an additional level of parallelism introduced by matrix substructuring.

The structure of this paper is organized as follows. Section 2 describes a technique based on the combination of randomized range finders, harmonic Rayleigh-Ritz projections and rational transformations. Section 3 presents two variants based on matrix partitioning which aim at reducing the computational cost associated with the construction of an efficient projection subspace. Section 4 discusses practical details and presents computational cost comparisons. Section 5 provides numerical experiments

on a few test problems. Finally, Section 6 presents our concluding remarks.

1.1. Notation. Throughout this paper we denote the spectrum of (A, M) by $\Lambda(A, M)$. The total number of eigenvalues located inside the disk $\mathcal{D}$ is assumed unknown and is denoted by n_{ev}. The eigentriplets of the matrix pencil (A, M) are denoted by $\left(\lambda_i, x^{(i)}, \hat{x}^{(i)}\right)$, $i = 1, \ldots, n$, where λ_i denotes the ith eigenvalue of smallest distance from the center of the disk $\mathcal{D}$, and $x^{(i)}$ and $\left(\hat{x}^{(i)}\right)^H$ denote the corresponding right and left eigenvectors, respectively. Notice that using the above definition we have $\lambda_1, \ldots, \lambda_{n_{ev}} \in \mathcal{D}$ and $\lambda_{n_{ev}+1}, \ldots, \lambda_n \notin \mathcal{D}$. The superscript "$H$" denotes the conjugate transpose of the corresponding matrix. Unless mentioned otherwise, the term "eigenvector" should be understood to refer to a right eigenvector. Throughout the rest of this paper we use the notation $\texttt{rank}(X)$, $\texttt{orth}(X)$, and $\texttt{range}(X)$ to denote the rank, orthonormalization, and range (column space) of the $m \times n$ matrix X, respectively. Moreover, we use the notation $\texttt{span}\left(r^{(1)}, \ldots, r^{(k)}\right)$ to denote the linear span of vectors $r^{(1)}, \ldots, r^{(k)}$.

2. Harmonic Rayleigh-Ritz projections and randomized range finders for column spaces of matrix functions. Computing a few exterior eigenvalues and associated eigenvectors of large and sparse matrix pencils is typically achieved via applying a Rayleigh-Ritz procedure (RR) onto a (nearly) invariant subspace associated with the sought eigenvalues [37]. For Hermitian eigenvalue problems, the RR procedure retains several optimality properties, e.g., see [29]. For non-Hermitian eigenvalue problems no such optimality is guaranteed, e.g., when the sought eigenvalues are located in the interior of the spectrum, e.g., inside a disk $\mathcal{D}$ surrounded by several unwanted eigenvalues, the RR procedure might provide poor results [36].

An alternative for the solution of interior eigenvalue problems is the harmonic Rayleigh-Ritz procedure (HRR) suggested in [35]. More specifically, let matrix Z represent a basis of some projection subspace $\mathcal{Z}$. The HRR procedure extracts approximate eigenpairs of the form (θ, Zq) by solving the following eigenvalue problem

$$Z^H(A - \zeta_c M)^H(A - \zeta_c M)Zq = (\theta - \zeta_c)Z^H(A - \zeta_c M)^H MZq, \quad \zeta_c \in \mathbb{C}. \tag{2.1}$$

For eigenvalue problems such as the ones considered in this paper, it is reasonable to set ζ_c equal to the center of the disk $\mathcal{D}$. The approximate eigenvalue θ and eigenvector Zq are referred to as (harmonic) Ritz value and Ritz vector, respectively. In practice, if the subspace $\mathcal{Z}$ includes the sought invariant subspace, then (2.1) will return accurate approximations of the corresponding eigenpairs provided that there are no spurious eigenvalues close to the Ritz values located inside the disk $\mathcal{D}$ [19, 35].

Based on the above discussion, we seek to compute a subspace $\mathcal{Z}$ which includes the invariant subspace associated with n_{ev} sought eigenvalues $\lambda_1, \ldots, \lambda_{n_{ev}}$. This section considers ansatz subspaces of the form $\mathcal{Z} = \texttt{range}\left(\rho(M^{-1}A)\right)$ for some scalar function ρ such that $\rho(M^{-1}A)$ is rank-deficient. The rest of this section considers such a function ρ while it also discusses a randomized algorithm to compute the range of rank-deficient matrices.

2.1. Fast randomized range finder for rank-deficient matrices. Let $X \in \mathbb{C}^{m\times n}$ be a rectangular matrix. The goal of a range finding procedure is to compute an orthonormal matrix Y such that $\|(I - YY^H)X\|$ is zero. In this paper we are interested in scenarios where matrix X is rank-deficient and accessible only through a Matrix-Vector product routine.

Let $k \in \mathbb{N}$, $k < \min(m, n)$, denote the rank of matrix X. The range of matrix X is equal to the span of the left-singular vectors corresponding to the k non-zero

singular values. The span of these left-singular vectors can be computed in a matrix-free fashion by Lanczos bidiagonalization (LBD) [14]. In the absence of round-off errors, LBD requires k Matrix-Vector products with each of the matrices X and X^H, in addition to the cost introduced by the chosen orthogonalization strategy, e.g., see [17]. Alternatively, we can apply k steps of the Lanczos process on XX^H [26], but this approach still requires $2k$ Matrix-Vector products overall.

ALGORITHM 2.1. *Randomized range finding algorithm*

0. *Inputs:* $X \in \mathbb{C}^{m\times n}$, $Y := 0$
1a. *For* $i = 1, \ldots, \min(m, n)$
2. *Fill* $r \in \mathbb{C}^n$ *with normally distributed random entries*
3. $Y = [Y, Xr]$
4. *Set the* $i \times 1$ *vector* $\sigma^{(Y)}$ *equal to the (sorted) singular values of matrix* Y
5. *If* $\sigma_i^{(Y)}/\sigma_1^{(Y)} \leq$ machine epsilon, *break;*
1b. *End*
6. *Orthonormalize and return* Y

The complexity of the range finding problem can be reduced by considering techniques from randomized linear algebra [12, 30, 34]. Randomized numerical algorithms have gained significant prominence over the last two decades due to their superior performance in several important numerical linear algebra problems, e.g., low-rank matrix approximations [16, 32] and principal component analysis [8, 40]. Returning to the range finding problem, let $R \in \mathbb{C}^{n\times k}$ be a matrix whose entries are drawn from a Gaussian distribution. Then, with probability one, we have $\texttt{rank}(XR) = k$ and $\texttt{range}(XR) = \texttt{range}(X)$ [33]. Thus, a randomized range finder requires only half of the Matrix-Vector products performed by LBD or Lanczos. Our interest lies in scenarios where the exact rank of matrix X is either unknown or expensive to estimate. To bypass this issue, next we consider a modification of the randomized range finder where the Matrix-Vector products with matrix X are performed in an incremental manner and no information regarding k is needed.

Let $r^{(i)} \in \mathbb{C}^n$, $i = 1, 2, \ldots$, denote a sequence of vectors with normally randomly distributed entries, and $\left[Xr^{(1)}, Xr^{(2)}, \ldots\right]$ denote the evolving matrix in which we accumulate the products of matrix X with $r^{(i)}$. After k such products, the rank (and range) of the evolving matrix is equal to that of matrix X. Since the following Matrix-Vector products $Xr^{(i)}$, $i = k+1, k+2, \ldots$, already lie in $\texttt{range}(X)$, the evolving matrix will become singular. Therefore, we can bypass the unknown rank of matrix X by monitoring the singular values of the evolving matrix. The above approach is listed as Algorithm 2.1. The procedure terminates when the ratio of the smallest to the largest singular value of the matrix $[Xr^{(1)}, Xr^{(2)}, \ldots]$ becomes zero, which in a numerical computing environment translates to smaller than or equal to the machine epsilon.[1] In the absence of round-off errors, Algorithm 2.1 terminates after $k+1$ iterations. The SVD of the evolving matrix in Step 3 can be updated on-the-fly each time a new column is added [22, 55]. We note here that Algorithm 2.1 can be also seen as a variation of the adaptive range finder described in [16, Section 4] with the exception that the stopping criterion is based on the magnitude of the condition number of the evolving matrix.

[1] We consider a number to be equal to zero if its numerical value is less than the machine epsilon of the IEEE 754 binary64 definition.

While our interest lies in computing the exact $\texttt{range}(X)$, in practice we stop the iterative procedure in Algorithm 2.1 when the ratio of the smallest to the largest singular value becomes less than a small threshold, e.g., 10^{-12}. This helps to avoid orthonormalizing an ill-conditioned basis at the last step of Algorithm 2.1, e.g., see [13]. If Algorithm 2.1 terminates after $k_0+k_1 < k+1$ iterations, then, with probability at least $1-6k_1^{-k_1}$, the matrix $Y = \texttt{orth}(X\left[r^{(1)},\ldots,r^{(k_0+k_1)}\right])$ satisfies [16]:

$$\|(I-YY^H)X\|_2 \leq \left(1+11\sqrt{k_0+k_1}\sqrt{\min(m,n)}\right)\sigma_{k_0+1}(X), \tag{2.2}$$

where $\sigma_j(X)$ denotes the jth singular value of matrix X. Therefore, when the singular values of matrix X decay fast enough, Algorithm 2.1 can still return a good approximation of $\texttt{range}(X)$ in less than $k+1$ iterations. Note though that we can not predict predict the number of iterations associated with a higher stop tolerance in Algorithm 2.1.

2.2. Column spaces of matrix functions as projection subspaces. Let $\rho : \mathbb{C}^* \to \mathbb{R}$, $\mathbb{C}^* \subseteq \mathbb{C}$, be a scalar function that is defined over $\Lambda(A,M)$. Since (A,M) is diagonalizable, applying the function ρ to matrix $M^{-1}A$ is equivalent to

$$\rho(M^{-1}A) = \sum_{i=1}^{n} \rho(\lambda_i) x^{(i)} \left(\hat{x}^{(i)}\right)^H M. \tag{2.3}$$

Notice now that $\texttt{span}\left(x^{(1)},\ldots,x^{(n_{ev})}\right) \subseteq \texttt{range}(\rho(M^{-1}A))$ for any function ρ such that $\rho(\lambda_i) \neq 0$, $i = 1,\ldots,n_{ev}$. Algorithm 2.2 outlines a two-step procedure to approximate the eigenvalues located inside the disk $\mathcal{D}$ and associated eigenvectors. The first step is to compute an orthonormal basis matrix Z of $\texttt{range}(\rho(M^{-1}A))$ by calling Algorithm 2.1. The number of iterations performed by Algorithm 2.1 is bounded by the number of eigenvalues λ that satisfy $\rho(\lambda) \neq 0$. Therefore, the scalar function ρ should be set such that $\rho(\lambda_i)$ is about equal to machine precision for as many eigenvalues $\lambda_i \notin \mathcal{D}$ as possible. The second step is to perform a HRR projection step to approximate the eigenvalues located inside $\mathcal{D}$ and their associated eigenvectors. Note that no information about the value of n_{ev} is required.

ALGORITHM 2.2. *Prototype algorithm*
0. *Inputs:* $\rho : \mathbb{C} \to \mathbb{R}$, $\mathcal{D}$
1. *Compute an orthonormal basis* Z *of* $\texttt{range}(\rho(M^{-1}A))$ *by Algorithm 2.1*
2. *Solve the eigenvalue problem in (2.1) and return all Ritz values* $\theta \in \mathcal{D}$ *and associated Ritz vectors*

Motivated by the above discussion, an ideal function ρ is defined by the contour integral

$$\mathcal{P}(\zeta) = \frac{-1}{2\pi i}\int_{\Gamma} \frac{1}{\zeta-\nu} d\nu \ , \tag{2.4}$$

where the complex contour Γ denotes the circumference of the disk $\mathcal{D}$, and the integration is performed counter-clockwise. By Cauchy's residue theorem it follows that $\mathcal{P}(\zeta) = 1$ for any $\zeta \in \mathcal{D}$, and zero otherwise. Applying (2.4) to (2.3) yields

$$\mathcal{P}(M^{-1}A) = \frac{-1}{2\pi i}\int_{\Gamma} \left(M^{-1}A-\nu I\right)^{-1} d\nu = \sum_{i=1}^{n_{ev}} x^{(i)} \left(\hat{x}^{(i)}\right)^H M. \tag{2.5}$$

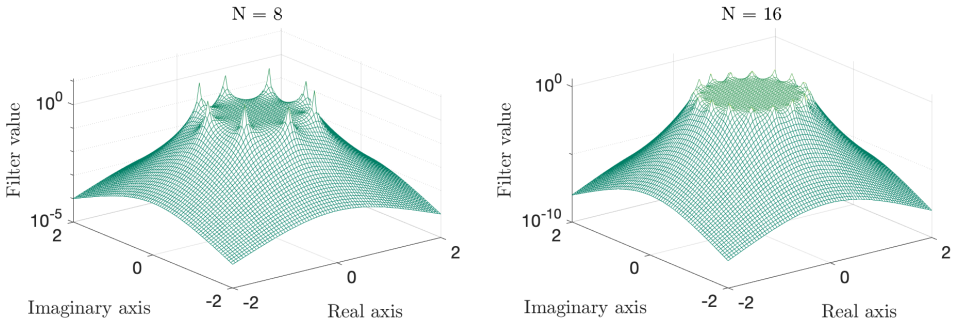


FIG. 2.1. *The modulus of the rational filter $\rho(\zeta)$ defined on the unit disk with the trapezoidal rule of order $N = 8$ (left) and $N = 16$ (right).*

Algorithm 2.2 then terminates after exactly n_{ev} iterations.

In practice, (2.4) will be approximated by numerical quadrature which leads to a rational "filter" function of the form

$$\rho(\zeta) = \sum_{j=1}^{N} \frac{\omega_j}{\zeta - \zeta_j} \ , \tag{2.6}$$

where the integer N denotes the order of the approximation, and the complex pairs $\{\omega_j, \zeta_j\}_{j=1,\ldots,N}$ denote the weights and nodes of the quadrature rule, respectively. Rational filter functions of this form were pioneered in the context of eigenvalue solvers first in [2, 6, 39, 44]. The application of (2.6) to the pencil (A, M) then gives

$$\rho(M^{-1}A) = \sum_{j=1}^{N} \omega_j (M^{-1}A - \zeta_j I)^{-1} = \sum_{j=1}^{N} \omega_j (A - \zeta_j M)^{-1} M, \tag{2.7}$$

and computing $\rho(M^{-1}A)r$ for a vector r at each iteration of Algorithm 2.2 involves: a) one Matrix-Vector product with matrix M, and b) the solution of one linear system with each matrix $A - \zeta_j M$, $j = 1, \ldots, N$. These N linear system solutions can be obtained in parallel by replicating matrices A and M in N different groups of processors.

Ideally, the function in (2.6) should decay to zero as ζ moves away from $\mathcal{D}$. Figure 2.1 plots the modulus of a rational filter $\rho(\zeta)$ defined on the unit disk ($\mathcal{D} \equiv \{|z| : |z| \leq 1\}$) with the trapezoidal rule of order $N = 8$ (left) and $N = 16$ (right). Increasing the value of N leads to a faster decay of the rational filter $\rho(\zeta)$ outside the boundary of $\mathcal{D}$. In particular, the approximation of $\mathcal{P}(\zeta)$ by $\rho(\zeta)$ at the center of the disk $\mathcal{D}$ converges exponentially[2] with respect to N [3, 46].

The convergence of Algorithm 2.1 is likely to be slow for small values of N since `range`$(\rho(M^{-1}A))$ could contain many eigenvector directions associated with a large number of eigenvalues located outside $\mathcal{D}$. As a result, subspace iteration might be a better alternative in this case, and this is exploited in the FEAST eigenvalue solver library [38, 39]. The drawback of subspace iteration as a projection scheme is that a good estimation of n_{ev} is necessary (e.g., see [10, 52, 53]), a condition which is bypassed by Algorithm 2.2.

[2]Note that eigenvalues λ located very close to the poles ζ_j can lead to values $\rho(\lambda)$ which are larger than one even if $\lambda \notin \mathcal{D}$.

Throughout the rest of this paper we describe two variants which aim at reducing the computational cost of Algorithm 2.2.

3. Algorithms based on matrix partitionings. Let $d,\ s \in \mathbb{N}$, such that $n = s + d$, and partition each eigenvector $x^{(i)},\ i = 1, \ldots, n$, of the pencil (A, M) as

$$x^{(i)} = \begin{pmatrix} u^{(i)} \\ y^{(i)} \end{pmatrix},\ u^{(i)} \in \mathbb{C}^d,\ y^{(i)} \in \mathbb{C}^s. \tag{3.1}$$

In addition, let $0_{\chi,\psi}$ denote the zero matrix of size $\chi \times \psi$. Then, we can write

$$\begin{aligned} \texttt{span}\left(x^{(1)}, \ldots, x^{(n_{ev})}\right) &= \texttt{span}\left(\begin{bmatrix} u^{(1)}, \ldots, u^{(n_{ev})} \\ 0_{s,n_{ev}} \end{bmatrix} + \begin{bmatrix} 0_{d,n_{ev}} \\ y^{(1)}, \ldots, y^{(n_{ev})} \end{bmatrix}\right) \qquad (3.2) \\ &\subseteq \texttt{span}\left(\begin{bmatrix} u^{(1)}, \ldots, u^{(n_{ev})} \\ 0_{s,n_{ev}} \end{bmatrix}\right) \oplus \texttt{span}\left(\begin{bmatrix} 0_{d,n_{ev}} \\ y^{(1)}, \ldots, y^{(n_{ev})} \end{bmatrix}\right). \qquad (3.3) \end{aligned}$$

The expression in (3.2) implies that $\texttt{span}\left(x^{(1)}, \ldots, x^{(n_{ev})}\right)$ is captured by the direct sum of $\texttt{span}\left(u^{(1)}, \ldots, u^{(n_{ev})}\right)$ and $\texttt{span}\left(y^{(1)}, \ldots, y^{(n_{ev})}\right)$. The rest of this section describes two variations of Algorithm 2.2, presented in Sections 3.2 and 3.3. These algorithms make use of matrix partitioning to reduce the computational costs associated with the construction of a good HRR projection subspace.

3.1. Two equivalent matrix resolvent representations. Consider the following 2×2 block-partitioning of the non-Hermitian matrices A and M:

$$A = \begin{pmatrix} B & F \\ E & C \end{pmatrix} \quad \text{and} \quad M = \begin{pmatrix} M_B & M_F \\ M_E & M_C \end{pmatrix}, \tag{3.4}$$

where $B,\ M_B \in \mathbb{C}^{d\times d},\ F,\ M_F \in \mathbb{C}^{d\times s},\ E,\ M_E \in \mathbb{C}^{s\times d}$, and $C,\ M_C \in \mathbb{C}^{s\times s}$. Moreover, define the following matrix-valued functions of $\zeta \in \mathbb{C}$:

$$B(\zeta) = B - \zeta M_B,\quad F(\zeta) = F - \zeta M_F,\quad E(\zeta) = E - \zeta M_E,\ \text{and}\ \ C(\zeta) = C - \zeta M_C.$$

For any $\zeta \notin \Lambda(A, M)$, the matrix $(A - \zeta M)^{-1}$ can be written as

$$(A - \zeta M)^{-1} = \begin{pmatrix} B(\zeta)^{-1}\left[I + F(\zeta)S(\zeta)^{-1}E(\zeta)B(\zeta)^{-1}\right] & -B(\zeta)^{-1}F(\zeta)S(\zeta)^{-1} \\ -S(\zeta)^{-1}E(\zeta)B(\zeta)^{-1} & S(\zeta)^{-1} \end{pmatrix}, \tag{3.5}$$

where the $s \times s$ matrix-valued function

$$S(\zeta) = C(\zeta) - E(\zeta)B(\zeta)^{-1}F(\zeta)$$

is the *Schur complement* of matrix $A - \zeta M$. Combining (3.5) with (2.7) then gives

$$\rho(M^{-1}A) = \sum_{j=1}^{N} \omega_j \begin{bmatrix} B(\zeta_j)^{-1}\left[I + F(\zeta_j)S(\zeta_j)^{-1}E(\zeta_j)B(\zeta_j)^{-1}\right] & -B(\zeta_j)^{-1}F(\zeta_j)S(\zeta_j)^{-1} \\ -S(\zeta_j)^{-1}E(\zeta_j)B(\zeta_j)^{-1} & S(\zeta_j)^{-1} \end{bmatrix} M. \tag{3.6}$$

Similarly, the matrix $(A - \zeta_j M)^{-1}$ can be expressed in terms of the eigenvectors of the matrix pencil (A, M) as

$$(A - \zeta_j M)^{-1} = \sum_{i=1}^{n} \frac{x^{(i)}\left(\hat{x}^{(i)}\right)^H}{\lambda_i - \zeta_j}. \tag{3.7}$$

Then, by partitioning the left eigenvectors of the pencil (A, M) as in (3.1),

$$\left(\hat{x}^{(i)}\right)^H = \begin{bmatrix} \left(\hat{u}^{(i)}\right)^H & \left(\hat{y}^{(i)}\right)^H \end{bmatrix}, \ \left(\hat{u}^{(i)}\right)^H \in \mathbb{C}^{1\times d}, \ \left(\hat{y}^{(i)}\right)^H \in \mathbb{C}^{1\times s},$$

and combining (2.7) with (3.7), we obtain the following identity:

$$\rho(M^{-1}A) = \sum_{i=1}^{n} \rho(\lambda_i) \begin{bmatrix} u^{(i)}\left(\hat{u}^{(i)}\right)^H & u^{(i)}\left(\hat{y}^{(i)}\right)^H \\ y^{(i)}\left(\hat{u}^{(i)}\right)^H & y^{(i)}\left(\hat{y}^{(i)}\right)^H \end{bmatrix} M. \tag{3.8}$$

3.2. First algorithm. In this section we present an algorithm which exploits the equivalent representations of $\rho(M^{-1}A)$ shown in (3.6) and (3.8) to build a subspace which captures $\texttt{span}\left(u^{(1)}, \ldots, u^{(n_{ev})}\right)$ and $\texttt{span}\left(y^{(1)}, \ldots, y^{(n_{ev})}\right)$.

Equating the (1,2) and (2,2) blocks on the right-hand sides of (3.6) and (3.8) gives

$$\begin{aligned} -\sum_{j=1}^{N} \omega_j B(\zeta_j)^{-1} F(\zeta_j) S(\zeta_j)^{-1} &= \sum_{i=1}^{n} \rho(\lambda_i) u^{(i)} \left(\hat{y}^{(i)}\right)^H, \\ \sum_{j=1}^{N} \omega_j S(\zeta_j)^{-1} &= \sum_{i=1}^{n} \rho(\lambda_i) y^{(i)} \left(\hat{y}^{(i)}\right)^H. \end{aligned} \tag{3.9}$$

These identities indicate that, under mild conditions, we can capture a superset of $\texttt{span}\left(u^{(1)}, \ldots, u^{(n_{ev})}\right)$ and $\texttt{span}\left(y^{(1)}, \ldots, y^{(n_{ev})}\right)$ by capturing the range of the matrices on the left-hand side in (3.9).

THEOREM 3.1. *Let* $\left[u^{(i)}\right]_{\rho(\lambda_i)\neq 0}, \left[y^{(i)}\right]_{\rho(\lambda_i)\neq 0}$, *and* $\left[\hat{y}^{(i)}\right]_{\rho(\lambda_i)\neq 0}$, *denote the matrices whose columns are formed by those vectors* $u^{(i)}$, $y^{(i)}$, *and* $\hat{y}^{(i)}$, *for which* $\rho(\lambda_i) \neq 0$, $i = 1, \ldots, n$, *respectively. If the rank of matrices* $\sum_{j=1}^{N} \omega_j B(\zeta_j)^{-1} F(\zeta_j) S(\zeta_j)^{-1}$ *and* $\sum_{j=1}^{N} \omega_j S(\zeta_j)^{-1}$ *is equal to that of matrices* $\left[u^{(i)}\right]_{\rho(\lambda_i)\neq 0}$ *and* $\left[y^{(i)}\right]_{\rho(\lambda_i)\neq 0}$, *respectively, then:*

$$\texttt{range}\left(\left[u^{(i)}\right]_{\rho(\lambda_i)\neq 0}\right) = \texttt{range}\left(\sum_{j=1}^{N} \omega_j B(\zeta_j)^{-1} F(\zeta_j) S(\zeta_j)^{-1}\right), \tag{3.10}$$

and

$$\texttt{range}\left(\left[y^{(i)}\right]_{\rho(\lambda_i)\neq 0}\right) = \texttt{range}\left(\sum_{j=1}^{N} \omega_j S(\zeta_j)^{-1}\right). \tag{3.11}$$

Proof. First, notice that $\texttt{range}\left(\left[\rho(\lambda_i)u^{(i)}\right]_{\rho(\lambda_i)\neq 0}\right) = \texttt{range}\left(\left[u^{(i)}\right]_{\rho(\lambda_i)\neq 0}\right)$, and $\texttt{range}\left(\left[\rho(\lambda_i)y^{(i)}\right]_{\rho(\lambda_i)\neq 0}\right) = \texttt{range}\left(\left[y^{(i)}\right]_{\rho(\lambda_i)\neq 0}\right)$. Second, we have

$$\sum_{j=1}^{N} \omega_j B(\zeta_j)^{-1} F(\zeta_j) S(\zeta_j)^{-1} = \left[\rho(\lambda_i)u^{(i)}\right]_{\rho(\lambda_i)\neq 0} \left[\hat{y}^{(i)}\right]_{\rho(\lambda_i)\neq 0}^{H},$$

and

$$\sum_{j=1}^{N} \omega_j S(\zeta_j)^{-1} = \left[\rho(\lambda_i)y^{(i)}\right]_{\rho(\lambda_i)\neq 0} \left[\hat{y}^{(i)}\right]_{\rho(\lambda_i)\neq 0}^{H}.$$

Recall now that for two matrices X_1 and X_2, if the rank of the matrix X_1X_2 is equal to that of X_1, then the span of the columns of X_1 is equal to the range of X_1X_2. The results in (3.10) and (3.11) follow directly by setting $X_1 = \left[\rho(\lambda_i)u^{(i)}\right]_{\rho(\lambda_i)\neq 0}$ in (3.10) and $X_1 = \left[\rho(\lambda_i)y^{(i)}\right]_{\rho(\lambda_i)\neq 0}$ in (3.11), respectively, while $X_2 = \left[\hat{y}^{(i)}\right]^H_{\rho(\lambda_i)\neq 0}$. ∎

Theorem 3.1 implies that a necessary condition for (3.10) and (3.11) to hold is

$$\max\left(\texttt{rank}\left(\left[u^{(i)}\right]_{\rho(\lambda_i)\neq 0}\right), \texttt{rank}\left(\left[y^{(i)}\right]_{\rho(\lambda_i)\neq 0}\right)\right) \leq \texttt{rank}\left(\left[\hat{y}^{(i)}\right]_{\rho(\lambda_i)\neq 0}\right). \tag{3.12}$$

For symmetric eigenvalue problems we have $y^{(i)} = \hat{y}^{(i)}$ and (3.12) is trivially satisfied [20]. In practice, the violation of (3.12) for non-Hermitian eigenvalue problems is quite rare in a finite-precision arithmetic environment.

Algorithm 3.1 outlines a matrix partitioning procedure to build the HRR projection subspace in (2.1) by setting the latter subspace equal to the direct sum of subspaces $\texttt{range}\left(\sum_{j=1}^N \omega_j B(\zeta_j)^{-1}F(\zeta_j)S(\zeta_j)^{-1}\right)$ and $\texttt{range}\left(\sum_{j=1}^N \omega_j S(\zeta_j)^{-1}\right)$. Each instance of Algorithm 2.1 called in Algorithm 3.1 performs a number of iterations which is at most equal to the number of eigenvalues λ for which $\rho(\lambda) \neq 0$. Moreover, the two instances of Algorithm 2.1 shown in Steps 1 and 2 are performed in parallel and thus the linear system solutions computed in Step 2 are exploited at Step 1 as well. Moreover, similarly to Algorithm 2.2, Algorithm 3.1 requires no estimation of the value of n_{ev}.

ALGORITHM 3.1.
0a. Inputs: N, $\mathcal{D}$
0b. Compute the complex pairs $\{\omega_j, \zeta_j\}_{j=1,2,\ldots,N}$, *set* $G := W := 0$
0c. (Optionally) Reorder (A, M) *as in Section 4.2*
1. Compute an orthonormal basis G *of* $\texttt{range}\left(\sum_{j=1}^N \omega_j S(\zeta_j)^{-1}\right)$ *by Algorithm 2.1*
2. Compute an orthonormal basis W *of* $\texttt{range}\left(\sum_{j=1}^N \omega_j B(\zeta_j)^{-1}F(\zeta_j)S(\zeta_j)^{-1}\right)$ *by Algorithm 2.1*
3. Set $Z = [\,W\ G\,]$, *solve the eigenvalue problem in (2.1) and return all Ritz values* $\theta \in \mathcal{D}$ *and associated Ritz vectors*

Unless mentioned otherwise, the default value of the number of poles in the rational filter ρ will be equal to $N = 16$.

3.3. Second algorithm. This section describes an alternative technique to construct the matrix W in Algorithm 3.1 under the assumption that the pencil (B, M_B) is diagonalizable. Throughout the rest of this section we will denote the eigentriplets of the pencil (B, M_B) by $\left(\delta_i, v^{(i)}, \hat{v}^{(i)}\right)$, $i = 1, 2, \ldots, d$, where δ_i denotes the eigenvalue of (B, M_B) with the ith shortest distance from the center of the disk $\mathcal{D}$, and $v^{(i)}$ and $\left(\hat{v}^{(i)}\right)^H$ denote the corresponding right and left eigenvectors, respectively.

By combining (3.2) and (3.4), we can write the top $d \times 1$ part of the eigenvector $x^{(i)} = \begin{pmatrix} u^{(i)} \\ y^{(i)} \end{pmatrix}$ associated with the eigenvalue λ_i as

$$u^{(i)} = -B(\lambda_i)^{-1}F(\lambda_i)y^{(i)}. \tag{3.13}$$

While expression (3.13) is not practical, it serves as a starting point for the construction of a subspace which (approximately) captures $\texttt{span}\left(u^{(i)}\right)$ without depending on the (unknown) quantities λ_i and $y^{(i)}$.

Let G be a matrix such that $y^{(i)} \in \texttt{range}(G)$, e.g., the matrix G constructed in Algorithm 3.1. In addition, define the matrices

$$V_\phi = \left[v^{(1)}, v^{(2)}, \ldots, v^{(\phi)}\right] \quad \text{and} \quad \hat{V}_\phi = \left[\hat{v}^{(1)}, \hat{v}^{(2)}, \ldots, \hat{v}^{(\phi)}\right],$$

where $\phi \in \mathcal{Z}^*$ is larger than or equal to the number of eigenvalues of (B, M_B) located inside the disk $\mathcal{D}$. Taking advantage of the identity $I = V_\phi \hat{V}_\phi^H M_B + (I - V_\phi \hat{V}_\phi^H M_B)$, and noticing that $\texttt{span}\left(V_\phi \hat{V}_\phi^H M_B B(\lambda_i)^{-1} F(\lambda_i) y^{(i)}\right) \subseteq \texttt{span}\left(v^{(1)}, v^{(2)}, \ldots, v^{(\phi)}\right)$, we can write

(3.14)
$$\begin{aligned}
\texttt{span}\left(u^{(i)}\right) &= \texttt{span}\left(B(\lambda_i)^{-1} F(\lambda_i) y^{(i)}\right) \\
&= \texttt{span}\left(V_\phi \hat{V}_\phi^H M_B B(\lambda_i)^{-1} F(\lambda_i) y^{(i)} + (I - V_\phi \hat{V}_\phi^H M_B) B(\lambda_i)^{-1} F(\lambda_i) y^{(i)}\right) \\
&\subseteq \texttt{span}\left(v^{(1)}, v^{(2)}, \ldots, v^{(\phi)}\right) + \texttt{span}\left((I - V_\phi \hat{V}_\phi^H M_B) B(\lambda_i)^{-1} F(\zeta) G\right) + \\
&\quad \texttt{span}\left((I - V_\phi \hat{V}_\phi^H M_B) B(\lambda_i)^{-1} M_F G\right),
\end{aligned}$$

where $\zeta \in \mathcal{D}$, and we replaced $F(\lambda_i)$ by its equivalent form

$$F(\lambda_i) = F(\zeta) - (\lambda_i - \zeta) M_F.$$

The expression in (3.14) still depends on λ_i through the term $B(\lambda_i)^{-1}$. Next, we show an equivalent expression of the matrix $(I - V_\phi \hat{V}_\phi^H M_B) B(\lambda_i)^{-1}$.

THEOREM 3.2. *Let $\zeta_c \in \mathbb{C}$ be the center of disk $\mathcal{D}$ and $\phi \in \mathbb{N}$ larger than or equal to the number of eigenvalues of (B, M_B) located inside $\mathcal{D}$. If we define the matrix*

$$\widetilde{B}(\zeta) := \left(I - V_\phi \hat{V}_\phi^H M_B\right) B(\zeta)^{-1},$$

then

$$\left(I - V_\phi \hat{V}_\phi^H M_B\right) B(\lambda_i)^{-1} = \widetilde{B}(\zeta_c) \sum_{k=0}^{\infty} \left[(\lambda_i - \zeta_c) M_B \widetilde{B}(\zeta_c)\right]^k, \tag{3.15}$$

for any $\lambda_i \in \mathcal{D}$.

Proof. Define the matrices

$$V = \left[V_\phi, v^{(\phi+1)}, \ldots, v^{(d)}\right] \quad \text{and} \quad \hat{V} = \left[\hat{V}_\phi, \hat{v}^{(\phi+1)}, \ldots, \hat{v}^{(d)}\right].$$

Recall that $\hat{V}^H M_B V = I$, and thus $M_B = \hat{V}^{-H} V^{-1}$ and $B = \hat{V}^{-H} \begin{pmatrix} \delta_1 & & \\ & \ddots & \\ & & \delta_d \end{pmatrix} V^{-1}$. Using the above identities we can write

$$\begin{aligned}
\widetilde{B}(\zeta) &= \left(I - V_\phi \hat{V}_\phi^H M_B\right) V \begin{pmatrix} \delta_1 - \zeta & & \\ & \ddots & \\ & & \delta_d - \zeta \end{pmatrix}^{-1} \hat{V}^H \\
&= V \begin{pmatrix} 0_{\phi,\phi} & & & \\ & \frac{1}{\delta_{\phi+1} - \zeta} & & \\ & & \ddots & \\ & & & \frac{1}{\delta_d - \zeta} \end{pmatrix} \hat{V}^H.
\end{aligned}$$

Let us now define the scalar $\gamma_j = \dfrac{\lambda_i - \zeta_c}{\delta_j - \zeta_c}$. We can write

$$\widetilde{B}(\zeta_c)\left[(\lambda_i-\zeta_c)M_B\widetilde{B}(\zeta_c)\right]^k = V\begin{pmatrix} 0_{\phi,\phi} & & & \\ & \frac{\gamma_{\phi+1}^k}{\delta_{\phi+1}-\zeta_c} & & \\ & & \ddots & \\ & & & \frac{\gamma_d^k}{\delta_d-\zeta_c}\end{pmatrix}\hat{V}^H.$$

Accounting for all powers $k = 0, 1, 2, \ldots$, gives

$$\widetilde{B}(\zeta_c)\sum_{k=0}^{\infty}\left[(\lambda_i-\zeta_c)M_B\widetilde{B}(\zeta_c)\right]^k = V\begin{pmatrix} 0_{\phi,\phi} & & & \\ & \frac{\sum_{k=0}^{\infty}\gamma_{\phi+1}^k}{\delta_{\phi+1}-\zeta_c} & & \\ & & \ddots & \\ & & & \frac{\sum_{k=0}^{\infty}\gamma_d^k}{\delta_d-\zeta_c}\end{pmatrix}\hat{V}^H.$$

Since ζ_c is the center of $\mathcal{D}$, it follows that $|\gamma_j| < 1$ for any $\delta_j \notin \mathcal{D}$. Therefore, the geometric series converges and $\sum_{k=0}^{\infty}\gamma_j^k = \dfrac{1}{1-\gamma_j} = \dfrac{\delta_j-\zeta_c}{\delta_j-\lambda_i}$. It follows that $\dfrac{1}{\delta_j-\zeta_c}\sum_{k=0}^{\infty}\gamma_j^k = \dfrac{1}{\delta_j-\lambda_i}$.

We finally have

$$\begin{aligned}\widetilde{B}(\zeta_c)\sum_{k=0}^{\infty}\left[(\lambda_i-\zeta_c)M_B\widetilde{B}(\zeta_c)\right]^k &= V\begin{pmatrix} 0_{\phi,\phi} & & & \\ & \frac{1}{\delta_{\phi+1}-\lambda_i} & & \\ & & \ddots & \\ & & & \frac{1}{\delta_d-\lambda_i}\end{pmatrix}\hat{V}^H \\ &= \left(I - V_\phi\hat{V}_\phi^H M_B\right)B(\lambda_i)^{-1}.\end{aligned}$$

This concludes the proof. ∎

Theorem 3.2 implies that we can approximate $\left(I - V_\phi\hat{V}_\phi^H M_B\right)B(\lambda_i)^{-1}$ through a finite truncation of the right-hand side in (3.15). The approximation error of this truncation is considered in the following proposition.

PROPOSITION 3.3. *Let ψ be a positive integer and define the error matrix*

$$R_\psi(\lambda_i) = (I - V_\phi\hat{V}_\phi^H M_B)B(\lambda_i)^{-1} - \widetilde{B}(\zeta_c)\sum_{k=0}^{\psi}\left[(\lambda_i-\zeta_c)M_B\widetilde{B}(\zeta_c)\right]^k.$$

Then

$$R_\psi(\lambda_i) = \sum_{k=\psi+1}^{\infty}\sum_{j=\phi+1}^{d}\left[\frac{(\lambda_i-\zeta_c)^k}{(\delta_j-\zeta_c)^{k+1}}\right]v^{(j)}\left(\hat{v}^{(j)}\right)^H M_B. \tag{3.16}$$

Proof. Recall the scalar $\gamma_j = \dfrac{\lambda_i - \zeta_c}{\delta_j - \zeta_c}$. The matrix $R_\psi(\lambda_i)$ is then equal to

$$R_\psi(\lambda_i) = V \begin{pmatrix} 0_{\phi,\phi} & & & \\ & \frac{\sum_{k=\psi+1}^{\infty} \gamma_{\phi+1}^k}{\delta_{\phi+1} - \zeta_c} & & \\ & & \ddots & \\ & & & \frac{\sum_{\psi+1}^{\infty} \gamma_d^k}{\delta_d - \zeta_c} \end{pmatrix} \hat{V}^H.$$

The proof concludes by replacing γ_j with its ratio. ∎

Proposition 3.3 indicates that when ζ_c is close to the sought eigenvalues $\lambda_1, \ldots, \lambda_{nev}$ and the eigenvalues $\delta_{\phi+1}, \ldots, \delta_d$ are located far away from the disk $\mathcal{D}$, then the matrix $\widetilde{B}(\zeta_c) \sum_{k=0}^{\psi} \left[(\lambda_i - \zeta_c) M_B \widetilde{B}(\zeta_c)\right]^k$ can be used as an accurate approximation of the matrix $(I - V_\phi \hat{V}_\phi^H M_B) B(\lambda_i)^{-1}$ even for small values of ψ.

Based on the above discussion, we expect to find a reasonable approximation of the subspace $\texttt{span}\left(u^{(1)}, \ldots, u^{(n_{ev})}\right)$ in the range of the matrix

$$W_{\phi,\psi} = \Big[V_\phi,\ \left(I - V_\phi \hat{V}_\phi^H M_B\right) \hat{B}_\psi(\zeta_c) \hat{G}_F,\ \underbrace{\left(I - V_\phi \hat{V}_\phi^H M_B\right) \hat{B}_\psi(\zeta_c) \hat{G}_{M_F}}_{\text{only if } M_F \neq 0}\Big], \tag{3.17}$$

where

$$\hat{B}_\psi(\zeta_c) = \left[\tilde{B}(\zeta_c), \tilde{B}(\zeta_c)\left[M_B \tilde{B}(\zeta_c)\right], \ldots, \tilde{B}(\zeta_c)\left[M_B \tilde{B}(\zeta_c)\right]^\psi\right],$$

and we set

$$\hat{G}_F = \begin{bmatrix} F(\zeta_c)G & & & \\ & F(\zeta_c)G & & \\ & & \ddots & \\ & & & F(\zeta_c)G \end{bmatrix},\ \hat{G}_{M_F} = \begin{bmatrix} M_F G & & & \\ & M_F G & & \\ & & \ddots & \\ & & & M_F G \end{bmatrix}.$$

ALGORITHM 3.2.
- 0a. *Inputs:* N, $\mathcal{D}$, ψ (optionally), ϕ *(optionally)*
- 0b. *Compute the complex pairs* $\{\omega_j, \zeta_j\}_{j=1,2,\ldots,N}$, *set* $G := 0$
- 0c. *(Optionally) Reorder* (A, M) *as in Section 4.2*
- 1. *Compute an orthonormal basis* G *of* $\texttt{range}\left(\sum_{j=1}^{N} \omega_j S(\zeta_j)^{-1}\right)$ *by Algorithm 2.1*
- 2. *Compute the eigenpairs associated with the* ϕ *eigenvalues of smallest modulus of the pencil* $(B(\zeta), M_B)$ *and form the matrix* V_ϕ
- 3. *Set the matrix* $W_{\phi,\psi}$ *as in (3.17)*
- 4. *Set* $Z = \begin{bmatrix} W_{\phi,\psi} \\ G \end{bmatrix}$, *solve the eigenvalue problem in (2.1) and return all Ritz values* $\theta \in \mathcal{D}$ *and associated Ritz vectors*

The complete algorithmic procedure is summarized in Algorithm 3.2. The accuracy in the approximation of the eigenpairs $(\lambda_i, x^{(i)})$, $i = 1, \ldots, n_{ev}$, depends on

TABLE 4.1

Total number of linear system solutions of the form $B(\zeta)x_d = b_d$ and $S(\zeta)x_s = b_s$ computed by Algorithm 2.2, Algorithm 3.1, Algorithm 3.2, and the algorithm used in the FEAST software package. The variables $\eta_1 \in \mathbb{N}$, $\eta_2 \in \mathbb{N}$, $\eta_3 \in \mathbb{N}$, denote the number of iterations performed by Algorithm 2.1 when called from Algorithm 2.2, Algorithm 3.1, and Algorithm 3.2, respectively. The variable τ_ϕ denotes the number of linear systems of the form $B(\zeta_c)x_d = b_d$ required to compute the ϕ sought eigenvectors of the pencil $(B - \zeta_c M_B, M_B)$ by Implicitly Restarted Arnoldi (IRA) combined with shift-and-invert [27]. The variable $\eta_4 \in \mathbb{N}$ denotes the number of iterations performed by subspace iteration.

			Alg. 3.2		
	Alg. 2.2	Alg. 3.1	$M_F = 0$	$M_F \neq 0$	Sub. It.
$B(\zeta)x_d = b_d$	$2N\eta_1$	$N\eta_2$	$\eta_3(\psi+1)+\tau_\phi$	$2\eta_3(\psi+1)+\tau_\phi$	$2mN\eta_4$
$S(\zeta)x_s = b_s$	$N\eta_1$	$N\eta_2$	$N\eta_3$	$N\eta_3$	$mN\eta_4$

the distance of the eigenvalues $\lambda_i \in \mathcal{D}$ from both the center of the disk $\mathcal{D}$ and the (non-deflated) eigenvalues of the matrix pencil (B, M_B). In contrast, the accuracy provided by Algorithm 3.1 is irrespective to the location of the eigenvalues $\lambda_i \in \mathcal{D}$. Thus, the latter should be the algorithm of choice when one seeks higher accuracy in the approximation of the n_{ev} sought eigenpairs of the pencil (A, M). On the other hand, Algorithm 3.2 should be preferred when a few digits of accuracy are deemed enough, and lower wall-clock execution time is critical.

Compared to Algorithm 3.1, Algorithm 3.2 introduces two new parameters, $\psi \in \mathbb{N}$ and $\phi \in \mathbb{N}$. Larger values of these two integers lead to higher accuracy but increase the associated computational cost. Increasing the value of ψ aims at reducing the error along all eigenvector directions of (B, M_B), while increasing the value of ϕ aims at eliminating the approximation error associated with eigenvectors corresponding closer to the center of the disk $\mathcal{D}$. Generally speaking, the main improvements in accuracy come from increasing the value of ψ. Our default choice is to set $\psi = 1$, and ϕ equal to the number of eigenvalues of the pencil (B, M_B) located inside the disk $\mathcal{D}$. If additional accuracy is needed, one can augment $W_{\phi,\psi}$ with either additional eigenvectors of the pencil (B, M_B) (i.e., increase ϕ), or additional resolvent approximation matrix terms (i.e., increase ψ) and only repeat the Rayleigh-Ritz projection step. This approach can be repeated more than once, i.e., until the residual norms of all n_{ev} approximate eigenpairs are less a chosen threshold.

4. Practical details.

4.1. Computational cost comparison.

The main computational bottleneck of the rational filtering algorithms discussed in this paper is the solution of complex-shifted sparse linear systems of the form $B(\zeta)x_d = b_d$ and $S(\zeta)x_s = b_s$. Therefore, an algorithm that requires fewer such linear system solutions will typically be faster as well.

Table 4.1 summarizes the computational costs of Algorithm 2.2, Algorithm 3.1, and Algorithm 3.2, where we assume that all linear systems are solved by a direct solver and their complexity is oblivious to the actual value of $\zeta \notin \Lambda(B, M_B)$. The variables $\eta_1 \in \mathbb{N}$, $\eta_2 \in \mathbb{N}$, $\eta_3 \in \mathbb{N}$, denote the number of iterations performed by Algorithm 2.1 when called from Algorithm 2.2, Algorithm 3.1, and Algorithm 3.2, respectively. It is straightforward to observe that when $\eta_1 \approx \eta_2 \approx \eta_3$, Algorithm 3.1 requires about half linear system solutions of the form $B(\zeta)x_d = b_d$ than what Algorithm 2.2 does. Moreover, Algorithm 3.2 requires a number of linear system solutions

which is independent of the number of poles N. Thus, larger values of N should increase the computational complexity gap in favor of Algorithm 3.2. For comparison purposes we also list the computational complexity of subspace iteration applied to matrix $\rho(M^{-1}A)$ with an initial subspace of dimension $m \geq n_{ev}$. In contrast to the algorithms proposed in this paper, the convergence of subspace iteration depends on the dimension m of its initial subspace.

4.2. Matrix partitionings. The matrix partitioning algorithms discussed in this paper can take advantage of a reordering of the pencil (A, M) so that the pencil (B, M_B) is block-diagonal. For Algorithm 3.2 this implies that the computation of the matrix V_ϕ then decouples into p independent generalized non-Hermitian eigenvalue problems. The eigenvalues of each one of these p matrix pencils can be then computed in parallel.

To obtain the above reordering we partition the adjacency graph of the matrix $|A| + \left|A^T\right| + |M| + \left|M^T\right|$ into $p \geq 2$ non-overlapping partitions [42]. We then reorder the equations/unknowns so that the interior variables across all partitions are ordered before the interface ones. The latter procedure is equivalent to transforming the original pencil (A, M) into the form $\left(PAP^T, PMP^T\right)$, where the $n \times n$ matrix P holds the row permutation of (A, M). The eigenpairs of (A, M) are connected with those of the matrix pencil $\left(PAP^T, PMP^T\right)$ through the formula

$$PAP^T\left(Px^{(i)}\right) = \lambda_i PMP^T\left(Px^{(i)}\right).$$

The matrices PAP^T and PMP^T can be written us

$$PAP^T = \begin{pmatrix} B_1 & & & & F_1 \\ & B_2 & & & F_2 \\ & & \ddots & & \vdots \\ & & & B_p & F_p \\ E_1 & E_2 & \dots & E_p & C \end{pmatrix}, \quad PMP^T = \begin{pmatrix} M_B^{(1)} & & & & M_F^{(1)} \\ & M_B^{(2)} & & & M_F^{(2)} \\ & & \ddots & & \vdots \\ & & & M_B^{(p)} & M_F^{(p)} \\ M_E^{(1)} & M_E^{(2)} & \dots & M_E^{(p)} & M_C \end{pmatrix}$$

where matrices B_i and $M_B^{(i)}$ are square matrices of size $d_i \times d_i$, matrices F_i (E_i) and $M_F^{(i)}$ $\left(M_E^{(i)}\right)$ are of size $d_i \times s_i$ $(s_i \times d_i)$, and the integers d_i and s_i denote the number of interior and interface nodes located in the ith subdomain of the adjacency graph of $|A| + \left|A^T\right| + |M| + \left|M^T\right|$, respectively. On the other hand, matrices C and M_C are of size $s \times s$ where $s = \sum_{i=1}^{p} s_i$.

5. Experiments. The numerical experiments presented in this section were performed in a Matlab environment (version R2018b), using 64-bit arithmetic, on a single core of a MacBook Pro equipped with a quad-core 2.5 GHz Intel Core i7 processor and 16 GB 1600 MHz DDR3 of system memory. The matrices used throughout our experiments are listed in Table 5.1 and can be retrieved from SuiteSparse Matrix Collection [9] and Matrix Market repository [7].

Throughout the rest of this section we consider the application of three different algorithms: a) Algorithm 3.1, b) Algorithm 3.2, and finally c) subspace iteration with the matrix $\rho(M^{-1}A)$, where the initial subspace is of dimension $m \geq n_{ev}$. We will refer to this approach as RSI. As a separate note, a high-performance implementation of subspace iteration with rational filtering can be found in the FEAST software package.

TABLE 5.1
n: size of matrices A and M, nnz(.): number of nonzero entries.

#	Matrix pencil	n	$nnz(A)/n$	$nnz(M)/n$	Application
1.	`bfw782`	782	9.6	7.6	Engineering
2.	`utm1700b`	1,700	12.7	1.0	Electromagnetics
3.	`wang1`	2,903	6.6	1.0	Semiconductors
4.	`rdb32001`	3,200	5.9	1.0	CFD
5.	`thermal`	3,456	19.2	1.0	Thermal
6.	`dw4096`	8,192	5.1	1.0	Engineering
7.	`big`	13,209	6.9	1.0	Directed weighted graph

The radius of the disk $\mathcal{D}$ is set equal to 1.001 times the radius of the minimal enclosing circle of eigenvalues $\lambda_1, \ldots, \lambda_{n_{ev}}$. The rational filter function in (2.6) is constructed through discretizing (2.4) by the trapezoidal rule of order N. Throughout the rest of this section we assume that the iterative loop in Algorithm 2.1 terminates when the ratio of the smallest to the largest singular value is less than or equal to 1.0×10^{-12}, and we set a maximum number of four hundred iterations. All matrix pencils were reordered as discussed in Section 4.2 using $p = 8$. The residual norm of each approximate eigenpair $(\hat{\lambda}, \hat{x})$ is computed as $\hat{\rho} = \dfrac{\|A\hat{x} - \hat{\lambda}M\hat{x}\|_2}{\|A\hat{x}\|_2 + |\hat{\lambda}|\|M\hat{x}\|_2}$. All algorithms discussed in this section return only those approximate eigenpairs $(\hat{\lambda}, \hat{x})$ for which $\hat{\lambda} \in \mathcal{D}$. When more than n_{ev} approximate eigenvalues are located in $\mathcal{D}$ we purge the spurious ones by keeping only those for which the associated residual norm is smaller than the threshold tolerance 1.0×10^{-3}. This approach was successful in all experiments we performed.

5.1. A detailed example. We consider the computation of the $n_{ev} = 20$ eigenvalues of smallest modulus (and their associated eigenvectors) of the matrices `wang1` and `thermal`. The size of the Schur complement matrices after the application of the graph partitioner is equal to $s = 576$, and $s = 668$, respectively. The application of Algorithm 3.1 is visualized in Figure 5.1. The first row of plots shows the n_{ev} sought and n_{ev} immediate unwanted eigenvalues of smallest modulus, while the second row of shows the ratio of the smallest to the largest singular value as determined at each iteration of Algorithm 2.1 during its application to matrix $\sum_{j=1}^{N} \omega_j S(\zeta_j)^{-1}$.[3] Notice that this ratio approximates zero and decreases faster for larger values of N as a consequence of the fact that the rank of the matrix $\sum_{j=1}^{N} \omega_j S(\zeta_j)^{-1}$ approaches that of the matrix $\sum_{i=1}^{n_{ev}} y^{(i)} \left(\hat{y}^{(i)}\right)^H$, which is bounded from above by n_{ev}. Increasing the value of N does not always lead to a proportional gain in terms of convergence rate. For example, increasing $N = 8$ to $N = 16$ reduces the number of iterations by a factor of at least four for the first two matrices considered. On the other hand, increasing $N = 16$ to $N = 32$ reduces the number of iterations by a factor which is smaller than two. Moreover, small values of N, e.g., $N = 4$, might lead to very slow convergence. The third and fourth rows of plots show the associated eigenvalue errors (left column) and residual norms (right column) returned by Algorithm 3.1. For the choice $N = 4$, the range of matrices $\sum_{j=1}^{N} \omega_j B(\zeta_j)^{-1} F(\zeta_j) S(\zeta_j)^{-1}$ and $\sum_{j=1}^{N} \omega_j S(\zeta_j)^{-1}$ was not captured to high precision and this is reflected in the approximation of the sought eigenpairs. For the choices $N = 8$, 16 and $N = 32$, the range of both matrices was

[3] Results for matrix $\sum_{j=1}^{N} \omega_j B(\zeta_j)^{-1} F(\zeta_j) S(\zeta_j)^{-1}$ were essentially identical and thus not reported.

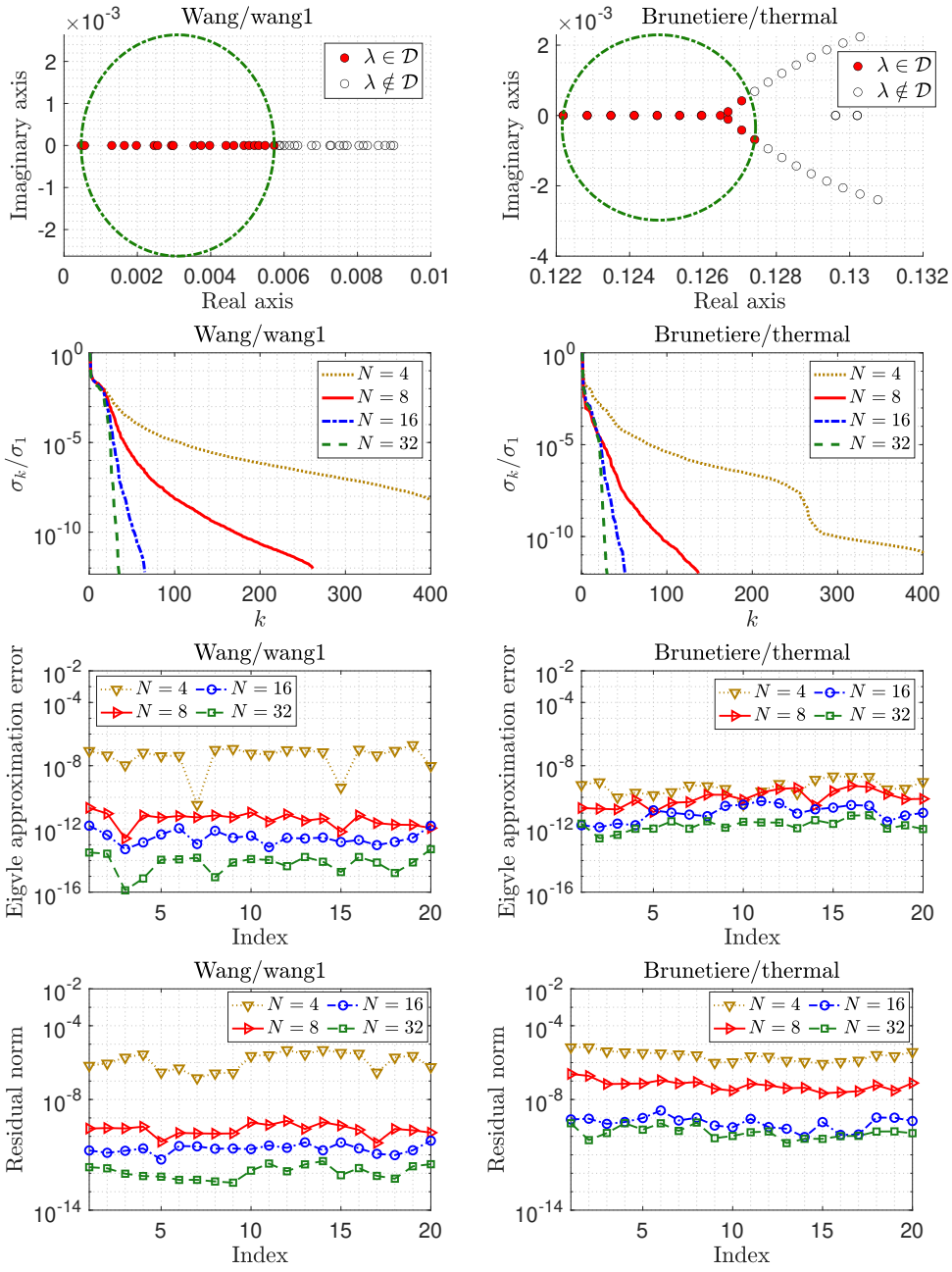


FIG. 5.1. *Application of Algorithm 3.1 to matrices* `wang1` *(left column) and* `thermal` *(right column). First row: the n_{ev} sought eigenvalues of (A, M) and n_{ev} immediate unwanted eigenvalues of smallest modulus. Second row: the ratio of smallest to largest singular value of matrix G as determined at each iteration of Algorithm 2.1 during its application to matrix $\sum_{j=1}^{N} \omega_j S(\zeta_j)^{-1}$. Third row: absolute eigenvalue error. Fourth row: residual norm. The indices of the x-axis are organized such that index 'i' corresponds to the sought eigenvalue with the ith smallest real part.*

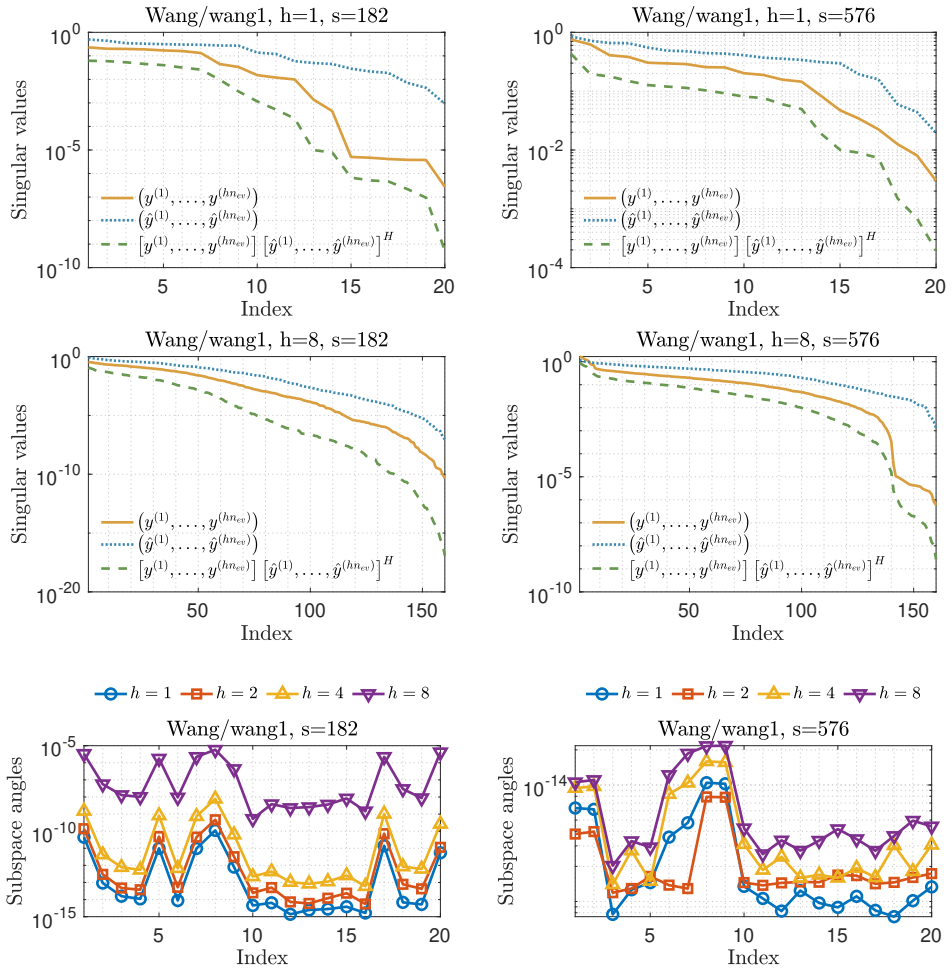


FIG. 5.2. *Singular values of matrices* $\left[y^{(1)},\dots,y^{(hn_{ev})}\right]$, $\left[\hat{y}^{(1)},\dots,\hat{y}^{(hn_{ev})}\right]$, *and their product* $\left[y^{(1)},\dots,y^{(hn_{ev})}\right]\left[\hat{y}^{(1)},\dots,\hat{y}^{(hn_{ev})}\right]^H$, *for different values of* $h\in\mathbb{N}$ *and size* s *of the matrix* $\sum_{j=1}^{N}\omega_j S(\zeta_j)^{-1}$. *In all figures we set* $n_{ev}=20$. *First row:* $h=1$ *and matrix* A *was partitioned so that the size of each Schur complement matrix in* $\sum_{j=1}^{N}\omega_j S(\zeta_j)^{-1}$ *is equal to* $s=182$ *(left) and* $s=576$ *(right). Second row: same as before but now we set* $h=8$. *Third row: angles between* $\texttt{range}\left(\left[y^{(1)},\dots,y^{(hn_{ev})}\right]\left[\hat{y}^{(1)},\dots,\hat{y}^{(hn_{ev})}\right]^H\right)$ *and vectors* $y^{(1)},\dots,y^{(n_{ev})}$.

captured up to the required tolerance and the sought eigenpairs were captured to higher accuracy.

The results in Figure 5.1 suggest that larger values of N can lead to higher accuracy in Algorithm 3.1 even though the range of matrices $\sum_{j=1}^{N}\omega_j B(\zeta_j)^{-1}F(\zeta_j)S(\zeta_j)^{-1}$ and $\sum_{j=1}^{N}\omega_j S(\zeta_j)^{-1}$ is captured highly accurately for all $N=8$, $N=16$, and $N=32$. Consider for example the matrix $\sum_{j=1}^{N}\omega_j S(\zeta_j)^{-1}=\left[\rho(\lambda_i)y^{(i)}\right]_{\rho(\lambda_i)\neq 0}\left[\hat{y}^{(i)}\right]_{\rho(\lambda_i)\neq 0}^{H}$. In practice, even though the condition in Proposition 3.1 holds, some of the trailing nonzero singular values of the matrix $\sum_{j=1}^{N}\omega_j S(\zeta_j)^{-1}$ might be (much) smaller than those of $\left[\rho(\lambda_i)y^{(i)}\right]_{\rho(\lambda_i)\neq 0}$, thus "suppressing" some directions of $\texttt{span}\left(\left[\rho(\lambda_i)y^{(i)}\right]_{\rho(\lambda_i)\neq 0}\right)$

in the range of the matrix $\sum_{j=1}^{N} \omega_j S(\zeta_j)^{-1}$. Since these directions generally have a nonzero projection to the subspace $\texttt{span}\left(y^{(1)}, \ldots, y^{(n_{ev})}\right)$, we expect that the accuracy to which we can capture the latter subspace might also be reduced. The same holds for $\texttt{span}\left(u^{(1)}, \ldots, u^{(n_{ev})}\right)$ and the range of matrix $\sum_{j=1}^{N} \omega_j B(\zeta_j)^{-1} F(\zeta_j) S(\zeta_j)^{-1}$ as well.

Figure 5.2 visualizes the above discussion for matrix `wang1`. In particular, we plot the singular values of the matrices $\left[y^{(1)}, \ldots, y^{(hn_{ev})}\right]$, $\left[\hat{y}^{(1)}, \ldots, \hat{y}^{(hn_{ev})}\right]$, as well as those of their matrix product, for $h = 1$ and $h = 8$, and $n_{ev} = 20$. Smaller values of h simulate larger values of N. The size of the Schur complement matrices was varied as $s = 182$ and $s = 576$. Observe that the singular values of the matrix $\left[y^{(1)}, \ldots, y^{(hn_{ev})}\right]\left[\hat{y}^{(1)}, \ldots, \hat{y}^{(hn_{ev})}\right]^H$ trail those of $\left[y^{(1)}, \ldots, y^{(hn_{ev})}\right]$. As a result, some directions of $\texttt{span}\left(y^{(1)}, \ldots, y^{(hn_{ev})}\right)$ are captured less accurately in $\texttt{range}\left(\left[y^{(1)}, \ldots, y^{(hn_{ev})}\right]\left[\hat{y}^{(1)}, \ldots, \hat{y}^{(hn_{ev})}\right]^H\right)$. The latter effect is sketched in the bottom row of plots where we plot the angle between the vectors $y^{(1)}, \ldots, y^{(n_{ev})}$ and $\texttt{range}\left(\left[y^{(1)}, \ldots, y^{(hn_{ev})}\right]\left[\hat{y}^{(1)}, \ldots, \hat{y}^{(hn_{ev})}\right]^H\right)$. Ideally, all angles should be equal to zero. However, we observe larger angles when h is larger (i.e., N gets smaller) and the vectors $y^{(i)}$ and $\hat{y}^{(i)}$ lie in a lower-dimensional subspace (i.e., s gets smaller).

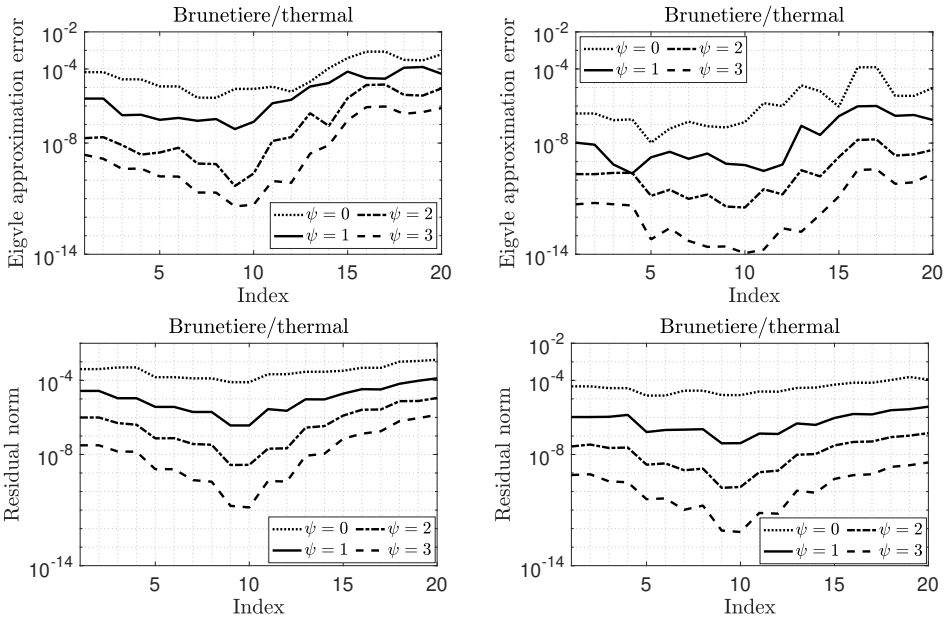


FIG. 5.3. *Application of Algorithm 3.2 to the* `thermal` *matrix. Top row: absolute eigenvalue errors for different values of* ψ *and* $\phi = 14$ *(left) and* $\phi = 160$ *(right). Bottom row: residual norms for the same values. The indices of the x-axis are organized such that index 'i' corresponds to the sought eigenvalue with the ith smallest real part.*

Figure 5.3 plots the eigenvalue approximation errors and corresponding residual norms in the approximation of the n_{ev} smallest modulus eigenvalues and associated eigenvectors of the `thermal` matrix by Algorithm 3.2. The number of computed matrix resolvent terms was set to $\psi = 0,\ 1,\ 2$, and $\psi = 3$, while the number of computed (deflated) eigenvectors was varied to $\phi = 14$ and $\phi = 160$. The choice $\phi = 14$ coincides with computing only those eigenvalues of the pencil (B, M_B) located

inside the disk $\mathcal{D}$. The number of poles was set equal to $N = 16$.[4] As expected, the accuracy in the approximation of the sought eigenpairs improves with larger values of ψ since the action of the matrix $\left(I - V_\phi \hat{V}_\phi^H M_B\right) B(\lambda_i)^{-1}$ is now approximated more accurately. Similarly, increasing $\phi = 14$ to $\phi = 160$ leads to enhanced accuracy. Note that the major improvements in accuracy come from increasing the value of ψ. This was a general trend observed for the rest of our test matrices as well. Moreover, as was also discussed in Section 3.3, the accuracy obtained by Algorithm 3.2 depends on the location of each eigenvalue $\lambda_i \in \mathcal{D}$ since the action of $\left(I - V_\phi \hat{V}_\phi^H M_B\right) B(\lambda_i)^{-1}$ is better approximated for those $\lambda_i \in \mathcal{D}$ located closer to the center of the disk $\mathcal{D}$. This is in contrast to Algorithm 3.1 which provides an almost uniform accuracy for all eigenpairs $(\lambda_i, x^{(i)})$ for which $\lambda_i \in \mathcal{D}$.

5.2. Comparisons against subspace iteration with rational filtering. We now consider the computation of the $n_{ev} = 40$ eigenvalues of smallest modulus (and their associated eigenvectors) of the matrix pencil `bfw782`, and the matrices `utm1700b`, `rdb32001`, `dw4096`, and `big`. Figure 5.4 plots the $2n_{ev}$ eigenvalues of smallest modulus of the last four matrices. Table 5.2 lists the maximum and minimum absolute

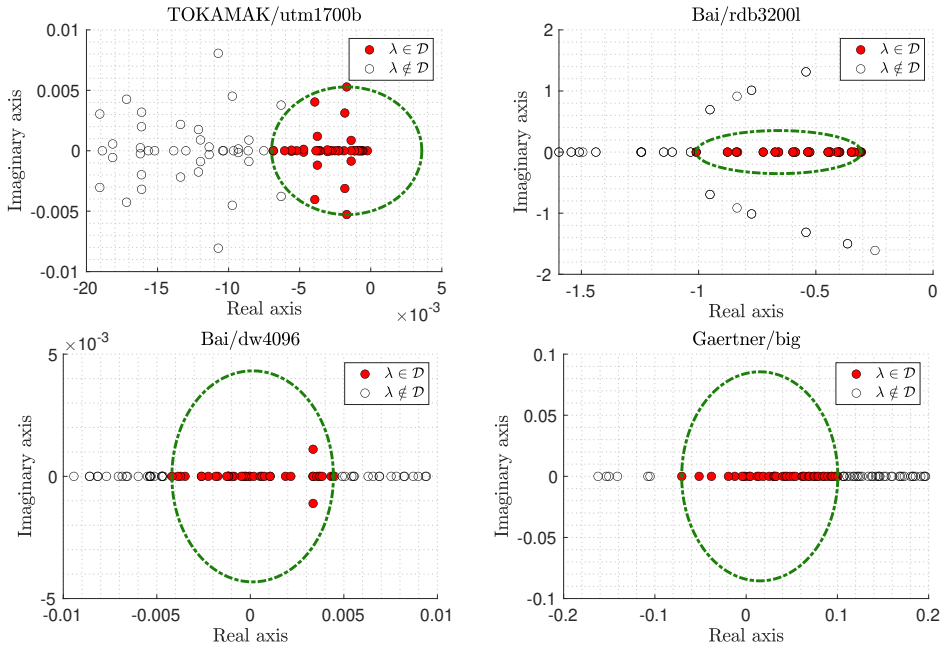


FIG. 5.4. *Plot of the $2n_{ev}$ eigenvalues of smallest modulus of some of the matrices listed in Table 5.1.*

eigenvalue errors and associated residual norms returned by Algorithm 3.1 as N varies. The number of iterations performed by Algorithm 3.1 is also listed. As expected, larger values of N lead to fewer iterations since the rational filter $\rho(\zeta)$ decays faster outside $\mathcal{D}$. In agreement with the results discussed in Figure 5.2, larger values of N also lead to higher accuracy. Additionally, Table 5.3 lists the maximum eigenvalue

[4]The results obtained for the choices $N = 8$ and $N = 32$ were essentially identical to those for the case $N = 16$, and thus are not reported.

TABLE 5.2

Minimum and maximum eigenvalue errors and residual norms returned by Algorithm 3.1 for the test matrices listed in Table 5.1. The total number of iterations performed by Algorithm 3.1 is also reported. A value 'F' indicates that the loop in Algorithm 3.1 did not terminate within four hundred iterations.

		Algorithm 3.1				
		$\min \lvert\lambda - \hat{\lambda}\rvert$	$\max \lvert\lambda - \hat{\lambda}\rvert$	$\min \hat{\rho}$	$\max \hat{\rho}$	It
bfw782	$\boldsymbol{N = 4}$	4.3×10^{-1}	2.1×10^{-0}	3.2×10^{-2}	2.4×10^{-1}	87
	$\boldsymbol{N = 8}$	1.5×10^{-2}	3.2×10^{-1}	2.9×10^{-4}	2.0×10^{-2}	87
	$\boldsymbol{N = 16}$	1.3×10^{-7}	4.5×10^{-4}	5.9×10^{-8}	1.0×10^{-6}	76
	$\boldsymbol{N = 32}$	1.1×10^{-9}	7.4×10^{-6}	6.6×10^{-10}	2.8×10^{-8}	55
utm1700b	$\boldsymbol{N = 4}$	1.4×10^{-8}	1.3×10^{-4}	2.0×10^{-7}	3.9×10^{-6}	235
	$\boldsymbol{N = 8}$	1.2×10^{-9}	3.2×10^{-6}	6.9×10^{-9}	2.5×10^{-7}	134
	$\boldsymbol{N = 16}$	8.0×10^{-12}	7.0×10^{-8}	5.9×10^{-10}	4.0×10^{-8}	72
	$\boldsymbol{N = 32}$	1.6×10^{-11}	3.1×10^{-8}	6.6×10^{-10}	6.0×10^{-8}	53
rdb32001	$\boldsymbol{N = 4}$	2.0×10^{-5}	1.7×10^{-3}	9.1×10^{-4}	1.5×10^{-2}	296
	$\boldsymbol{N = 8}$	7.2×10^{-9}	5.7×10^{-4}	2.3×10^{-7}	3.5×10^{-6}	161
	$\boldsymbol{N = 16}$	1.6×10^{-12}	3.9×10^{-9}	7.5×10^{-10}	4.5×10^{-8}	77
	$\boldsymbol{N = 32}$	1.3×10^{-15}	2.3×10^{-13}	2.1×10^{-11}	5.6×10^{-10}	52
dw4096	$\boldsymbol{N = 4}$	2.4×10^{-8}	4.0×10^{-5}	2.9×10^{-5}	1.4×10^{-2}	F
	$\boldsymbol{N = 8}$	7.2×10^{-12}	6.1×10^{-9}	1.9×10^{-8}	3.1×10^{-5}	329
	$\boldsymbol{N = 16}$	2.6×10^{-15}	1.8×10^{-10}	2.7×10^{-10}	5.5×10^{-6}	147
	$\boldsymbol{N = 32}$	9.8×10^{-15}	3.4×10^{-13}	2.6×10^{-12}	1.5×10^{-11}	75
big	$\boldsymbol{N = 4}$	4.8×10^{-6}	3.0×10^{-2}	3.8×10^{-4}	1.1×10^{-2}	377
	$\boldsymbol{N = 8}$	1.5×10^{-11}	3.6×10^{-6}	2.1×10^{-6}	1.2×10^{-4}	226
	$\boldsymbol{N = 16}$	6.2×10^{-13}	1.7×10^{-9}	1.0×10^{-8}	2.8×10^{-6}	108
	$\boldsymbol{N = 32}$	1.3×10^{-14}	6.3×10^{-11}	1.8×10^{-9}	1.3×10^{-6}	68

TABLE 5.3

Maximum eigenvalue error and residual norm returned by Algorithm 3.2 for some of the test matrices listed in Table 5.1. These results were obtained by setting $N = 16$ and varying the value of computed resolvent terms ψ.

		bfw782	utm1700b	rdb32001	wd4096	big
$\lvert\lambda - \hat{\lambda}\rvert$	$\psi = 0$	9.0×10^{-2}	5.1×10^{-3}	1.5×10^{-1}	2.8×10^{-1}	7.2×10^{-2}
	$\psi = 1$	5.9×10^{-3}	1.0×10^{-5}	9.1×10^{-3}	1.2×10^{-1}	2.0×10^{-3}
	$\psi = 2$	1.5×10^{-4}	2.8×10^{-7}	2.4×10^{-4}	2.9×10^{-3}	1.4×10^{-5}
	$\psi = 3$	8.4×10^{-6}	9.0×10^{-8}	7.5×10^{-6}	9.7×10^{-5}	8.9×10^{-7}
$\hat{\rho}$	$\psi = 0$	4.3×10^{-2}	1.2×10^{-3}	1.0×10^{-0}	4.6×10^{-1}	2.6×10^{-1}
	$\psi = 1$	4.1×10^{-3}	4.7×10^{-4}	6.0×10^{-2}	4.3×10^{-1}	1.1×10^{-1}
	$\psi = 2$	2.4×10^{-3}	2.8×10^{-6}	1.6×10^{-3}	3.0×10^{-2}	4.0×10^{-3}
	$\psi = 3$	3.8×10^{-5}	6.6×10^{-7}	7.3×10^{-5}	1.2×10^{-4}	6.2×10^{-5}

error and residual norm returned by Algorithm 3.2 when the value of ϕ is set equal to the number of eigenvalues located inside the disk $\mathcal{D}$ and ψ varies.

Table 5.4 lists the average number (with respect to N) of linear systems of the form $B(\zeta)x_d = b_d$ and $S(\zeta)x_s = b_s$ solved by Algorithm 3.1, Algorithm 3.2, and RSI.[5] The loop in RSI terminates when the maximum residual in the approximation of the n_{ev} sought eigenpairs becomes smaller than or equal to the maximum residual norm achieved by Algorithm 3.1. These residual norms listed in Table 5.2. Notice that the number of linear systems $B(\zeta)x_d = b_d$ solved by Algorithm 3.2 is independent of N, and thus the average value becomes smaller as N increases. On the other hand, the

[5]These numbers do not include the cost to compute a good approximation of n_{ev} in RSI.

TABLE 5.4

Average number of linear systems of the form $B(\zeta_j)x_d = b_d$ and $S(\zeta_j)x_s = b_s$ solved per pole ζ_j, $j = 1, \ldots, N$. For Algorithm 3.2 we consider only the case $\psi = 3$. For RSI we set $m := m_1 = 1.1n_{ev}$, $m := m_2 = 1.5n_{ev}$, and $m := m_3 = 2n_{ev}$.

		bfw782		utm1700b		rdb32001		dw4096		big	
		$B(\zeta_j)$	$S(\zeta_j)$	$B(\zeta_j)$	$S(\zeta_j)$	$B(\zeta_j)$	$S(\zeta_j)$	$B(\zeta_j)$	$S(\zeta_j)$	$B(\zeta_j)$	$S(\zeta_j)$
$N = 8$	Alg. 3.1	87	87	134	134	161	161	329	329	226	226
	Alg. 3.2	112	87	76	134	71	161	176	329	127	226
	RSI(m_1)	264	132	616	308	616	308	616	308	704	352
	RSI(m_2)	120	60	240	120	240	120	480	240	480	240
	RSI(m_3)	160	80	320	160	320	160	320	160	320	160
$N = 16$	Alg. 3.1	76	76	72	72	77	77	147	147	108	108
	Alg. 3.2	50	76	23	72	26	77	42	147	33	108
	RSI(m_1)	440	220	352	176	352	176	616	308	352	176
	RSI(m_2)	360	180	120	60	240	120	240	120	240	120
	RSI(m_3)	320	160	160	80	160	80	160	80	160	80
$N = 32$	Alg. 3.1	55	55	53	53	52	52	75	75	68	68
	Alg. 3.2	20	55	9	53	10	52	12	75	12	68
	RSI(m_1)	264	132	176	88	264	132	616	308	352	176
	RSI(m_2)	240	120	120	60	120	60	240	120	240	120
	RSI(m_3)	160	80	160	80	160	80	160	80	160	80

accuracy achieved by Algorithm 3.2 is also lower. For RSI we consider three different dimensions of the starting subspace, set as $m = 1.1n_{ev}$, $m = 1.5n_{ev}$, and $m = 2n_{ev}$. The rate of convergence of RSI is dictated by the ratio $\rho(\lambda_m)/\rho(\lambda_{n_{ev}})$, and thus the convergence rate improves as m increases. We observe two main trends. First, for small values of m, e.g., $m = 1.1n_{ev}$, RSI is considerably more expensive than both Algorithm 3.1 and Algorithm 3.2. This is because for small values of m the ratio $\rho(\lambda_m)/\rho(\lambda_{n_{ev}})$ might not be close to zero, thus leading to slower convergence in RSI. Second, as m increases, the average number of linear systems solved by RSI generally decreases, however the large value of m might lead to a few unnecessary solves, i.e., for $N = 32$ choosing $m = 2n_{ev}$ sometimes provides the same accuracy with $m = 1.5n_{ev}$.

In the previous experiment we assumed that the tolerance threshold in RSI was dictated by the maximum residual norm achieved by Algorithm 3.1. Since Algorithm 3.1 is a one-shot method, its maximum attainable accuracy is lower than that of RSI since the latter is an iterative approach. Nonetheless, the accuracy of the approximate eigenpairs returned by Algorithm 3.1 can improve by using the corresponding eigenvectors as an initial subspace in a separate run of RSI. While this enhancement comes at an increased computational cost, it is generally still cheaper than applying subspace iteration with a random starting subspace since it avoids the overhead associated with the computation of a good approximation of n_{ev} through the techniques described in [43] and [10]. Figure 5.5 plots the maximum residual norm achieved at the end of each iteration when RSI is applied to matrices `utm1700b` and `dw4096` with $m = 1.5n_{ev}$ and $n_{ev} = 40$. The initial subspace in RSI was set using "a)" m random vectors as in the results reported above (dashed lines), and "b)" the n_{ev} approximate eigenvectors returned by Algorithm 3.1 augmented by $m - n_{ev}$ random vectors (solid lines). In the latter case, the value at the origin denotes the accuracy achieved by Algorithm 3.1 before post-processing by RSI. The combination of RSI with Algorithm 3.1 leads to faster convergence, since the initial subspace is of much higher quality. What approach will be faster overall depends on the particular problem. For example, choosing $N = 8$ and a random initial subspace leads to very slow convergence in the case of matrix `dw4096`.

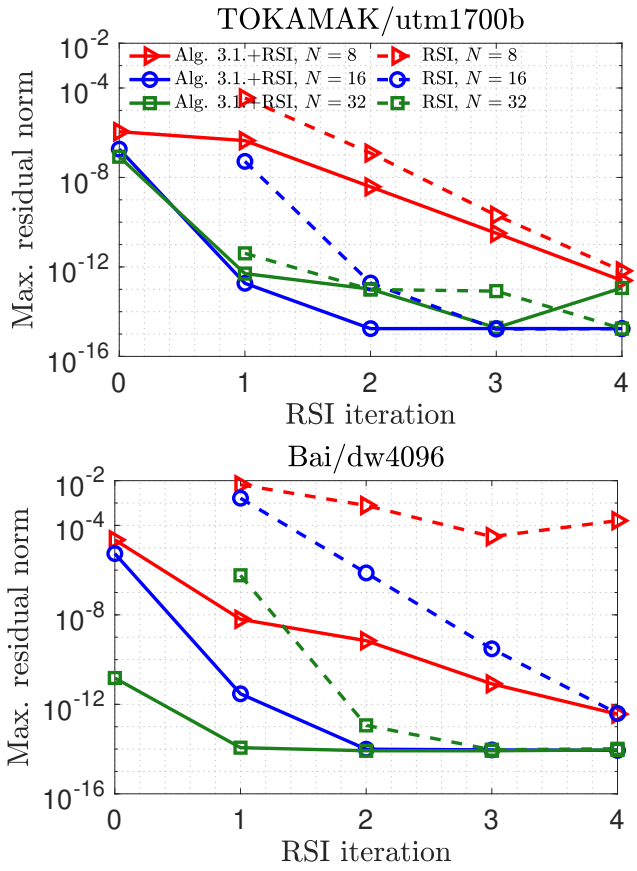


FIG. 5.5. *Maximum residual norm achieved at each iteration of RSI as N varies and $m = 1.5n_{ev}$, $n_{ev} = 40$. The values listed at the origin denote the maximum residual norm achieved by Algorithm 3.1 before post-processing by RSI.*

6. Conclusion. This paper presents a class of algorithms for the computation of all eigenvalues (and associated eigenvectors) of non-Hermitian matrix pencils located inside a disk. The proposed algorithms approximate the sought eigenpairs by harmonic Rayleigh-Ritz projections on subspaces built by computing range spaces of rational matrix functions through randomized range finders. These rational matrix functions are designed so that directions associated with non-sought eigenvalues are dampened to (approximately) zero. Moreover, the proposed algorithms do not require any a priori estimation of the number of eigenvalues located inside the disk. The competitiveness of the proposed algorithms was demonstrated through numerical experiments performed on a few test problems.

Several research directions are left as future work. One such direction is the extension of the algorithms presented in this paper with non-Hermitian Krylov subspace iterative solvers and hierarchical preconditioners such as those discussed in [11]. Moreover, although this paper focused on algorithms, rational filtering eigenvalue solvers owe a large portion of their appeal in the ample parallelism they offer, and a distributed memory implementation of the proposed technique would be also of interest. Another interesting research direction is the extension of the algorithms presented in

this paper for the computation of a partial Schur decomposition, or the simultaneous computation of both left and right eigenvectors.

Acknowledgments. The authors are grateful to the two anonymous referees for their helpful comments and suggestions which improved the readability and overall quality of the present manuscript. The contribution of the first author was partially supported by the Herman H. Goldstine Postdoctoral Fellowship program of the International Business Machines Corporation. The work of Yuanzhe Xi was supported by NSF grant OAC 2003720. Finally, the authors are indebted to Anthony P. Austin for his comments on an earlier draft of the present manuscript.

REFERENCES

[1] H. M. Aktulga, L. Lin, C. Haine, E. G. Ng, and C. Yang, *Parallel eigenvalue calculation based on multiple shift–invert Lanczos and contour integral based spectral projection method*, Parallel Computing, 40 (2014), pp. 195–212.

[2] J. Asakura, T. Sakurai, H. Tadano, T. Ikegami, and K. Kimura, *A numerical method for nonlinear eigenvalue problems using contour integrals*, JSIAM Letters, 1 (2009), pp. 52–55.

[3] A. Austin, P. Kravanja, and L. Trefethen, *Numerical algorithms based on analytic function values at roots of unity*, SIAM Journal on Numerical Analysis, 52 (2014), pp. 1795–1821.

[4] A. Austin and L. Trefethen, *Computing eigenvalues of real symmetric matrices with rational filters in real arithmetic*, SIAM Journal on Scientific Computing, 37 (2015), pp. A1365–A1387.

[5] M. V. Barel, *Designing rational filter functions for solving eigenvalue problems by contour integration*, Linear Algebra and its Applications, 502 (2016), pp. 346 – 365. Structured Matrices: Theory and Applications.

[6] W.-J. Beyn, *An integral method for solving nonlinear eigenvalue problems*, Linear Algebra and its Applications, 436 (2012), pp. 3839–3863.

[7] R. F. Boisvert, R. Pozo, K. Remington, R. F. Barrett, and J. J. Dongarra, *Matrix market: a web resource for test matrix collections*, in Quality of Numerical Software, Springer, 1997, pp. 125–137.

[8] A. Bose, V. Kalantzis, E.-M. Kontopoulou, M. Elkady, P. Paschou, and P. Drineas, *Terapca: a fast and scalable software package to study genetic variation in tera-scale genotypes*, Bioinformatics, 35 (2019), pp. 3679–3683.

[9] T. A. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Trans. Math. Softw., 38 (2011), pp. 1:1–1:25.

[10] E. Di Napoli, E. Polizzi, and Y. Saad, *Efficient estimation of eigenvalue counts in an interval*, Numerical Linear Algebra with Applications, 23 (2016), pp. 674–692.

[11] G. Dillon, V. Kalantzis, Y. Xi, and Y. Saad, *A hierarchical low rank Schur complement preconditioner for indefinite linear systems*, SIAM Journal on Scientific Computing, 40 (2018), pp. A2234–A2252.

[12] P. Drineas and M. W. Mahoney, *Randnla: randomized numerical linear algebra*, Communications of the ACM, 59 (2016), pp. 80–90.

[13] L. Giraud and J. Langou, *When modified gram–schmidt generates a well-conditioned set of vectors*, IMA Journal of Numerical Analysis, 22 (2002), pp. 521–528.

[14] G. Golub and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis, 2 (1965), pp. 205–224.

[15] S. Güttel, E. Polizzi, P. T. P. Tang, and G. Viaud, *Zolotarev quadrature rules and load balancing for the FEAST eigensolver*, SIAM Journal on Scientific Computing, 37 (2015), pp. A2100–A2122.

[16] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Review, 53 (2011), pp. 217–288.

[17] V. Hernandez, J. E. Roman, and A. Tomas, *Restarted lanczos bidiagonalization for the svd in slepc*.

[18] S. Iwase, Y. Futamura, A. Imakura, T. Sakurai, and T. Ono, *Efficient and scalable calculation of complex band structure using Sakurai-Sugiura method*, in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2017, pp. 1–12.

[19] Z. Jia, *The convergence of harmonic Ritz values, harmonic Ritz vectors and refined harmonic Ritz vectors*, Mathematics of Computation, 74 (2005), pp. 1441–1456.
[20] V. Kalantzis, *Domain decomposition algorithms for the solution of sparse symmetric generalized eigenvalue problems*, PhD Thesis, University of Minnesota, (2018).
[21] V. Kalantzis, J. Kestyn, E. Polizzi, and Y. Saad, *Domain decomposition approaches for accelerating contour integration eigenvalue solvers for symmetric eigenvalue problems*, Numerical Linear Algebra with Applications, 25 (2018), p. e2154.
[22] V. Kalantzis, G. Kollias, S. Ubaru, A. N. Nikolakopoulos, L. Horesh, and K. L. Clarkson, *Projection techniques to update the truncated svd of evolving matrices*, arXiv preprint arXiv:2010.06392, (2020).
[23] V. Kalantzis, Y. Xi, and Y. Saad, *Beyond automated multilevel substructuring: Domain decomposition with rational filtering*, SIAM Journal on Scientific Computing, 40 (2018), pp. C477–C502.
[24] J. Kestyn, V. Kalantzis, E. Polizzi, and Y. Saad, *PFEAST: a high performance sparse eigenvalue solver using distributed-memory linear solvers*, in High Performance Computing, Networking, Storage and Analysis, SC16: International Conference for, IEEE, 2016, pp. 178–189.
[25] K. Kollnig, P. Bientinesi, and E. Di Napoli, *Rational spectral filters with optimal convergence rate*, arXiv:2001.04184, (2020).
[26] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, United States Governm. Press Office Los Angeles, CA, 1950.
[27] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, vol. 6, SIAM, 1998.
[28] R. Li, Y. Xi, L. Erlandson, and Y. Saad, *The eigenvalues slicing library (evsl): Algorithms, implementation, and software*, SIAM Journal on Scientific Computing, 41 (2019), pp. C393–C415.
[29] R.-C. Li, *Rayleigh quotient based optimization methods for eigenvalue problems*, in Matrix Functions and Matrix Equations, World Scientific, 2015, pp. 76–108.
[30] E. Liberty, F. Woolfe, P.-G. Martinsson, V. Rokhlin, and M. Tygert, *Randomized algorithms for the low-rank approximation of matrices*, Proceedings of the National Academy of Sciences, 104 (2007), pp. 20167–20172.
[31] X. Liu, Y. Xi, Y. Saad, and M. V. de Hoop, *Solving the three-dimensional high-frequency helmholtz equation using contour integration and polynomial preconditioning*, SIAM Journal on Matrix Analysis and Applications, 41 (2020), pp. 58–82.
[32] M. W. Mahoney and P. Drineas, *Cur matrix decompositions for improved data analysis*, Proceedings of the National Academy of Sciences, 106 (2009), pp. 697–702.
[33] P.-G. Martinsson, *Randomized methods for matrix computations*, The Mathematics of Data, 25, pp. 187–231.
[34] P.-G. Martinsson and J. Tropp, *Randomized numerical linear algebra: Foundations & algorithms*, arXiv preprint arXiv:2002.01387, (2020).
[35] R. B. Morgan, *Computing interior eigenvalues of large matrices*, Linear Algebra and its Applications, 154 (1991), pp. 289–309.
[36] R. B. Morgan and M. Zeng, *Harmonic projection methods for large non-symmetric eigenvalue problems*, Numerical Linear Algebra with Applications, 5 (1998), pp. 33–55.
[37] B. N. Parlett, *The symmetric eigenvalue problem*, vol. 20, SIAM, 1998.
[38] P. T. Peter Tang and E. Polizzi, *FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection*, SIAM Journal on Matrix Analysis and Applications, 35 (2014), pp. 354–390.
[39] E. Polizzi, *Density-matrix-based algorithm for solving eigenvalue problems*, Phys. Rev. B, 79 (2009), pp. 115–112.
[40] V. Rokhlin, A. Szlam, and M. Tygert, *A randomized algorithm for principal component analysis*, SIAM Journal on Matrix Analysis and Applications, 31 (2010), pp. 1100–1124.
[41] A. Ruhe, *Rational Krylov: A practical algorithm for large sparse nonsymmetric matrix pencils*, SIAM Journal on Scientific Computing, 19 (1998), pp. 1535–1551.
[42] Y. Saad, *ILUM: a multi-elimination ILU preconditioner for general sparse matrices*, SIAM Journal on Scientific Computing, 17 (1996), pp. 830–847.
[43] T. Sakurai, Y. Futamura, and H. Tadano, *Efficient parameter estimation and implementation of a contour integral-based eigensolver*, Journal of Algorithms & Computational Technology, 7 (2013), pp. 249–269.
[44] 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.

[45] T. Sakurai, H. Tadano, et al., *CIRR: a Rayleigh-Ritz type method with contour integral for generalized eigenvalue problems*, Hokkaido Mathematical Journal, 36 (2007), pp. 745–757.
[46] L. N. Trefethen and J. Weideman, *The exponentially convergent trapezoidal rule*, SIAM Review, 56 (2014), pp. 385–458.
[47] J. Winkelmann and E. Di Napoli, *Non-linear least-squares optimization of rational filters for the solution of interior hermitian eigenvalue problems*, Frontiers in Applied Mathematics and Statistics, 5 (2019), p. 5.
[48] Y. Xi and Y. Saad, *Computing partial spectra with least-squares rational filters*, SIAM Journal on Scientific Computing, 38 (2016), pp. A3020–A3045.
[49] ———, *A rational function preconditioner for indefinite sparse linear systems*, SIAM Journal on Scientific Computing, 39 (2017), pp. A1145–A1167.
[50] I. Yamazaki, H. Tadano, T. Sakurai, and T. Ikegami, *Performance comparison of parallel eigensolvers based on a contour integral method and a Lanczos method*, Parallel Computing, 39 (2013), pp. 280 – 290.
[51] X. Ye, J. Xia, R. H. Chan, S. Cauley, and V. Balakrishnan, *A fast contour-integral eigensolver for non-Hermitian matrices*, SIAM Journal on Matrix Analysis and Applications, 38 (2017), pp. 1268–1297.
[52] G. Yin, *A contour-integral based method with Schur–Rayleigh–Ritz procedure for generalized eigenvalue problems*, Journal of Scientific Computing, 81 (2019), pp. 252–270.
[53] ———, *A harmonic feast algorithm for non-hermitian generalized eigenvalue problems*, Linear Algebra and its Applications, 578 (2019), pp. 75–94.
[54] 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.
[55] H. Zha and H. D. Simon, *On updating problems in latent semantic indexing*, SIAM Journal on Scientific Computing, 21 (1999), pp. 782–791.