Beyond Automated Multilevel Substructuring: Domain Decomposition with Rational Filtering

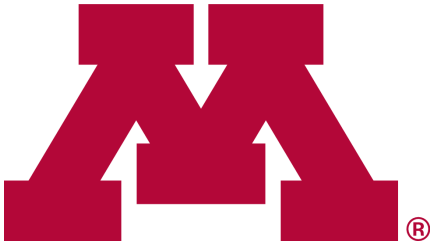
Vassilis Kalantzis, Yuanzhe Xi, and Y. Saad

June 2018

EPrint ID: 2018.2

Department of Computer Science and Engineering
University of Minnesota, Twin Cities

Preprints available from: `http://www-users.cs.umn.edu/kalantzi`



UNIVERSITY OF MINNESOTA

Supercomputing Institute

SIAM J. SCI. COMPUT.
Vol. 0, No. 0, pp. 000–000

BEYOND AUTOMATED MULTILEVEL SUBSTRUCTURING: DOMAIN DECOMPOSITION WITH RATIONAL FILTERING*

VASSILIS KALANTZIS†, YUANZHE XI‡, AND YOUSEF SAAD†

Abstract. This paper proposes a rational filtering domain decomposition technique for the solution of large and sparse symmetric generalized eigenvalue problems. The proposed technique is purely algebraic and decomposes the eigenvalue problem associated with each subdomain into two disjoint subproblems. The first subproblem is associated with the interface variables and accounts for the interaction among neighboring subdomains. To compute the solution of the original eigenvalue problem at the interface variables we leverage ideas from contour integral eigenvalue solvers. The second subproblem is associated with the interior variables in each subdomain and can be solved in parallel among the different subdomains using real arithmetic only. Compared to rational filtering projection methods applied to the original matrix pencil, the proposed technique integrates only a part of the matrix resolvent while it applies any orthogonalization necessary to vectors whose length is equal to the number of interface variables. In addition, no estimation of the number of eigenvalues located inside the interval of interest is needed. Numerical experiments performed in distributed memory architectures illustrate the competitiveness of the proposed technique against rational filtering Krylov approaches.

Key words. domain decomposition, Schur complement, symmetric generalized eigenvalue problem, rational filtering, parallel computing

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

DOI. 10.1137/17M1154527

1. Introduction.

The typical approach to solve large and sparse symmetric eigenvalue problems of the form $Ax = \lambda Mx$ is via a Rayleigh–Ritz (projection) process on a low-dimensional subspace that spans an invariant subspace associated with the $nev \geq 1$ eigenvalues of interest, e.g., those located inside the real interval $[\alpha, \beta]$.

One of the main bottlenecks of Krylov projection methods in large-scale eigenvalue computations is the cost to maintain the orthonormality of the basis of the Krylov subspace, especially when nev runs in the order of hundreds or thousands. To reduce the orthonormalization and memory costs, it is typical to enhance the convergence rate of the Krylov projection method of choice by a filtering acceleration technique so that eigenvalues located outside the interval of interest are damped to (approximately) zero. For generalized eigenvalue problems, a standard choice is to exploit rational filtering techniques, i.e., to transform the original matrix pencil into a complex, rational matrix-valued function [7, 8, 22, 23, 30, 32, 37, 40, 42]. While rational filtering approaches reduce orthonormalization costs, their main bottleneck is the application of the transformed pencil, i.e., the solution of the associated linear systems.

An alternative to reduce the computational costs (especially that of orthogonalization) in large-scale eigenvalue computations is to consider domain decomposition-type approaches. (We refer to [34, 38] for an in-depth discussion of domain decomposition.)

*Submitted to the journal's Software and High-Performance Computing section October 30, 2017; accepted for publication (in revised form) April 19, 2018; published electronically DATE.
http://www.siam.org/journals/sisc/x-x/M115452.html
Funding: This work was supported by NSF (award CCF-1505970). The work of the first author was partially supported by a Gerondelis Foundation Fellowship.
†Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455 (kalan019@umn.edu, saad@umn.edu).
‡Department of Mathematics and Computer Science, Emory University, Atlanta, GA 30322 (yxi26@emory.edu).

Domain decomposition decouples the original eigenvalue problem into two separate subproblems, one defined locally in the interior of each subdomain, and one defined on the interface region connecting neighboring subdomains. Once the original eigenvalue problem is solved at the interface region, the solution associated with the interior of each subdomain is computed independently of the rest of the subdomains [4, 9, 10, 20, 26, 27, 29]. When the number of variables associated with the interface region is much smaller than the number of global variables, domain decomposition approaches can provide approximations to thousands of eigenpairs while avoiding excessive orthogonalization costs. One prominent such example is the automated multilevel substructuring (AMLS) method [9, 10, 16], originally developed by the structural engineering community for the frequency response analysis of finite element automobile bodies. AMLS has been shown to be considerably faster than the NASTRAN industrial package [24] in applications where $nev \gg 1$. However, the accuracy provided by AMLS is good typically only for eigenvalues located close to a user-given real-valued shift [9].

In this paper we describe the rational filtering domain decomposition eigenvalue solver (RF-DDES), an approach which combines domain decomposition with rational filtering. Below, we list the main characteristics of RF-DDES:

(1) *Reduced complex arithmetic and orthgonalization costs.* Standard rational filtering techniques apply the rational filter to the entire matrix pencil, i.e., they require the solution of linear systems with complex coefficient matrices of the form $A-\zeta M$ for different complex values of ζ. In contrast, RF-DDES applies the rational filter only to that part of $A-\zeta M$ that is associated with the interface variables. As we show later, this approach has several advantages: (a) if a Krylov projection method is applied, orthonormalization needs to be applied to vectors whose length is equal to the number of interface variables only, (b) while RF-DDES also requires the solution of complex linear systems, the associated computational cost is lower than that of standard rational filtering approaches, and (c) focusing on the interface variables only makes it possible to achieve convergence of the Krylov projection method in even fewer than nev iterations. In contrast, any Krylov projection method applied to a rational transformation of the original matrix pencil must perform at least nev iterations.

(2) *Controllable approximation accuracy.* Domain decomposition approaches like AMLS might fail to provide high accuracy for all eigenvalues located inside $[\alpha, \beta]$. This is because AMLS solves only for an approximation of the eigenvalue problem associated with the interface variables of the domain. In contrast, RF-DDES can compute the part of the solution associated with the interface variables highly accurately. As a result, if not satisfied with the accuracy obtained by RF-DDES, one can simply refine the part of the solution that is associated with the interior variables.

(3) *Multilevel parallelism.* The solution of the original eigenvalue problem associated with the interior variables of each subdomain can be approximated independently in each subdomain exploiting only real arithmetic. Moreover, being a combination of domain decomposition and rational filtering techniques, RF-DDES can take advantage of different levels of parallelism, making itself appealing for execution in high-end computers. We report results of experiments performed in distributed memory environments and verify the effectiveness of RF-DDES.

Throughout this paper we are interested in computing the $nev \geq 1$ eigenpairs $(\lambda_i, x^{(i)})$ of $Ax^{(i)} = \lambda_i M x^{(i)},\ i = 1, \ldots, n$, for which $\lambda_i \in [\alpha, \beta],\ \alpha \in \mathbb{R},\ \beta \in \mathbb{R}$. The $n \times n$ matrices A and M are assumed large, sparse, and symmetric while M is also positive-definite (SPD). For brevity, we will refer to the linear SPD matrix pencil $A - \lambda M$ simply as (A, M).

The structure of this paper is as follows: Section 2 describes the general working of rational filtering and domain decomposition eigenvalue solvers. Section 3 describes computational approaches for the solution of the eigenvalue problem associated with the interface variables. Section 4 describes the solution of the eigenvalue problem associated with the interior variables in each subdomain. Section 5 combines all previous discussion into the form of an algorithm. Section 6 presents experiments performed on model and general matrix pencils. Finally, section 7 contains our concluding remarks.

2. Rational filtering and domain decomposition eigensolvers. In this section we review the concept of rational filtering for the solution of real symmetric generalized eigenvalue problems. In addition, we present a prototype Krylov-based rational filtering approach to serve as a baseline algorithm, while also discussing the solution of symmetric generalized eigenvalue problems from a domain decomposition viewpoint.

Throughout the rest of this paper we will assume that the eigenvalues of (A, M) are ordered so that eigenvalues $\lambda_1, \ldots, \lambda_{nev}$ are located inside $[\alpha, \beta]$ while eigenvalues $\lambda_{nev+1}, \ldots, \lambda_n$ are located outside $[\alpha, \beta]$.

2.1. Construction of the rational filter. The classic approach to construct a rational filter function $\rho(\zeta)$ is to exploit the Cauchy integral representation of the indicator function $I_{[\alpha,\beta]}$, where $I_{[\alpha,\beta]}(\zeta) = 1$ iff $\zeta \in [\alpha, \beta]$, and $I_{[\alpha,\beta]}(\zeta) = 0$ iff $\zeta \notin [\alpha, \beta]$; see also [5, 18, 40, 39] for other filter functions not based on Cauchy's formula.

In particular, let $\Gamma_{[\alpha,\beta]}$ be a smooth, closed contour that encloses only those nev eigenvalues of (A, M) which are located inside $[\alpha, \beta]$, e.g., a circle centered at $(\alpha+\beta)/2$ with radius $(\beta - \alpha)/2$. We then have

$$I_{[\alpha,\beta]}(\zeta) = \frac{-1}{2\pi i} \int_{\Gamma_{[\alpha,\beta]}} \frac{1}{\zeta - \nu} d\nu, \tag{2.1}$$

where the integration is performed counterclockwise. The filter function $\rho(\zeta)$ can be obtained by applying a quadrature rule to discretize the right-hand side in (2.1):

$$\rho(\zeta) = \sum_{\ell=1}^{2N_c} \frac{\omega_\ell}{\zeta - \zeta_\ell}, \tag{2.2}$$

where $\{\zeta_\ell, \omega_\ell\}_{1 \le \ell \le 2N_c}$ are the poles and weights of the quadrature rule. If the $2N_c$ poles in (2.2) come in conjugate pairs, and the first N_c poles lie on the upper half plane, (2.2) can be simplified into

$$\rho(\zeta) = 2\Re e \left\{ \sum_{\ell=1}^{N_c} \frac{\omega_\ell}{\zeta - \zeta_\ell} \right\} \text{ when } \zeta \in \mathbb{R}. \tag{2.3}$$

Figure 2.1 plots the modulus of the rational function $\rho(\zeta)$ (scaled such that $\rho(\alpha) \equiv \rho(\beta) \equiv 1/2$) in the interval $\zeta \in [-2, 2]$, where $\{\zeta_\ell, \omega_\ell\}_{1 \le \ell \le 2N_c}$ are obtained by numerically approximating $I_{[-1,1]}(\zeta)$ by the Gauss–Legendre rule (left) and midpoint rule (right). Notice that as N_c increases, $\rho(\zeta)$ becomes a more accurate approximation of $I_{[-1,1]}(\zeta)$ [40]. Throughout the rest of this paper, we will only consider the midpoint rule [2].

2.2. Rational filtered Arnoldi procedure. Now, consider the rational matrix function $\rho(M^{-1}A)$ with $\rho(.)$ defined as in (2.2):

$$\rho(M^{-1}A) = 2\Re e \left\{ \sum_{\ell=1}^{N_c} \omega_\ell (A - \zeta_\ell M)^{-1} M \right\}. \tag{2.4}$$

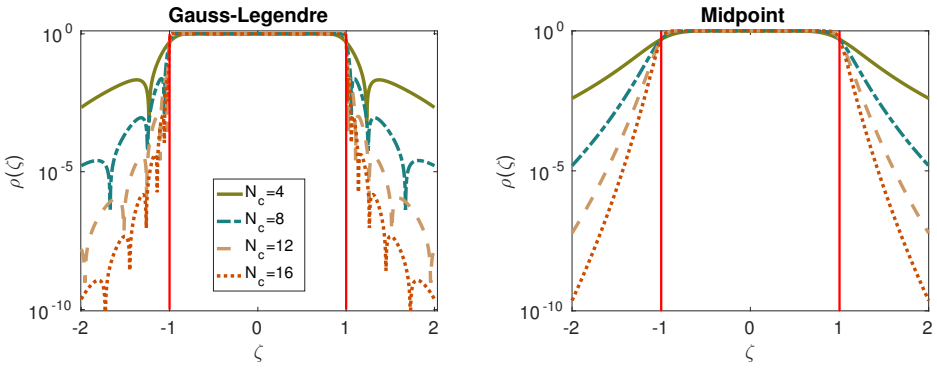


FIG. 2.1. *The modulus of the rational filter function $\rho(\zeta)$ when $\zeta \in [-2, 2]$. Left: Gauss–Legendre rule. Right: midpoint rule.*

The eigenvectors of $\rho(M^{-1}A)$ are identical to those of (A, M), while the corresponding eigenvalues are transformed to $\{\rho(\lambda_j)\}_{j=1,\ldots,n}$. Since $\rho(\lambda_1), \ldots, \rho(\lambda_{nev})$ are all larger than $\rho(\lambda_{nev+1}), \ldots, \rho(\lambda_n)$, the eigenvalues of (A, M) located inside $[\alpha, \beta]$ become the dominant ones in $\rho(M^{-1}A)$. Applying a projection method to $\rho(M^{-1}A)$ can then lead to fast convergence toward an invariant subspace associated with the eigenvalues of (A, M) located inside $[\alpha, \beta]$.

One popular choice as the projection method in rational filtering approaches is that of subspace iteration, e.g., as in the FEAST package [22, 23, 30]. In this paper, we exploit Krylov subspace methods, instead, in order to try and reduce the total number of linear system solutions performed.

ALGORITHM 2.1. *RF-KRYLOV*

0. *Start with* $q^{(1)} \in \mathbb{R}^n$ *s.t.* $\|q^{(1)}\|_2 = 1$
1. *For* $\mu = 1, 2, \ldots$
2. *Compute* $w = 2\Re e \left\{ \sum_{\ell=1}^{N_c} \omega_\ell (A - \zeta_\ell M)^{-1} M q^{(\mu)} \right\}$
3. *For* $\kappa = 1, \ldots, \mu$
4. $h_{\kappa,\mu} = w^T q^{(\kappa)}$
5. $w = w - h_{\kappa,\mu} q^{(\kappa)}$
6. *End*
7. $h_{\mu+1,\mu} = \|w\|_2$
8. *If* $h_{\mu+1,\mu} = 0$
9. *generate a unit-norm* $q^{(\mu+1)}$ *orthogonal to* $q^{(1)}, \ldots, q^{(\mu)}$
10. *Else*
11. $q^{(\mu+1)} = w / h_{\mu+1,\mu}$
12. *EndIf*
13. *If the sum of eigenvalues of* H_μ *no less than* $1/2$ *is unchanged during the last few iterations; BREAK; EndIf*
14. *End*
15. *Compute the eigenvectors of* H_μ *and form the Ritz vectors of* (A, M)
16. *For each Ritz vector* $\hat{q}$*, compute the corresponding approximate Ritz value as the Rayleigh quotient* $\hat{q}^T A \hat{q} / \hat{q}^T M \hat{q}$

Algorithm 2.1 sketches the Arnoldi procedure applied to $\rho(M^{-1}A)$ for the computation of all *nev* eigenvalues of (A, M) located inside $[\alpha, \beta]$ and associated eigenvectors.

Note that no a priori estimation of *nev* is necessary. Line 2 computes the "filtered" vector w by applying the matrix function $\rho(M^{-1}A)$ to $q^{(\mu)}$, which in turn requires the solution of the N_c linear systems associated with matrices $A - \zeta_\ell M,\ \ell = 1, \ldots, N_c$. Lines 3–12 orthonormalize w against the previous Arnoldi vectors $q^{(1)}, \ldots, q^{(\mu)}$ to produce the next Arnoldi vector $q^{(\mu+1)}$. Line 13 checks the sum of those eigenvalues of the upper-Hessenberg matrix H_μ which are no less than $1/2$. If this sum remains constant up to a certain tolerance, the outer loop stops. Finally, line 16 computes the Rayleigh quotients associated with the approximate eigenvectors of $\rho(M^{-1}A)$ (the Ritz vectors obtained in line 15).

Throughout the rest of this paper, Algorithm 2.1 will be abbreviated as RF-KRYLOV.

2.3. Domain decomposition framework. Domain decomposition eigenvalue solvers [9, 10, 19, 20] compute spectral information of (A, M) by decoupling the original eigenvalue problem into two separate subproblems: one defined locally in the interior of each subdomain, and one restricted to the interface region connecting neighboring subdomains. Algebraic domain decomposition eigensolvers start by calling a graph partitioner [21, 28] to decompose the adjacency graph of $|A| + |M|$ into p nonoverlapping subdomains. If we then order the interior variables in each subdomain before the interface variables across all subdomains, matrices A and M then take the following block structures:

$$A = \begin{pmatrix} B_1 & & & & E_1 \\ & B_2 & & & E_2 \\ & & \ddots & & \vdots \\ & & & B_p & E_p \\ E_1^T & E_2^T & \ldots & E_p^T & C \end{pmatrix}, \tag{2.5}$$

$$M = \begin{pmatrix} M_B^{(1)} & & & & M_E^{(1)} \\ & M_B^{(2)} & & & M_E^{(2)} \\ & & \ddots & & \vdots \\ & & & M_B^{(p)} & M_E^{(p)} \\ \left(M_E^{(1)}\right)^T & \left(M_E^{(2)}\right)^T & \ldots & \left(M_E^{(p)}\right)^T & M_C \end{pmatrix}.$$

If we denote the number of interior and interface variables lying in the jth subdomain by d_j and s_j, respectively, and set $s = \sum_{j=1}^{p} s_j$, then B_j and $M_B^{(j)}$ are square matrices of size $d_j \times d_j$, E_j and $M_E^{(j)}$ are rectangular matrices of size $d_j \times s_j$, and C and M_C are square matrices of size $s \times s$. Matrices $E_i,\ M_E^{(j)}$ have a special nonzero pattern of the form $E_j = [0_{d_j,\ell_j}, \hat{E}_j, 0_{d_j,\nu_j}]$, and $M_E^{(j)} = [0_{d_j,\ell_j}, \hat{M}_E^{(j)}, 0_{d_j,\nu_j}]$, where $\ell_j = \sum_{k=1}^{k<j} s_k$, $\nu_j = \sum_{k>j}^{k=p} s_k$, and $0_{\chi,\psi}$ denotes the zero matrix of size $\chi \times \psi$.

Under the permutation (2.5), A and M can be also written in a compact form as

$$A = \begin{pmatrix} B & E \\ E^T & C \end{pmatrix}, \quad M = \begin{pmatrix} M_B & M_E \\ M_E^T & M_C \end{pmatrix}. \tag{2.6}$$

The block-diagonal matrices B and M_B are of size $d \times d$, where $d = \sum_{i=1}^{p} d_i$, while E and M_E are of size $d \times s$.

2.3.1. Invariant subspaces from a Schur complement viewpoint. Domain decomposition eigenvalue solvers decompose the construction of the Rayleigh–Ritz projection subspace $\mathcal{Z}$ into two separate parts. More specifically, $\mathcal{Z}$ can be written as

$$\mathcal{Z} = \mathcal{U} \oplus \mathcal{Y}, \tag{2.7}$$

where $\mathcal{U}$ and $\mathcal{Y}$ are (structurally) orthogonal subspaces that approximate the part of the solution associated with the interior and interface variables, respectively.

Let the ith eigenvector of (A, M) be partitioned as

$$x^{(i)} = \begin{pmatrix} u^{(i)} \\ y^{(i)} \end{pmatrix}, \ i = 1, \ldots, n, \tag{2.8}$$

where $u^{(i)} \in \mathbb{R}^d$ and $y^{(i)} \in \mathbb{R}^s$ correspond to the eigenvector parts associated with the interior and interface variables, respectively. We can then rewrite $Ax^{(i)} = \lambda_i M x^{(i)}$ in the following block form:

$$\begin{pmatrix} B - \lambda_i M_B & E - \lambda_i M_E \\ E^T - \lambda_i M_E^T & C - \lambda_i M_C \end{pmatrix} \begin{pmatrix} u^{(i)} \\ y^{(i)} \end{pmatrix} = 0. \tag{2.9}$$

Eliminating $u^{(i)}$ from the second equation in (2.9) leads to the following nonlinear eigenvalue problem of size $s \times s$:

$$\Big[C - \lambda_i M_C - (E - \lambda_i M_E)^T (B - \lambda_i M_B)^{-1} (E - \lambda_i M_E)\Big] y^{(i)} = 0. \tag{2.10}$$

Once $y^{(i)}$ is computed in the above equation, $u^{(i)}$ can be recovered by the following linear system solution:

$$(B - \lambda_i M_B) u^{(i)} = -(E - \lambda_i M_E) y^{(i)}. \tag{2.11}$$

In practice, since matrices B and M_B in (2.5) are block-diagonal, the p subvectors $u_j^{(i)} \in \mathbb{R}^{d_j}$ of $u^{(i)} = [(u_1^{(i)})^T, \ldots, (u_p^{(i)})^T]^T$ can be computed in a decoupled fashion among the p subdomains as

$$\left(B_j - \lambda_i M_B^{(j)}\right) u_j^{(i)} = -\left(\hat{E}_j - \lambda_i \hat{M}_E^{(j)}\right) y_j^{(i)}, \ j = 1, \ldots, p, \tag{2.12}$$

where $y_j^{(i)} \in \mathbb{R}^{s_j}$ is the subvector of $y^{(i)} = [(y_1^{(i)})^T, \ldots, (y_p^{(i)})^T]^T$ that corresponds to the jth subdomain.

By (2.10) and (2.11) we see that the subspaces $\mathcal{U}$ and $\mathcal{Y}$ in (2.7) should ideally be chosen as

$$\mathcal{Y} = \text{span}\left\{\left[y^{(1)}, \ldots, y^{(nev)}\right]\right\}, \tag{2.13}$$

$$\mathcal{U} = \text{span}\left\{\left[(B-\lambda_1 M_B)^{-1}(E-\lambda_1 M_E)y^{(1)}, \ldots, (B-\lambda_{nev} M_B)^{-1}(E-\lambda_{nev} M_E)y^{(nev)}\right]\right\}. \tag{2.14}$$

The following two sections propose efficient numerical schemes to approximate these two subspaces.

3. Approximation of span$\{y^{(1)},\ldots,y^{(nev)}\}$. In this section we propose a numerical scheme to approximate span$\{y^{(1)},\ldots,y^{(nev)}\}$.

3.1. Rational filtering restricted to the interface region. We start by defining the following matrices:

$$B_{\zeta_\ell} = B - \zeta_\ell M_B, \quad E_{\zeta_\ell} = E - \zeta_\ell M_E, \quad C_{\zeta_\ell} = C - \zeta_\ell M_C.$$

Then, each matrix $(A-\zeta_\ell M)^{-1}$ in (2.4) can be expressed as

$$(A-\zeta_\ell M)^{-1} = \begin{pmatrix} B_{\zeta_\ell}^{-1} + B_{\zeta_\ell}^{-1}E_{\zeta_\ell}S_{\zeta_\ell}^{-1}E_{\zeta_\ell}^T B_{\zeta_\ell}^{-1} & -B_{\zeta_\ell}^{-1}E_{\zeta_\ell}S_{\zeta_\ell}^{-1} \\ -S_{\zeta_\ell}^{-1}E_{\zeta_\ell}^T B_{\zeta_\ell}^{-1} & S_{\zeta_\ell}^{-1} \end{pmatrix}, \tag{3.1}$$

where

$$S_{\zeta_\ell} = C_{\zeta_\ell} - E_{\zeta_\ell}^T B_{\zeta_\ell}^{-1} E_{\zeta_\ell} \tag{3.2}$$

denotes the corresponding Schur complement matrix.

Substituting (3.1) into (2.4) leads to

$$\rho(M^{-1}A) = 2\Re e\left\{\sum_{\ell=1}^{N_c}\omega_\ell \begin{bmatrix} \left[B_{\zeta_\ell}^{-1} + B_{\zeta_\ell}^{-1}E_{\zeta_\ell}S_{\zeta_\ell}^{-1}E_{\zeta_\ell}^T B_{\zeta_\ell}^{-1}\right] & -B_{\zeta_\ell}^{-1}E_{\zeta_\ell}S_{\zeta_\ell}^{-1} \\ -S_{\zeta_\ell}^{-1}E_{\zeta_\ell}^T B_{\zeta_\ell}^{-1} & S_{\zeta_\ell}^{-1} \end{bmatrix}\right\} M. \tag{3.3}$$

On the other hand, we have that for any $\zeta \notin \Lambda(A,M)$,

$$(A-\zeta M)^{-1} = \sum_{i=1}^{n} \frac{x^{(i)}\left(x^{(i)}\right)^T}{\lambda_i - \zeta}. \tag{3.4}$$

The above equality yields another expression for $\rho(M^{-1}A)$:

$$\rho(M^{-1}A) = \sum_{i=1}^{n}\rho(\lambda_i)x^{(i)}\left(x^{(i)}\right)^T M \tag{3.5}$$

$$= \sum_{i=1}^{n}\rho(\lambda_i)\begin{bmatrix} u^{(i)}\left(u^{(i)}\right)^T & u^{(i)}\left(y^{(i)}\right)^T \\ y^{(i)}\left(u^{(i)}\right)^T & y^{(i)}\left(y^{(i)}\right)^T \end{bmatrix} M. \tag{3.6}$$

Equating the (2,2) blocks of the right-hand sides in (3.3) and (3.6) yields

$$2\Re e\left\{\sum_{\ell=1}^{N_c}\omega_\ell S_{\zeta_\ell}^{-1}\right\} = \sum_{i=1}^{n}\rho(\lambda_i)y^{(i)}\left(y^{(i)}\right)^T. \tag{3.7}$$

Equation (3.7) provides a way to approximate span$\{y^{(1)},\ldots,y^{(nev)}\}$ through the information in $2\Re e\{\sum_{\ell=1}^{N_c}\omega_\ell S_{\zeta_\ell}^{-1}\}$. In particular, the magnitude of $\rho(\lambda_i)$ can be interpreted as the contribution of the direction $y^{(i)}$ in $2\Re e\{\sum_{\ell=1}^{N_c}\omega_\ell S_{\zeta_\ell}^{-1}\}$. In the ideal case where $\rho(\zeta) \equiv \pm I_{[\alpha,\beta]}(\zeta)$, we have $\sum_{i=1}^{n}\rho(\lambda_i)y^{(i)}(y^{(i)})^T = \pm\sum_{i=1}^{nev}y^{(i)}(y^{(i)})^T$. In practice, $\rho(\zeta)$ will only be an approximation to $\pm I_{[\alpha,\beta]}(\zeta)$, and since $\rho(\lambda_1),\ldots,\rho(\lambda_{nev})$ are all nonzero, the following relation holds:

$$\text{span}\left\{y^{(1)},\ldots,y^{(nev)}\right\} \subseteq \text{range}\left(\Re e\left\{\sum_{\ell=1}^{N_c}\omega_\ell S_{\zeta_\ell}^{-1}\right\}\right). \tag{3.8}$$

The above relation suggests to compute an approximation to span$\{y^{(1)},\ldots,y^{(nev)}\}$ by capturing the range space of $\Re e\{\sum_{\ell=1}^{N_c}\omega_\ell S_{\zeta_\ell}^{-1}\}$.

3.2. A Krylov-based approach. To capture range($\Re e\{\sum_{\ell=1}^{N_c} \omega_\ell S_{\zeta_\ell}^{-1}\}$) we consider the numerical scheme outlined in Algorithm 3.1. In contrast with RF-KRYLOV, Algorithm 3.1 is based on the Lanczos process [31]. Variable T_μ denotes a $\mu \times \mu$ symmetric tridiagonal matrix with $\alpha_1, \ldots, \alpha_\mu$ as its diagonal entries, and $\beta_2, \ldots, \beta_\mu$ as its off-diagonal entries, respectively. Line 2 computes the "filtered" vector w by applying $\Re e\{\sum_{\ell=1}^{N_c} \omega_\ell S_{\zeta_\ell}^{-1}\}$ to $q^{(\mu)}$ by solving the N_c linear systems associated with matrices S_{ζ_ℓ}, $\ell = 1, \ldots, N_c$. Lines 4–12 orthonormalize w against vectors $q^{(1)}, \ldots, q^{(\mu)}$ in order to generate the next vector $q^{(\mu+1)}$. Throughout this paper we apply full orthogonalization but other choices, e.g., partial orthgonalization [33], are possible. Algorithm 3.1 terminates when the trace of the tridiagonal matrices T_μ and $T_{\mu-1}$ remains the same up to a certain tolerance.

ALGORITHM 3.1. *Krylov restricted to the interface variables*

0. *Start with* $q^{(1)} \in \mathbb{R}^s$, *s.t.* $\|q^{(1)}\|_2 = 1$, $q_0 := 0$, $\beta_1 = 0$, tol $\in \mathbb{R}$
1. *For* $\mu = 1, 2, \ldots$
2. *Compute* $w = \Re e \left\{ \sum_{\ell=1}^{N_c} \omega_\ell S_{\zeta_\ell}^{-1} q^{(\mu)} \right\} - \beta_\mu q^{(\mu-1)}$
3. $\alpha_\mu = w^T q^{(\mu)}$
4. *For* $\kappa = 1, \ldots, \mu$
5. $w = w - q^{(\kappa)} \left(w^T q^{(\kappa)} \right)$
6. *End*
7. $\beta_{\mu+1} = \|w\|_2$
8. *If* $\beta_{\mu+1} = 0$
9. *generate a unit-norm* $q^{(\mu+1)}$ *orthogonal to* $q^{(1)}, \ldots, q^{(\mu)}$
10. *Else*
11. $q^{(\mu+1)} = w/\beta_{\mu+1}$
12. *EndIf*
13. *If the sum of the eigenvalues of* T_μ *remains unchanged (up to* tol*) during the last few iterations; BREAK; EndIf*
14. *End*
15. *Return* $Q_\mu = \left[q^{(1)}, \ldots, q^{(\mu)} \right]$

Algorithm 3.1 has a few key differences with RF-KRYLOV. First, Algorithm 3.1 restricts orthonormalization to vectors of length s instead of n. In addition, Algorithm 3.1 only requires linear system solutions with S_ζ instead of $A - \zeta M$. As can be verified by (3.1), a computation of the form $(A - \zeta M)^{-1} v = w$ requires (in addition to a linear system solution with matrix S_ζ) two linear system solutions with B_ζ as well as two matrix-vector multiplications with E_ζ. Finally, in contrast to RF-KRYLOV, which requires at least *nev* iterations to compute any *nev* eigenpairs of the pencil (A, M), Algorithm 3.1 might terminate in fewer than *nev* iterations. This possible "early termination" of Algorithm 3.1 is explained in more detail by Proposition 3.1.

PROPOSITION 3.1. *The rank of the matrix* $\Re e\{\sum_{\ell=1}^{N_c} \omega_\ell S_{\zeta_\ell}^{-1}\}$,

$$r(S) = \text{rank} \left(\Re e \left\{ \sum_{\ell=1}^{N_c} \omega_\ell S_{\zeta_\ell}^{-1} \right\} \right) \tag{3.9}$$

satisfies the inequality

$$\text{rank} \left(\left[y^{(1)}, \ldots, y^{(nev)} \right] \right) \leq r(S) \leq s. \tag{3.10}$$

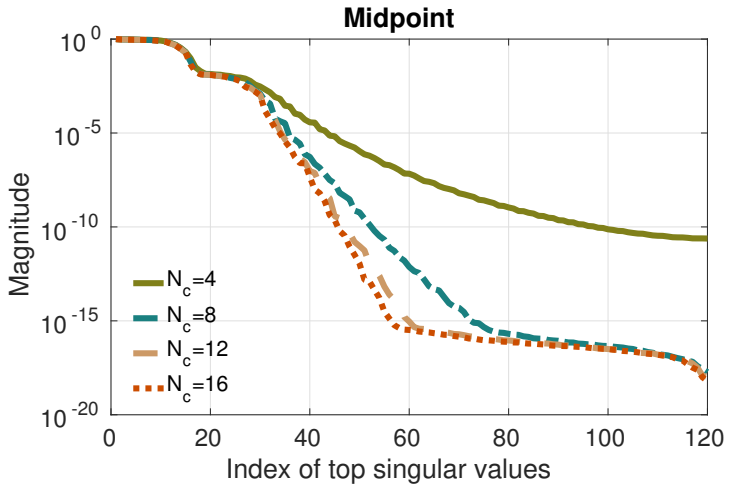


FIG. 3.1. *The leading singular values of* $\Re e\{\sum_{\ell=1}^{N_c} \omega_\ell S_{\zeta_\ell}^{-1}\}$ *for different values of* N_c *when* (2.1) *is discretized by the midpoint rule.*

Proof. We first prove the upper bound of $r(S)$. Since $\Re e\{\sum_{\ell=1}^{N_c} \omega_\ell S_{\zeta_\ell}^{-1}\}$ is of size $s \times s$, $r(S)$ cannot exceed s. To get the lower bound, let $\rho(\lambda_i) = 0,\ i = nev+\kappa, \ldots, n$, where $0 \le \kappa \le n - nev$. We then have

$$\Re e\left\{\sum_{\ell=1}^{N_c} \omega_\ell S_{\zeta_\ell}^{-1}\right\} = \sum_{i=1}^{nev+\kappa} \rho(\lambda_i) y^{(i)} \left(y^{(i)}\right)^T,$$

and $\operatorname{rank}(\Re e\{\sum_{\ell=1}^{N_c} \omega_\ell S_{\zeta_\ell}^{-1}\}) = \operatorname{rank}([y^{(1)}, \ldots, y^{(nev+\kappa)}])$. Since $\rho(\lambda_i) \neq 0,\ i = 1, \ldots, nev$, we have

$$r(S) = \operatorname{rank}\left(\left[y^{(1)}, \ldots, y^{(nev+\kappa)}\right]\right) \ge \operatorname{rank}\left(\left[y^{(1)}, \ldots, y^{(nev)}\right]\right). \qquad \square$$

By Proposition 3.1, Algorithm 3.1 will perform at most $r(S)$ iterations, and $r(S)$ can be as small as $\operatorname{rank}([y^{(1)}, \ldots, y^{(nev)}])$. We quantify this with a short example for a two-dimensional Laplacian matrix generated by a finite difference discretization with Dirichlet boundary conditions (for more details on this matrix see entry "FDmesh1" in Table 6.1) where we set $[\alpha, \beta] = [\lambda_1, \lambda_{100}]$ (thus $nev = 100$). After computing the vectors $y^{(1)}, \ldots, y^{(nev)}$ explicitly, we found that $\operatorname{rank}([y^{(1)}, \ldots, y^{(nev)}]) = 48$. Figure 3.1 plots the 120 (after normalization) leading singular values of matrix $\Re e\{\sum_{\ell=1}^{N_c} \omega_\ell S_{\zeta_\ell}^{-1}\}$. As N_c increases, the trailing $s - \operatorname{rank}([y^{(1)}, \ldots, y^{(nev)}])$ singular values approach zero. Moreover, even for those singular values which are not zero, their magnitude might be small, in which case Algorithm 3.1 might still converge in fewer than $r(S)$ iterations. Indeed, when $N_c = 16$, Algorithm 3.1 terminates after exactly 36 iterations, which is lower than $r(S)$ and only one third of the minimum number of iterations required by RF-KRYLOV for any value of N_c. As a sidenote, when $N_c = 2$, Algorithm 3.1 terminates after 70 iterations.

4. Approximation of span$\{u^{(1)}, \ldots, u^{(nev)}\}$. Recall the eigenvector partitioning $(x^{(i)})^T = \left[(u^{(i)})^T, (y^{(i)})^T\right]^T$ for each $x^{(i)},\ \lambda_i,\ i = 1, \ldots, nev$. A straightforward approach to compute the part $u^{(i)}$ of $x^{(i)}$ is then to solve the block-diagonal linear

system in (2.11). However, this approach entails two main drawbacks. First, solving the linear systems with matrices $B - \lambda_i M_B$ for all $\lambda_i,\ i = 1, \ldots, nev$ becomes prohibitively expensive when $nev \gg 1$. In addition, Algorithm 3.1 only returns an approximation of $\text{span}\{y^{(1)}, \ldots, y^{(nev)}\}$, rather than the individual vectors $y^{(1)}, \ldots, y^{(nev)}$, or the eigenvalues $\lambda_1, \ldots, \lambda_{nev}$.

In the remaining part of this section we consider alternative approaches to approximate the subspace $\text{span}\{u^{(1)}, \ldots, u^{(nev)}\}$. Since the following discussion applies to any of the n eigenpairs $(\lambda, x^{(i)})$ of (A, M), we will drop the superscripts in vectors $u^{(i)}$ and $y^{(i)}$.

4.1. The basic approximation. Let $x^T = \left[u^T, y^T\right]^T$ be the eigenvector associated with (the unknown) eigenvalue λ of (A, M). The part associated with the interior variables, u, can be then approximated as

$$\hat{u} = -B_\sigma^{-1} E_\sigma y, \tag{4.1}$$

where $\sigma \in \mathbb{R}$. As it is trivial to verify, when $\sigma \equiv \lambda$, the formula in (4.1) leads to $\hat{u} \equiv u$. In the general case, the approximation error is of the following form.

LEMMA 4.1. *Suppose u and $\hat{u}$ are computed as in* (2.11) *and* (4.1)*, respectively. Then*

$$u - \hat{u} = -\left[B_\lambda^{-1} - B_\sigma^{-1}\right] E_\sigma y + (\lambda - \sigma) B_\lambda^{-1} M_E y. \tag{4.2}$$

Proof. We can write u as

$$\begin{aligned} u &= -B_\lambda^{-1} E_\lambda y \\ &= -B_\lambda^{-1}(E_\sigma - (\lambda - \sigma) M_E) y \\ &= -B_\lambda^{-1} E_\sigma y + (\lambda - \sigma) B_\lambda^{-1} M_E y. \end{aligned} \tag{4.3}$$

The result in (4.2) follows by combining (4.1) and (4.3). ∎

We are now ready to compute an upper bound of $u - \hat{u}$ measured in the M_B-norm.[1]

THEOREM 4.2. *Let the eigendecomposition of (B, M_B) be written as*

$$BV = M_B V D, \tag{4.4}$$

where $D = \text{diag}(\delta_1, \ldots, \delta_d)$ and $V = [v^{(1)}, \ldots, v^{(d)}]$. If $\hat{u}$ is defined as in (4.1) *and $(\delta_\ell, v^{(\ell)})$, $\ell = 1, \ldots, d$, denote the eigenpairs of (B, M_B) where each eigenvector $v^{(\ell)}$ is scaled such that $(v^{(\ell)})^T M_B v^{(\ell)} = 1$, then*

$$\|u - \hat{u}\|_{M_B} \leq \max_\ell \frac{|\lambda - \sigma|}{|(\lambda - \delta_\ell)(\sigma - \delta_\ell)|} ||E_\sigma y||_{M_B^{-1}} + \max_\ell \frac{|\lambda - \sigma|}{|\lambda - \delta_\ell|} ||M_E y||_{M_B^{-1}}. \tag{4.5}$$

Proof. Since M_B is SPD, vectors $E_\sigma y$ and $M_E y$ in (4.3) can be expanded in the basis $M_B v^{(\ell)}$ as

$$E_\sigma y = M_B \sum_{\ell=1}^{\ell=d} \epsilon_\ell v^{(\ell)}, \quad M_E y = M_B \sum_{\ell=1}^{\ell=d} \gamma_\ell v^{(\ell)}, \tag{4.6}$$

[1] We define the X-norm of any nonzero vector y and SPD matrix X as $||y||_X = \sqrt{y^T X y}$.

where $\epsilon_\ell,\ \gamma_\ell \in \mathbb{R}$ are the expansion coefficients. By recalling (4.4) and noticing that $V^T M_B V = I$, we get

$$B_\sigma^{-1} = V(D-\sigma I)^{-1}V^T, \qquad B_\lambda^{-1} = V(D-\lambda I)^{-1}V^T. \tag{4.7}$$

Substituting the expressions in (4.6) and (4.7) into the right-hand side of (4.2) gives

$$\begin{aligned} u-\hat{u} =& -V\left[(D-\lambda I)^{-1}-(D-\sigma I)^{-1}\right]V^T\left(M_B\sum_{\ell=1}^{\ell=d}\epsilon_\ell v^{(\ell)}\right) \\ &+(\lambda-\sigma)V(D-\lambda I)^{-1}V^T\left(M_B\sum_{\ell=1}^{\ell=d}\gamma_\ell v^{(\ell)}\right) \\ =& -\sum_{\ell=1}^{\ell=d}\frac{\epsilon_\ell(\lambda-\sigma)}{(\delta_\ell-\lambda)(\delta_\ell-\sigma)}v^{(\ell)}+\sum_{\ell=1}^{\ell=d}\frac{\gamma_\ell(\lambda-\sigma)}{\delta_\ell-\lambda}v^{(\ell)}. \end{aligned} \tag{4.8}$$

Now, taking the M_B-norm of (4.8), we finally obtain

$$\begin{aligned} ||u-\hat{u}||_{M_B} &\le \left\|\sum_{\ell=1}^{\ell=d}\frac{-\epsilon_\ell(\lambda-\sigma)}{(\delta_\ell-\lambda)(\delta_\ell-\sigma)}v^{(\ell)}\right\|_{M_B}+\left\|\sum_{\ell=1}^{\ell=d}\frac{\gamma_\ell(\lambda-\sigma)}{\delta_\ell-\lambda}v^{(\ell)}\right\|_{M_B} \\ &= \left\|\sum_{\ell=1}^{\ell=d}\left|\frac{(\lambda-\sigma)}{(\delta_\ell-\lambda)(\delta_\ell-\sigma)}\right|\epsilon_\ell v^{(\ell)}\right\|_{M_B}+\left\|\sum_{\ell=1}^{\ell=d}\left|\frac{(\lambda-\sigma)}{\delta_\ell-\lambda}\right|\gamma_\ell v^{(\ell)}\right\|_{M_B} \\ &\le \max_\ell\frac{|\lambda-\sigma|}{|(\lambda-\delta_\ell)(\sigma-\delta_\ell)|}\left\|\sum_{\ell=1}^{\ell=d}\epsilon_\ell v^{(\ell)}\right\|_{M_B}+\max_\ell\frac{|\lambda-\sigma|}{|\lambda-\delta_\ell|}\left\|\sum_{\ell=1}^{\ell=d}\gamma_\ell v^{(\ell)}\right\|_{M_B} \\ &= \max_\ell\frac{|\lambda-\sigma|}{|(\lambda-\delta_\ell)(\sigma-\delta_\ell)|}\left\|M_B^{-1}E_\sigma y\right\|_{M_B}+\max_\ell\frac{|\lambda-\sigma|}{|\lambda-\delta_\ell|}\left\|M_B^{-1}M_E y\right\|_{M_B} \\ &= \max_\ell\frac{|\lambda-\sigma|}{|(\lambda-\delta_\ell)(\sigma-\delta_\ell)|}\left\|E_\sigma y\right\|_{M_B^{-1}}+\max_\ell\frac{|\lambda-\sigma|}{|\lambda-\delta_\ell|}\left\|M_E y\right\|_{M_B^{-1}}. \end{aligned}$$

∎

Theorem 4.2 indicates that the upper bound of $||u-\hat{u}||_{M_B}$ depends on the distance between σ and λ, as well as the distance of these values from the eigenvalues of (B, M_B). In particular, this upper bound becomes small when λ and σ lie close to each other, and far from the eigenvalues of (B, M_B).

4.2. Enhancing accuracy by resolvent expansions. Consider the resolvent expansion of B_λ^{-1} around $\sigma \in \mathbb{R}$:

$$B_\lambda^{-1} = B_\sigma^{-1}\sum_{\theta=0}^{\infty}\left[(\lambda-\sigma)M_B B_\sigma^{-1}\right]^\theta. \tag{4.9}$$

By recalling (4.2), the error $u-\hat{u}$ consists of two components: (i) $(B_\lambda^{-1}-B_\sigma^{-1})E_\sigma y$; and (ii) $(\lambda-\sigma)B_\lambda^{-1}M_E y$. A straightforward idea then is to approximate B_λ^{-1} by also considering higher-order terms in (4.9) instead of B_σ^{-1} only. Furthermore, the same idea can be repeated for the second error component. Thus, we can extract $\hat{u}$ by a projection step from the following subspace:

$$\hat{u} \in \left\{B_\sigma^{-1}E_\sigma y,\ldots,B_\sigma^{-1}\left(M_B B_\sigma^{-1}\right)^{\psi-1}E_\sigma y, B_\sigma^{-1}M_E y,\ldots,B_\sigma^{-1}\left(M_B B_\sigma^{-1}\right)^{\psi-1}M_E y\right\}. \tag{4.10}$$

The following theorem refines the upper bound of $\|u-\hat{u}\|_{M_B}$ computed in Theorem 4.2.

THEOREM 4.3. *Let* $\mathcal{U} = \text{span}\{U_1, U_2\}$*, where*

$$U_1 = \left[B_\sigma^{-1} E_\sigma y, \ldots, B_\sigma^{-1}\left(M_B B_\sigma^{-1}\right)^{\psi-1} E_\sigma y\right], \tag{4.11}$$

$$U_2 = \left[B_\sigma^{-1} M_E y, \ldots, B_\sigma^{-1}\left(M_B B_\sigma^{-1}\right)^{\psi-1} M_E y\right]. \tag{4.12}$$

If $\hat{u} := \arg\min_{g\in\mathcal{U}} \|u - g\|_{M_B}$*, and* $(\delta_\ell, v^{(\ell)})$*,* $\ell = 1, \ldots, d$*, denote the eigenpairs of* (B, M_B)*, then*

$$\|u-\hat{u}\|_{M_B} \leq \max_\ell \frac{|\lambda-\sigma|^\psi}{|(\lambda-\delta_\ell)(\sigma-\delta_\ell)^\psi|}||E_\sigma y||_{M_B^{-1}} + \max_\ell \frac{|\lambda-\sigma|^{\psi+1}}{|(\lambda-\delta_\ell)(\sigma-\delta_\ell)^\psi|}||M_E y||_{M_B^{-1}}. \tag{4.13}$$

Proof. Define the vector $g := U_1\mathbf{c}_1 + U_2\mathbf{c}_2$, where

$$\mathbf{c}_1 = -\Big[1, \lambda-\sigma, \ldots, (\lambda-\sigma)^{\psi-1}\Big]^T, \quad \mathbf{c}_2 = \Big[\lambda-\sigma, \ldots, (\lambda-\sigma)^\psi\Big]^T.$$

The difference $u - g$ then is equal to

$$\begin{aligned} u - g = &- \left[B_\lambda^{-1} - B_\sigma^{-1}\sum_{\theta=0}^{\psi-1}\left[(\lambda-\sigma)M_B B_\sigma^{-1}\right]^\theta\right] E_\sigma y \\ &+ (\lambda-\sigma)\left[B_\lambda^{-1} - B_\sigma^{-1}\sum_{\theta=0}^{\psi-1}\left[(\lambda-\sigma)M_B B_\sigma^{-1}\right]^\theta\right] M_E y. \end{aligned} \tag{4.14}$$

Expanding B_σ^{-1} and B_λ^{-1} in the eigenbasis of (B, M_B) gives

$$B_\lambda^{-1} - B_\sigma^{-1}\sum_{\theta=0}^{\psi-1}\left[(\lambda-\sigma)M_B B_\sigma^{-1}\right]^\theta = (\lambda-\sigma)^\psi V(D-\lambda I)^{-1}(D-\sigma I)^{-\psi}V^T, \tag{4.15}$$

and thus (4.14) can be simplified as

$$\begin{aligned} u - g = &-(\lambda-\sigma)^\psi V(D-\lambda I)^{-1}(D-\sigma I)^{-\psi}V^T E_\sigma y \\ &+ (\lambda-\sigma)^{\psi+1}V(D-\lambda I)^{-1}(D-\sigma I)^{-\psi}V^T M_E y. \end{aligned}$$

Plugging in the expansion of $E_\sigma y$ and $M_E y$ defined in (4.6) finally leads to

$$u - g = -\sum_{\ell=1}^{\ell=d}\frac{\epsilon_\ell(\lambda-\sigma)^\psi}{(\delta_\ell-\lambda)(\delta_\ell-\sigma)^\psi}v^{(\ell)} + \sum_{\ell=1}^{\ell=d}\frac{\gamma_\ell(\lambda-\sigma)^{\psi+1}}{(\delta_\ell-\lambda)(\delta_\ell-\sigma)^\psi}v^{(\ell)}. \tag{4.16}$$

Similarly to Theorem 4.2, we consider the M_B-norm of (4.16):

$$\begin{aligned}\|u-g\|_{M_B} &\le \left\|\sum_{\ell=1}^{\ell=d}\frac{-\epsilon_\ell(\lambda-\sigma)^\psi}{(\delta_\ell-\lambda)(\delta_\ell-\sigma)^\psi}v^{(\ell)}\right\|_{M_B}+\left\|\sum_{\ell=1}^{\ell=d}\frac{\gamma_\ell(\lambda-\sigma)^{\psi+1}}{(\delta_\ell-\lambda)(\delta_\ell-\sigma)^\psi}v^{(\ell)}\right\|_{M_B}\\ &\le \max_\ell\frac{|\lambda-\sigma|^\psi}{|(\lambda-\delta_\ell)(\sigma-\delta_\ell)^\psi|}\left\|\sum_{\ell=1}^{\ell=d}\epsilon_\ell v^{(\ell)}\right\|_{M_B}\\ &\quad+\max_\ell\frac{|\lambda-\sigma|^{\psi+1}}{|(\lambda-\delta_\ell)(\sigma-\delta_\ell)^\psi|}\left\|\sum_{\ell=1}^{\ell=d}\gamma_\ell v^{(\ell)}\right\|_{M_B}\\ &= \max_\ell\frac{|\lambda-\sigma|^\psi}{|(\lambda-\delta_\ell)(\sigma-\delta_\ell)^\psi|}||E_\sigma y||_{M_B^{-1}}+\max_\ell\frac{|\lambda-\sigma|^{\psi+1}}{|(\lambda-\delta_\ell)(\sigma-\delta_\ell)^\psi|}||M_E y||_{M_B^{-1}}.\end{aligned}$$

Recalling that $\hat{u}:=\arg\min_{g\in\mathcal{U}}\|u-g\|_{M_B}$ finishes the proof. ∎

A comparison of the bound in Theorem 4.3 with the bound in Theorem 4.2 indicates that one may expect an improved approximation when σ is close to λ. The numerical examples in section 6 will verify that this approach enhances accuracy even when $|\sigma-\lambda|$ is not very small.

4.3. Enhancing accuracy by eigenvector deflation. Both Theorems 4.2 and 4.3 imply that the approximation error $u-\hat{u}$ might have its largest components along those eigenvector directions associated with those eigenvalues of (B,M_B) located the closest to σ. We can remove these error directions explicitly by augmenting the projection subspace with the corresponding eigenvectors of (B,M_B).

THEOREM 4.4. *Let $\delta_1,\delta_2,\ldots,\delta_\kappa$ be the κ eigenvalues of (B,M_B) that lie the closest to σ, and let $v^{(1)},v^{(2)},\ldots,v^{(\kappa)}$ denote the corresponding eigenvectors. Moreover, let $\mathcal{U}=\operatorname{span}\{U_1,U_2,U_3\}$, where*

$$U_1=\left[B_\sigma^{-1}E_\sigma y,\ldots,B_\sigma^{-1}\left(M_BB_\sigma^{-1}\right)^{\psi-1}E_\sigma y\right], \tag{4.17}$$

$$U_2=\left[B_\sigma^{-1}M_E y,\ldots,B_\sigma^{-1}\left(M_BB_\sigma^{-1}\right)^{\psi-1}M_E y\right], \tag{4.18}$$

$$U_3=\left[v^{(1)},v^{(2)},\ldots,v^{(\kappa)}\right]. \tag{4.19}$$

If $\hat{u}:=\arg\min_{g\in\mathcal{U}}\|u-g\|_{M_B}$ and $(\delta_\ell,v^{(\ell)})$, $\ell=1,\ldots,d$, denote the eigenpairs of (B,M_B), then

$$\|u-\hat{u}\|_{M_B}\le\max_{\ell>\kappa}\frac{|\lambda-\sigma|^\psi}{|(\lambda-\delta_\ell)(\sigma-\delta_\ell)^\psi|}||E_\sigma y||_{M_B^{-1}}+\max_{\ell>\kappa}\frac{|\lambda-\sigma|^{\psi+1}}{|(\lambda-\delta_\ell)(\sigma-\delta_\ell)^\psi|}||M_E y||_{M_B^{-1}}. \tag{4.20}$$

Proof. Let us define the vector $g:=U_1\mathbf{c}_1+U_2\mathbf{c}_2+U_3\mathbf{c}_3$, where

$$\mathbf{c}_1=-\left[1,\lambda-\sigma,\ldots,(\lambda-\sigma)^{\psi-1}\right]^T,\quad \mathbf{c}_2=\left[\lambda-\sigma,\ldots,(\lambda-\sigma)^\psi\right]^T,$$
$$\mathbf{c}_3=\left[\frac{\gamma_1(\lambda-\sigma)^{\psi+1}-\epsilon_1(\lambda-\sigma)^\psi}{(\delta_1-\lambda)(\delta_1-\sigma)^\psi},\ldots,\frac{\gamma_\kappa(\lambda-\sigma)^{\psi+1}-\epsilon_\kappa(\lambda-\sigma)^\psi}{(\delta_\kappa-\lambda)(\delta_\kappa-\sigma)^\psi}\right]^T.$$

We then have

$$\begin{aligned} u - g =& \sum_{\ell=1}^{\ell=d} \frac{-\epsilon_\ell(\lambda-\sigma)^\psi}{(\delta_\ell-\lambda)(\delta_\ell-\sigma)^\psi} v^{(\ell)} + \sum_{\ell=1}^{\ell=d} \frac{\gamma_\ell(\lambda-\sigma)^{\psi+1}}{(\delta_\ell-\lambda)(\delta_\ell-\sigma)^\psi} v^{(\ell)} \\ & - \sum_{\ell=1}^{\ell=\kappa} \frac{-\epsilon_\ell(\lambda-\sigma)^\psi}{(\delta_\ell-\lambda)(\delta_\ell-\sigma)^\psi} v^{(\ell)} - \sum_{\ell=1}^{\ell=\kappa} \frac{\gamma_\ell(\lambda-\sigma)^{\psi+1}}{(\delta_\ell-\lambda)(\delta_\ell-\sigma)^\psi} v^{(\ell)} \\ =& \sum_{\ell=\kappa+1}^{\ell=d} \frac{-\epsilon_\ell(\lambda-\sigma)^\psi}{(\delta_\ell-\lambda)(\delta_\ell-\sigma)^\psi} v^{(\ell)} + \sum_{\ell=\kappa+1}^{\ell=d} \frac{\gamma_\ell(\lambda-\sigma)^{\psi+1}}{(\delta_\ell-\lambda)(\delta_\ell-\sigma)^\psi} v^{(\ell)}, \end{aligned}$$

and taking the M_B-norm of $u-g$ finally leads to

$$\begin{aligned} \|u-g\|_{M_B} \leq & \left\| \sum_{\ell=\kappa+1}^{\ell=d} \frac{-\epsilon_\ell(\lambda-\sigma)^\psi}{(\delta_\ell-\lambda)(\delta_\ell-\sigma)^\psi} v^{(\ell)} \right\|_{M_B} + \left\| \sum_{\ell=\kappa+1}^{\ell=d} \frac{\gamma_\ell(\lambda-\sigma)^{\psi+1}}{(\delta_\ell-\lambda)(\delta_\ell-\sigma)^\psi} v^{(\ell)} \right\|_{M_B} \\ & \leq \max_{\ell>\kappa} \frac{|\lambda-\sigma|^\psi}{|(\lambda-\delta_\ell)(\sigma-\delta_\ell)^\psi|} ||E_\sigma y||_{M_B^{-1}} + \max_{\ell>\kappa} \frac{|\lambda-\sigma|^{\psi+1}}{|(\lambda-\delta_\ell)(\sigma-\delta_\ell)^\psi|} ||M_E y||_{M_B^{-1}}. \end{aligned}$$

Recalling that $\hat{u} := \arg\min_{g\in\mathcal{U}} \|u-g\|_{M_B}$ concludes the proof. ∎

5. The RF-DDES algorithm. In this section we describe RF-DDES in terms of a formal algorithm.

RF-DDES starts by calling a graph partitioner to partition the graph of $|A|+|M|$ into p subdomains and reorders the matrix pencil (A, M) as in (2.5). RF-DDES then proceeds to the computation of those eigenvectors associated with the $nev_B^{(j)}$ smallest (in magnitude) eigenvalues of each matrix pencil $(B_j - \sigma M_B^{(j)} M_B^{(j)})$ and stores these eigenvectors in $V_j \in \mathbb{R}^{d_j \times nev_B^{(j)}}$, $j = 1, \ldots, p$. As our current implementation stands, these eigenvectors are computed by Lanczos combined with shift-and-invert [17], but other options, e.g., Davidson-based techniques, are possible [36]. Moreover, while in this paper we do not consider any special mechanisms to set the value of $nev_B^{(j)}$, it is possible to adapt the work in [41]. The next step of RF-DDES is to call Algorithm 3.1 and approximate $\text{span}\{y^{(1)}, \ldots, y^{(nev)}\}$ by $\text{range}\{Q\}$, where Q denotes the orthonormal matrix returned by Algorithm 3.1. RF-DDES then builds an approximation subspace as described in section 5.1 and performs a Rayleigh–Ritz projection to extract approximate eigenpairs of (A, M). The complete procedure is shown in Algorithm 5.1.

ALGORITHM 5.1. *RF-DDES*

0. *Input:* A, M, α, β, σ, p, $\{\omega_\ell, \zeta_\ell\}_{\ell=1,\ldots,N_c}$, $\left\{nev_B^{(j)}\right\}_{j=1,\ldots,p}$, ψ
1. *Reorder* A *and* M *as in* (2.5)
2. *For* $j = 1, \ldots, p$*:*
3. *Compute the eigenvectors associated with the* $nev_B^{(j)}$ *smallest (in magnitude) eigenvalues of* $\left(B_\sigma^{(j)}, M_B^{(j)}\right)$ *and store them in* V_j
4. *End*
5. *Compute* Q *by Algorithm* 3.1
6. *Form* Z *as in* (5.3)
7. *Solve the Rayleigh–Ritz eigenvalue problem:* $Z^T A Z G = Z^T M Z G \hat{\Lambda}$
8. *If eigenvectors were also sought, permute the entries of each approximate eigenvector back to their original ordering*

The Rayleigh–Ritz eigenvalue problem in line 7 of RF-DDES can be solved either by a shift-and-invert procedure or by the appropriate routine in LAPACK [11].

5.1. The projection matrix Z. Let matrix Q returned by Algorithm 3.1 be written in its distributed form among the p subdomains,

$$Q = \begin{pmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_p \end{pmatrix}, \tag{5.1}$$

where $Q_j \in \mathbb{R}^{s_i \times \mu},\ j = 1, \ldots, p$, is local to the jth subdmonain and $\mu \in \mathbb{N}^*$ denotes the total number of iterations performed by Algorithm 3.1. By defining

$$\begin{aligned} B_\sigma^{(j)} &= B_j - \sigma M_B^{(j)}, \\ \Phi_\sigma^{(j)} &= -\left(\hat{E}_j - \sigma \hat{M}_E^{(j)}\right) Q_j, \\ \Psi^{(j)} &= \hat{M}_E^{(j)} Q_j, \end{aligned} \tag{5.2}$$

the Rayleigh–Ritz projection matrix Z in RF-DDES can be written as

$$Z = \begin{pmatrix} V_1 & & & & \Sigma_1^{(\psi)} & \Gamma_1^{(\psi)} \\ & V_2 & & & \Sigma_2^{(\psi)} & \Gamma_2^{(\psi)} \\ & & \ddots & & \vdots & \vdots \\ & & & V_p & \Sigma_p^{(\psi)} & \Gamma_p^{(\psi)} \\ & & & & [Q, 0_{s,(\psi-1)\mu}] & \end{pmatrix}, \tag{5.3}$$

where $0_{\chi,\zeta}$ denotes a zero matrix of size $\chi \times \zeta$, and

(5.4)

$$\Sigma_j^{(\psi)} = \left[(B_\sigma^{(j)})^{-1} \Phi_\sigma^{(j)}, (B_\sigma^{(j)})^{-1} M_B^{(j)} (B_\sigma^{(j)})^{-1} \Phi_\sigma^{(j)}, \ldots, (B_\sigma^{(j)})^{-1} \left(M_B^{(j)} (B_\sigma^{(j)})^{-1} \right)^{\psi-1} \Phi_\sigma^{(j)} \right],$$

$$\Gamma_j^{(\psi)} = \left[(B_\sigma^{(j)})^{-1} \Psi^{(j)}, (B_\sigma^{(j)})^{-1} M_B^{(j)} (B_\sigma^{(j)})^{-1} \Psi^{(j)}, \ldots, (B_\sigma^{(j)})^{-1} \left(M_B^{(j)} (B_\sigma^{(j)})^{-1} \right)^{\psi-1} \Psi^{(j)} \right].$$

When M_E is a nonzero matrix, the size of matrix Z is $n \times (\kappa + 2\psi\mu)$, $\kappa = \sum_{j=1}^{p} nev_B^{(j)}$. When $M_E \equiv 0_{d,s}$ (as is the case, for example, when M is the identity matrix) the size of Z reduces to $n \times (\kappa + \psi\mu)$ since $\Gamma_j^{(\psi)} \equiv 0_{d_j,\psi\mu}$. The total memory overhead of RF-DDES associated with the jth subdomain is then at most that of storing $d_j(nev_B^{(j)} + 2\psi\mu) + s_i\mu$ floating-point numbers.

5.2. Main differences with AMLS. Both RF-DDES and AMLS exploit the domain decomposition framework discussed in section 2.3. However, the two methods differ in a few points.

In contrast to RF-DDES which exploits Algorithm 3.1, AMLS approximates the part of the solution associated with the interface variables of (A, M) by solving a generalized eigenvalue problem stemming by a first-order approximation of the nonlinear eigenvalue problem in (2.10). More specifically, AMLS approximates

span$\{[y^{(1)}, \ldots, y^{(nev)}]\}$ by the span of the eigenvectors associated with a few of the eigenvalues of smallest magnitude of the SPD pencil $(S_\sigma, -S'_\sigma)$, where σ is some real shift and S'_σ denotes the derivative of the Schur complement matrix at σ. As such, only the span of those vectors $y^{(i)}$ for which λ_i lies sufficiently close to σ can be captured very accurately. In the standard AMLS method the shift σ is zero. In contrast, RF-DDES can capture all of span$\{[y^{(1)}, \ldots, y^{(nev)}]\}$ to high accuracy regardless of where λ_i is located inside the interval $[\alpha, \beta]$ of interest.

Another difference between RF-DDES and AMLS concerns the way in which the two schemes approximate span$\{[u^{(1)}, \ldots, u^{(nev)}]\}$. As can be easily verified, AMLS is similar to RF-DDES with the choice $\psi = 1$ [9]. While it is possible to combine AMLS with higher values of ψ, this might not always lead to a significant increase in the accuracy of the approximate eigenpairs of (A, M) due to the inaccuracies in the approximation of span$\{[y^{(1)}, \ldots, y^{(nev)}]\}$. In contrast, because RF-DDES can compute a good approximation to the entire space span$\{[y^{(1)}, \ldots, y^{(nev)}]\}$, the accuracy of the approximate eigenpairs of (A, M) can be improved by simply increasing ψ and/or $nev_B^{(j)}$ and repeating the Rayleigh–Ritz projection.

From a computational viewpoint, AMLS computes the factorization of S_σ in real arithmetic and proceeds to compute a partial solution of the eigenvalue problem with the matrix pencil $(S_\sigma, -S'_\sigma)$. When the accuracy provided by AMLS is deemed accurate enough without computing a large number of eigenvectors of $(S_\sigma, -S'_\sigma)$, AMLS can be potentially faster than RF-DDES. Note, however, that even in this scenario RF-DDES might also be an appealing approach due to its good parallelization properties, e.g., when a distributed memory environment is available.

6. Experiments. In this section we present numerical experiments performed in serial and distributed memory computing environments. The RF-KRYLOV and RF-DDES schemes were written in C/C++ and built on top of the PETSc [6, 14, 15] and Intel Math Kernel (MKL) scientific libraries. The source files were compiled with the Intel MPI compiler `mpiicpc`, using the -O3 optimization level. For RF-DDES, the computational domain was partitioned to p nonoverlapping subdomains by the METIS graph partitioner [21], and each subdomain was then assigned to a distinct processor group. Communication among different processor groups was achieved by means of the Message Passing Interface (MPI) standard [35]. The linear system solutions with matrices $A - \zeta_1 M, \ldots, A - \zeta_{N_c} M$ and $S_{\zeta_1}, \ldots, S_{\zeta_{N_c}}$ were computed by the Multifrontal Massively Parallel Sparse Direct Solver [3], while those with the block-diagonal matrices $B_{\zeta_1}, \ldots, B_{\zeta_{N_c}}$, and B_σ by MKL PARDISO [1].

The quadrature node-weight pairs $\{\omega_\ell, \zeta_\ell\}$, $\ell = 1, \ldots, N_c$ were computed by the midpoint quadrature rule of order $2N_c$, retaining only the N_c quadrature nodes (and associated weights) with positive imaginary part. Unless stated otherwise, the default values used throughout the experiments were $p = 2$, $N_c = 2$, and $\sigma = 0$, while we set $nev_B^{(1)} = \ldots = nev_B^{(p)} = nev_B$. The stopping criterion in Algorithm 3.1, was set to tol = 1e-6. All computations were carried out in 64-bit (double) precision, and all wall-clock times reported throughout the rest of this section will be listed in seconds.

6.1. Computational system. The experiments were performed on the `Mesabi` Linux cluster at the Minnesota Supercomputing Institute. `Mesabi` consists of 741 nodes of various configurations with a total of 17,784 compute cores that are part of Intel Haswell E5-2680v3 processors. Each node features two sockets, each socket with twelve physical cores at 2.5 GHz. Moreover, each node is equipped with 64 GB of system memory.

TABLE 6.1
n: size of A and M, nnz(X): number of nonzero entries in matrix X.

#	Mat. pencil	n	$nnz(A)/n$	$nnz(M)/n$	$[\alpha, \beta]$	nev
1.	bcsst24	3,562	44.89	1.00	[0, 352.55]	100
2.	Kuu/Muu	7,102	47.90	23.95	[0, 934.30]	100
3.	FDmesh1	24,000	4.97	1.00	[0, 0.0568]	100
4.	bcsst39	46,772	44.05	1.00	[-11.76, 3915.7]	100
5.	qa8fk/qa8fm	66,127	25.11	25.11	[0, 15.530]	100

TABLE 6.2
Maximum relative errors of the approximation of the lowest $nev = 100$ eigenvalues returned by RF-DDES for the matrix pencils listed in Table 6.1.

	$nev_B = 50$			$nev_B = 100$			$nev_B = 200$		
Mat. pencil	$\psi = 1$	$\psi = 2$	$\psi = 3$	$\psi = 1$	$\psi = 2$	$\psi = 3$	$\psi = 1$	$\psi = 2$	$\psi = 3$
bcsst24	2.2e-2	1.8e-3	3.7e-5	9.2e-3	1.5e-5	1.4e-7	7.2e-4	2.1e-8	4.1e-11
Kuu/Muu	2.4e-2	5.8e-3	7.5e-4	5.5e-3	6.6e-5	1.5e-6	1.7e-3	2.0e-6	2.3e-8
FDmesh1	1.8e-2	5.8e-3	5.2e-3	6.8e-3	2.2e-4	5.5e-6	2.3e-3	1.3e-5	6.6e-8
bcsst39	2.5e-2	1.1e-2	8.6e-3	1.2e-2	7.8e-5	2.3e-6	4.7e-3	4.4e-6	5.9e-7
qa8fk/qa8fm	1.6e-1	9.0e-2	2.0e-2	7.7e-2	5.6e-3	1.4e-4	5.9e-2	4.4e-4	3.4e-6

6.2. Numerical illustration of RF-DDES. We tested RF-DDES on the matrix pencils listed in Table 6.1. For each pencil, the interval of interest $[\alpha, \beta]$ was chosen so that $nev = 100$. Matrix pencils 1, 2, 4, and 5 can be found in the SuiteSparse matrix collection (https://sparse.tamu.edu/) [12]. Matrix pencil 3 was obtained by a discretization of a differential eigenvalue problem associated with a membrane on the unit square with Dirichlet boundary conditions on all four edges using finite differences and is of the standard form, i.e., $M = I$, where I denotes the identity matrix of appropriate size.

Table 6.2 lists the maximum (worst-case) relative error among all nev approximate eigenvalues returned by RF-DDES. In agreement with the discussion in section 4, exploiting higher values of ψ and/or nev_B leads to enhanced accuracy. Figure 6.1 plots the relative errors among all nev approximate eigenvalues (not just the worst-case errors) for the largest matrix pencil listed in Table 6.1. Since all eigenvalues of "qa8fk/qa8fm" are positive, and $\sigma = 0$, we expect the algebraically smallest eigenvalues of (A, M) to be approximated more accurately. Increasing the value of ψ and/or nev_B then will mainly improve the accuracy in the approximation of those eigenvalues λ located farther away from σ. A similar pattern was also observed for the rest of the matrix pencils listed in Table 6.1.

Table 6.3 lists the number of iterations performed by Algorithm 3.1 as the value of N_c increases. Observe that for matrix pencils 2, 3, 4, and 5 this number can be less than nev (recall the "early termination" property discussed in Proposition 3.1), even for values of N_c as low as $N_c = 2$. Moreover, Figure 6.2 plots the 150 leading[2] singular values of matrix $\Re e\{\sum_{\ell=1}^{N_c} \omega_\ell S(\zeta_\ell)^{-1}\}$ for matrix pencils "bcsst24" and "Kuu/Muu" as $N_c = 4$, 8, 12, and $N_c = 16$. In agreement with the discussion in section 3.2, as the value of N_c increases the magnitude of the trailing $s - \text{rank}([y^{(1)}, \ldots, y^{(nev)}])$ singular values approaches zero.

Except the value of N_c, the number of subdomains p might also affect the number of iterations performed by Algorithm 3.1. Figure 6.3 shows the total number

[2]After normalization by the spectral norm.

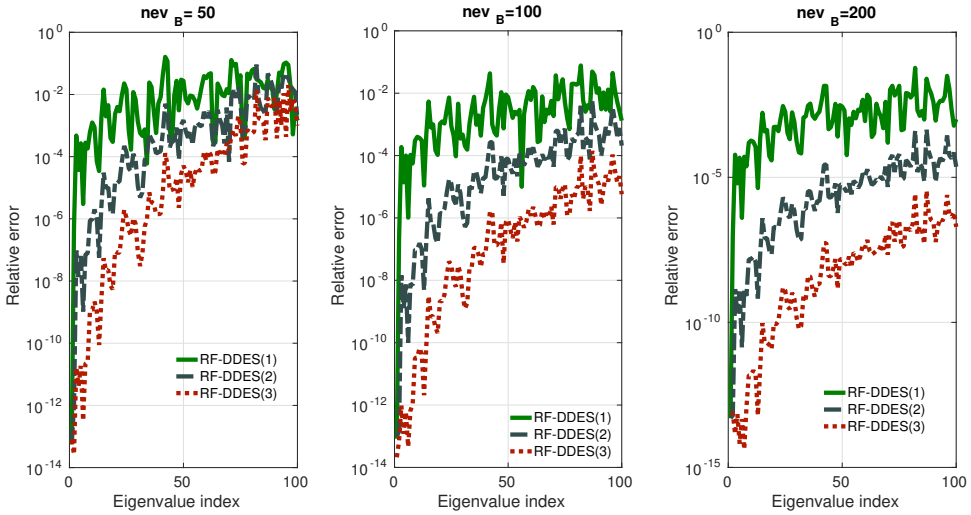


FIG. 6.1. *Relative errors of the approximation of the lowest* $nev = 100$ *eigenvalues for the "qa8fk/qafm" matrix pencil. Left:* $nev_B = 50$. *Center:* $nev_B = 100$. *Right:* $nev_B = 200$.

TABLE 6.3
Number of iterations performed by Algorithm 3.1 *for the matrix pencils listed in Table* 6.1. *"s" denotes the number of interface variables.*

Mat. pencil	s	s/n	$N_c = 2$	$N_c = 4$	$N_c = 8$	$N_c = 12$	$N_c = 16$
bcsst24	449	0.12	164	133	111	106	104
Kuu/Muu	720	0.10	116	74	66	66	66
FDmesh1	300	0.01	58	40	36	35	34
bcsst39	475	0.01	139	93	75	73	72
qa8fk/qa8fm	1272	0.01	221	132	89	86	86

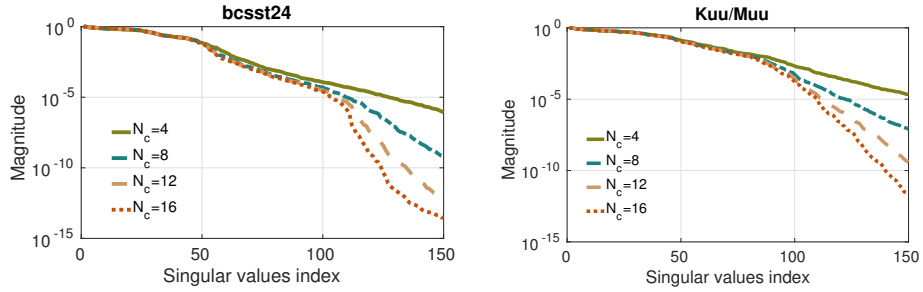


FIG. 6.2. *The* 150 *leading singular values of* $\Re e\{\sum_{\ell=1}^{N_c} \omega_\ell S_{\zeta_\ell}^{-1}\}$ *for the matrix pencils "bcsst24" and "Kuu/Muu".*

of iterations performed by Algorithm 3.1 when applied to matrix "FDmesh1" for $p = 2$, 4, 8, and $p = 16$ subdomains. For each different value of p we considered $N_c = 2$, 4, 8, 12, and $N_c = 16$ quadrature nodes. The interval $[\alpha, \beta]$ was set so that it included only eigenvalues $\lambda_1, \ldots, \lambda_{200}$ ($nev = 200$). Observe that higher values of p might lead to an increase in the number of iterations performed by Algorithm 3.1. For example, when the number of subdomains is set to $p = 2$ or $p = 4$, setting $N_c = 2$ is sufficient for Algorithm 3.1 to terminate in less than nev iterations. On

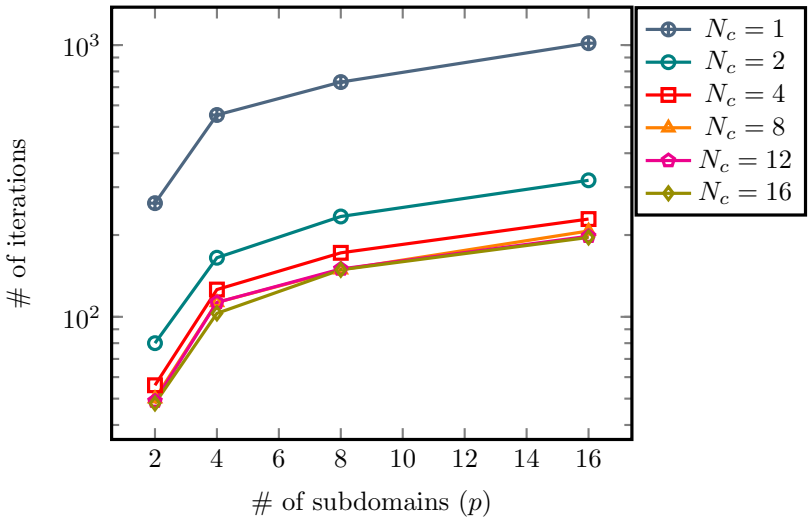


FIG. 6.3. *Total number of iterations performed by Algorithm* 3.1 *when applied to matrix "FDmesh1" with* $[\alpha, \beta] = [\lambda_1, \lambda_{200}]$. *Results reported are for all different combinations of* $p = 2,\ 4,\ 8,$ *and* $p = 16,$ *and* $N_c = 1,\ 2,\ 4,\ 8,$ *and* $N_c = 16$.

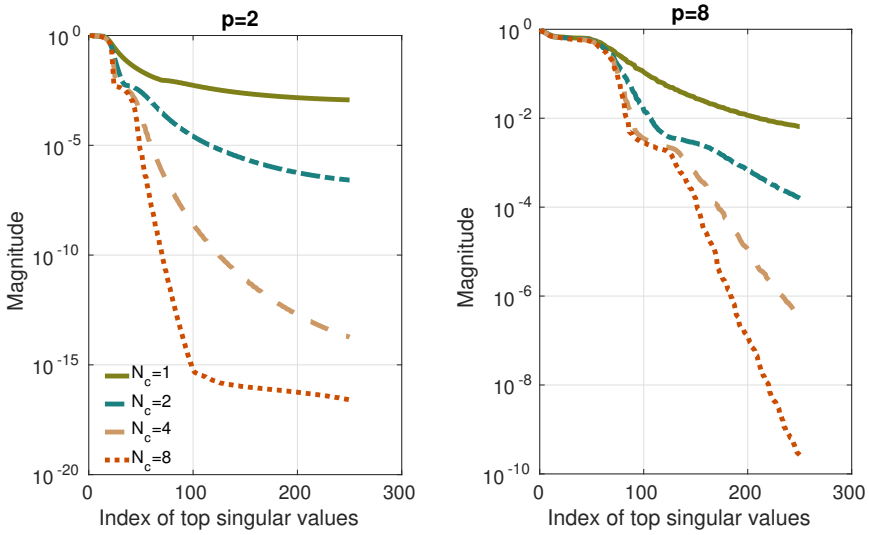


FIG. 6.4. *The leading* 250 *singular values of* $\Re e\{\sum_{\ell=1}^{N_c} \omega_\ell S(\zeta_\ell)^{-1}\}$ *for the same problem as in Figure* 6.3. *Left:* $p = 2$. *Right:* $p = 8$. *For both values of* p *we set* $N_c = 1,\ 2,\ 4,$ *and* $N_c = 8$.

the other hand, when $p \geq 8$, we need at least $N_c \geq 4$ if a similar number of iterations is to be performed. This potential increase in the number of iterations performed by Algorithm 3.1 for larger values of p is a consequence of the fact that the columns of matrix $Y = [y^{(1)}, \ldots, y^{(nev)}]$ now lie in a higher-dimensional subspace. This might not only increase the rank of Y, but also affect the decay of the singular values of $\Re e\{\sum_{\ell=1}^{N_c} \omega_\ell S(\zeta_\ell)^{-1}\}$. This can be seen more clearly in Figure 6.4, where we plot the leading 250 singular values of $\Re e\{\sum_{\ell=1}^{N_c} \omega_\ell S(\zeta_\ell)^{-1}\}$ of the problem in Figure 6.3 for

two different values of p, $p = 2$ and $p = 8$. Notice how the leading singular values decay more slowly for the case $p = 8$. Similar results were observed for different values of p and for all matrix pencils listed in Table 6.1.

6.3. A comparison of RF-DDES and RF-KRYLOV in distributed computing environments. In this section we compare the performance of RF-KRYLOV and RF-DDES on distributed computing environments for the matrices listed in Table 6.4. All eigenvalue problems in this section are of the form (A, I), i.e., standard eigenvalue problems. Matrices "boneS01" and "shipsec8" can be found in the SuiteSparse matrix collection. Similarly to "FDmesh1," matrices "FDmesh2" and "FDmesh3" were generated by a finite differences discretization of the Laplacian operator on the unit plane using Dirichlet boundary conditions and two different mesh sizes so that $n = 250,000$ ("FDmesh2") and $n = 1,000,000$ ("FDmesh3").

Throughout the rest of this section we will keep $N_c = 2$ fixed, since this option was found the best both for RF-KRYLOV and RF-DDES.

6.3.1. Wall-clock time comparisons. We now consider the wall-clock times achieved by RF-KRYLOV and RF-DDES when executing both schemes on $\tau = 2,\ 4,\ 8,\ 16$, and $\tau = 32$ compute cores. For RF-KRYLOV, the value of τ will denote the number of single-threaded MPI processes. For RF-DDES, the number of MPI processes will be equal to the number of subdomains, p, and each MPI process will utilize τ/p compute threads. Unless mentioned otherwise, we will assume that RF-DDES is executed with $\psi = 3$ and $nev_B = 100$.

Table 6.5 lists the number of iterations performed by RF-KRYLOV and Algorithm 3.1 in RF-DDES. For all matrices but "boneS01," Algorithm 3.1 required fewer iterations than RF-KRYLOV. Table 6.6 lists the maximum relative error of the approximate eigenvalues returned by RF-DDES when $p = 4$. The decrease in the accuracy of RF-DDES as nev increases is due the fact that nev_B remains constant. More specifically, an increase in the value of nev should be also accompanied by an increase in the value of nev_B if the same level of maximum relative error need be retained.

TABLE 6.4

n: size of A, $nnz(A)$: number of nonzero entries in matrix A. s_2 and s_4 denote the number of interface variables when $p = 2$ and $p = 4$, respectively.

#	Matrix	n	$nnz(A)/n$	s_2	s_4	$[\lambda_1, \lambda_{101}, \lambda_{201}, \lambda_{300}]$
1.	shipsec8	114,919	28.74	4,534	9,001	[3.2e-2, 1.14e-1, 1.57e-2, 0.20]
2.	boneS01	172,224	32.03	10,018	20,451	[2.8e-3, 24.60, 45.42, 64.43]
3.	FDmesh2	250,000	4.99	1,098	2,218	[7.8e-5, 5.7e-3, 1.08e-2, 1.6e-2]
4.	FDmesh3	1,000,000	4.99	2,196	4,407	[1.97e-5, 1.4e-3, 2.7e-3, 4.0e-3]

TABLE 6.5

Number of iterations performed by RF-KRYLOV (denoted as RFK) and Algorithm 3.1 *in RF-DDES (denoted by RFD(2) and RFD(4), with the number inside the parentheses denoting the value of p) for the matrix pencils listed in Table* 6.4. *The convergence criterion in both RF-KRYLOV and Algorithm* 3.1 *was tested every ten iterations.*

	$nev = 100$			$nev = 200$			$nev = 300$		
Matrix	RFK	RFD(2)	RFD(4)	RFK	RFD(2)	RFD(4)	RFK	RFD(2)	RFD(4)
shipsec8	280	170	180	500	180	280	720	190	290
boneS01	240	350	410	480	520	600	620	640	740
FDmesh2	200	100	170	450	130	230	680	160	270
FDmesh3	280	150	230	460	180	290	690	200	380

TABLE 6.6
Maximum relative error of the approximate eigenvalues returned by RF-DDES for the matrix pencils listed in Table 6.4.

	$nev = 100$			$nev = 200$			$nev = 300$		
Matrix	nev_B=25	nev_B=50	nev_B=100	nev_B=25	nev_B=50	nev_B=100	nev_B=25	nev_B=50	nev_B=100
shipsec8	1.4e-3	2.2e-5	2.4e-6	3.4e-3	1.9e-3	1.3e-5	4.2e-3	1.9e-3	5.6e-4
boneS01	5.2e-3	7.1e-4	2.2e-4	3.8e-3	5.9e-4	4.1e-4	3.4e-3	9.1e-4	5.1e-4
FDmesh2	4.0e-5	2.5e-6	1.9e-7	3.5e-4	9.6e-5	2.6e-6	3.2e-4	2.0e-4	2.6e-5
FDmesh3	6.2e-5	8.5e-6	4.3e-6	6.3e-4	1.1e-4	3.1e-5	9.1e-4	5.3e-4	5.3e-5

TABLE 6.7
Wall-clock times of RF-KRYLOV and RF-DDES using $\tau = 2,\ 4,\ 8,\ 16$, *and* $\tau = 32$ *computational cores. RFD(2) and RFD(4) denote RF-DDES with* $p = 2$ *and* $p = 4$ *subdomains, respectively.*

	$nev = 100$			$nev = 200$			$nev = 300$		
Matrix	RFK	RFD(2)	RFD(4)	RFK	RFD(2)	RFD(4)	RFK	RFD(2)	RFD(4)
shipsec8($\tau = 2$)	114	195	-	195	207	-	279	213	-
($\tau = 4$)	76	129	93	123	133	103	168	139	107
($\tau = 8$)	65	74	56	90	75	62	127	79	68
($\tau = 16$)	40	51	36	66	55	41	92	57	45
($\tau = 32$)	40	36	28	62	41	30	75	43	34
boneS01($\tau = 2$)	94	292	-	194	356	-	260	424	-
($\tau = 4$)	68	182	162	131	230	213	179	277	260
($\tau = 8$)	49	115	113	94	148	152	121	180	187
($\tau = 16$)	44	86	82	80	112	109	93	137	132
($\tau = 32$)	51	66	60	74	86	71	89	105	79
FDmesh2($\tau = 2$)	241	85	-	480	99	-	731	116	-
($\tau = 4$)	159	34	63	305	37	78	473	43	85
($\tau = 8$)	126	22	23	228	24	27	358	27	31
($\tau = 16$)	89	16	15	171	17	18	256	20	21
($\tau = 32$)	51	12	12	94	13	14	138	15	20
FDmesh3($\tau = 2$)	1021	446	-	2062	502	-	3328	564	-
($\tau = 4$)	718	201	281	1281	217	338	1844	237	362
($\tau = 8$)	423	119	111	825	132	126	1250	143	141
($\tau = 16$)	355	70	66	684	77	81	1038	88	93
($\tau = 32$)	177	47	49	343	51	58	706	62	82

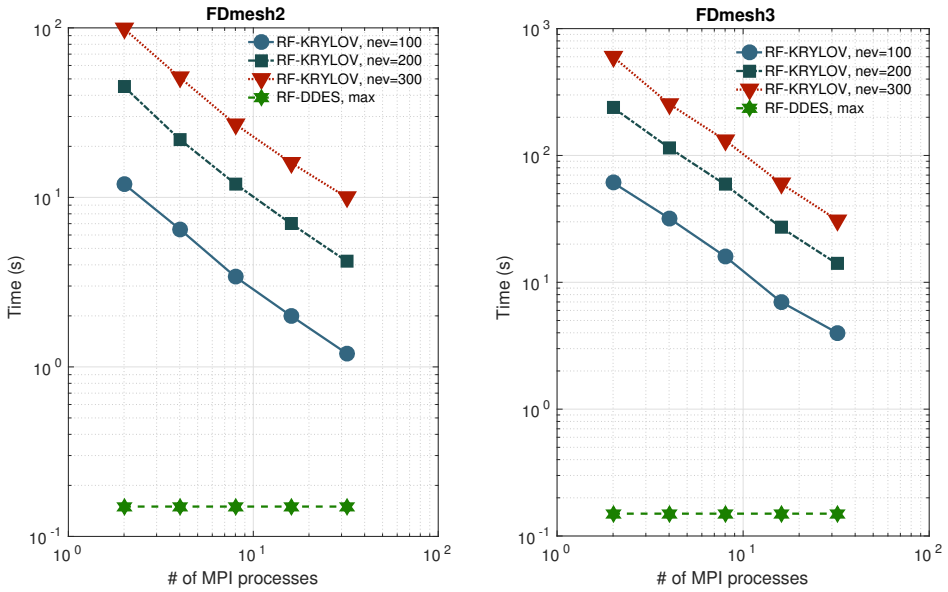
On the other hand, RF-KRYLOV computed all nev eigenpairs within an accuracy which was of the order $O(10^{-13})$ and below for all four matrices considered. Table 6.7 lists the wall-clock time required by RF-KRYLOV and RF-DDES to approximate the $nev = 100$, $nev = 200$, and $nev = 300$ algebraically smallest eigenvalues of the matrices listed in Table 6.4. For RF-DDES we considered two different values of p, $p = 2$ and $p = 4$. Overall, RF-DDES was found to be faster than RF-KRYLOV, especially for higher values of nev.

Table 6.8 lists the amount of time spent on the triangular substitutions required to apply the rational filter in RF-KRYLOV, as well as the amount of time spent on forming and factorizing the Schur complement matrices and applying the rational filter in RF-DDES. For the values of nev tested in this section, these procedures were found to be the computationally most expensive ones. Figure 6.5 plots the total amount of time spent on orthonormalization by RF-KRYLOV and RF-DDES when applied to matrices "FDmesh2" and "FDmesh3." For RF-KRYLOV, we report results for all different values of nev and number of MPI processes. For RF-DDES we only report the highest times across all different values of nev, τ, and p. RF-DDES was found to spend a considerably smaller amount of time on orthonormalization than what RF-KRYLOV did, mainly because s was much smaller than n. (The values of

TABLE 6.8

Time elapsed to apply the rational filter in RF-KRYLOV and RF-DDES using $\tau = 2,\ 4,\ 8,\ 16,$ and $\tau = 32$ compute cores. RFD(2) and RFD(4) denote RF-DDES with $p = 2$ and $p = 4$ subdomains, respectively. For RF-KRYLOV the times listed also include the amount of time spent on factorizing matrices $A - \zeta_\ell M,\ \ell = 1, \ldots, N_c$. For RF-DDES, the times listed also include the amount of time spent in forming and factorizing matrices $S_{\zeta_\ell},\ \ell = 1, \ldots, N_c$.

	$nev = 100$			$nev = 200$			$nev = 300$		
Matrix	RFK	RFD(2)	RFD(4)	RFK	RFD(2)	RFD(4)	RFK	RFD(2)	RFD(4)
shipsec8($\tau = 2$)	104	153	-	166	155	-	222	157	-
($\tau = 4$)	71	93	75	107	96	80	137	96	82
($\tau = 8$)	62	49	43	82	50	45	110	51	47
($\tau = 16$)	38	32	26	61	33	28	83	34	20
($\tau = 32$)	39	21	19	59	23	20	68	24	22
boneS01($\tau = 2$)	86	219	-	172	256	-	202	291	-
($\tau = 4$)	64	125	128	119	152	168	150	178	199
($\tau = 8$)	46	77	88	84	95	117	104	112	140
($\tau = 16$)	43	56	62	75	70	85	86	84	102
($\tau = 32$)	50	42	44	72	51	60	82	63	61
FDmesh2($\tau = 2$)	227	52	-	432	59	-	631	65	-
($\tau = 4$)	152	22	36	287	24	42	426	26	45
($\tau = 8$)	122	13	14	215	14	16	335	15	18
($\tau = 16$)	85	9	8	164	10	10	242	11	11
($\tau = 32$)	50	6	6	90	7	8	127	8	10
FDmesh3($\tau = 2$)	960	320	-	1817	341	-	2717	359	-
($\tau = 4$)	684	158	174	1162	164	192	1582	170	201
($\tau = 8$)	406	88	76	764	91	82	1114	94	88
($\tau = 16$)	347	45	43	656	48	49	976	51	52
($\tau = 32$)	173	28	26	328	28	32	674	31	41



(a) FDmesh2 ($n = 250{,}000$). (b) FDmesh3 ($n = 1{,}000{,}000$).

FIG. 6.5. *Time spent on orthonormalization in RF-KRYLOV and RF-DDES when computing the $nev = 100,\ 200,$ and $nev = 300$ algebraically smallest eigenvalues and associated eigenvectors of matrices "FDmesh2" and "FDmesh3."*

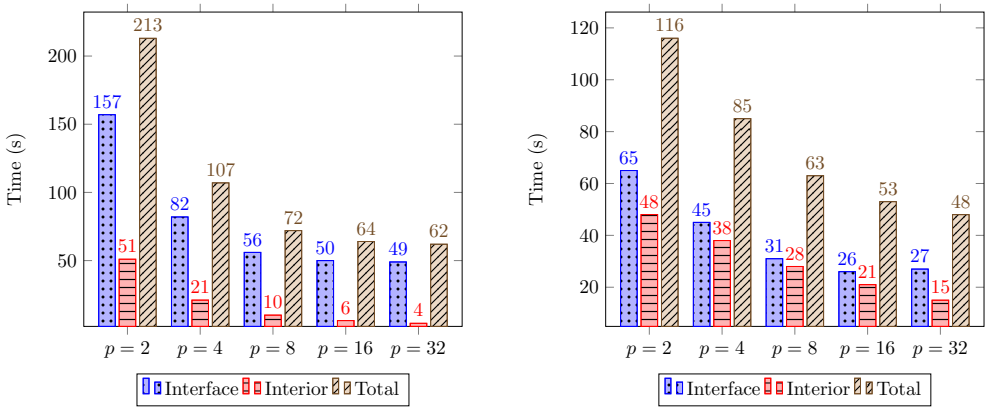


FIG. 6.6. *Amount of time required to apply the rational filter ("Interface"), form the subspace associated with the interior variables ("Interior"), and total wall-clock time ("Total") obtained by an MPI-only execution of RF-DDES for the case where* $nev = 300$. *Left: "shipsec8." Right: "FDmesh2."*

s for $p = 2$ and $p = 4$ can be found in Table 6.4.) Indeed, if both RF-KRYLOV and Algorithm 3.1 in RF-DDES perform a similar number of iterations, we expect the former to spend roughly n/s more time on orthonormalization compared to RF-DDES.

Figure 6.6 lists the wall-clock times achieved by an MPI-only implementation of RF-DDES, i.e., p still denotes the number of subdomains but each subdomain is handled by a separate (single-threaded) MPI process, for matrices "shipsec8" and "FDmesh2." In all cases, the MPI-only implementation of RF-DDES led to higher wall-clock times than those achieved by the hybrid implementations discussed in Tables 6.7 and 6.8. More specifically, while the MPI-only implementation reduced the cost to construct and factorize the distributed S_{ζ_ℓ} matrices, the application of the rational filter in Algorithm 3.1 became more expensive due to the following: (a) each linear system solution with S_{ζ_ℓ} required more time, and (b) a larger number of iterations had to be performed as p increased. (Algorithm 3.1 required 190, 290, 300, 340, and 370 iterations for "shipsec8," and 160, 270, 320, 350, and 410 iterations for "FDmesh2" as $p = 2,\ 4,\ 8,\ 16,$ and $p = 32$, respectively.) Note that the scalability of the MPI-only version of RF-DDES for increasing values of p is limited by the scalability of the linear system used, which in this particular case was not high. This suggests that reducing p and applying RF-DDES recursively to the local pencils $(B_\sigma^{(j)}, M_B^{(j)})$, $j = 1, \ldots, p$, might be the best combination when only distributed memory parallelism is considered.

7. Summary and future work. In this paper we proposed a rational filtering domain decomposition approach (termed as RF-DDES) for the computation of all eigenpairs of real symmetric pencils located inside a given interval $[\alpha, \beta]$. In contrast with rational filtering Krylov approaches, RF-DDES applies the rational filter only to the interface variables. This has several advantages. First, orthogonalization is performed on vectors whose length is equal to the number of interface variables only. Second, the Krylov projection method may converge in fewer than *nev* iterations. Third, it is possible to solve the original eigenvalue problem associated with the interior variables in real arithmetic and with trivial parallelism with respect to each

subdomain. RF-DDES can be considerably faster than rational filtering Krylov approaches, especially when nev is large.

As part of our future work, we aim to extend RF-DDES by considering additional levels of parallelism. In addition to the ability to divide the initial interval $[\alpha, \beta]$ into nonoverlapping subintervals and apply RF-DDES to each one of them in parallel (e.g., see [22, 25]), we can also assign the complex linear system solutions associated with different quadrature nodes to distinct processor groups. Moreover, these linear systems can be solved by distributed preconditioned Krylov subspace approaches which could be helpful when RF-DDES is applied to the solution of symmetric eigenvalue problems arising from discretization of three-dimensional domains. To this end, one option we are currently exploring is to combine RF-DDES with the distributed memory extension of the work in [13].

Another interesting approach would also be to explore recursive implementations of RF-DDES. For example, RF-DDES could be applied individually to each matrix pencil $(B_\sigma^{(j)}, M_B^{(j)})$, $j = 1, \ldots, p$. This could be particularly helpful when either d_j, the number of interior variables of the jth subdomain, or $nev_B^{(j)}$, are large. On the algorithmic side, it would be of interest to develop more efficient criteria to set the value of $nev_B^{(j)}$, $j = 1, \ldots, p$ in each subdomain, perhaps by adapting the work in [41].

Acknowledgments. The authors acknowledge the Minnesota Supercomputing Institute (MSI; http://www.msi.umn.edu) at the University of Minnesota for providing resources that contributed to the research results reported within this paper. Finally, the authors are grateful to the referees for their careful reading and comments, which helped to improve the quality of the paper.

REFERENCES

[1] *Intel Math Kernel Library. Reference Manual*, Intel, Santa Clara, CA, 2009.

[2] M. Abramowitz, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover, New York, 1974.

[3] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster, *A fully asynchronous multifrontal solver using distributed dynamic scheduling*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 15–41, https://doi.org/10.1137/S0895479899358194.

[4] A. L. S. Andrew V. Knyazev, *Preconditioned gradient-type iterative methods in a subspace for partial generalized symmetric eigenvalue problems*, SIAM J. Numer. Anal., 31 (1994), pp. 1226–1239.

[5] A. P. Austin and L. N. Trefethen, *Computing eigenvalues of real symmetric matrices with rational filters in real arithmetic*, SIAM J. Sci. Comput., 37 (2015), pp. A1365–A1387, https://doi.org/10.1137/140984129.

[6] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *Efficient management of parallelism in object oriented numerical software libraries*, in Modern Software Tools in Scientific Computing, E. Arge, A. M. Bruaset, and H. P. Langtangen, eds., Birkhäuser, Basel 1997, pp. 163–202.

[7] M. V. Barel, *Designing rational filter functions for solving eigenvalue problems by contour integration*, Linear Algebra Appl., 502 (2016), pp. 346–365, http://dx.doi.org/10.1016/j.laa.2015.05.029.

[8] M. V. Barel and P. Kravanja, *Nonlinear eigenvalue problems and contour integrals*, J. Comput. Appl. Math., 292 (2016), pp. 526–540, http://dx.doi.org/10.1016/j.cam.2015.07.012.

[9] C. Bekas and Y. Saad, *Computation of smallest eigenvalues using spectral Schur complements*, SIAM J. Sci. Comput., 27 (2006), pp. 458–481.

[10] J. K. Bennighof and R. B. Lehoucq, *An automated multilevel substructuring method for eigenspace computation in linear elastodynamics*, SIAM J. Sci. Comput., 25 (2004), pp. 2084–2106.

[11] L. S. Blackford, J. Choi, A. Cleary, E. D'Azeuedo, J. Demmel, I. Dhillon, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley, *ScaLAPACK User's Guide*, SIAM Philadelphia, 1997.

[12] T. A. Davis and Y. Hu, *The university of Florida sparse matrix collection*, ACM Trans. Math. Software, 38 (2011), pp. 1:1–1:25, https://doi.org/10.1145/2049662.2049663.
[13] G. Dillon, V. Kalantzis, Y. Xi, and Y. Saad, *A hierarchical low-rank Schur complement preconditioner for indefinite linear systems*, SIAM J. Sci. Comput., to appear.
[14] S. B. et al., *PETSc Users Manual*, Technical report ANL-95/11—Revision 3.6, Argonne National Laboratory, 2015, http://www.mcs.anl.gov/petsc.
[15] S. B. et al., *PETSc*, http://www.mcs.anl.gov/petsc, 2015.
[16] W. Gao, X. S. Li, C. Yang, and Z. Bai, *An implementation and evaluation of the AMLS method for sparse eigenvalue problems*, ACM Trans. Math. Software, 34 (2008), pp. 20:1–20:28, https://doi.org/10.1145/1377596.1377600.
[17] R. G. Grimes, J. G. Lewis, and H. D. Simon, *A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 228–272, https://doi.org/10.1137/S0895479888151111.
[18] S. Güttel, E. Polizzi, P. T. P. Tang, and G. Viaud, *Zolotarev quadrature rules and load balancing for the FEAST eigensolver*, SIAM J. Sci. Comput., 37 (2015), pp. A2100–A2122, https://doi.org/10.1137/140980090.
[19] V. Kalantzis, J. Kestyn, E. Polizzi, and Y. Saad, *Domain decomposition approaches for accelerating contour integration eigenvalue solvers for symmetric eigenvalue problems*, Numer. Linear Algebra Appl. (2018), e2154, https://doi.org/10.1002/nla.2154.
[20] V. Kalantzis, R. Li, and Y. Saad, *Spectral Schur complement techniques for symmetric eigenvalue problems*, Electron. Trans. Numer. Anal., 45 (2016), pp. 305–329.
[21] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392, https://doi.org/10.1137/S1064827595287997.
[22] J. Kestyn, V. Kalantzis, E. Polizzi, and Y. Saad, *PFEAST: A high performance sparse eigenvalue solver using distributed-memory linear solvers*, in Proceedings of the ACM/IEEE Supercomputing Conference (SC16), 2016.
[23] J. Kestyn, E. Polizzi, and P. T. P. Tang, *FEAST eigensolver for non-Hermitian problems*, SIAM J. Sci. Comput., 38 (2016), pp. S772–S799, https://doi.org/10.1137/15M1026572.
[24] L. Komzsik and T. Rose, *Substructuring in MSC/NASTRAN for large scale parallel applications*, Comput. Syst. Eng., 2 (1991), pp. 167–173.
[25] R. Li, Y. Xi, E. Vecharynski, C. Yang, and Y. Saad, *A thick-restart Lanczos algorithm with polynomial filtering for Hermitian eigenvalue problems*, SIAM J. Sci. Comput., 38 (2016), pp. A2512–A2534, https://doi.org/10.1137/15M1054493.
[26] S. Lui, *Kron's method for symmetric eigenvalue problems*, J. Comput. Appl. Math., 98 (1998), pp. 35–48, http://dx.doi.org/10.1016/S0377-0427(98)00110-1.
[27] S. Lui, *Domain decomposition methods for eigenvalue problems*, J. Comput. Appl. Math., 117 (2000), pp. 17–34, http://dx.doi.org/10.1016/S0377-0427(99)00326-X.
[28] F. Pellegrini, Scotch *and* libScotch 5.1 *User's Guide*, INRIA Bordeaux Sud-Ouest, IPB & LaBRI, UMR CNRS 5800, 2010, http://gforge.inria.fr/docman/view.php/248/7104/scotch_user5.1.pdf.
[29] B. Philippe and Y. Saad, *On correction equations and domain decomposition for computing invariant subspaces*, Comput. Methods Appl. Mech. Engrg., 196 (2007), pp. 1471–1483, http://dx.doi.org/10.1016/j.cma.2006.03.026.
[30] E. Polizzi, *Density-matrix-based algorithm for solving eigenvalue problems*, Phys. Rev. B, 79 (2009), pp. 115–112, https://doi.org/10.1103/PhysRevB.79.115112.
[31] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, SIAM, Philadelphia, 2011, https://doi.org/10.1137/1.9781611970739.
[32] T. Sakurai and H. Sugiura, *A projection method for generalized eigenvalue problems using numerical integration*, J. Comput. Appl. Math., 159 (2003), pp. 119–128, http://dx.doi.org/10.1016/S0377-0427(03)00565-X.
[33] H. D. Simon, *The Lanczos algorithm with partial reorthogonalization*, Math. Comp., 42 (1984), pp. 115–142.
[34] B. F. Smith, P. E. Bjørstad, and W. D. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, New York, 1996.
[35] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, *MPI—The Complete Reference, Volume* 1: *The MPI Core*, 2nd ed., MIT Press, Cambridge, MA, 1998.
[36] A. Stathopoulos and J. R. McCombs, *PRIMME: Preconditioned iterative multimethod eigensolver: Methods and software description*, ACM Trans. Math. Software, 37 (2010), pp. 21:1–21:30, https://doi.org/10.1145/1731022.1731031.
[37] P. T. P. Tang and E. Polizzi, *FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 354–390, https://doi.org/10.1137/13090866X.

[38] A. Toselli and O. Widlund, *Domain Decomposition Methods: Algorithms and Theory*, Vol. 3, Springer, New York, 2005.

[39] J. Winkelmann and E. Di Napoli, *Non-Linear Least-Squares Optimization of Rational Filters for the Solution of Interior Eigenvalue Problems*, preprint, arXiv:1704.03255, 2017.

[40] Y. Xi and Y. Saad, *Computing partial spectra with least-squares rational filters*, SIAM J. Sci. Comput., 38 (2016), pp. A3020–A3045, https://doi.org/10.1137/16M1061965.

[41] C. Yang, W. Gao, Z. Bai, X. S. Li, L.-Q. Lee, P. Husbands, and E. Ng, *An algebraic substructuring method for large-scale eigenvalue calculation*, SIAM J. Sci. Comput., 27 (2005), pp. 873–892, https://doi.org/10.1137/040613767.

[42] X. Ye, J. Xia, R. H. Chan, S. Cauley, and V. Balakrishnan, *A fast contour-integral eigensolver for non-Hermitian matrices*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 1268–1297, https://doi.org/10.1137/16M1086601.