A Domain Decomposition Rayleigh-Ritz Algorithm for Symmetric Generalized Eigenvalue Problems

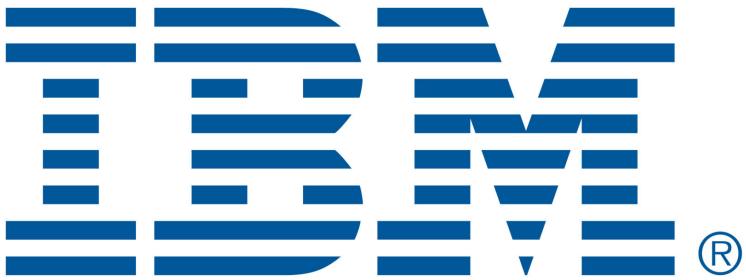
Vassilis Kalantzis

September 2020

EPrint ID: 2020.2

IBM Research
Thomas J. Watson Research Center

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



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

A DOMAIN DECOMPOSITION RAYLEIGH–RITZ ALGORITHM FOR SYMMETRIC GENERALIZED EIGENVALUE PROBLEMS*

VASSILIS KALANTZIS†

Abstract. This paper proposes a parallel domain decomposition Rayleigh–Ritz projection scheme to compute a selected number of eigenvalues (and, optionally, associated eigenvectors) of large and sparse symmetric pencils. The projection subspace associated with interface variables is built by computing a few of the eigenvectors and associated leading derivatives of a zeroth-order approximation of the nonlinear matrix-valued interface operator. On the other hand, the projection subspace associated with interior variables is built independently in each subdomain by exploiting local eigenmodes and matrix resolvent approximations. The sought eigenpairs are then approximated by a Rayleigh–Ritz projection onto the subspace formed by the union of these two subspaces. Several theoretical and practical details are discussed, and upper bounds of the approximation errors are provided. Our numerical experiments demonstrate the efficiency of the proposed technique on sequential/distributed memory architectures as well as its competitiveness against schemes such as shift-and-invert Lanczos and automated multilevel substructuring combined with p-way vertex-based partitionings.

Key words. symmetric generalized eigenvalue problem, domain decomposition, high-performance computing, spectral Schur complement, Rayleigh–Ritz

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

DOI. 10.1137/19M1280004

1. Introduction.

This paper proposes a Rayleigh–Ritz projection scheme based on algebraic domain decomposition to compute eigenvalues (and, optionally, associated eigenvectors) of large and sparse symmetric matrix pencils. In particular, our focus lies in the computation of a large number of eigenvalues located immediately on the right of some real scalar. Eigenvalue problems of this form appear in applications such as low-frequency response analysis [16, 31] and spectral clustering [39], among others. Extensions to the case where the sought eigenvalues are located immediately on the left of some real scalar are straightforward.

Computing eigenvalues located the closest to a given scalar is typically achieved by enhancing the projection method of choice by shift-and-invert [13, 41]. When the required accuracy in the approximation of the sought eigenvalues is not high, an alternative to Krylov subspace techniques is the automated multilevel substructuring technique (AMLS) [6, 7, 10, 28]. From an algebraic perspective, AMLS performs a Rayleigh–Ritz projection on a large projection subspace while avoiding excessive orthogonalization costs and has been demonstrated as a superior alternative to shift-and-invert Krylov subspace techniques when a very large number of eigenvalues is sought [29]. An analysis of the AMLS algorithm for elliptic PDE problems from a variational viewpoint can be found in [7]. Therein, tools from the mathematical theory of substructuring domain decomposition for elliptic PDEs are considered so as to understand how AMLS scales as the mesh is refined.

*Submitted to the journal's Software and High-Performance Computing section August 8, 2019; accepted for publication (in revised form) September 29, 2020; published electronically DATE.
https://doi.org/10.1137/19M1280004

Funding: This work was supported by the International Business Machines Corporation through its Herman H. Goldstine Postdoctoral Fellowship program and by the Minnesota Supercomputing Institute (MSI) at the University of Minnesota.

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

This paper focuses on algebraic domain decomposition eigenvalue solvers where the concept of domain decomposition is applied directly to the eigenvalue equation. Algebraic domain decomposition eigenvalue solvers start by calling an algebraic graph partitioner to partition the graph associated with the given matrix pencil into a number of nonoverlapping subdomains. The variables within each subdomain are then classified into two different categories: (a) (interior) variables which are coupled with variables located only in the same subdomain, and (b) (interface) variables which are both coupled with local variables and variables located in neighboring subdomains. The sought eigenpairs are then approximated by a Rayleigh–Ritz projection onto a subspace formed by the combination of subspaces associated with the two different types of variables. An in-depth analysis of algebraic domain decomposition eigenvalue solvers can be found in [20].

The main challenge of algebraic domain decomposition eigenvalue solvers is the construction of the projection subspace associated with interface variables due to the nonlinear nature of the interface matrix operator, also known as "spectral Schur complement" [21, 23]. From a purely algebraic perspective, AMLS builds this subspace by computing a few of the eigenvectors of a linear generalized eigenvalue problem involving the Schur complement matrix and its first derivative [6]. As a result, the accuracy provided by AMLS can deteriorate considerably as we look to compute eigenvalues located away from the origin. An alternative suggested recently is to form the projection subspace associated with interface variables by applying a complex rational transformation to the original pencil so as to annihilate components associated with unwanted eigenvalues, e.g., see the RF-DDES algorithm [24] and the discussion in [22, 27]. While these techniques can indeed lead to enhanced accuracy, multiple matrix factorizations computed in complex arithmetic are required.

In this paper we propose an algorithm which preserves advantages of algebraic domain decomposition eigenvalue solvers such as reduced orthogonalization costs and inherent parallelism while, at the same time, increases their accuracy without considering more than one shift.

The key characteristics of the proposed scheme are summarized below.

(1) *Zeroth-order truncation of the interface matrix operator.* In contrast to AMLS, the algorithm proposed in this paper generates the projection subspace associated with interface variables by solving partially a standard eigenvalue problem with the Schur complement matrix. This approach avoids the need to compute/apply the derivative of the spectral Schur complement.

(2) *Exploiting Taylor series expansions of interface eigenvectors.* The accuracy of the projection subspace associated with interface variables is enhanced by expanding the (analytic) eigenvectors of the spectral Schur complement through their Taylor series and injecting a few leading eigenvector derivatives into the subspace. We show theoretically (and verify experimentally) that injecting up to second-order eigenvector derivatives leads to eigenvalue approximations for which the upper bound of the absolute error reduces quartically. These eigenvector derivatives are computed independently of each other by exploiting deflated Krylov subspace solvers.

(3) *Reduced orthogonalization costs and enhanced parallelism.* Similarly to domain decomposition schemes such as AMLS and RF-DDES, orthonormalization is applied only to vectors whose length is equal to the number of interface variables instead of the entire set of domain variables. This becomes especially important when both a large number of eigenvalues is sought and the number of interface variables is much smaller compared to the total number of equations/unknowns. In addition, the projection subspaces associated with interior variables in each subdomain are

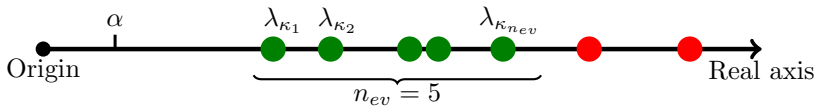


FIG. 1.1. *An illustration of the eigenvalue problem considered in this paper. Our goal is to compute the $n_{ev} = 5$ (real) eigenvalues located immediately on the right of $\alpha \in \mathbb{R}$. The sought eigenvalues are denoted by $\lambda_{\kappa_1}, \ldots, \lambda_{\kappa_{n_{ev}}}$, $1 \le \kappa_1 \le \kappa_{n_{ev}} \le n$.*

built independently of each other by computing local eigenmodes and computing up to second-order resolvent expansions. We report experiments performed on sequential/distributed memory architectures and demonstrate the suitability of the proposed technique in high-performance computing settings.

1.1. Notation and outline. The eigenvalue problem considered in this paper is of the form $Ax = \lambda Mx$ where the matrices $A \in \mathbb{R}^{n\times n}$ and $M \in \mathbb{R}^{n\times n}$ are assumed large, sparse, and symmetric, and matrix M is also positive-definite (SPD). Our goal is to compute the $n_{ev} \ll n$ eigenvalues located immediately on the right of a user-given scalar α. An illustrative example is shown in Figure 1.1. Extensions of the proposed technique to the computation of eigenvalues located immediately on the left of a user-given scalar are straightforward. Throughout the rest of this paper we will denote the pencil $L - \lambda K$ by (L, K), and for any such pencil we define $\Lambda(L, K) := \{\lambda \mid \mathbf{det}[L - \lambda K] = 0\}$.

The outline of this paper is as follows. Section 2 gives a brief introduction to Rayleigh–Ritz projection subspaces from the viewpoint of algebraic domain decomposition. Section 3 presents a spectral Schur complement approach to approximate a single eigenpair of the pencil (A, M). Section 4 extends these results to a Rayleigh–Ritz projection that approximates all sought eigenpairs from a common projection subspace. Section 5 presents numerical experiments performed on sequential and distributed memory environments. Finally, in section 6 we give our concluding remarks.

2. Rayleigh–Ritz projections from a domain decomposition viewpoint. The standard approach to compute a few eigenpairs of sparse and symmetric matrix pencils is by applying the Rayleigh–Ritz technique onto a carefully constructed subspace $\mathcal{Z}$ of $\mathbb{R}^n$ [33]. The goal is to find a subspace $\mathcal{Z}$ that includes an invariant subspace associated with the n_{ev} sought eigenvalues $\lambda_{\kappa_1}, \ldots, \lambda_{\kappa_{n_{ev}}}$. The sought eigenpairs can then be recovered (in the absence or roundoff errors) as a subset of the Ritz pairs of the matrix pencil $(Z^T AZ, Z^T MZ)$, where matrix Z represents a basis of $\mathcal{Z}$.

Let the matrices A and M be partitioned in a 2×2 block form

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

where B and M_B are square matrices of size $d \times d$, E and M_E are rectangular matrices of size $d \times s$, C and M_C are square matrices of size $s \times s$, and $n = d + s$. Without loss of generality we assume $n_{ev} \le s$. Similarly, the eigenvector $x^{(i)}$ associated with eigenvalue λ_i can be written as

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

Rewriting $Ax^{(i)} = \lambda_i Mx^{(i)}$ using the above block form gives

$$\begin{pmatrix} B-\lambda_i M_B & E-\lambda_i M_E \\ E^T-\lambda_i M_E^T & C-\lambda_i M_C \end{pmatrix} \begin{pmatrix} u^{(i)} \\ y^{(i)} \end{pmatrix} = 0, \tag{2.1}$$

while eliminating $u^{(i)}$ from the second row equation in (2.1) leads to the $s \times s$ nonlinear eigenvalue problem

$$\Big[C-\lambda_i M_C - (E-\lambda_i M_E)^T (B-\lambda_i M_B)^{-1}(E-\lambda_i M_E)\Big] y^{(i)} = 0, \tag{2.2}$$

from which λ_i and $y^{(i)}$ can be determined. The missing $d \times 1$ part of $x^{(i)}$, $u^{(i)}$, is then recovered by solving the linear system

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

From a domain decomposition perspective, an ideal choice is to set $\mathcal{Z} = \mathcal{U} \oplus \mathcal{Y}$ where

$$\mathcal{U} = \mathbf{span}\left(\begin{bmatrix} u^{(\kappa_1)}, \ldots, u^{(\kappa_{n_{ev}})} \\ 0_{s,n_{ev}} \end{bmatrix} \right), \quad \mathcal{Y} = \mathbf{span}\left(\begin{bmatrix} 0_{d,n_{ev}} \\ y^{(\kappa_1)}, \ldots, y^{(\kappa_{n_{ev}})} \end{bmatrix} \right), \tag{2.3}$$

and $0_{\chi,\psi}$ denotes the zero matrix of size $\chi \times \psi$.

The main goal of algebraic domain decomposition eigenvalue solvers is to build a projection subspace which, ideally, includes the subspace in (2.3).

2.1. Domain decomposition reordering. Practical applications of algebraic domain decomposition eigenvalue solvers rely on relabeling the unknowns/equations of the eigenvalue problem $Ax = \lambda Mx$ such that the matrix $B - \lambda M_B$ is block-diagonal. This can be easily achieved by applying a graph partitioner to the adjacency graph of the matrix $|A| + |M|$, e.g., [25, 34].

In this paper we only consider p-way partitionings, and the partitioner divides the graph into $p > 1$ nonoverlapping subdomains. The rows/columns of matrices A and M are then reordered so that unknowns/equations associated with interior variables (i.e., nodes of the adjacency graph which are connected only to nodes located in the same partition) are listed before those associated with interface variables (i.e., nodes of the adjacency graph which are connected with nodes located in neighboring partitions). The permuted matrices A and M can be written[1] as

$$A := PAP^T = \begin{pmatrix} B_1 & & & E_1 \\ & B_2 & & E_2 \\ & & \ddots & \vdots \\ & & B_p & E_p \\ E_1^T & E_2^T & \ldots & E_p^T \quad C \end{pmatrix}, \quad \text{and}$$

$$M := PMP^T = \begin{pmatrix} M_B^{(1)} & & & & M_E^{(1)} \\ & M_B^{(2)} & & & M_E^{(2)} \\ & & \ddots & & \vdots \\ & & & M_B^{(p)} & M_E^{(p)} \\ \left(M_E^{(1)}\right)^T & \left(M_E^{(2)}\right)^T & \ldots & \left(M_E^{(p)}\right)^T & M_C \end{pmatrix}. \tag{2.4}$$

[1] The eigenvalues of the pencil (A, M) in (2.4) are identical to those prior to the symmetric permutation. If eigenvectors are also of interest, these need be postmultiplied by P^T. Throughout the rest of this paper we will work with the reordered matrices shown in (2.4).

Let us denote the number of interior and interface variables residing in the jth subdomain by d_j and s_j, respectively. Matrices B_j and $M_B^{(j)}$ are of size $d_j \times d_j$, while E_j and $M_E^{(j)}$ are rectangular matrices of size $d_j \times s$ where $s = \sum_{j=1}^{p} s_j$. In particular, E_j and $M_E^{(j)}$ have a special nonzero pattern of the form $E_j = [0_{d_j,\ell_j}, \hat{E}_j, 0_{d_j,\xi_j}]$, and $M_E^{(j)} = [0_{d_j,\ell_j}, \hat{M}_E^{(j)}, 0_{d_j,\xi_j}]$, where $\ell_j = \sum_{k=1}^{k<j} s_k$, and $\xi_j = \sum_{k>j}^{k=p} s_k$. Note that typically the value of p is chosen such that $s \ll d$.

Taking advantage of the Schur complements framework, the vectors $u^{(i)}$ and $y^{(i)}$ can be further partitioned as

$$u^{(i)} = \begin{pmatrix} u_1^{(i)} \\ \vdots \\ u_p^{(i)} \end{pmatrix} \quad \text{and} \quad y^{(i)} = \begin{pmatrix} y_1^{(i)} \\ \vdots \\ y_p^{(i)} \end{pmatrix},$$

where $u_j^{(i)} \in \mathbb{R}^{d_j}$ and $y_j^{(i)} \in \mathbb{R}^{s_j}$ denote the components of vectors $u^{(i)}$ and $y^{(i)}$ which are associated with the jth subdomain, respectively. Once the subvector $y_j^{(i)}$ becomes available, $u_j^{(i)}$ is computed independently of the rest of the subvectors of $u^{(i)}$ by solving the local linear system

$$\left(B_j - \lambda_i M_B^{(j)}\right) u_j^{(i)} = -\left(\hat{E}_j - \lambda_i \hat{M}_E^{(j)}\right) y_j^{(i)}.$$

3. A scheme to approximate a single eigenpair.

This section considers the approximation of a single eigenpair (λ, x) by $(\hat{\lambda}, \hat{x})$, where $\hat{\lambda} = \frac{\hat{x}^T A \hat{x}}{\hat{x}^T M \hat{x}}$ is the Rayleigh quotient associated with the approximate eigenvector $\hat{x}$.

3.1. Spectral Schur complements.

We begin by defining the univariate, nonlinear, matrix-valued function

$$S: \ \zeta \in \mathbb{R} \to S(\zeta) \in \mathbb{R}^{s\times s}, \quad S(\zeta) = C - \zeta M_C - (E - \zeta M_E)^T (B - \zeta M_B)^{-1} (E - \zeta M_E).$$

For each $\zeta \in \mathbb{R} \backslash \Lambda(B, M_B)$, there exist s real eigenvalues and corresponding orthogonal eigenvectors of $S(\zeta)$. When $\zeta \in \Lambda(B, M_B)$, there exist at least $s - d$ (if $s \geq d$) eigenvalues of $S(\zeta)$ which are well-defined (see also section 3.2).

DEFINITION 3.1. *For $j = 1, \ldots, s$, we define the scalar-vector pairs*

$$(\mu_j, y_j): \quad \zeta \in \mathbb{R} \backslash \mathcal{D}_j \ \longrightarrow \ (\mu_j(\zeta), y_j(\zeta)) \in \{\mathbb{R}, \mathbb{R}^s\}, \quad \text{where} \ \ \mathcal{D}_j \subseteq \Lambda(B, M_B),$$

such that for any $\zeta \notin \mathcal{D}_j$, the pair $(\mu_j(\zeta), y_j(\zeta))$ satisfies

$$S(\zeta) y_j(\zeta) = \mu_j(\zeta) y_j(\zeta).$$

Each function $\mu_j(\zeta)$ (from now on referred to as "eigenvalue curve") has either no or a finite number of poles $\mathcal{D}_j$, and these poles are eigenvalues of the pencil (B, M_B). Moreover, the corresponding eigenvector $y_j(\zeta)$ is uniquely defined (up to a normalizing factor) and the associated spectral projector is analytic [4, 26]. Throughout the rest of this paper we assume that the spectrum of $S(\zeta)$ is implicitly arranged so that the eigenvalue curves $\mu_1(\zeta), \ldots, \mu_s(\zeta)$ are analytic functions of ζ everywhere within their domain of definition.[2] This assumption corresponds to a mere reordering of the eigenpairs of $S(\zeta)$ and does not affect the practicality of the algorithm proposed throughout this paper nor require any additional work.

[2] See also the discussion in [38, section 4].

Remark 1. One idea to order the subscripts of the eigenvalue curves is to denote by $\mu_j(\zeta)$ the eigenvalue curve for which $\mu_j(\zeta_0),\ \zeta_0 \in \mathbb{R}\backslash\Lambda(B, M_B)$ is equal to the jth algebraically smallest eigenvalue of matrix $S(\zeta_0)$. Throughout this paper we denote by $\mu_j(\zeta)$ the eigenvalue curve for which $\mu_j(0)$ is equal to the jth algebraically smallest eigenvalue of matrix $S(0)$.

3.2. Behavior of eigenvalue curves at their poles. In general, it is not possible to determine what eigenvalues (if any) of (B, M_B) are poles of $\mu_j(\zeta)$, nor does the algorithm proposed in this paper require such information. Nonetheless, we can determine the number of eigenvalue curves that remain analytic as ζ approaches an eigenvalue of (B, M_B).

DEFINITION 3.2. *The eigenpairs of the matrix pencil (B, M_B) will be denoted by $(\delta_\ell, v^{(\ell)}),\ \ell = 1, \ldots, d$, where $V = [v^{(1)}, v^{(2)}, \ldots, v^{(d)}]$ is scaled so that $V^T M_B V = I$.*

Let $w_\zeta^{(\ell)} = (E - \zeta M_E)^T v^{(\ell)},\ \ell = 1, \ldots, d$, and assume that δ_k is an eigenvalue of (B, M_B) with multiplicity $\rho_\delta \leq s$, and corresponding eigenvectors $v_k^{(1)}, \ldots, v_k^{(\rho_\delta)}$. Then, if $\mathbf{rank}(\mathbf{lim}_{\zeta\to\delta_k}(E - \zeta M_E)^T[v_k^{(1)}, \ldots, v_k^{(\rho_\delta)}]) = \theta \leq \rho_\delta$, there exist integers $j_1, \ldots, j_\theta \in \{1, \ldots, s\}$ such that

$$\begin{cases} \mathbf{lim}_{\zeta\to\delta_k}\, \mu_{\{j_1,\ldots,j_\theta\}}(\zeta) = -\infty, & \text{when } \ \zeta < \delta_k, \ \text{ and} \\ \mathbf{lim}_{\zeta\to\delta_k}\, \mu_{\{j_1,\ldots,j_\theta\}}(\zeta) = +\infty, & \text{when } \ \zeta > \delta_k. \end{cases}$$

As $\zeta \to \delta_k$, all but the above eigenvalue curves cross δ_k in an analytical manner. The eigenvalue curves are strictly decreasing in their entire domain of definition. Details on the above discussion can be found in [23, Theorem 4.1] and [30, Theorem 2.1] when $M = I$. Extensions to the case $M \neq I$ are straightforward.

3.3. Taylor series expansion of $y_j(\lambda)$. Following (2.2), a scalar $\lambda \notin \Lambda(B, M_B)$ is an eigenvalue[3] of the matrix pencil (A, M) if and only if there exists an integer $1 \leq j \leq s$ such that $\mu_j(\lambda) = 0$ (e.g., see Figure 3.1). The eigenvector $y_j(\lambda)$ associated with the eigenvalue $\mu_j(\lambda)$ is then equal to the bottom $s \times 1$ subvector of eigenvector x associated with λ. Therefore, computing $y_j(\lambda)$ is the first step toward approximating the eigenvector x.

Let now $\sigma \in [\lambda^-, \lambda^+]$ be an approximation of λ, where $[\lambda^-, \lambda^+]$ is located between two consecutive poles of $\mu_j(\zeta)$ (if such poles exist).[4] In the ideal scenario, we have $\sigma \equiv \lambda$, which leads to $y_j(\sigma) = y_j(\lambda)$. In practice, we can only hope that $\sigma \approx \lambda$, and thus $y_j(\sigma)$ is only an approximation of $y_j(\lambda)$. To improve this approximation, we exploit the analyticity of the eigenpair $(\mu_j(\zeta), y_j(\zeta))$.

Let $\frac{d^i y_j(\zeta)}{d\zeta^i}$ denote the ith derivative of the univariate vector-valued function $y_j(\zeta)$. Expanding $y_j(\lambda)$ through its Taylor series around σ gives

$$y_j(\lambda) = \sum_{i=0}^{\infty} \frac{(\lambda - \sigma)^i}{i!} \left(\frac{d^i y_j(\zeta)}{d\zeta^i} \right)_{\zeta=\sigma}. \tag{3.1}$$

The above expression suggests that even when σ is not very close to λ, we can improve the approximation of $y_j(\lambda)$ by considering higher-order derivatives of $y_j(\zeta)$ evaluated at σ.

[3] The case where $\lambda \in \Lambda(B, M_B)$ is more involved as it is possible that $\mathbf{det}\,[S(\lambda)] \neq 0$. Nonetheless, such scenarios can be easily detected, e.g., see [30].

[4] In practice a pole of $\mu_j(\zeta)$ located inside the interval $[\lambda^-, \lambda^+]$ poses no threat as long as σ is chosen carefully.

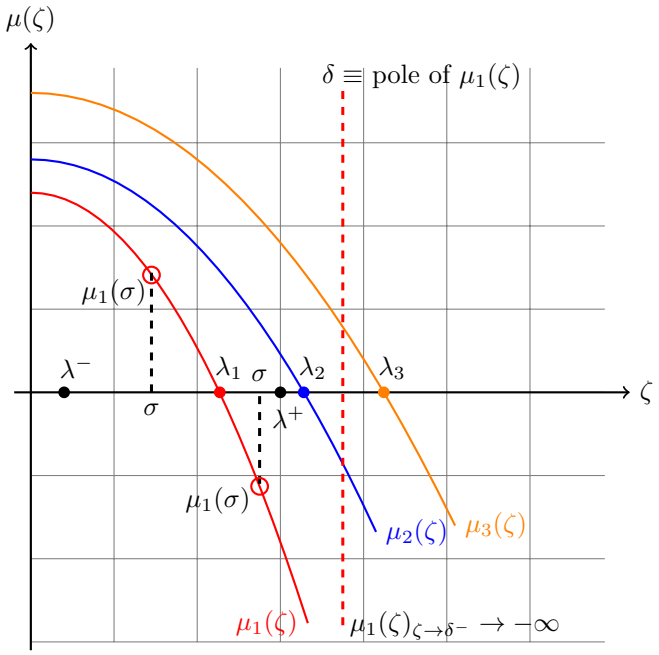


FIG. 3.1. *Illustration of the concept of eigenvalue curves. We assume that $\lambda \equiv \lambda_1$ is a root of the eigenvalue curve $\mu_1(\zeta)$. The figure shows two potential choices of the variable $\sigma \in [\lambda^-, \lambda^+]$.*

Following (3.1), the eigenvector x can be written as

$$
\begin{aligned}
x &= \begin{pmatrix} -(B-\lambda M_B)^{-1}(E-\lambda M_E) y_j(\lambda) \\ y_j(\lambda) \end{pmatrix} \\
&= \begin{pmatrix} -(B-\lambda M_B)^{-1}(E-\lambda M_E) \left[\sum_{i=0}^{\infty} \frac{(\lambda-\sigma)^i}{i!} \left(\frac{d^i y_j(\zeta)}{d\zeta^i} \right)_{\zeta=\sigma} \right] \\ \sum_{i=0}^{\infty} \frac{(\lambda-\sigma)^i}{i!} \left(\frac{d^i y_j(\zeta)}{d\zeta^i} \right)_{\zeta=\sigma} \end{pmatrix} \\
&= \begin{pmatrix} -(B-\lambda M_B)^{-1}(E-\lambda M_E) \left[y_j(\sigma), \left(\frac{dy_j(\zeta)}{d\zeta} \right)_{\zeta=\sigma}, \left(\frac{d^2 y_j(\zeta)}{d\zeta^2} \right)_{\zeta=\sigma}, \dots \right] \\ \left[y_j(\sigma), \left(\frac{dy_j(\zeta)}{d\zeta} \right)_{\zeta=\sigma}, \left(\frac{d^2 y_j(\zeta)}{d\zeta^2} \right)_{\zeta=\sigma}, \dots \right] \end{pmatrix} \begin{pmatrix} 1 \\ (\lambda-\sigma)/1! \\ (\lambda-\sigma)^2/2! \\ \vdots \end{pmatrix}.
\end{aligned}
$$

3.4. Rayleigh quotient approximation of λ.

DEFINITION 3.3. *We define the following matrix-valued functions of $\zeta \in \mathbb{R}$:*

$$B_\zeta = B - \zeta M_B, \qquad E_\zeta = E - \zeta M_E, \qquad \text{and } C_\zeta = C - \zeta M_C.$$

DEFINITION 3.4. *We define the M-norm of an SPD $n \times n$ matrix M and a nonzero vector $x \in \mathbb{R}^n$ to be equal to $\|x\|_M = \sqrt{x^T M x}$.*

3.4.1. A basic approximation.

PROPOSITION 3.5. *Let $\lambda \in \Lambda(A, M)$ satisfy $\mu_j(\lambda) = 0$, and $\lambda^- \le \sigma \le \lambda \le \lambda^+$, where λ is the only root of $\mu_j(\zeta)$ located inside the interval $[\lambda^-, \lambda^+]$. Additionally, let $\tau \in \mathbb{Z}$, and define the vectors*

$$\hat{y} = \left[y_j(\sigma), \left(\frac{dy_j(\zeta)}{d\zeta}\right)_{\zeta=\sigma}, \ldots, \left(\frac{d^\tau y_j(\zeta)}{d\zeta^\tau}\right)_{\zeta=\sigma} \right] \begin{pmatrix} 1 \\ \lambda-\sigma/1! \\ \vdots \\ (\lambda-\sigma)^\tau/\tau! \end{pmatrix} \tag{3.2}$$

and

$$\hat{x} = \begin{pmatrix} -B_\sigma^{-1} E_\sigma \hat{y} \\ \hat{y} \end{pmatrix}.$$

Then, if $\hat{\lambda} = \frac{\hat{x}^T A \hat{x}}{\hat{x}^T M \hat{x}}$,

$$\left|\lambda - \hat{\lambda}\right| = O\left((\lambda-\sigma)^2\right),$$

where the big-O symbol is to be interpreted in the limit as $\sigma \to \lambda$.

Proof. Let $\rho(z) = \frac{z^T A z}{z^T M z}$ be the Rayleigh quotient of any nonzero vector $z \in \mathbb{R}^n$. Expanding $\rho(z)$ through its Taylor series expansion around the eigenvector x gives

$$\rho(z) = \rho(x) + (z-x)^T \nabla\rho(x) + O(\|z-x\|_M^2) \quad \text{as} \quad \hat{z} \to x.$$

The gradient of the Rayleigh quotient is equal to $\nabla\rho(z) = 2\frac{Az(z^TMz) - Mz(z^TAz)}{(z^TMz)^2}$, and thus $\nabla\rho(x) = 0$. It follows that

$$\left|\lambda - \hat{\lambda}\right| = |\rho(x) - \rho(\hat{x})| = O\left(\|\hat{x}-x\|_M^2\right) \quad \text{as} \quad \hat{x} \to x.$$

Write now $E_\lambda = E_\sigma - (\lambda-\sigma)M_E$ and define the vector

$$r = \begin{pmatrix} -B_\lambda^{-1} E_\lambda \left[\left(\frac{d^{\tau+1} y_j(\zeta)}{d\zeta^{\tau+1}}\right)_{\zeta=\sigma}, \left(\frac{d^{\tau+2} y_j(\zeta)}{d\zeta^{\tau+2}}\right)_{\zeta=\sigma}, \cdots \right] \\ \left[\left(\frac{d^{\tau+1} y_j(\zeta)}{d\zeta^{\tau+1}}\right)_{\zeta=\sigma}, \left(\frac{d^{\tau+2} y_j(\zeta)}{d\zeta^{\tau+2}}\right)_{\zeta=\sigma}, \cdots \right] \end{pmatrix} \begin{pmatrix} (\lambda-\sigma)^{\tau+1}/(\tau+1)! \\ (\lambda-\sigma)^{\tau+2}/(\tau+2)! \\ \vdots \end{pmatrix}.$$

The difference $x - \hat{x}$ can then be written as

$$x - \hat{x} = r - \begin{pmatrix} \left[B_\lambda^{-1} - B_\sigma^{-1}\right] E_\sigma \hat{y} \\ 0 \end{pmatrix} + \begin{pmatrix} (\lambda-\sigma) B_\lambda^{-1} M_E \hat{y} \\ 0 \end{pmatrix}.$$

Let now $E_\sigma\hat{y}$ and $M_E\hat{y}$ be expanded in the basis $\{M_B v^{(\ell)}\}_{\ell=1,\ldots,d}$:

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

where $[\epsilon_\ell, \gamma_\ell]^T \in \mathbb{R}^2$ are the expansion coefficients associated with the direction $v^{(\ell)}$. Taking advantage of the identity $B_\lambda^{-1} - B_\sigma^{-1} = (\lambda-\sigma)B_\lambda^{-1} M_B B_\sigma^{-1}$ and noticing that $\|r\|_M = O((\lambda-\sigma)^{\tau+1})$ leads to

(3.3)

$$\begin{aligned} \|x-\hat{x}\|_M^2 &= \left\| r - \begin{pmatrix} (\lambda-\sigma) B_\lambda^{-1} \left(M_B B_\sigma^{-1} E_\sigma - M_E\right) \hat{y} \\ 0 \end{pmatrix} \right\|_M^2 \\ &= \|r\|_M^2 + \left\|(\lambda-\sigma) B_\lambda^{-1} \left(M_B B_\sigma^{-1} E_\sigma - M_E\right) \hat{y}\right\|_{M_B}^2 \\ &\quad - 2(\lambda-\sigma) r^T M \begin{pmatrix} B_\lambda^{-1} \left(M_B B_\sigma^{-1} E_\sigma - M_E\right) \hat{y} \\ 0 \end{pmatrix} \\ &= O\left((\lambda-\sigma)^{2\tau+2}\right) + (\lambda-\sigma)^2 \sum_{\ell=1}^{d} \left(\frac{\epsilon_\ell - \gamma_\ell(\delta_\ell - \sigma)}{(\delta_\ell - \sigma)(\delta_\ell - \lambda)} \right)^2 + O\left((\lambda-\sigma)^{\tau+2}\right), \end{aligned}$$

where the big-O symbol is to be interpreted in the limit as $\sigma \to \lambda$, and we made use of the relations

$$(\lambda-\sigma)B_\lambda^{-1}\left(M_B B_\sigma^{-1}E_\sigma - M_E\right)\hat{y} = (\lambda-\sigma)\sum_{\ell=1}^{d}\left(\frac{\epsilon_\ell - \gamma_\ell(\delta_\ell-\sigma)}{(\delta_\ell-\sigma)(\delta_\ell-\lambda)}\right)v^{(\ell)}$$

and

$$\left\|\begin{pmatrix}(\lambda-\sigma)B_\lambda^{-1}\left(M_B B_\sigma^{-1}E_\sigma - M_E\right)\hat{y}\\ 0\end{pmatrix}\right\|_M^2 = \left\|(\lambda-\sigma)B_\lambda^{-1}\left(M_B B_\sigma^{-1}E_\sigma - M_E\right)\hat{y}\right\|_{M_B}^2$$
$$= (\lambda-\sigma)^2\sum_{\ell=1}^{d}\left(\frac{\epsilon_\ell - \gamma_\ell(\delta_\ell-\sigma)}{(\delta_\ell-\sigma)(\delta_\ell-\lambda)}\right)^2. \qquad \square$$

Proposition 3.5 remains valid even when the interval $[\lambda^-,\lambda^+]$ includes more than one root of $\mu_j(\zeta)$ (including multiple eigenvalues). However, only one of these eigenvalues can be approximated. Moreover, the interval $[\lambda^-,\lambda^+]$ can include poles of $\mu_j(\zeta)$ but σ needs to be algebraically smaller than these poles.

3.4.2. Improving accuracy by deflation. The bound on $\|x-\hat{x}\|_M^2$ appearing in Proposition 3.5 can be improved (reduced) by explicitly removing those directions in which $\|x-\hat{x}\|_M^2$ is large, i.e., the directions corresponding to the eigenvectors associated with the few smallest (in magnitude) eigenvalues of the matrix pencil (B_σ, M_B). This is especially true for the second and third terms shown in (3.3), both of which depend on the distance of $\delta_\ell,\ \ell=1,\ldots,d$, from both σ and λ.

More specifically, let the approximate eigenvector $\hat{x}$ set as

$$\hat{x} = \begin{pmatrix}-B_\sigma^{-1}E_\sigma\hat{y}\\ \hat{y}\end{pmatrix} - \begin{pmatrix}\left[v^{(1)},\ldots,v^{(\hat{\kappa})}\right]\\ 0\end{pmatrix}\begin{pmatrix}\nu_1\\ \vdots\\ \nu_{\hat{\kappa}}\end{pmatrix}, \tag{3.4}$$

where $\hat{y}$ is defined in (3.2), and $\nu_j = (\lambda-\sigma)(\frac{\epsilon_j-\gamma_j(\delta_j-\sigma)}{(\delta_j-\lambda)(\delta_j-\sigma)}),\ j=1,\ldots,\hat{\kappa}\le d$. Following the reasoning in Proposition 3.5 (we omit the intermediate steps), we can show

$$\|x-\hat{x}\|_M^2 = O\left((\lambda-\sigma)^{2\tau+2}\right) + (\lambda-\sigma)^2\sum_{\ell=\hat{\kappa}+1}^{d}\left(\frac{\epsilon_\ell-\gamma_\ell(\delta_\ell-\sigma)}{(\delta_\ell-\sigma)(\delta_\ell-\lambda)}\right)^2 + O\left((\lambda-\sigma)^{\tau+2}\right).$$

The eigenvector approximation $\hat{x}$ shown in (3.4) becomes appealing when $\hat{\kappa}$ is set such that the eigenvalues $\delta_{\hat{\kappa}+1},\ldots,\delta_d$ lie far away from both λ and σ.

3.4.3. Reducing the asymptotic order of the upper bound. The analysis in Proposition 3.5 suggests that regardless of the value of τ, at the limit $\sigma\to\lambda$ the term $|\lambda-\hat{\lambda}|$ will be of the order $O((\lambda-\sigma)^2)$ due to the approximation of the term $B_\lambda^{-1}E_\lambda$ by $B_\sigma^{-1}E_\sigma$.

Let us write $B_\lambda^{-1} = B_{\sigma+\epsilon}^{-1}$ where $\epsilon < \mathbf{min}_{1\le\ell\le d}|\delta_\ell-\sigma|$. The matrix resolvent can then be written as $B_\lambda^{-1} = ((I-\epsilon M_B B_\sigma^{-1})B_\sigma)^{-1} = B_\sigma^{-1}\sum_{k=0}^{\infty}[\epsilon M_B B_\sigma^{-1}]^k$. The approximation $u\approx -B_\sigma^{-1}E_\sigma\hat{y}$ can be replaced by $u\approx -(B_\sigma^{-1}+\epsilon B_\sigma^{-1}M_B B_\sigma^{-1})E_\sigma\hat{y}$. When $M_E\neq 0$, the error term associated with $B_\lambda^{-1}M_E\hat{y}$ can also be smoothed out by adding the vector $B_\sigma^{-1}M_E\hat{y}$, i.e., $u\approx -(B_\sigma^{-1}E_\sigma+\epsilon B_\sigma^{-1}M_B B_\sigma^{-1}E_\sigma - B_\sigma^{-1}M_E)\hat{y}$.

PROPOSITION 3.6. *Let $\lambda \in \Lambda(A, M)$ satisfy $\mu_j(\lambda) = 0$, and $\lambda^- \leq \sigma \leq \lambda \leq \lambda^+$, where λ is the only root of $\mu_j(\zeta)$ located inside the interval $[\lambda^-, \lambda^+]$. Additionally, let $\tau \in \mathbb{Z}$ and define the vectors*

$$\hat{y} = \left[y_j(\sigma), \left(\frac{dy_j(\zeta)}{d\zeta}\right)_{\zeta=\sigma}, \ldots, \left(\frac{d^\tau y_j(\zeta)}{d\zeta^\tau}\right)_{\zeta=\sigma} \right] \begin{pmatrix} 1 \\ \lambda-\sigma/1! \\ \vdots \\ (\lambda-\sigma)^\tau/\tau! \end{pmatrix}$$

and

$$\hat{x} = \begin{pmatrix} -\left[B_\sigma^{-1} + (\lambda-\sigma)B_\sigma^{-1}M_B B_\sigma^{-1}\right] E_\sigma \hat{y} \\ \hat{y} \end{pmatrix} + \begin{pmatrix} B_\sigma^{-1} M_E \hat{y} \\ 0 \end{pmatrix} - \left(\begin{bmatrix} v^{(1)}, \ldots, v^{(\hat{\kappa})} \end{bmatrix} \right) \begin{pmatrix} \nu_1 \\ \vdots \\ \nu_{\hat{\kappa}} \end{pmatrix},$$

where $\nu_j = (\lambda-\sigma)^2 (\frac{\epsilon_j - \gamma_j(\delta_j - \sigma)}{(\delta_j - \lambda)(\delta_j - \sigma)^2})$, $j = 1, \ldots, \hat{\kappa} \leq d$.

Then, if $\hat{\lambda} = \frac{\hat{x}^T A \hat{x}}{\hat{x}^T M \hat{x}}$,

$$\left|\lambda - \hat{\lambda}\right| = O\left((\lambda-\sigma)^\chi\right), \quad \textit{where} \quad \begin{cases} \chi = 2, & \text{when} \quad \tau = 0, \quad \text{and} \\ \chi = 4, & \text{when} \quad \tau \geq 1, \end{cases}$$

and the big-O symbol is to be interpreted in the limit as $\sigma \to \lambda$.

Proof. First notice that

$$(\lambda-\sigma)\left(B_\lambda^{-1} - B_\sigma^{-1}\right) = (\lambda-\sigma)^2 B_\lambda^{-1} M_B B_\sigma^{-1},$$
$$\left(B_\lambda^{-1} - B_\sigma^{-1} - (\lambda-\sigma)B_\sigma^{-1}M_B B_\sigma^{-1}\right) = (\lambda-\sigma)^2 B_\lambda^{-1} M_B B_\sigma^{-1} M_B B_\sigma^{-1}.$$

We can then write the difference $x - \hat{x}$ as

$$\begin{aligned} x - \hat{x} &= r - (\lambda-\sigma)^2 \left[\begin{pmatrix} B_\lambda^{-1} M_B B_\sigma^{-1} M_B B_\sigma^{-1} E_\sigma \hat{y} \\ 0 \end{pmatrix} - \begin{pmatrix} B_\lambda^{-1} M_B B_\sigma^{-1} M_E \hat{y} \\ 0 \end{pmatrix} \right] \\ &\quad + \left(\begin{bmatrix} v^{(1)}, \ldots, v^{(\hat{\kappa})} \end{bmatrix} \right) \begin{pmatrix} \nu_1 \\ \vdots \\ \nu_{\hat{\kappa}} \end{pmatrix} \\ &= r - (\lambda-\sigma)^2 \begin{pmatrix} B_\lambda^{-1} M_B B_\sigma^{-1} \left(M_B B_\sigma^{-1} E_\sigma - M_E\right) \hat{y} \\ 0 \end{pmatrix} + \left(\begin{bmatrix} v^{(1)}, \ldots, v^{(\hat{\kappa})} \end{bmatrix} \right) \begin{pmatrix} \nu_1 \\ \vdots \\ \nu_{\hat{\kappa}} \end{pmatrix}. \end{aligned}$$

Following the same reasoning as in Proposition 3.5 (we omit the intermediate steps), we get

$$\|x - \hat{x}\|_M^2 = O\left((\lambda-\sigma)^{2(\tau+1)}\right) + (\lambda-\sigma)^4 \sum_{\ell=\hat{\kappa}+1}^{d} \left(\frac{\epsilon_\ell - \gamma_\ell(\delta_\ell - \sigma)}{(\delta_\ell - \sigma)^2(\delta_\ell - \lambda)} \right)^2 + O\left((\lambda-\sigma)^{\tau+3}\right). \square$$

Proposition 3.6 tells us that a first-order approximation of the resolvent B_λ^{-1} combined with $\tau = 1$ leads to eigenvalue approximation errors of the order $O((\lambda-\sigma)^4)$ as $\sigma \to \lambda$. Figure 3.2 plots the approximation error of eigenvalues λ_1, λ_2, and λ_3, obtained by Proposition 3.6 for some Dirichlet discretization of the Laplacian operator in the two-dimensional space. In agreement with Proposition 3.6, the true error curves follow nicely those of $(\sigma-\lambda_j)^2$ (when $\tau = 0$) and $(\sigma-\lambda_j)^4$ (when $\tau = 1$), respectively. The error reduction remains quartic even when $\tau \geq 2$. More generally, an increase by one in the value of τ should be accompanied by the addition of one more term in the Neumann series approximation of the resolvent B_λ^{-1} if the order of the upper bound is to be decreased.

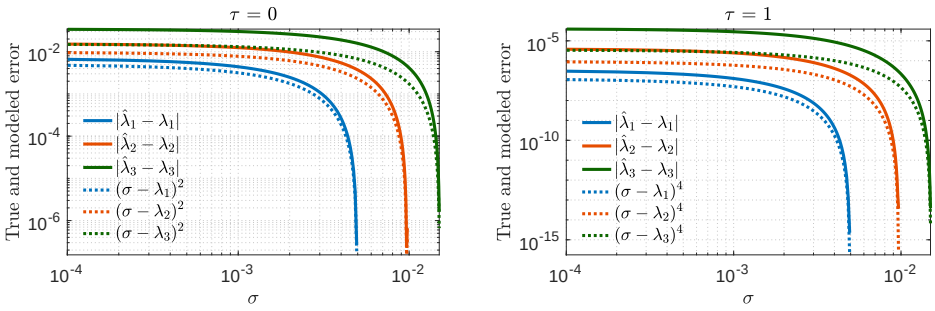


FIG. 3.2. *Approximation error of eigenvalues λ_1, λ_2, and λ_3, obtained by Proposition* 3.6 *as $\sigma \in [0, \lambda_3)$. Matrix A is formed by a regular Dirichlet discretization of the Laplacian operator over a square domain where the grid is partitioned into $p = 4$ subdomains ($M = I$). Left: $\tau = 0$. Right: $\tau = 1$.*

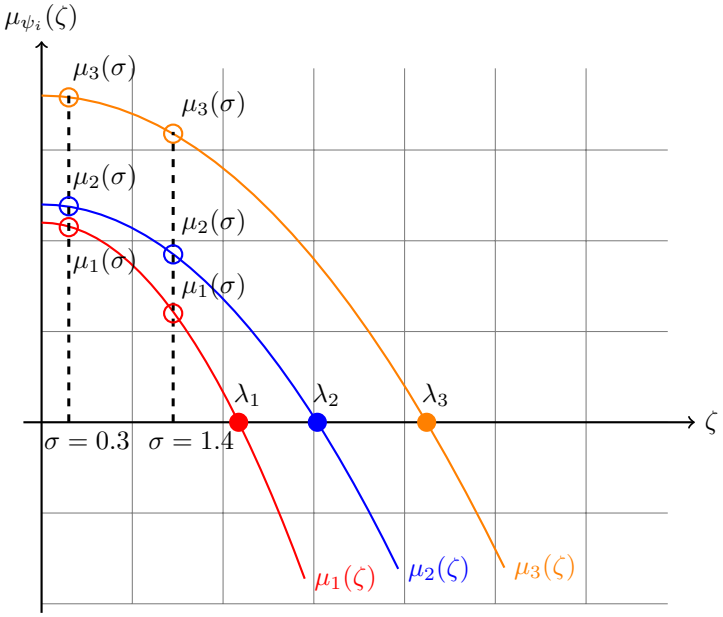


FIG. 4.1. *The eigenpairs of the matrix $S(\sigma)$ can be exploited for the approximation of more than one eigenpair of the matrix pencil (A, M). In this example we have $n_{ev} = 3$ and $\psi_i = \kappa_i = i$. Two different choices of σ are shown.*

4. Computing a large number of eigenpairs. The technique discussed in section 3 can be extended to the simultaneous approximation of all n_{ev} sought eigenpairs. More specifically, denote the n_{ev} sought eigenvalues located immediately on the right of a real scalar α by $\lambda_{\kappa_1} \leq \cdots \leq \lambda_{\kappa_{n_{ev}}}$, and let eigenvalue λ_{κ_i} be a root of the eigenvalue curve $\mu_{\psi_i}(\zeta)$, i.e., $\psi_i = \mathbf{arg}\{j \mid \mu_j(\lambda_{\kappa_i}) = 0\}$. The eigenpair associated with eigenvalue λ_{κ_i} can then be approximated independently of the rest by computing the eigenpair $(\mu_{\psi_i}(\sigma), y_{\psi_i}(\sigma))$ and considering eigenvector $y_{\psi_i}(\sigma)$ as a zeroth-order approximation of the eigenvector $y_{\psi_i}(\lambda_{\kappa_i}) \equiv y^{(\kappa_i)}$. Figure 4.1 illustrates an example where $n_{ev} = 3$ and $\psi_i = \kappa_i = i$. Eigenvalue λ_i is a root of eigenvalue curve $\mu_i(\zeta)$, and the theory presented in section 3 applies to each eigenvalue curve independently.

Assume for the moment that no eigenvalue curves cross each other and let $\sigma = \alpha$. Since the eigenvalue curves are strictly decreasing, we can infer that the eigenvalues

$\lambda_{\kappa_1}, \ldots, \lambda_{\kappa_{n_{ev}}}$ are roots of consecutive eigenvalue curves, i.e., $\psi_i = \psi_1 + (i-1)$. Additionally, since $\alpha \leq \lambda_{\kappa_1}$, $\mu_{\psi_1}(\zeta)$ is precisely the eigenvalue curve which crosses the algebraically smallest nonnegative eigenvalue of $S(\alpha)$. The two above observations tell us that the eigenvectors $y_{\psi_1}(\alpha), \ldots, y_{\psi_{n_{ev}}}(\alpha)$ associated with the n_{ev} algebraically smallest nonnegative eigenvalues of the matrix $S(\alpha)$ form a zeroth-order approximation of the vectors $y_{\psi_1}(\lambda_{\kappa_1}) \equiv y^{(\lambda_{\kappa_1})}, \ldots, y_{\psi_{n_{ev}}}(\lambda_{\kappa_{n_{ev}}}) \equiv y^{(\lambda_{\kappa_{n_{ev}}})}$. Note that the value of ψ_1 itself is not needed.

In practice, eigenvalue curves which intersect each other pose no threat as long as each one of the n_{ev} algebraically smallest nonnegative eigenvalues of the matrix $S(\alpha)$ coincides with one of the values $\mu_{\psi_1}(\alpha), \ldots, \mu_{\psi_{n_{ev}}}(\alpha)$. Our default strategy throughout the rest of this paper is to set $\sigma = \alpha$.

4.1. A Rayleigh–Ritz algorithm. The theoretical results presented in section 3 focused on the eigenvalue approximation error resulting by a Rayleigh quotient approximation with an approximate eigenvector $\hat{x}$. In practice, we cannot form $\hat{x}$ since we do not know the quantity $\lambda - \sigma$. Nonetheless, we can overcome this drawback by approximating all n_{ev} eigenvalues from a single subspace by means of a Rayleigh–Ritz projection.

ALGORITHM 4.1. *The complete procedure*

0. *Input:* A, M, p, α, n_{ev}, κ
1. *Reorder* A *and* M *as in* (2.4)
2. *Call Algorithm* 4.2 *(subspace associated with interface variables)*
3. *Call Algorithm* 4.3 *(subspace associated with interior variables)*
4. *Build the projection matrix* Z *as described in section* 4.3.1
5. *Solve* $(Z^T A Z)\tilde{x} = \hat{\lambda}(Z^T M Z)\tilde{x}$ *for the* n_{ev} *sought* $(\tilde{\lambda}, \tilde{x})$
6. *Form the Ritz vectors* $\hat{x} = Z\tilde{x}$ *associated with the* n_{ev} *computed eigenpairs from step 5*

Algorithm 4.1 describes the complete algorithmic procedure to compute the n_{ev} eigenvalues located immediately on the right of the user-given scalar $\alpha \in \mathbb{R}$. The Rayleigh–Ritz eigenvalue problem shown in step 5 must be solved only for the eigenpairs associated with the n_{ev} smallest eigenvalues that are greater than or equal to α. Except for the matrices A and M, and the scalars n_{ev} and α, Algorithm 4.1 also requires a number of partitions p and a nonzero integer κ denoting the number of eigenvectors computed from each matrix pencil $(B_\sigma^{(j)}, M_B^{(j)})$, $j = 1, \ldots, p$.

4.2. Building the projection subspace associated with interface variables. Algorithm 4.2 begins by computing the eigenvectors associated with the n_{ev} smallest nonnegative eigenvalues of the matrix $S(\alpha)$. These can be computed by any appropriate sparse eigenvalue solver, e.g., the (block) Lanczos method combined with shift-and-invert [13] or (generalized) Davidson [36]. The algorithm proceeds by computing the derivatives $y'_{\psi_i}(\alpha) \equiv (\frac{dy_j(\zeta)}{d\zeta})_{\zeta=\alpha}, y''_{\psi_i}(\alpha) \equiv (\frac{d^2 y_j(\zeta)}{d\zeta^2})_{\zeta=\alpha}, \ldots,$ $i = 1, 2, \ldots, n_{ev}$. Details on the practical computation of these derivatives up to a second order are provided in the appendix.

ALGORITHM 4.2. *Projection subspace associated with interface variables*

0. *Input:* α, n_{ev}
1. *Solve* $S(\alpha)y(\alpha) = \mu(\alpha)y(\alpha)$ *for the* n_{ev} *smallest nonnegative eigenvalues* $\mu(\alpha)$ *and associated eigenvectors* $y(\alpha)$
-. *Denote these eigenpairs as* $(\mu_{\psi_i}(\alpha), y_{\psi_i}(\alpha))$, $i = 1, 2, \ldots, n_{ev}$
2. *Form* $Y = [y_{\psi_1}(\alpha), y'_{\psi_1}(\alpha), y''_{\psi_1}(\alpha), \ldots, y_{\psi_2}(\alpha), y'_{\psi_2}(\alpha), y''_{\psi_2}(\alpha), \ldots]$

4.3. Building the projection subspace associated with interior variables. Algorithm 4.3 provides a formal description of the procedure followed for the construction of the projection subspace associated with interior variables. The routine "$\texttt{eigs}(B_\alpha^{(j)}, M_B^{(j)}, \kappa, \mathrm{sm})$" listed in step 2 denotes the computation of the eigenvectors associated with the κ smallest (in magnitude) eigenvalues of each matrix pencil $(B_\alpha^{(j)}, M_B^{(j)})$, $j = 1, \ldots, p$, and can be performed by any appropriate sparse eigenvalue solver. These eigenvectors form the columns of the $d_j \times \kappa$ matrix V_j. The value of κ can be set either a priori or adaptively, e.g., see the related discussion in [40].

ALGORITHM 4.3. *Projection subspace associated with interior variables*

0. *Input:* $\alpha,\ p,\ \kappa,\ Y :=$ orthonormal basis returned by Algorithm 4.2
1a. *For* $j = 1, \ldots, p$
2. *Compute* $V_j = \texttt{eigs}\left(B_\alpha^{(j)}, M_B^{(j)}, \kappa, \mathrm{sm}\right)$
1b. *End*
3. *Form* $U = \left[\bar{U}^{(1)}, \ldots, \bar{U}^{(p)}, -B_\alpha^{-1}E_\alpha Y, -B_\alpha^{-1}M_B B_\alpha^{-1}E_\alpha Y, B_\alpha^{-1}M_E Y\right]$
where $\bar{U}^{(j)} = \begin{pmatrix} 0_{\ell_j,\kappa} \\ V_j \\ 0_{\xi_j,\kappa} \end{pmatrix}$ *and we recall* $\ell_j = \sum_{k=1}^{k<j} s_k$, and $\xi_j = \sum_{k>j}^{k=p} s_k$

4.3.1. The Rayleigh–Ritz eigenvalue problem. The matrix Y returned by Algorithm 4.2 is distributed among the p subdomains and can be written as

$$Y = \begin{bmatrix} Y_1^T & Y_2^T & \cdots & Y_p^T \end{bmatrix}^T, \tag{4.1}$$

where $Y_j \in \mathbb{R}^{s_j \times \eta}$ is the row block of matrix Y associated with the jth subdomain and $\eta \in \mathbb{Z}^*$ denotes the column dimension of matrix Y. By definition η is equal to an integer multiple of n_{ev}.

Define now the matrices

$$\left(\hat{B}_\alpha^{(j)}\right)^{-1} = \left(B_\alpha^{(j)}\right)^{-1}\left(I - V_j V_j^T M_B^{(j)}\right) \text{ and}$$

$$P_j = \left[E_\alpha^{(j)}, -M_E^{(j)}\right]\begin{pmatrix} Y_j & 0 \\ 0 & Y_j \end{pmatrix}.$$

The projection matrix Z can be then written as

$$Z = \begin{bmatrix} V_1 & & & -\left(\hat{B}_\alpha^{(1)}\right)^{-1}P_1 & -\left(B_\alpha^{(1)}\left(M_B^{(1)}\right)^{-1}\hat{B}_\alpha^{(1)}\right)^{-1}E_\alpha^{(1)}Y_1 \\ & V_2 & & -\left(\hat{B}_\alpha^{(2)}\right)^{-1}P_2 & -\left(B_\alpha^{(2)}\left(M_B^{(2)}\right)^{-1}\hat{B}_\alpha^{(2)}\right)^{-1}E_\alpha^{(2)}Y_2 \\ & & \ddots & \vdots & \vdots \\ & & & V_p & -\left(\hat{B}_\alpha^{(p)}\right)^{-1}P_p & -\left(B_\alpha^{(p)}\left(M_B^{(p)}\right)^{-1}\hat{B}_\alpha^{(p)}\right)^{-1}E_\alpha^{(p)}Y_p \\ & & & & [Y, 0_{s,\eta}] & \end{bmatrix}.$$

The total memory overhead associated with the jth subdomain is equal to that of storing $\kappa d_j + (3d_j + s_j)\eta$ floating-point scalars. The dimension of the Rayleigh–Ritz pencil $(Z^T A Z, Z^T M Z)$ is equal to $\kappa p + 3\eta$ and can be solved by the appropriate routine in LAPACK, e.g., `dsygv` [3]. As a sidenote, when $M_E^{(j)} = 0$ we have $P_j = E_\alpha^{(j)} Y_j$ and the bottom row block of matrix Z becomes $[0_{s,p\kappa}, Y, 0_{s,\eta}]$. The dimension of the Rayleigh–Ritz pencil then reduces to $\kappa p + 2\eta$.

4.4. Comparing Algorithm 4.1 with shift-and-invert Lanczos. A natural question is how Algorithm 4.1 compares against shift-and-invert Lanczos when the latter is applied directly to the pencil $(A - \alpha M, M)$.

The first key difference between these two techniques is *orthogonalization cost*. Applying k steps of shift-and-invert Lanczos to matrix pencils $(S(\alpha), I)$ and $(A - \alpha M, M)$ leads to a total orthogonalization cost of $O(k^2 s)$ and $O(k^2 n)$, respectively. Thus, Algorithm 4.1 reduces orthogonalization costs by a factor of n/s, and this difference becomes more pronounced as n_{ev} increases (since $k \geq n_{ev}$).

The second key difference between Algorithm 4.1 and shift-and-invert Lanczos is *the number of linear system solutions with B_α as the coefficient matrix*. It is straightforward to verify that applying k steps of shift-and-invert Lanczos to the pencil $(A - \alpha M, M)$ requires $2k$ such linear system solutions. In contrast, Algorithm 4.1 requires 3η (2η if $M_E = 0$) linear solves with B_α. Nonetheless, these 3η linear solves can be performed simultaneously, since all right-hand sides are available at the same time. Thus, the associated cost can be marginally higher than that of solving for a few right-hand sides. To the above cost we also need to add the computational cost required to compute the eigenvectors of the block-diagonal pencil (B_α, M_B) in Algorithm 4.3.

On the other hand, the accuracy achieved by shift-and-invert Lanczos can be significantly higher than that of Algorithm 4.1. Nonetheless, in many applications the sought eigenpairs need not be computed to high accuracy, e.g., eigenpairs of problems originating from discretizations need be approximated up to the associated discretization error.

4.5. Parallel computing. Algorithm 4.1 is based on domain decomposition and is well-suited for execution in modern distributed memory environments. For example, each one of the p subdomains can be mapped to a separate MPI process. Performing any type of operation with the matrices $B_\sigma^{(j)}$, $E_\sigma^{(j)}$, and $M_E^{(j)}$ is then an entirely local process in the jth subdomain (i.e., Algorithm 4.3 is embarrassingly parallel). An additional layer of parallelism can be realized in each subdomain by further exploiting shared memory parallelism to perform the required computations with the aforementioned matrices; e.g., see [24] for a similar discussion in the context of domain decomposition eigenvalue solvers and [15] for a general discussion on parallel computing and domain decomposition.

In contrast to Algorithm 4.3, Algorithm 4.2 involves computations with the Schur complement matrix $S(\alpha)$, which is distributed among the p subdomains. In this case, point-to-point communication among neighboring subdomains is necessary. Finally, the Rayleigh–Ritz eigenvalue problem is typically small enough so that it can be replicated in all MPI processes and solved redundantly.

5. Numerical experiments. Our sequential experiments are conducted in a MATLAB environment (version R2018b), using 64-bit arithmetic, on a single core of a computing system equipped with an Intel Haswell E5-2680v3 processor and 32 GB of system memory.

The eigenvalues of the pencil (A, M) are ordered as $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, and throughout the rest of this section we focus in computing the n_{ev} algebraically smallest eigenvalues $\lambda_1, \ldots, \lambda_{n_{ev}}$. Unless mentioned otherwise, the default values in Algorithm 4.1 will be set as $p = 8$, $\kappa = 5$, and $\alpha = 0$. For indefinite pencils we first shift the spectrum so that all eigenvalues become positive, and then apply Algorithm 4.1 with $\alpha = 0$. Throughout the rest of this paper we assume $\psi_i = i,\ i = 1, \ldots, n_{ev}$.

We consider three different choices to set $\mathbf{span}(Y)$ in Algorithm 4.2:

$$\{y\} := \mathbf{span}\left([y_i(\alpha)]_{i=1,\ldots,3n_{ev}}\right),$$

$$\{y, dy\} := \mathbf{span}\left(\left[y_i(\alpha), \left(\frac{dy_i(\zeta)}{d\zeta}\right)_{\zeta=\alpha}\right]_{i=1,\ldots,n_{ev}}\right),$$

$$\{y, dy, d^2y\} := \mathbf{span}\left(\left[y_i(\alpha), \left(\frac{dy_i(\zeta)}{d\zeta}\right)_{\zeta=\alpha}, \left(\frac{d^2y_i(\zeta)}{d\zeta^2}\right)_{\zeta=\alpha}\right]_{i=1,\ldots,n_{ev}}\right).$$

The eigenpairs of matrix $S(\alpha)$ are computed up to a residual norm of 1.0×10^{-6}.

5.1. A model problem. Our first test case consists of a five-point stencil finite difference discretization of the Dirichlet eigenvalue problem

$$\Delta u + \lambda u = 0 \text{ in } \Omega := (0,1) \times (0,1), \quad u_{|\partial\Omega} = 0, \tag{5.1}$$

where Δ denotes the Laplace operator. Our goal is not to compute the actual eigenvalues of the Laplace operator but rather to assess the performance of Algorithm 4.1. To this end, the Dirichlet eigenvalue problem is discretized on a 250×250 mesh, and we set $n_{ev} = 150$. Note that A has many eigenvalues of multiplicity $\rho_\lambda = 2$. The proposed method can naturally capture multiple eigenvalues in contrast to (nonblock) Krylov-based approaches such as shift-and-invert Lanczos.

Figure 5.1 plots the relative eigenvalue errors and associated residual norms of the approximate eigenpairs returned by Algorithm 4.1 when $U = \left[u^{(1)}, \ldots, u^{(n_{ev})}\right]$, i.e., there is no error associated with interior variables. In agreement with the discussion in section 3, adding more eigenvector derivatives leads to higher accuracy, especially for those eigenvalues located closest to α. Table 5.1 lists the maximum/minimum

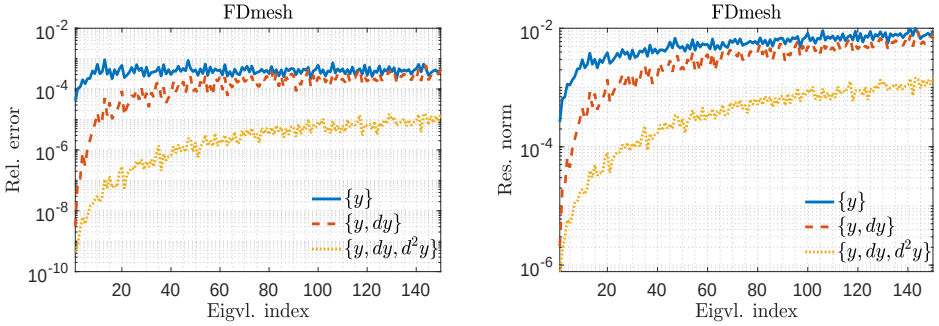


FIG. 5.1. *Relative eigenvalue errors (left) and residual norms (right) returned by Algorithm 4.1 in the approximation of the $n_{ev} = 150$ algebraically smallest eigenvalues of the 250×250 finite difference discretization of the Dirichlet eigenvalue problem when $U = \left[u^{(1)}, \ldots, u^{(n_{ev})}\right]$.*

TABLE 5.1
Maximum/minimum relative eigenvalue error achieved by Algorithm 4.1 in the approximation of the $n_{ev} = 50$ algebraically smallest eigenvalues of the 250×250 finite difference discretization of the Dirichlet eigenvalue problem for $U = \left[u^{(1)}, \ldots, u^{(n_{ev})}\right]$ and various values of p.

	$\{y\}$		$\{y, dy\}$		$\{y, dy, d^2y\}$	
	max	min	max	min	max	min
$p = 2$	1.1×10^{-6}	9.0×10^{-9}	5.7×10^{-7}	5.2×10^{-12}	2.9×10^{-9}	1.7×10^{-15}
$p = 4$	7.5×10^{-6}	2.8×10^{-8}	8.0×10^{-6}	1.9×10^{-11}	4.5×10^{-7}	1.6×10^{-13}
$p = 8$	4.7×10^{-5}	7.2×10^{-8}	8.4×10^{-5}	2.0×10^{-11}	6.7×10^{-6}	2.8×10^{-12}
$p = 16$	1.5×10^{-4}	1.1×10^{-7}	3.4×10^{-4}	5.8×10^{-11}	5.3×10^{-5}	9.4×10^{-12}

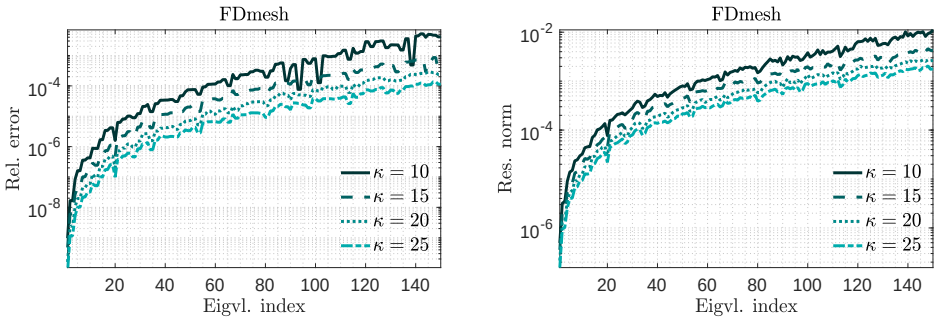


FIG. 5.2. *Plots of the relative eigenvalue errors (left) and residual norms (right) returned by Algorithm* 4.1 *in the approximation of the* $n_{ev} = 150$ *algebraically smallest eigenvalues of the* 250×250 *finite difference discretization of the Dirichlet eigenvalue problem when* $Y = \left[y^{(1)}, \ldots, y^{(n_{ev})}\right]$, *and* κ *varies.*

TABLE 5.2
n: size of (A, M)*; s: number of interface variables* ($p = 8$)*; nnz*(.)*: number of nonzero entries.*

#	Matrix pencil	n	$nnz(A)/n$	$nnz(M)/n$	s	Application	Source
1.	Si2	769	23.1	1.0	547	Quantum Chemistry	[9]
2.	nos3	960	16.5	1.0	170	Structural	[9]
3.	FEmesh	2,689	6.9	6.9	190	Finite Element	-
4.	VCNT_4000	4,000	40.0	1.0	640	Quantum Mechanics	[19]
5.	Kuu/Muu	7,102	47.9	24.0	488	Structural	[9]
6.	fv1	9,604	8.9	1.0	453	2D/3D	[9]
7.	FDmesh	62,500	5.0	1.0	1,032	2D	-
8.	qa8fk/qa8fm	66,172	25.1	25.1	5,270	3D Acoustic	[9]

relative eigenvalue error for the same problem where $n_{ev} = 50$ and p varies. In summary, increasing the number of interface variables leads to lower accuracy unless more eigenvector derivatives are computed.

Figure 5.2 considers the opposite scenario where U is set as in Algorithm 4.3 and $Y = [y^{(1)}, \ldots, y^{(n_{ev})}]$. Here, the asymptotic order of the approximation error is fixed (i.e., quartic), and increasing the value of κ reduces the upper bound of the approximation error.

5.2. General pencils. Throughout the rest of this section we assume that matrices U and Y are set as described in Algorithm 4.1. Details on the numerical approximation of the first and second derivatives of each computed eigenvector of matrix $S(\alpha)$ are given in the appendix.[5]

We consider the application of Algorithm 4.1 on a set of model problems and matrix pencils obtained by the SuiteSparse[6] Matrix Collection [9], and the Elses[7] matrix library [19]. Details can be found in Table 5.2. The matrix pencil FEmesh represents a finite elements discretization of the Dirichlet eigenvalue problem on a $[-1, 1] \times [-1, 1]$ plane where the Laplacian is discretized using linear elements with target maximum mesh edge length of $h = 0.05$. The matrix FDmesh represents the

[5]The linear system solver we choose is preconditioned MINRES with a stopping tolerance of 1.0×10^{-3} and a maximum number of five preconditioned iterations, whichever occurs first. The convergence criterion is applied on the unpreconditioned residuals. For preconditioning we use matrix $S(\alpha)$ combined with deflation.

[6]https://sparse.tamu.edu/.

[7]http://www.elses.jp/matrix/.

TABLE 5.3
Maximum relative error of the approximation of the $n_{ev} = 50$ algebraically smallest eigenvalues returned by Algorithm 4.1 as κ varies.

	$\{y\}$	$\{y, dy\}$			$\{y, dy, d^2y\}$		
	$\kappa = 20$	$\kappa = 10$	$\kappa = 20$	$\kappa = 40$	$\kappa = 10$	$\kappa = 20$	$\kappa = 40$
Si2	5.0×10^{-4}	1.4×10^{-3}	9.4×10^{-4}	6.1×10^{-4}	7.7×10^{-4}	1.2×10^{-4}	2.5×10^{-5}
nos3	3.4×10^{-3}	1.4×10^{-3}	5.4×10^{-4}	2.7×10^{-4}	4.8×10^{-3}	3.6×10^{-4}	9.9×10^{-5}
FEmesh	1.7×10^{-3}	2.1×10^{-3}	1.4×10^{-3}	8.9×10^{-4}	9.1×10^{-4}	2.4×10^{-4}	1.1×10^{-4}
VCNT_4000	4.2×10^{-3}	3.6×10^{-3}	2.6×10^{-3}	9.0×10^{-4}	6.7×10^{-3}	9.4×10^{-4}	2.7×10^{-4}
{K,M}uu	7.3×10^{-3}	1.4×10^{-2}	9.6×10^{-3}	7.1×10^{-3}	2.6×10^{-3}	4.8×10^{-4}	1.5×10^{-4}
fv1	3.4×10^{-3}	1.4×10^{-2}	7.4×10^{-4}	3.0×10^{-4}	6.7×10^{-3}	4.1×10^{-4}	3.1×10^{-5}
FDmesh	1.9×10^{-3}	1.9×10^{-3}	1.7×10^{-3}	9.4×10^{-4}	2.4×10^{-3}	9.1×10^{-3}	2.4×10^{-4}
qa8f{k,m}	4.7×10^{-3}	9.0×10^{-3}	5.5×10^{-3}	4.2×10^{-3}	6.1×10^{-3}	2.2×10^{-3}	9.5×10^{-4}

250×250 finite difference discretization of the Dirichlet eigenvalue problem discussed in the previous section. With the exception of the matrix pencils FEmesh, Kuu/Muu, and qa8fk/qa8fm, all other pencils are of standard form.

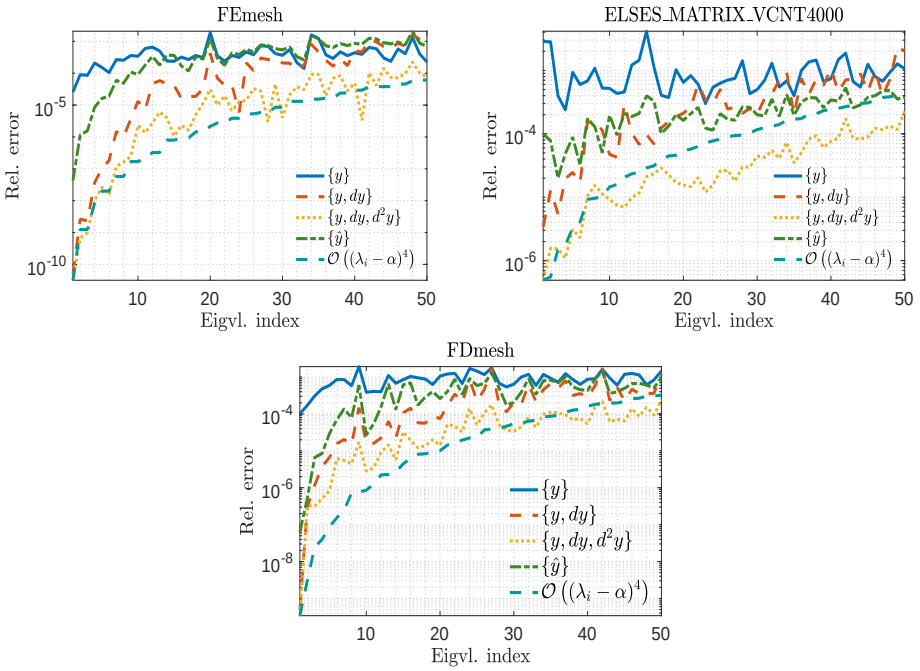
Table 5.3 lists the maximum relative error in the approximation of the $n_{ev} = 50$ algebraically smallest eigenvalues returned by Algorithm 4.1 for all pencils listed in Table 5.2. Naturally, increasing κ enhances overall accuracy, and this enhancement becomes greater as the interface projection subspace also improves. Note though that the entries associated with the choices $\{y, dy\}$ and $\{y, dy, d^2y\}$ are now closer to each other than what was shown in section 5.1. This is owed to (a) the inexact computation of the first and second derivatives and (b) the fact that the asymptotic order of the error is the same in both cases since only a first-order approximation of the resolvent B_λ^{-1} is considered.

Finally, we compare the accuracy achieved by Algorithm 4.1 against that of our own variant of the AMLS method (termed "p-way AMLS" and denoted by $\{\hat{y}\}$). For reasons of fairness and easiness in the interpretation of our comparisons, our own variant of AMLS is identical to Algorithm 4.1 except that matrix Y is formed as $Y = [\hat{y}^{(1)}, \ldots, \hat{y}^{(3n_{ev})}]$ where $\hat{y}^{(i)} \in \mathbb{R}^s$ denotes the eigenvector associated with the ith algebraically smallest eigenvalue of the pencil $(S(\alpha), -S'(\alpha))$. Our variant of AMLS is more accurate (but slower) than standard AMLS described in [7, 10].

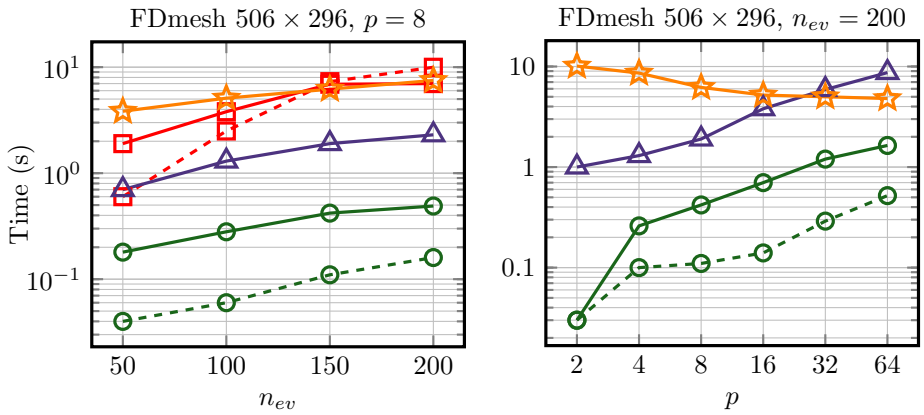
Figure 5.3 plots the relative error in the approximation of the $n_{ev} = 50$ algebraically smallest eigenvalues returned by Algorithm 4.1 and p-way AMLS when applied on a subset of the pencils listed in Table 5.2. In summary, p-way AMLS is more accurate than the $\{y\}$ variant of Algorithm 4.1 but not as good as the variants $\{y, dy\}$ and $\{y, dy, d^2y\}$, especially for those eigenvalues located closer to α. This performance gap increases favorably for Algorithm 4.1 as the projection subspace associated with interior variables improves.

5.3. Comparisons against shift-and-invert Lanczos. This section presents wall-clock time comparisons between Algorithm 4.1 (variant $\{y, dy, d^2y\}$) and implicitly restarted shift-and-invert Lanczos with full orthogonalization applied directly to the pencil (A, M). We will refer to the latter as IRSIL. The maximum dimension of the Krylov subspace was set to $2n_{ev}$. As our test matrix we choose a five-point 506×296 finite difference discretization of the Dirichlet eigenvalue problem.

Figure 5.4 plots sequential wall-clock times of the individual steps of IRSIL and Algorithm 4.1 when exploiting both schemes to approximate the n_{ev} algebraically smallest eigenvalues of the discretized Laplacian (left subfigure). In total, IRSIL required 2.5, 6.3, 14.2, and 17.0 seconds to approximate the $n_{ev} = 50,\ 100,\ 150$, and



570 FIG. 5.3. *Relative error of the approximation of the $n_{ev} = 50$ algebraically smallest eigenvalues*
571 *returned by Algorithm* 4.1 *and p-way AMLS ($\{\hat{y}\}$). We also plot the curve $(\lambda - \alpha)^4$ adjusted so that*
572 *its first entry is equal to that of the curve $\{y, dy, d^2y\}$.*



573 FIG. 5.4. *Sequential wall-clock times of IRSIL (applied to the pencil $(A - \alpha M, M)$) and Al-*
574 *gorithm* 4.1 *(variant $\{y, dy, d^2y\}$) for various values of n_{ev}. For IRSIL, we report the amount of*
575 *time spent on applying $(A - \alpha M)^{-1}$ (solid) and orthogonalization (dashed) separately (□). For*
576 *Algorithm* 4.1 *we report the amount of time spent on* (a) *computing eigenvectors $y_1(\alpha), \ldots, y_{n_{ev}}(\alpha)$*
577 *(o),* (b) *approximating the two leading eigenvector derivatives by pseudoblock MINRES (△), and*
578 (c) *executing Algorithm* 4.3 *with $\kappa = 40$ (☆). Steps* (a) *and* (b) *form Algorithm* 4.2. *Notice that for*
579 *step* (a) *we report the amount of time spent on applying the operator (in this case $S(\alpha)^{-1}$) (solid)*
580 *separately from that spent on orthogonalization (dashed). Left: fix $p = 8$ and vary n_{ev}. Right: fix*
581 *$n_{ev} = 200$ and vary p.*

$n_{ev} = 200$, algebraically smallest eigenvalues and associated eigenvectors up to a residual norm 1.0×10^{-8}. On the other hand, Algorithm 4.1 with the default choice $p = 8$ required 4.9, 6.7, 8.7, and 10.5 seconds, respectively. As n_{ev} increases, IRSIL becomes increasingly slower than Algorithm 4.1 due to its increasing orthogonalization cost. For example, when $n_{ev} = 200$, IRSIL spent about 10 seconds in orthogonalization, while Algorithm 4.1 only required about a quarter of a second. The maximum eigenvalue error returned by Algorithm 4.1 for any value of n_{ev} tested was $O(10^{-4})$. Figure 5.4 also plots the sequential execution timing of Algorithm 4.1 as p varies and $n_{ev} = 200$ is fixed (right subfigure). Increasing the value of p leads to a greater number of interface variables (and thus increased orthogonalization costs), while solving linear systems with matrix $S(\alpha)$ also becomes more expensive. On the other hand, larger values of p generally decrease the amount of time spent in Algorithm 4.3. Note that for $p = 64$ the maximum error returned by Algorithm 4.1 was of the order $O(10^{-3})$.

Table 5.4 lists the total number of iterations required by implicitly restarted Lanczos to (a) compute the sought eigenvectors of matrix $S(\alpha)$ and (b) solve the original eigenvalue problem (i.e., compute the n_{ev} sought eigenpairs of (A, M)). For Algorithm 4.1 we considered setting $p = 4, 16$, and $p = 32$, respectively. In summary, we observe two patterns. First, as p increases so does the number of iterations required by Lanczos. In particular, increasing the size of the Schur complement matrix typically leads to a denser spectrum in matrix $S(\alpha)$ since the eigenvalues of the latter interlace those of $(A - \alpha M, M)$. This can be verified in Figure 5.5, where we plot the 250 algebraically smallest eigenvalues of matrices FDmesh 506×296 and $S(\alpha \equiv 0)$ as $p = 4, 16$, and $p = 32$, respectively. Second, working with the Schur complement matrix becomes the only practical approach if shift-and-invert is not possible, since applying implicitly restarted Lanczos to (A, M) can lead to very slow convergence and thus high orthogonalization costs.

A preliminary distributed memory implementation of Algorithm 4.1 was built on top of the PETSc library [5].[8] Message passing between different processes was achieved by means of the Message Passing Interface (MPI), a standard application programming interface for message passing applications [14]. We considered only one thread per MPI process, and the number of MPI processes will be equal to the number of subdomains p. To allow for a fair comparison we implemented our own version of IRSIL instead of that in the PETSc-based, state-of-the-art eigenvalue package

TABLE 5.4

Total number of iterations performed by implicitly restarted Lanczos to compute the sought eigenvectors of (a) *matrix* $S(\alpha)$ *and* (b) *the original eigenvalue problem. Flag* `sa` *(*`sm`*) indicates the absence (presence) of shift-and-invert acceleration. An* **F** *flag indicates that not all eigenvectors were computed after* 20 *restarts, where the maximum Krylov subspace dimension was set equal to* $2n_{ev}$.

	$n_{ev} = 100$				$n_{ev} = 200$			
	Alg. 4.1: $p = 4$	$p = 16$	$p = 32$	(A, I)	Alg. 4.1: $p = 4$	$p = 16$	$p = 32$	(A, I)
"`sa`"	605	1,027	1,289	**F**	775	1,440	1,723	**F**
"`sm`"	304	310	315	317	400	440	472	553

[8] Our code builds on top of the implementation featured in [24] and was compiled using real arithmetic. The source files were compiled with the Intel MPI compiler `mpiicpc`, using the -O3 optimization level. The linear system solutions with the distributed matrices $A - \alpha M$ and $S(\alpha)$ were computed by the Multifrontal Massively Parallel Sparse Direct Solver [2] and those with the block-diagonal matrix B_α by MKL PARDISO [1].

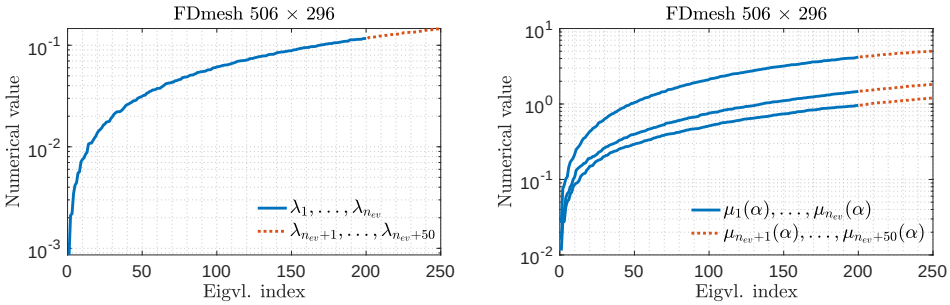
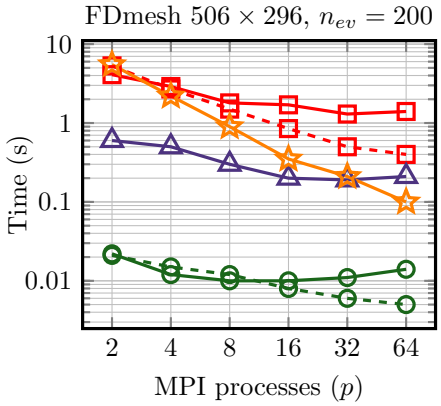


FIG. 5.5. *Plot of the* 250 *algebraically smallest eigenvalues of matrix* FDmesh 506×296 *(left), and matrix* $S(\alpha \equiv 0)$ *setting* $p = 4, 16, 32$ *(right). On the right subfigure, the top, middle, and bottom curves indicate the choice* $p = 4, 16,$ *and* $p = 32$*, respectively. The solid part of each curve indicates the algebraically smallest* $n_{ev} = 200$ *eigenvalues of each matrix.*



Module	$p = 4$	$p = 16$	$p = 64$
IRSIL (sol.)	0.60	0.28	0.16
IRSIL (orth.)	0.89	0.73	0.40
IRSIL	0.76	0.41	0.15
Alg. 4.3	1.25	1.52	1.71
Alg. 4.2	0.55	0.34	0.14
Alg. 4.1	1.18	1.40	0.66

FIG. 5.6. *Left: Distributed memory wall-clock times (strong scaling). Right: Ratio of true speedup to theoretical speedup (efficiency) computed as* $\mathrm{T}_s/(p\mathrm{T}_p)$*, where* T_p *denotes the wall-clock time using* p *MPI processes. For IRSIL we set* T_s *equal to its sequential wall-clock time shown in Figure* 5.4*. For Algorithm* 4.1 *and its individual steps we set* $\mathrm{T}_s = 2\mathrm{T}_2$*.*

SLEPc [17]. Our experiments were performed at the `Mesabi` cluster of the Minnesota Supercomputing Institute (https://www.msi.umn.edu/).

Figure 5.6 plots the distributed memory wall-clock times of different modules of IRSIL and Algorithm 4.1 as $n_{ev} = 200$ and $p = 2, 4, 8, 16, 32,$ and $p = 64$. The corresponding efficiency for some of these values of p is listed in the accompanying table. Algorithm 4.3 is the most scalable operation of Algorithm 4.1 and in this example provides superlinear speedups. As p increases, solving the distributed linear systems in Lanczos becomes the least scalable operation for both algorithms. Nonetheless, increasing p can still lead to a higher efficiency in Algorithm 4.1 up to a point where the amount of time spent in Algorithm 4.3 is small and thus the efficiency of the former is mainly determined by the efficiency of performing computations with the Schur complement matrix. This is the reason the efficiency of Algorithm 4.1 drops sharply from $p = 16$ to $p = 64$. This implies that increasing parallelism through increasing p is nonoptimal, and additional parallel resources should be exploited in a hierarchical fashion. In total, Algorithm 4.1 is about $3\times$ $(5\times)$ faster than IRSIL when 16 (64) MPI processes are used.

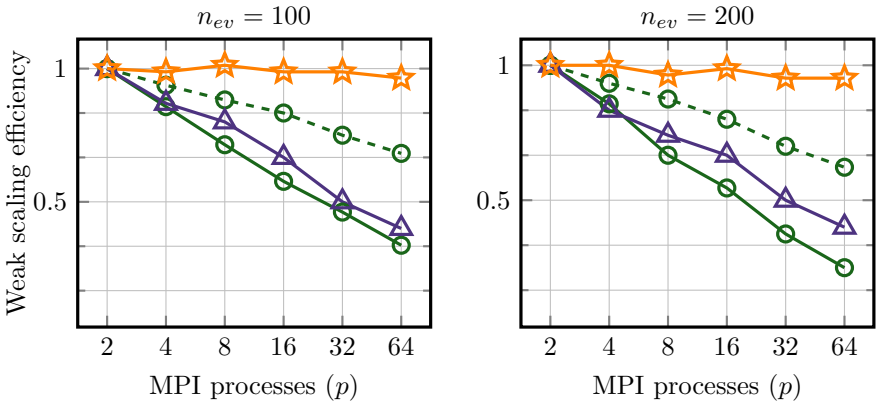


FIG. 5.7. *Weak scaling efficiency (computed as* T_2/T_p *where* T_p *denotes the wall-clock time of Algorithm* 4.1 *for* $p = 2, 4, 8, 16, 32, 64$*) of different modules of Algorithm* 4.1 *as* p *varies. The mesh size of the discretized Laplacian scaled as* $\sqrt{p}50 \times \sqrt{p}50$*. Left:* $n_{ev} = 100$*. Right:* $n_{ev} = 200$*.*

Finally, Figure 5.7 plots the weak scaling efficiency of different modules of Algorithm 4.1 for a Laplacian discretized on a square mesh with mesh size $\sqrt{p}50 \times \sqrt{p}50$. The number of sought eigenpairs was set to $n_{ev} = 100$, and $n_{ev} = 200$. Again we observe that the least scalable parts are those enabling the solution of distributed linear systems. Moreover, larger values of n_{ev} are more challenging in terms of scalability.

6. Summary and future work. This paper presented a domain decomposition technique for the computation of a few eigenpairs of symmetric generalized eigenvalue problems. The proposed technique is based on Rayleigh–Ritz projections onto subspaces formed by decoupling the original eigenvalue problem into two distinct subproblems, each one associated with the interface and interior variables of the domain, respectively. The part of the subspace associated with the interface variables of the global domain is formed by the eigenvectors and associated derivatives of a zeroth-order approximation of the nonlinear interface matrix-valued operator. On the other hand, the part of the subspace associated with the interior variables is formed in parallel among the different subdomains by exploiting local eigenmodes and approximations of resolvent expansions.

Future work includes a study on the effects which the number of interface variables has in the accuracy of the technique proposed in this paper. A parallel implementation with additional levels of parallelism (both distributed and shared memory), possibly combined with harmonic Rayleigh–Ritz projections, is in our short-term plans.

Appendix A. Analytical formulas. In this section we present analytical formulas for the computation of the first two derivatives of the eigenpairs $(\mu_i(\sigma), y_i(\sigma))$ of the matrix $S(\sigma),\ \sigma \in \mathbb{R}$.

PROPOSITION A.1. *The first and second derivatives of the matrix* $S(\sigma),\ \sigma \in \mathbb{R}$*, are given by*

$$S'(\sigma) = \frac{dS(\sigma)}{d\sigma} = -M_C - E_\sigma^T B_\sigma^{-1} M_B B_\sigma^{-1} E_\sigma + E_\sigma^T B_\sigma^{-1} M_E + M_E^T B_\sigma^{-1} E_\sigma$$

and

$$S''(\sigma) = \frac{d^2 S(\sigma)}{d\sigma^2} = 2\left(E_\sigma^T B_\sigma^{-1} M_B B_\sigma^{-1} M_B B_\sigma^{-1} E_\sigma + M_E^T B_\sigma^{-1} M_E\right) - 2\left(M_E^T B_\sigma^{-1} M_B B_\sigma^{-1} E_\sigma + E_\sigma^T B_\sigma^{-1} M_B B_\sigma^{-1} M_E\right),$$

respectively. The first and second derivatives of each eigenvalue curve $\mu_i(\sigma)$ are given by

$$\mu_i'(\sigma) = \frac{d\mu_i(\sigma)}{d\sigma} = \frac{y_i^T(\sigma) S'(\sigma) y_i(\sigma)}{y_i^T(\sigma) y_i(\sigma)}$$

and

$$\mu_i''(\sigma) = \frac{d^2\mu_i(\sigma)}{d\sigma^2} = \frac{y_i^T(\sigma) S''(\sigma) y_i(\sigma) + 2 y_i^T(\sigma) S'(\sigma) y_i'(\sigma)}{y_i^T(\sigma) y_i(\sigma)},$$

respectively. Finally, the first and second derivatives of each eigenvector $y_i(\sigma)$ satisfy the equations

$$\left(S(\sigma) - \mu_i(\sigma) I\right) y_i'(\sigma) = \left(\mu_i'(\sigma) I - S'(\sigma)\right) y_i(\sigma) \tag{A.1}$$

and

$$\left(S(\sigma) - \mu_i(\sigma) I\right) y_i''(\sigma) = \Big[\left(\mu_i''(\sigma) I - S''(\sigma)\right) y_i(\sigma) + 2\left(\mu_i'(\sigma) I - S'(\sigma)\right) y_i'(\sigma)\Big], \tag{A.2}$$

respectively, where $y_i'(\sigma) = (\frac{dy_i(\zeta)}{d\zeta})_{\zeta=\sigma}$ and $y_i''(\sigma) = (\frac{d^2 y_i(\zeta)}{d\zeta^2})_{\zeta=\sigma}$.

Differentiating the normalization condition $y_i^T(\sigma) y_i(\sigma) = 1$ gives $y_i^T(\sigma) y_i'(\sigma) = 0$, and thus the leading derivative of the eigenvector $y_i(\sigma)$ can be computed by solving the linear system in (A.1). On the other hand, solving the linear system in (A.2) will only provide the second derivative up to the direction $y_i(\sigma)$ (note that $y_i^T(\sigma) y_i''(\sigma) = -\|y_i'(\sigma)\|_2^2$). Nonetheless, the latter eigenvector direction already exists in the subspace $\mathbf{span}\,(Y)$ (Y is defined in Algorithm 4.2). Throughout the rest of this paper, when we refer to "$y_i''(\sigma)$" it should be understood that we actually refer to the solution of the linear system in (A.2).

Remark 2. When M is equal to the identity matrix, the first and second derivatives of the matrix-valued function $S(\zeta)$ evaluated at σ simplify to the block-diagonal matrices $S'(\sigma) = -I - E_\sigma^T B_\sigma^{-2} E_\sigma$, and $S''(\sigma) = -E_\sigma^T B_\sigma^{-3} E_\sigma$.

Appendix B. Computation of eigenvector derivatives. The computation of the first and second derivatives of each eigenvector $y_i(\sigma)$, $i = 1, 2, \ldots, s$, requires the application of an iterative solver, e.g., the minimum residual (MINRES) Krylov subspace method [8, 32], to the solution of the singular linear system

$$\left(S(\sigma) - \mu_i(\sigma) I\right) x = b^{(i)}, \tag{B.1}$$

where $b^{(i)} \in \mathbb{R}^s$ is defined as

$$b^{(i)} := \begin{cases} \left(\mu_i'(\sigma) I - S'(\sigma)\right) y_i(\sigma) & \text{(to compute } y_i'(\sigma)\text{)}, \\ \left(\mu_i''(\sigma) I - S''(\sigma)\right) y_i(\sigma) + 2\left(\mu_i'(\sigma) I - S'(\sigma)\right) y_i'(\sigma) & \text{(to compute } y_i''(\sigma)\text{)}. \end{cases}$$

The eigenvalues of the matrix $S(\sigma) - \mu_i(\sigma) I$ are equal to $\{\mu_k(\sigma) - \mu_i(\sigma)\}_{k=1,2,\ldots,s}$.

Let the n_{ev} computed eigenvalues of $S(\sigma)$ be indexed as $\mu_{\psi_1}(\sigma), \ldots, \mu_{\psi_{n_{ev}}}(\sigma)$, where $\psi_1 \leq i \leq \psi_{n_{ev}}$. We can enhance the convergence rate of MINRES applied to

(B.1) by explicitly removing the components along the directions associated with the computed eigenvectors of matrix $S(\sigma)$. In particular, the solution of the linear system with each matrix $S(\sigma) - \mu_i(\sigma)I$ is split into two phases. During the first phase we apply MINRES to the deflated linear equation

$$\mathcal{P}(S(\sigma) - \mu_i(\sigma)I)\bar{x} = \mathcal{P}b^{(i)}, \tag{B.2}$$

where $\mathcal{P} = I - W(W^TW)^{-1}W^T$, $C = [y_{\psi_1}(\sigma), \ldots, y_{\psi_{i-1}}(\sigma), y_{\psi_{i+1}}(\sigma), \ldots, y_{\psi_{n_{ev}}}(\sigma)]$, and $W = (S(\sigma) - \mu_i(\sigma)I)C$. As soon as the deflated linear equation in (B.2) is solved, the solution of the original linear system is formed as

$$x = \mathcal{Q}\bar{x} + (I - \mathcal{Q})b^{(i)}, \tag{B.3}$$

where $\mathcal{Q} = I - C(W^TW)^{-1}W^T$. Details on deflated MINRES can be found in [11, 12].

In case a symmetric factorization of $S(\sigma)$ is already at hand, this can be further exploited to compute the solution of the linear system $(S(\sigma) - \mu_i(\sigma)I)x = b^{(i)}$ by solving the linear system[9]

$$S(\sigma)^{-1}(S(\sigma) - \mu_i(\sigma)I)x = S(\sigma)^{-1}b^{(i)}.$$

The matrix $S(\sigma)^{-1}(S(\sigma) - \mu_i(\sigma)I)$ is symmetric, and its eigenvalues are equal to $\{\frac{\mu_k(\sigma) - \mu_i(\sigma)}{\mu_k(\sigma)}\}_{k=1,\ldots,s}$. This preconditioned linear system can also be combined with deflation. In the latter case MINRES is applied to the equation $\mathcal{P}S(\sigma)^{-1}(S(\sigma) - \mu_i(\sigma)I)x = \mathcal{P}S(\sigma)^{-1}b^{(i)}$ and the original linear system solution is obtained exactly as in (B.3).

COROLLARY B.1. *Let the eigenvalues of matrix $S(\sigma)$ be ordered as*

$$\mu_1(\sigma) \leq \cdots \leq \mu_{\psi_1 - 1}(\sigma) \leq \mu_{\psi_1}(\sigma) \leq \cdots \leq \mu_{\psi_{n_{ev}}}(\sigma) \leq \mu_{\psi_{n_{ev}}+1}(\sigma) \leq \cdots \leq \mu_s(\sigma).$$

Then, the effective condition number of the matrix $\mathcal{P}(S(\sigma) - \mu_i(\sigma)I)$ is equal to

$$\kappa_{MR,i} = \frac{\mathbf{max}\{|\mu_1(\sigma) - \mu_i(\sigma)|, |\mu_s(\sigma) - \mu_i(\sigma)|\}}{\mathbf{min}\{|\mu_{\psi_1 - 1}(\sigma) - \mu_i(\sigma)|, |\mu_{\psi_{n_{ev}}+1}(\sigma) - \mu_i(\sigma)|\}}, \quad i = \psi_1, \ldots, \psi_{n_{ev}}.$$

Similarly, the effective condition number of the preconditioned matrix $\mathcal{P}S(\sigma)^{-1}(S(\sigma) - \mu_i(\sigma)I)$ is equal to

$$\kappa_{PMR,i} = \frac{\mathbf{max}\left\{\left|1 - \frac{\mu_i(\sigma)}{\mu_1(\sigma)}\right|, \left|1 - \frac{\mu_i(\sigma)}{\mu_s(\sigma)}\right|\right\}}{\mathbf{min}\left\{\left|1 - \frac{\mu_i(\sigma)}{\mu_{\psi_1 - 1}(\sigma)}\right|, \left|1 - \frac{\mu_i(\sigma)}{\mu_{\psi_{n_{ev}}+1}(\sigma)}\right|\right\}}.$$

If we do not deflate the n_{ev} computed eigenvectors of the matrix $S(\sigma)$, the denominators in Corollary B.1 become $\mathbf{min}\{|\mu_{i-1}(\sigma) - \mu_i(\sigma)|, |\mu_{i+1}(\sigma) - \mu_i(\sigma)|\}$ and $\mathbf{min}\{|1 - \frac{\mu_i(\sigma)}{\mu_{i-1}(\sigma)}|, |1 - \frac{\mu_i(\sigma)}{\mu_{i+1}(\sigma)}|\}$, respectively. When $0 \leq \sigma \leq \lambda_{\mathbf{min}}(A, M)$, deflated MINRES can be replaced by a deflated variant of the conjugate gradient method [18]; see, for example, [35, 37].

Figure B.1 plots the number of iterations required by MINRES to compute the eigenvector derivatives $y_i'(\sigma)$, $i = 1, \ldots, n_{ev}$, up to a tolerance equal to 1.0×10^{-8}, for a 253×148 finite difference discretization of the Dirichlet eigenvalue problem.

[9] Recall that MINRES requires an SPD preconditioner. If $S(\sigma)$ is not SPD, then alternative Kyrlov subspace iterative linear system solvers should be considered.

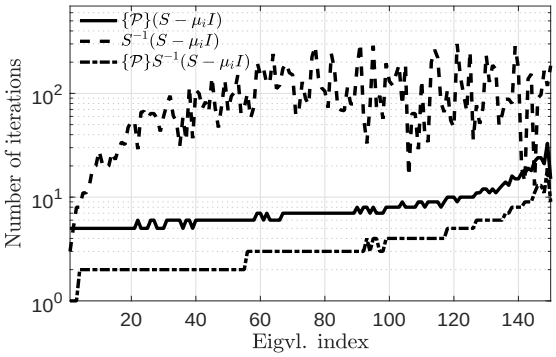


FIG. B.1. *Number of (deflated) MINRES iterations with/without preconditioning to compute the eigenvector derivatives* $y_i'(\sigma)$, $i = 1, \ldots, n_{ev}$. *Legend: "-" (deflation without preconditioning), "-" (preconditioning without deflation), and "-." (preconditioning with deflation).*

Combining preconditioning by matrix $S(\sigma)$ while deflating the invariant subspace associated with the n_{ev} computed eigenvectors of the matrix $S(\sigma)$ proved to be the fastest scheme in terms of iterations required to achieve convergence. Moreover, linear systems corresponding to eigenvector derivatives associated with eigenvalues that lie closer to σ converge faster due to a smaller effective condition number.

Acknowledgments. The author would like to thank the anonymous referees, whose detailed comments and suggestions led to considerable improvements in the quality of the present manuscript. In addition, the author is indebted to Daniel Kressner, Anthony Austin, and Yousef Saad for their feedback and encouragement to pursue the work featured in this manuscript.

REFERENCES

[1] *Fortran Compiler XE* 14.0 *for Linux*, Intel Corporation, 2018.
[2] P. R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent, *A fully asynchronous multifrontal solver using distributed dynamic scheduling*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 15–41.
[3] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, Vol. 9, SIAM, Philadelphia, 1999.
[4] A. Andrew and R. Tan, *Computation of derivatives of repeated eigenvalues and the corresponding eigenvectors of symmetric matrix pencils*, SIAM J. Matrix Anal. Appl., 20 (1998), pp. 78–100, https://doi.org/10.1137/S0895479896304332.
[5] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, *PETSc Users Manual*, Tech. Report ANL-95/11, Revision 3.11, Argonne National Laboratory, 2019, https://www.mcs.anl.gov/petsc.
[6] C. Bekas and Y. Saad, *Computation of smallest eigenvalues using spectral Schur complements*, SIAM J. Sci. Comput., 27 (2006), pp. 458–481.
[7] 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.
[8] S.-C. T. Choi, C. C. Paige, and M. A. Saunders, *MINRES-QLP: A Krylov subspace method for indefinite or singular symmetric systems*, SIAM J. Sci. Comput., 33 (2011), pp. 1810–1836.
[9] 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.

[10] 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.

[11] A. Gaul, M. H. Gutknecht, J. Liesen, and R. Nabben, *A framework for deflated and augmented Krylov subspace methods*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 495–518.

[12] A. Gaul and N. Schlömer, *Preconditioned recycling Krylov subspace methods for self-adjoint problems*, Electron. Trans. Numer. Anal., 44 (2015), pp. 522–547.

[13] 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.

[14] W. Gropp, W. D. Gropp, E. Lusk, A. D. F. E. E. Lusk, and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message-passing Interface*, Vol. 1, MIT Press, Cambridge, MA, 1999.

[15] W. D. Gropp, *Parallel computing and domain decomposition*, in Proceedings of the Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations, Philadelphia, 1992, pp. 349–361.

[16] A. Hannukainen, J. Malinen, and A. Ojalammi, *Efficient Solution of Symmetric Eigenvalue Problems from Families of Coupled Systems*, preprint, arXiv:1806.07235, 2018.

[17] V. Hernandez, J. E. Roman, and V. Vidal, *SLEPC: A scalable and flexible toolkit for the solution of eigenvalue problems*, ACM Trans. Math. Software, 31 (2005), pp. 351–362.

[18] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. National Bureau of Standards, 49 (1952), pp. 409–436.

[19] T. Hoshi, H. Imachi, A. Kuwata, K. Kakuda, T. Fujita, and H. Matsui, *Numerical aspect of large-scale electronic state calculation for flexible device material*, Jpn. J. Ind. Appl. Math., 36 (2019), pp. 685–698.

[20] V. Kalantzis, *Domain Decomposition Algorithms for the Solution of Sparse Symmetric Generalized Eigenvalue Problems*, Ph.D. thesis, University of Minnesota, 2018.

[21] V. Kalantzis, *A spectral Newton-Schur algorithm for the solution of symmetric generalized eigenvalue problems*, Electron. Trans. Numer. Anal., 52 (2020), pp. 132–153.

[22] 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., 25 (2018), e2154.

[23] V. Kalantzis, R. Li, and Y. Saad, *Spectral Schur complement techniques for symmetric eigenvalue problems*, Electron. Trans. Numer. Anal., 45 (2016), pp. 305–329.

[24] V. Kalantzis, Y. Xi, and Y. Saad, *Beyond automated multilevel substructuring: Domain decomposition with rational filtering*, SIAM J. Sci. Comput., 40 (2018), pp. C477–C502.

[25] 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.

[26] T. Kato, *Perturbation Theory for Linear Operators*, 2nd ed., Grundlehren Math. Wiss. 132, Springer, Berlin, 1976, https://cds.cern.ch/record/101545.

[27] J. Kestyn, V. Kalantzis, E. Polizzi, and Y. Saad, *PFEAST: A high performance sparse eigenvalue solver using distributed-memory linear solvers*, in SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, 2016, pp. 178–189.

[28] J. H. Ko and Z. Bai, *High-frequency response analysis via algebraic substructuring*, Internat. J. Numer. Methods Engrg., 76 (2008), pp. 295–313.

[29] L. Komzsik and T. Rose, *Parallel methods on large-scale structural analysis and physics applications substructuring in MSC/NASTRAN for large scale parallel applications*, Computing Systems in Engineering, 2 (1991), pp. 167–173, http://dx.doi.org/10.1016/0956-0521(91)90017-Y.

[30] S. Lui, *Kron's method for symmetric eigenvalue problems*, J. Comput. Appl. Math., 98 (1998), pp. 35–48.

[31] K. Meerbergen and Z. Bai, *The Lanczos method for parameterized symmetric linear systems with multiple right-hand sides*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1642–1662.

[32] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[33] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Classics in Appl. Math. 20, SIAM, Philadelphia, 1998, https://doi.org/10.1137/1.9781611971163.

[34] Y. Saad, *ILUM: A multi-elimination ILU preconditioner for general sparse matrices*, SIAM J. Sci. Comput., 17 (1996), pp. 830–847.

[35] Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc'h, *A deflated version of the conjugate gradient algorithm*, SIAM J. Sci. Comput., 21 (2000), pp. 1909–1926, https://doi.org/10.1137/S1064829598339761.

[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] A. Stathopoulos and K. Orginos, *Computing and deflating eigenvalues while solving multiple right-hand side linear systems with an application to quantum chromodynamics*, SIAM J. Sci. Comput., 32 (2010), pp. 439–462.

[38] Y. Su, T. Lu, and Z. Bai, 2*D Eigenvalue Problems* I*: Existence and Number of Solutions*, preprint, arXiv:1911.08109, 2019.

[39] U. Von Luxburg, *A tutorial on spectral clustering*, Stat. Comput., 17 (2007), pp. 395–416.

[40] 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.

[41] H. Zhang, B. Smith, M. Sternberg, and P. Zapol, *SIPs: Shift-and-invert parallel spectral transformations*, ACM Trans. Math. Software, 33 (2007), pp. 111–127.