MULTISCALE MODEL. SIMUL.
Vol. 17, No. 1, pp. 260–306

A FAST HIERARCHICALLY PRECONDITIONED EIGENSOLVER BASED ON MULTIRESOLUTION MATRIX DECOMPOSITION[*]

THOMAS Y. HOU[†], DE HUANG[‡], KA CHUN LAM[§], AND ZIYUN ZHANG[¶]

Abstract. In this paper we propose a new iterative method to hierarchically compute a relatively large number of leftmost eigenpairs of a sparse symmetric positive matrix under the multiresolution operator compression framework. We exploit the well-conditioned property of every decomposition component by integrating the multiresolution framework into the implicitly restarted Lanczos method. We achieve this combination by proposing an extension-refinement iterative scheme, in which the intrinsic idea is to decompose the target spectrum into several segments such that the corresponding eigenproblem in each segment is well-conditioned. Theoretical analysis and numerical illustration are also reported to illustrate the efficiency and effectiveness of this algorithm.

Key words. leftmost eigenpairs, sparse symmetric positive definite, multiresolution matrix decomposition, implicitly restarted Lanczos method, preconditioned conjugate gradient method, eigenpair refinement

AMS subject classifications. 15A18, 15A12, 65F08, 65F15

DOI. 10.1137/18M1180827

1. Introduction.

The computation of eigenpairs for large and sparse matrices is one of the most fundamental tasks in many scientific applications. For example, the leftmost eigenpairs (i.e., the N smallest eigenpairs for some $N \in \mathbb{N}$) of a graph Laplacian L help in revealing the topological information of the corresponding network from real data. One illustrative example is that the multiplicity of the smallest eigenvalue λ_1 of L coincides with the number of connected components of the corresponding graph G. In particular, the second-smallest eigenvalue of L is well known as the algebraic connectivity or the Fiedler value of the graph G, which is applied to develop algorithms for graph partitioning [6, 18, 19]. Another important example regarding the use of leftmost eigenpairs is the computation of betweenness centrality of graphs as mentioned in [3, 4, 1]. Computing the leftmost eigenpairs of large and sparse symmetric positive definite (SPD) matrices also stemmed from the problem of predicting electronic properties in complex structural systems [9]. Such a prediction is achieved by solving the Schrödinger equation $\mathcal{H}\Psi = \mathcal{E}\Psi$, where $\mathcal{H}$ is the Hamiltonian operator for the system, $\mathcal{E}$ corresponds to the total energy, and $|\Psi(r)|^2$ represents the charge density at location r. Solving this equation using the self-consistent field requires computing the eigenpairs of $\mathcal{H}$ repeatedly, which dominates the overall computation cost of the overall iterations. Thus, an efficient algorithm to solve the eigenproblem is indispensable. Usage of leftmost eigenpairs can also be found in vibrational analysis in mechanical engineering [17]. In [7], authors also suggest that the leftmost eigenpairs of the covariance matrix between residues are important to extract functional

[*]Received by the editors April 16, 2018; accepted for publication (in revised form) December 3, 2018; published electronically January 30, 2019.
http://www.siam.org/journals/mms/17-1/M118082.html
Funding: This research was supported in part by the NSF grants DMS-1318377 and DMS-1613861.
[†]Applied and Computational Mathematics, Caltech, Pasadena, CA, 91125 (hou@cms.caltech.edu).
[‡]ACM, Caltech, Pasadena, CA 91125 (dhuang@caltech.edu).
[§]CMS, Caltech, Pasadena, CA 91125 (kclam@caltech.edu).
[¶]SMS, Peking University, Pasadena, CA 91125 (zhangziyun@pku.edu.cn).

and structural information about protein families. Efficient algorithms for computing p smallest eigenpairs for relatively large p are therefore crucial in various applications.

As most of the linear systems from engineering problems or networks are typically large and sparse in nature, iterative methods are preferred. Recently, several efficient algorithms have been developed to obtain leftmost eigenpairs of A. These include the Jacobi–Davidson method [26], implicit restarted Arnoldi/Lanczos method [5, 28, 13], and the deflation-accelerated conjugate gradient (DACG) method [2]. All these methods give promising results [1, 16], especially for finding a small number of leftmost eigenpairs. Other methods for computing accurate leftmost eigenpairs using hierarchical refinement/correction techniques were proposed in [30, 14, 31]. However, as reported in [16], the implicit restarted Lanczos method (IRLM) is still the best performing algorithm when a large number of smallest eigenpairs are required. Therefore, it is highly desirable to develop a new algorithm, based on the architecture of the IRLM, that can further optimize the performance.

The main purpose of this paper is to explore the possibility of exploiting the advantageous energy decomposition framework under the architecture of the IRLM. In particular, we propose a new spectrum-preserving preconditioned hierarchical eigensolver for computing a large number of smallest eigenpairs. This eigensolver takes full advantage of the intrinsic structure of the given matrix, the nice spectral property in the Lanczos procedure, and also the preconditioning characteristics of the conjugate gradient method. Given a sparse symmetric positive matrix A which is assumed to be energy decomposable (see subsection 2.1 or section 2 for details), we integrate the well-behaved matrix properties that are inherited from the multiresolution matrix decomposition (MMD) with IRLM. The preconditioner we propose for the conjugate gradient method can also preserve the narrowed residual spectrum of A during the Lanzcos procedure. Throughout this paper, theoretical performance of our proposed algorithm is analyzed rigorously and we conduct a number of numerical experiments to verify the efficiency and effectiveness of the algorithm in practice. To summarize, our contributions are threefold:

- We propose a hierarchical framework to compute a relatively large number of leftmost eigenpairs of a sparse symmetric positive matrix. This framework employs the MMD algorithm to further optimize the performance of IRLM. In particular, a specially designed spectrum-preserving preconditioner is introduced for the conjugate gradient method to compute $x = A^{-1}b$ for some vector b.
- The proposed framework improves the running time of finding m_{tar} smallest eigenpairs of a matrix $A \in \mathbb{R}^{n\times n}$ from $O(m_{tar} \cdot \kappa(A) \cdot nnz(A) \log \frac{1}{\varepsilon})$ (which is achieved by the classical IRLM) to $O(m_{tar} \cdot nnz(A) \cdot (\log \frac{1}{\varepsilon} + \log n)^C)$, where $\kappa(A)$ is the condition number of A, $nnz(\cdot)$ is the number of nonzero entries, and C is some small constant independent of $m_{tar}, nnz(A)$, and $\kappa(A)$.
- We also provide a rigorous analysis on both the accuracy and the asymptotic computational complexity of our proposed algorithm. This ensures the correctness and efficiency of the algorithm even in large-scale, ill-conditioned scenarios.

Remark 1.1. In many real applications, the operator A may not be explicitly stored entrywisely and only the evaluation of Ax is available. In this situation, our proposed algorithm also works as it only requires the storage of the stiffness matrices corresponding to some hierarchical basis $\boldsymbol{\Psi}$ according to the accuracy requirement. To construct these stiffness matrices, we only need to evaluate Ax. Therefore, for the ease of discussion, we simply assume that the given operator A is a finite-dimensional accessible matrix.

1.1. Overview of the algorithm. In this paper, we propose and develop an iterative scheme under the framework of energy decomposition introduced in [10]. Under this framework, we can decompose $A^{-1} \in \mathbb{R}^{n\times n}$ into

$$A^{-1} = P_{\mathcal{U}}^A A^{-1} + P_{\Psi}^A A^{-1} := P_{\mathcal{U}}^A A^{-1} + \Theta,$$

where $[\mathcal{U}, \Psi]$ corresponds to a basis of $\mathbb{R}^n$; $P_{\mathcal{U}}^A$ and P_{Ψ}^A are the corresponding subspace projections. Recursively, we can also consider Θ as a "new" A^{-1} and decompose Θ in the same manner. This will give an MMD of $A^{-1} = \sum_{k=1}^{K} P_{\mathcal{U}^{(k)}}^A A^{-1} + \Theta^{(K)}$. To illustrate, we first consider a one-level decomposition, i.e., $K = 1$. One important observation regarding this decomposition is that the spectrum of the original operator A^{-1} resembles that of the compressed operator Θ. In particular, if $\lambda_{i,\Theta}$ is the ith smallest eigenvalue of Θ and $\zeta_{i,\Theta}$ is the corresponding eigenvector, then $(\lambda_{i,\Theta}^{-1}, \zeta_{i,\Theta})$ is a good approximation of (λ_i^{-1}, q_i) for small λ_i, where (λ_i^{-1}, q_i) denotes the ith eigenpair of A^{-1}. These approximate eigenpairs $(\lambda_{i,\Theta}^{-1}, \zeta_{i,\Theta})$ can then be used as the initial approximation of the required eigenpairs. Notice that compression errors are introduced into these eigenpairs by the matrix decomposition. Therefore, a refinement procedure should be carried out to diminish these errors up to the prescribed accuracy. Once we obtain the refined eigenpairs, we may extend the spectrum in order to obtain the required number of eigenpairs. As observed in [16], the IRLM is the best performing algorithm when large eigenpairs are considered, so we therefore employ the Krylov subspace extension technique to extend the spectrum up to some prescribed control of the well-posedness. Intuitively, the MMD decomposes the spectrum of A^{-1} into different segments of different scales. Using a subset of the decomposed components to approximate A^{-1} yields a great reduction of the relative condition number. Thus, we can further trim down the complexity of the IRLM by approximating A^{-1} during the shifting process.

To generalize, we propose a hierarchical scheme to compute the leftmost eigenpairs of an energy decomposable matrix. Given the K-level multiresolution decomposition $\{\Theta^{(k)}\}_{k=1}^{K}$ of an energy decomposable matrix A, we first compute the eigendecomposition $[V_{ex}^{(K)}, D_{ex}^{(K)}]$ of $\Theta^{(K)}$ (with dimension $N^{(K)}$) corresponding to the coarsest level by using some standard direct method. Then we propose a compatible refinement scheme for both $V_{ex}^{(K)}$ and $D_{ex}^{(K)}$ to obtain $V_{ini}^{(K-1)}$ and $D_{ini}^{(K-1)}$, which will then be the initial spectrum in the consecutive finer level. The efficiency of the cross-level refinement is achieved by a modified version of the orthogonal iteration with the Ritz acceleration, where we exploit the proximity of the eigenspace across levels to accelerate the conjugate gradient (CG) method within the refinement step. Using this refined initial spectrum, our second stage is to extend the spectrum up to some prescribed control of the well-posedness using the implicit restarted Lanczos architecture. Recall that a shifting approach is introduced to reduce the iteration number for the extension, which again requires solving $A^{(K-1)}x = w$ with the CG method in each iteration. However, the preconditioner for CG when we are solving for $A^{(K-1)}w$ must be chosen carefully. Otherwise the orthogonal property brought about by the Krylov subspace methods may not be utilized and a large CG iteration number will be observed (see section 8). In view of this, we propose a spectrum-preserving hierarchical preconditioner $M^{(K-1)} := (\Psi^{(K-1)})^T \Psi^{(K-1)}$ for accelerating the CG iteration during the Lanczos iteration. In particular, we can show that using the preconditioner $M^{(K-1)}$, the number of preconditioned CG (PCG) iterations to achieve a relative ε

in the $A^{(K-1)}$-norm can be controlled in terms of the condition factor $\delta(\mathcal{P})$ (from the energy decomposition of the matrix) and an extension threshold $\mu_{ex}^{(K-1)}$.

This process then repeats hierarchically until we reach the finest level. Under this framework, the condition number of every engaged operator is controlled. The overall accuracy of our proposed algorithm is also determined by the prescribed compression error at the highest level.

1.2. Previous works. Several important iterative methods have been proposed to tackle the eigenproblems of SPD matrices. One of the well established algorithms is the IRLM (or the implicitly restarted Arnoldi method for unsymmetric sparse matrices), which has been implemented in various popular scientific computing packages like MATLAB, R, and ARPACK. The IRLM combines both the techniques of the implicitly shifted QR method and the shifting of the operators to avoid the difficulties for obtaining the leftmost eigenpairs. Another popular algorithm for finding leftmost eigenpairs is the Jacobi–Davidson method. The main idea is to minimize the Rayleigh quotient $q(x) = \frac{x^TAx}{x^Tx}$ using a Newton-type methodology. Efficacy and stability of the algorithm are then achieved by using a projected simplification of the Hessian of the Rayleigh quotient, namely, $\tilde{J}(x_k) := (I - x_kx_k^T)(A - q(x_k)I)(I - x_kx_k^T)$ with the update of x_k to be

$$x_{k+1} = x_k - \tilde{J}(x_k)^{-1}(Ax_k - q(x_k)x_k). \tag{1}$$

Notice that the advantage of such an approach is the low accuracy requirement for solving (1). A parallelization was also proposed in [23]. In [2], the authors proposed the DACG method designed for solving the eigenproblem of SPD matrices. The main idea is to replace Newton's minimization procedure of the Rayleigh quotient $r(x)$ by the nonlinear CG method which avoids solving linear systems within the algorithm. A comprehensive numerical comparison between the three algorithms was reported in [1]. Recently, Martínez [16] studied a class of tuned preconditioners for accelerating both the DACG and the IRLM for the computation of the smallest set of eigenpairs of large and sparse SPD matrices. However, as reported in [16], the IRLM still outperforms the others when a relatively large number of leftmost eigenpairs is desired. By virtue of this, we are motivated to develop a more efficient algorithm particularly designed for computing a considerable amount of leftmost eigenpairs.

Another class of methods related to the localized spectrum is the compression of the eigenmodes. One of the representative pioneer works is proposed by Ozoliņš et al. in [22]. The goal of this work is to obtain a spatially localized solution of a class of problems in mathematical physics by constructing the compressed modes. In particular, finding these localized modes can be formulated as an optimization problem

$$\Psi_N = \arg\min_{\hat{\Psi}^N} \frac{1}{\mu}\|\Psi_N\|_1 + \mathrm{Tr}(\hat{\Psi}_N^T H \hat{\Psi}_N) \quad \text{such that} \quad \hat{\Psi}_N^T\hat{\Psi}_N = I.$$

The authors in [22] proposed an algorithm based on the split Bregman iteration to solve the L_1 minimization problem. By replacing the discrete operator H by the graph Laplacian matrix A, one obtains the L_1 regularized principal component analysis (PCA). In particular, if there is no L_1 regularization term in the optimization problem, the optimal Ψ_N will be the first m_{tar} eigenvectors of A. In other words, this procedure provides an effective way to obtain N (where $N \geq m_{tar}$) localized basis functions that can approximately span the m_{tar} leftmost eigenspace (i.e., eigenspace spanned by the m_{tar} eigenvectors corresponding to the leftmost eigenvalues). Similarly, the MMD framework provides us with the hierarchical and sparse/localized basis Ψ. These localized basis functions capture the compressed modes and eventually provide a convenient way to control the complexity of the eigensolver.

Stiffness matrices discretizing heterogeneous and rough elliptic operators, or graph Laplacians representing general sparse networks, are commonly found in practice. Recently, the problem of compressing these SPD matrices has been tackled in different perspectives. Målqvist and Peterseim [15] proposed the use of a modified coarse space in order to handle roughness of the coefficients when solving elliptic equations with finite element methods. They construct localized multiscale basis functions from the modified coarse space $V_H^{ms} = V_H - \mathfrak{F}V_H$, where V_H is the original coarse space spanned by the nodal basis, and $\mathfrak{F}$ is the energy projection onto the space $(V_H)^{\perp}$. The exponential decaying property of these modified basis functions has been shown both theoretically and numerically. In [20], Owhadi reformulated the problem from the decision theory perspective using the idea of *gamblets* as the modified basis. In particular, a coarse space Φ of measurement functions is constructed from the Bayesian perspective, and the gamblet space is explicitly given as $\Psi = A^{-1}(\Phi)$, which turns out to be a counterpart of the modified coarse space in [15]. The exponential decaying property of these localized basis functions is also proved independently using the idea of gamblets. Hou and Zhang in [11] further extended these works and constructed localized basis functions for higher order strongly elliptic operators. To further promote the operator compression for situations where the physical domain is unknown or is embedded in some nontrivial high-dimensional manifolds, Hou et al. [10] propose to exploit the local spectrum information of a general class of SPD matrices to bypass the needs of adopting knowledge of the computational domain during the construction of a local basis. Recently, Schäfer, Sullivan, and Owhadi [24] proposed a near-linear running time algorithm to compress a large class of dense kernel matrices $\Theta \in \mathbb{R}^{n\times n}$. The authors also provided rigorous complexity analyses and showed that the complexity of the proposed algorithm is $O(n \log(n) \log^d(n/\epsilon))$ in space and $O(n \log^2(n) \log^{2d}(n/\epsilon))$ in time, where d is the intrinsic dimension of the problem.

1.3. Outline. The layout of the rest of this paper is as follows: In section 2 we review the energy decomposition framework for SPD matrices proposed in [10] and, in particular, a brief review of the operator compression and MMD is summarized. This is then followed by the review of the implicitly restarted Arnoldi iteration procedure. Some error analysis and perturbation theories subject to our operator compression framework are discussed. Theoretical developments and algorithms of the hierarchical spectrum extension/compression and the eigenpair refinement are then proposed in sections 4 and 5, respectively. Combining these two methods, we propose our hierarchical eigensolver in section 6, where details of the choice of parameters are discussed. Section 7 is devoted to experimental results to justify the effectiveness of our proposed algorithm. In section 8, we provide a quantitative numerical comparison with the IRLM. The numerical results show that our proposed algorithm gives promising results in terms of runtime complexity. Discussion of future works and the conclusion are given in section 9.

2. Preliminaries.

The purpose of this section is to provide a general summary of the *energy decomposition* framework for operator compression and MMD. One may refer to [10] for a detailed numerical analysis and experimental results.

2.1. Energy decomposition. Let A be an $n \times n$ SPD matrix. We call $\mathcal{E} = \{E_k\}_{k=1}^m$ an energy decomposition of A and E_k to be an *energy element* of A if we can express $A = \sum_{k=1}^m E_k$, where $E_k \succeq 0\ \forall k = 1, \dots, m$. For the ease of discussion, we always assume that the given $\mathcal{E} = \{E_k\}_{k=1}^m$ is the finest underlying energy decomposition of A, meaning that no $E_k \in \mathcal{E}$ can be further decomposed as $E_k = E_{k,1} + E_{k,2}$.

Let $\mathcal{V}$ be a basis of $\mathbb{R}^n$. For any subset $\mathcal{S} \subset \mathcal{V}$, we denote $P_\mathcal{S}$ as the orthogonal projection onto $\mathcal{S}$. Following the notations in [10], we also denote $A_\mathcal{S}$, $\underline{A}_\mathcal{S}$, and $\overline{A}_\mathcal{S}$ as the *restricted*, *interior* and *closed energy* of $\mathcal{S}$ with respect to A and $\mathcal{E}$.

2.2. Operator compression. The procedures for compressing the solver A^{-1} with a broad-banded spectrum are (i) constructing a partition of the computational basis using local information of A; (ii) constructing the coarse space Φ that is locally computable and has a good interpolation property; (iii) constructing the modified coarse space $\Psi = A^{-1}(\Phi)$ of $\mathbb{R}^n$ as proposed in [11, 15, 20]. If an appropriate partitioning is given, we have the following error estimate for operator compression.

THEOREM 2.1. *Let Φ be an N-dimensional subspace of $\mathbb{R}^n$ such that for some $\epsilon > 0$,*

$$\|x - P_\Phi x\|_2 \le \sqrt{\epsilon}\|x\|_A \quad \forall x \in \mathbb{R}^n, \tag{2}$$

where P_Φ is the orthogonal projection onto Φ. Let Ψ be a subspace of $\mathbb{R}^n$ given by $\Psi = A^{-1}(\Phi)$. Denote P_Ψ^A as the orthogonal projection onto Ψ with respect to $\langle \cdot, \cdot \rangle_A$, and $\Theta = P_\Psi^A A^{-1}$ as the rank-N compressed approximation of A^{-1}. Then for any $x \in \mathbb{R}^n$ and $b = Ax$, we have

$$\|x - P_\Psi^A x\|_A \le \sqrt{\epsilon}\|b\|_2 \quad \text{and} \quad \|x - P_\Psi^A x\|_2 \le \epsilon\|b\|_2, \tag{3}$$

and thus

$$\|A^{-1} - \Theta\|_2 \le \epsilon. \tag{4}$$

As discussed in [10], to satisfy (2), Φ can be constructed by choosing some optimal local basis Φ_j on each patch P_j, where $\mathcal{P} = \{P_j\}_{j=1}^M$ is a partition of $\mathcal{V}$. To minimize $\dim \Phi$, the local basis Φ_j is chosen to be the eigenvectors corresponding to the smallest interior eigenvalues (i.e., eigenvalues of $\underline{A}_{P_j}$) $\lambda_1(P_j) \le \lambda_2(P_j) \le \cdots \le \lambda_{q_j(\epsilon)}(P_j)$, where $q_j(\epsilon)$ is the smallest integer such that $\frac{1}{\epsilon} \le \lambda_{q_j(\epsilon)}(P_j)$. By reversing the statement, we introduce the *error factor* $\varepsilon(\mathcal{P}) = \max_j(\lambda_{q+1}(P_j))^{-1}$ of partition $\mathcal{P}$, where q is some prescribed uniform integer for all patches. Then locally on each patch we have $\|x - P_{\Phi_j}x\|_2 \le \sqrt{\varepsilon(\mathcal{P})}\|x\|_{\underline{A}_{P_j}}\ \forall x \in \operatorname{span}\{P_j\}$, and by collecting $\Phi = \bigoplus_j \Phi_j$ we have globally $\|x - P_\Phi x\|_2 \le \sqrt{\varepsilon(\mathcal{P})}\|x\|_A\ \forall x \in \mathbb{R}^n$. In the following, we assume that $q = 1$ in all cases. Under this setting, the problem of minimizing $\dim \Phi$ subject to (2) is transformed into finding a partition $\mathcal{P} = \{P_j\}_{j=1}^N$ with minimal patch number and which satisfies $\varepsilon(\mathcal{P}) \le \epsilon$.

Following the notations in [10], we also use Φ, Ψ to denote the matrices whose columns are the basis vectors of the subspaces Φ, Ψ, respectively. We remark that using the matrix form, the A-orthogonal projection P_Ψ^A can be written as

$$P_\Psi^A = \Psi(\Psi^T A \Psi)^{-1}\Psi^T A = A^{-1}\Phi(\Phi^T A^{-1}\Phi)^{-1}\Phi^T, \tag{5}$$

and the rank-N compressed approximation is explicitly $\Theta = P_\Psi^A A^{-1} = \Psi A_{st}^{-1}\Psi^T$, where

$$A_{st} = \Psi^T A \Psi \tag{6}$$

is the stiffness matrix in the basis Ψ. Once the coarse space/basis Φ is constructed, the next step is to find $\Psi = [\psi_1, \psi_2, \ldots, \psi_N] = A^{-1}(\Phi)$ such that (i) the stiffness matrix A_{st} has a relatively small condition number, or the condition number can be bounded by some local information; (ii) each ψ_i is locally computable, or can be approximated by some $\widetilde{\psi}_i$ that is locally computable. To achieve these two requirements, we impose the correlation condition $\Phi^T\Psi = I_N$, which is equivalent to choosing $\Psi = [\psi_1, \psi_2, \ldots, \psi_N]$ to be

$$\Psi = A^{-1}\Phi(\Phi A^{-1}\Phi)^{-1} \tag{7}$$

and we have the following theorem for the well-posedness of A_{st}.

Theorem 2.2. *Let A_{st} be the stiffness matrix given by* (6). *Let $\lambda_{\min}(A_{st})$ and $\lambda_{\max}(A_{st})$ denote the smallest and largest eigenvalues of A_{st}, respectively, then we have*

$$\lambda_{\min}(A_{st}) \geq \lambda_{\min}(A), \qquad \lambda_{\max}(A_{st}) \leq \delta(\mathcal{P}) \tag{8}$$

with

$$\delta(\mathcal{P}) = \delta(\mathcal{P}, \Phi) = \max_{P_j \in \mathcal{P}} \delta(P_j, \Phi_j) \quad \textit{and} \quad \delta(P_j, \Phi_j) = \max_{x \in \Phi_j} \frac{x^T x}{x^T \overline{A}_{P_j}^{-1} x},$$

where $\delta(\mathcal{P})$ is called the condition factor *of the partition $\mathcal{P}$.*

In other words, by defining Ψ as in (7), the first requirement can be satisfied. Moreover, such a choice of Ψ also satisfies the second requirement. In fact, we can prove the spatial exponential decaying property of every basis function ψ_i (see [10, 20] for details). This fast decay feature makes it possible to approximate Ψ by some localized basis $\widetilde{\Psi}$ that preserves the good properties of Ψ. In particular, we can construct a basis $\widetilde{\Psi} = [\widetilde{\psi}_1, \widetilde{\psi}_2, \ldots, \widetilde{\psi}_N]$ such that each $\widetilde{\psi}_i$ satisfies $\|\psi_i - \widetilde{\psi}_i\|_A \leq C\sqrt{\frac{\epsilon}{N}}$ for some constant C, and has support size $O((\log\frac{1}{\epsilon} + \log N)^d)$, where d is the intrinsic dimension of the problem that characterizes its connectivity. For this localized $\widetilde{\Psi}$, we have an analogy of Theorem 2.1 stating that the operator compression error can be bounded by $\|A^{-1} - \widetilde{\Theta}\|_2 \leq (1 + C\|A^{-1}\|_2)^2\varepsilon(\mathcal{P})$ (where $\widetilde{\Theta} := P_{\widetilde{\Psi}}^A A^{-1} = \widetilde{\Psi}(\widetilde{\Psi}^T A\widetilde{\Psi})^{-1}\widetilde{\Psi}^T$), and the condition bound of the localized stiffness matrix can be estimated by

$$\kappa(\widetilde{A}_{\mathrm{st}}) = \frac{\lambda_{\max}(\widetilde{A}_{\mathrm{st}})}{\lambda_{\min}(\widetilde{A}_{\mathrm{st}})} \leq \left(1 + C\sqrt{\frac{\epsilon}{\delta(\mathcal{P})}}\right)^2 \delta(\mathcal{P})\|A^{-1}\|_2, \tag{9}$$

where $\kappa(\widetilde{A}_{\mathrm{st}})$ is the condition number of $\widetilde{A}_{\mathrm{st}} := \widetilde{\Psi}^T A\widetilde{\Psi}$. Therefore the burden of controlling the accuracy, sparsity, and well-posedness of the compressed operator A_{st} falls into the procedure of partitioning. We then propose a nearly linear time algorithm using the indicators error factor and condition factor to obtain an appropriate partition $\mathcal{P}$ subject to $\varepsilon(\mathcal{P})\delta(\mathcal{P}) \leq c$ for some prescribed upper bound c. For details of the notations and the algorithm, please refer to [10].

2.3. Multiresolution matrix decomposition. Recall that the main purpose of decomposing A^{-1} into hierarchical resolutions is to resolve the difficulty of a large condition number $\kappa(A)$ when solving the linear system $Ax = b$. Through decomposition, the relative condition number in each scale/level can be bounded by some prescribed value. Using the notation as in the previous subsections, we denote $U = [U_1, U_2, \dots, U_M]$ and therefore $[U, \Psi]$ forms a basis of $\mathbb{R}^n$. We also have $U^T A\Psi = U^T\Phi(\Phi^T A^{-1}\Phi)^{-1} = 0$. Thus the inverse of A can be written as

$$\begin{aligned} A^{-1} &= \left(\begin{bmatrix} U^T \\ \Psi^T \end{bmatrix}^{-1} \begin{bmatrix} U^T \\ \Psi^T \end{bmatrix} A \begin{bmatrix} U & \Psi \end{bmatrix} \begin{bmatrix} U & \Psi \end{bmatrix}^{-1} \right)^{-1} \\ &= U(\underbrace{U^T AU}_{B_{\mathrm{st}}})^{-1}U^T + \Psi(\underbrace{\Psi^T A\Psi}_{A_{\mathrm{st}}})^{-1}\Psi^T. \end{aligned} \tag{10}$$

Therefore, solving $A^{-1}b$ is equivalent to solving $A_{\mathrm{st}}^{-1}(\Psi^T b)$ and $B_{\mathrm{st}}^{-1}(U^T b)$ separately. For B_{st}, since the sparsity of U will be inherited by B_{st}, it will be efficient to solve $B_{\mathrm{st}}^{-1}b$ if $\kappa(B_{\mathrm{st}})$ is bounded. The following lemma estimates such an upper bound.

LEMMA 2.3. *If Φ satisfies the condition as in Theorem* 2.1 *with $\epsilon(\mathcal{P})$ and $B_{st} = U^T AU$, then*

$$\lambda_{\max}(B_{st}) \le \lambda_{\max}(A)\cdot\lambda_{\max}(U^T U), \qquad \lambda_{\min}(B_{st}) \ge \frac{1}{\varepsilon(\mathcal{P})}\cdot\lambda_{\min}(U^T U), \tag{11}$$

and thus

$$\kappa(B_{st}) \le \varepsilon(\mathcal{P})\cdot\lambda_{\max}(A)\cdot\kappa(U^T U). \tag{12}$$

Notice that $U^T U$ is block diagonal with blocks $U_j^T U_j$, therefore,

$$\kappa(U^T U) = \frac{\lambda_{\max}(U^T U)}{\lambda_{\min}(U^T U)} = \frac{\max_{1\le j\le M}\lambda_{\max}(U_j^T U_j)}{\min_{1\le j\le M}\lambda_{\min}(U_j^T U_j)}. \tag{13}$$

In particular, if we extend Φ_j to an orthonormal basis of span$\{P_j\}$ to get U_j using the QR factorization, we have $\kappa(U^T U) = 1$. So if the condition number of A is huge, we can first set a small enough ε to sufficiently bound $\kappa(B_{\mathrm{st}})$; if $\kappa(A_{\mathrm{st}})$ is still large, we apply the decomposition to A_{st}^{-1} again to further decompose $\kappa(A_{\mathrm{st}})$. In order to further decompose the stiffness matrix A_{st}, we need to construct the corresponding energy decomposition of A_{st}.

DEFINITION 2.4 (inherited energy decomposition). *Let $\mathcal{E} = \{E_k\}_{k=1}^m$ be the energy decomposition of A, then the inherited energy decomposition of $A_{st} = \Psi^T A\Psi$ with respect to $\mathcal{E}$ is simply given by $\mathcal{E}^\Psi = \{E_k^\Psi\}_{k=1}^m$, where $E_k^\Psi = \Psi^T E_k\Psi, \quad k = 1, 2, \dots, m$.*

Once we have the underlying energy decomposition of A_{st}, we can repeat the procedure to decompose A_{st}^{-1} in $\mathbb{R}^N$ as what we have done to A^{-1} in $\mathbb{R}^n$ and, furthermore, to obtain a multilevel decomposition of A^{-1}. In particular, at level k, we construct the partition $\mathcal{P}^{(k)}$ and the basis $\Phi^{(k)}, U^{(k)}, \Psi^{(k)}$ accordingly, and decompose $(A^{(k)})^{-1}$ as

$$\begin{aligned} (A^{(k)})^{-1} = {}& U^{(k+1)}\big((U^{(k+1)})^T A^{(k)} U^{(k+1)}\big)^{-1}(U^{(k+1)})^T \\ &+ \Psi^{(k+1)}\big((\Psi^{(k+1)})^T A^{(k)} \Psi^{(k+1)}\big)^{-1}(\Psi^{(k+1)})^T, \end{aligned}$$

and then define $A^{(k+1)} = (\Psi^{(k+1)})^T A^{(k)} \Psi^{(k+1)}$ and $B^{(k+1)} = (U^{(k+1)})^T A^{(k)} U^{(k+1)}$. We also recall the following notations:

$$\boldsymbol{\Phi}^{(k)} = \Phi^{(1)} \cdots \Phi^{(k-1)} \Phi^{(k)}, \quad k \geq 1, \tag{14a}$$

$$\mathcal{U}^{(k)} = \Psi^{(1)} \cdots \Psi^{(k-1)} U^{(k)}, \quad k \geq 1, \tag{14b}$$

$$\boldsymbol{\Psi}^{(k)} = \Psi^{(1)} \cdots \Psi^{(k-1)} \Psi^{(k)}, \quad k \geq 1. \tag{14c}$$

Using these notations and noticing that $(\Phi^{(k)})^T \Phi^{(k)} = (\Phi^{(k)})^T \Psi^{(k)} = I_{N^{(k)}}$, we have

$$A^{(k)} = (\boldsymbol{\Psi}^{(k)})^T A \boldsymbol{\Psi}^{(k)} = \big((\boldsymbol{\Phi}^{(k)})^T A^{-1} \boldsymbol{\Phi}^{(k)}\big)^{-1}, \quad B^{(k)} = (\mathcal{U}^{(k)})^T A \mathcal{U}^{(k)},$$
$$(\boldsymbol{\Phi}^{(k)})^T \boldsymbol{\Phi}^{(k)} = (\boldsymbol{\Phi}^{(k)})^T \boldsymbol{\Psi}^{(k)} = I_{N^{(k)}}, \quad \boldsymbol{\Psi}^{(k)} = A^{-1} \boldsymbol{\Phi}^{(k)} \big((\boldsymbol{\Phi}^{(k)})^T A^{-1} \boldsymbol{\Phi}^{(k)}\big)^{-1},$$

and for any integer K,

$$\begin{aligned} A^{-1} = (A^{(0)})^{-1} = & \sum_{k=1}^{K} \mathcal{U}^{(k)} \big((\mathcal{U}^{(k)})^T A \mathcal{U}^{(k)}\big)^{-1} (\mathcal{U}^{(k)})^T \\ & + \boldsymbol{\Psi}^{(K)} \big((\boldsymbol{\Psi}^{(K)})^T A \boldsymbol{\Psi}^{(K)}\big)^{-1} (\boldsymbol{\Psi}^{(K)})^T. \end{aligned} \tag{15}$$

We call (15) the MMD of A^{-1}. We remark that as k increases, the compressed dimension $N^{(k)}$ decreases, and the scale of the subspace spanned by $\boldsymbol{\Psi}^{(k)}$ becomes coarser. In the subspace spanned by $\boldsymbol{\Psi}^{(k-1)}$, the basis $\mathcal{U}^{(k)}$ represents the features that are finer than $\boldsymbol{\Psi}^{(k)}$. This decomposition helps separate A that has a large condition number into a sequence of matrices with more controllable conditioned numbers. This is stated in the following corollary.

Corollary 2.5. *We have*

$$\kappa(A^{(k)}) \leq \delta(\mathcal{P}^{(k)}) \|A^{-1}\|_2,$$
$$\kappa(B^{(k)}) \leq \varepsilon(\mathcal{P}^{(k)}) \delta(\mathcal{P}^{(k-1)}) \kappa\big((U^{(k)})^T U^{(k)}\big).$$

For consistency, we write $\delta(\mathcal{P}^{(0)}) = \lambda_{\max}(A^{(0)}) = \lambda_{\max}(A)$.

The following theorem provides an estimation of the total compression error under K levels of matrix decomposition.

Theorem 2.6. *Assume we have constructed* $\Phi^{(k)}, k = 1, 2, \ldots, K$ *on each level accordingly, then we have*

$$\|x - P_{\boldsymbol{\Phi}^{(k)}} x\|_2^2 \leq \varepsilon_k \|x\|_A^2 \quad \forall x \in \mathbb{R}^n, \quad \textit{where } \varepsilon_k = \sum_{k'=1}^{k} \varepsilon(\mathcal{P}^{(k')}), \tag{16}$$

and thus for any $x \in \mathbb{R}^n$ *and* $b = Ax$*, we have*

$$\|x - P^A_{\boldsymbol{\Psi}^{(k)}} x\|_A^2 \leq \varepsilon_k \|b\|_2^2, \quad \|x - P^A_{\boldsymbol{\Psi}^{(k)}} x\|_2 \leq \varepsilon_k \|b\|_2, \quad \textit{and} \quad \|A^{-1} - P^A_{\boldsymbol{\Psi}^{(k)}} A^{-1}\|_2 \leq \varepsilon_k.$$

Notice that the compression error ε_k is in a cumulative form. However, we can restrict $\varepsilon(\mathcal{P}^{(k)})$ to increase with k at a certain rate, i.e., $\frac{\varepsilon(\mathcal{P}^{(k+1)})}{\varepsilon(\mathcal{P}^{(k)})} = \frac{1}{\eta}$ for some $\eta \in (0,1)$, which gives

$$\varepsilon_k \leq \frac{1}{1-\eta} \varepsilon(\mathcal{P}^{(k)}). \tag{17}$$

With the above framework for the MMD, the original matrix A can be decomposed into bounded pieces, such that the condition number $\kappa(B^{(k)})$ is controlled by choosing an appropriating partition $\mathcal{P}$ with $\varepsilon(\mathcal{P}^{(k)})\delta(\mathcal{P}^{(k)}) \leq c$ for some constant c. Therefore, we can apply the MMD to solve a linear system. Notice that the difference between ε_k and $\varepsilon(\mathcal{P}^{(k)})$ is very small and can be neglected; in this manuscript, we will treat $\varepsilon(\mathcal{P}^{(k)})$ as ε_k and denote them simply by ε_k. To be coherent, we also replace the notation of $\delta(\mathcal{P}^{(k)})$ by δ_k to avoid confusion that may arise due to various notations.

In practice, we also introduce a local approximator $\widetilde{\Psi}^{(k)}$, with which the sparsity of $\widetilde{A}^{(k)}$ and $\widetilde{B}^{(k)}$ can be preserved. In particular, we require $nnz(\tilde{A}^{(k)}) = O(nnz(A))$, where nnz denotes the number of nonzero entries. We remark that, since $\widetilde{B}^{(k)} = (U^{(k)})^T\widetilde{A}^{(k-1)}U^{(k)}$, any multiplication operation concerning $\widetilde{B}^{(k)}$ only requires the applying of $(U^{(k)})^T, U^{(k)}$, and $\widetilde{A}^{(k-1)}$ separately. The applying of $(U^{(k)})^T, U^{(k)}$ can be done implicitly by performing a local Householder transform with cost linear in n. So only the sparsity of $\widetilde{A}^{(k)}$ matters. From the estimates for the MMD in [10], we can preserve the sparsity of $\widetilde{A}^{(k)}$ by choosing the scale ratio η^{-1} to be

$$\eta^{-1} = \left(\log\frac{1}{\varepsilon} + \log n\right)^p, \tag{18}$$

where we remark that $p = 1$ for graph Laplacian cases. Such a choice of η also gives us the estimate of the total level number as

$$K = O\left(\frac{\log n}{\log(\log\frac{1}{\varepsilon} + \log n)}\right). \tag{19}$$

Moreover, the uniform condition bound $\kappa(\mathcal{P}^{(k)}, q^{(k)}) \leq c$ can be imposed directly through the MMD algorithm. For more details, please refer to [10, section 6]. For the ease of discussion in this paper, we presume using the localized decomposition to control the sparsity throughout levels and simply write $\widetilde{\psi}^{(k)}$, $\widetilde{A}^{(k)}$, and $\widetilde{B}^{(k)}$ as $\Psi^{(k)}$, $A^{(k)}$, and $B^{(k)}$.

2.4. Implicitly restarted Lanczos method. The Arnoldi iteration is a widely used method to find eigenvalues of unsymmetric sparse matrices. It belongs to the family of Krylov subspace methods. For the symmetric case, we can further simplify it as the Lanczos iteration. A direct application of the Lanczos iteration gives the largest eigenvalues of an operator by calculating the eigenvalues of its projection on a Krylov subspace. In each step the algorithm expands the Krylov subspace and finds an orthogonal basis of the space. Namely, after k steps, the factorization is

$$AV_k = V_kT_k + f_ke_k^T, \tag{20}$$

where we recall that T_k is a tridiagonal matrix when A is symmetric. Denote (θ, y) as an eigenpair of T_k. Let $x = V_ky$. Then we have

$$\|Ax - x\theta\|_2 = \|AV_ky - V_ky\theta\|_2 = \|f_k\|_2|e_k^Ty|. \tag{21}$$

Therefore θ is a good approximation of the eigenvalue of A if and only if $\|f_k\|_2|e_k^Ty|$ is small. The latter is called the Ritz residual. An analogy to the power method shows that, to compute the largest m eigenvalues, the convergence rate of the largest m eigenvalues of A is $(\lambda_{m+1}/\lambda_m)^k$, where λ_i is the ith largest eigenvalue of A.

The direct Lanczos method is not practical due to the fact that $\|f_k\|_2$ rarely becomes small enough until the size of T_k approaches that of A. An improvement is the IRLM [27, 12]. The IRLM employs the idea analogously to the implicitly shifted QR-iteration [8]. With this approach, the "unwanted" eigenvalues (in this case the leftmost ones) are shifted away implicitly in each round of implicit restart, and T_k is kept with a small size equal to the number of desired eigenvalues. This is one of the state-of-the-art algorithms for large-scale partial eigenproblems.

Yet, it is still complicated if we want to find the leftmost eigenvalues. One possible approach is to use a shifted IRLM. Namely, to find eigenvalues nearest to σ, we can replace A with $(A-\sigma I)^{-1}$ as the target operator. By taking $\sigma = 0$ we get the eigenvalues with the smallest magnitude. Such an approach usually converges within a few iterations, but it requires solving A^{-1} in every iteration. For large sparse problems, A^{-1} is usually solved by the CG method. The complexity of CG is the complexity of the matrix-vector product times the number of CG iterations. The former is equal to the number of nonzero entries of A (denoted as $nnz(A)$), while the latter is controlled by the condition number $\kappa(A)$. Therefore, the total complexity of the shifted IRLM for solving m_{tar} smallest eigenvalues is

$$O(R_{\mathrm{IRLM}} \cdot m_{tar} \cdot nnz(A) \cdot \kappa(A)), \tag{22}$$

where R_{IRLM} is the number of IRLM rounds. In the following, we will develop the extension-refinement algorithm to integrate the MMD framework with the shifted IRLM which gives considerable improvement in terms of the iteration numbers of CG and PCG throughout the algorithm.

Algorithm 1 Lanczos iteration (p-step extension).

```
Input: $V$, $T$, $f$, target operator $op(\cdot)$, $p$.
Output: $V$, $T$, $f$.
 1: $k$ = column number of $V$;
 2: for $i = 1 : p$ do
 3:     $\beta = \|f\|_2$;
 4:     if $\beta < \epsilon$ then
 5:         generate a new random $f$, $\beta = \|f\|_2$;
 6:     end if
 7:     $T \leftarrow \begin{pmatrix} T \\ \beta e_{k+i-1}^T \end{pmatrix}$,   $v = f/\beta$,   $V \leftarrow [V, v]$;
 8:     $w = op(v)$;
 9:     $h = V^T w$,   $T \leftarrow [T, h]$;
10:     $f = w - Vh$;
11:     Reorthogonalize to adjust $f$;
12: end for
```

3. The compressed eigenproblem.

In the previous section, we introduced an effective compression technique for an SPD matrix A subject to a prescribed compression error ϵ. The compressed operator is also being SPD. Therefore, by the well-known eigenvalue perturbation theory, we know that the eigenpairs of the compressed operator can be used as good approximations for the eigenpairs of the original matrix. In particular, we have the following estimate.

LEMMA 3.1. *Let $\Theta = \Psi(\Psi^T A \Psi)^{-1}\Psi^T$ be the rank-N compressed approximation of A^{-1} introduced in Theorem* 2.1 *such that $\|A^{-1} - \Theta\|_2 \le \varepsilon$. Let $\mu_1 \ge \mu_2 \ge \cdots \ge \mu_n > 0$*

Algorithm 2 Inner iteration of the IRLM.

```
Input: $V$, $T$, $f$, $p$.
Output: $V$, $T$, $f$.
1: $k$ = column number of $V$;
2: Perform Algorithm 1 on $V$, $T$, and $f$ for $p$ steps;
3: Set $Q = I_{k+p}$ and $\{\sigma_j\}$ to be the $p$ smallest eigenvalues of $T$;
4: for $j = 1 : p$ do
5:     $T - \sigma_j I = Q_j R_j$;
6:     $T = Q_j^T T Q_j, Q \leftarrow Q Q_j$;
7: end for
8: $f \leftarrow V \cdot Q(:, k+1) \cdot T(k+1, k) + f \cdot Q(k+p, k)$;
9: $V \leftarrow V \cdot Q(:, 1:k)$, $T \leftarrow T(1:k, 1:k)$;
```

be the eigenvalues of A^{-1} in descending order, and $\tilde{\mu}_1 \geq \tilde{\mu}_2 \geq \cdots \geq \tilde{\mu}_N > 0$ be the nonzero eigenvalues of Θ in descending order. Then we have

$$|\mu_i - \tilde{\mu}_i| \leq \varepsilon, \quad 1 \leq i \leq N, \qquad \mu_i \leq \varepsilon, \quad N < i \leq n.$$

Moreover, let $\tilde{v}_i$, $i = 1, \ldots, N$, be the corresponding normalized eigenvectors of Θ such that $\Theta\tilde{v}_i = \tilde{\mu}_i\tilde{v}_i$, then we have

$$\|A^{-1}\tilde{v}_i - \mu_i\tilde{v}_i\|_2 \leq 2\varepsilon, \quad 1 \leq i \leq N.$$

Since the nonzero eigenvalues of Θ and the corresponding eigenvectors actually result from the nonsingular stiffness matrix $A_{st} = \Psi^T A \Psi$, we will call these eigenpairs the *essential eigenpairs* of Θ in what follows. We will also need the following lemma for developing our algorithms.

LEMMA 3.2. *Let $(\tilde{\mu}_i, \tilde{v}_i)$, $i = 1, \ldots, N$, be the N essential eigenpairs of Θ given in Lemma* 3.1.

(i) *Let $w_i = \Psi^T\tilde{v}_i$, then*

$$\Psi^T\Psi A_{st}^{-1} w_i = \tilde{\mu}_i w_i, \quad 1 \leq i \leq N.$$

(ii) *Let $z_i = \Psi^\dagger\tilde{v}_i = (\Psi^T\Psi)^{-1}\Psi^T\tilde{v}_i$, then*

$$A_{st}^{-1}\Psi^T\Psi z_i = \tilde{\mu}_i z_i, \quad 1 \leq i \leq N,$$

where $A_{st} = \Psi^T A \Psi$ is the stiffness matrix. Conversely, if either (i) *or* (ii) *is true, then $(\tilde{\mu}_i, \tilde{v}_i)$, $i = 1, \ldots, N$, are eigenpairs of Θ.*

Similarly to Lemma 3.1, we have the following estimates for multiresolution decomposition.

LEMMA 3.3. *Given an integer K, let $\Theta^{(k)} = \mathbf{\Psi}^{(k)}((\mathbf{\Psi}^{(k)})^T A \mathbf{\Psi}^{(k)})^{-1}(\mathbf{\Psi}^{(k)})^T$, $k = 1, 2, \ldots, K$, with $\mathbf{\Psi}^{(k)}$ given in* (14). *Write $A^{-1} = \Theta^{(0)}$. Let $(\mu_i^{(k)}, v_i^{(k)})$, $i = 1, 2, \ldots, N^{(k)}$, be the essential eigenpairs of $\Theta^{(k)}$, where $\mu_1^{(k)} \geq \mu_2^{(k)} \geq \cdots \geq \mu_{N^{(k)}}^{(k)} > 0$. Then for any $0 \leq k' < k \leq K$, we have*

$$|\mu_i^{(k')} - \mu_i^{(k)}| \leq \varepsilon_k, \quad 1 \leq i \leq N^{(k)}, \qquad |\mu_i^{(k')}| \leq \varepsilon_k, \quad N^{(k)} < i \leq N^{(k')},$$

and

$$\|\Theta^{(k')} v_i^{(k)} - \mu_i^{(k')} v_i^{(k)}\|_2 \leq 2\varepsilon_k, \quad 1 \leq i \leq N^{(k)}.$$

Proof. By Theorem 2.6 we have that $\|\Theta^{(0)} - \Theta^{(k)}\|_2 = \|A^{-1} - \Theta^{(k)}\|_2 \leq \varepsilon_k$, $k = 1, 2, \ldots, K$. From the definition of $\Theta^{(k)}$ and the decomposition (15), one can easily check that

$$A^{-1} = \Theta^{(0)} \succeq \Theta^{(1)} \succeq \cdots \succeq \Theta^{(K-1)} \succeq \Theta^{(K)}.$$

Then the results follow immediately. ∎

On compressed eigenproblems. We should remark that the efficiency of constructing the compressed operator we propose relies on the exponential decay property of the basis Ψ. This spatial exponential decay feature allows us to localize Ψ and to construct sparse stiffness matrix $A_{st} = \Psi^T A \Psi$ without compromising compression accuracy ε in $O(nnz(A) \cdot (\log(\frac{1}{\varepsilon}) + \log n)^c)$ time. In fact, the problem of using a spatially localized/compact basis to compress a high-dimensional operator and to approximate the eigenspace of the smallest eigenvalues has long been studied in different ways. A representative pioneer work is the method of compressed modes proposed by Ozoliņš et al. [22], intended originally for Schrödinger's equation in quantum physics. By adding an L_1 regularization to the variational form of an eigenproblem, they obtained spatially compressed basis modes that span the desired eigenspace well. Though the way they obtain sparsity is quite different from what we do, both methods obtain interestingly similar results for some model problems. It can be inspiring to make a comparison between their method and ours, so that readers can have a better understanding of our approach. We leave the detailed comparison to the appendix.

4. Hierarchical spectrum completion.

Now that we have a sequence of compressed approximations, we next seek to use this decomposition to compute the dominant spectrum of A^{-1} down to a prescribed value in a hierarchical manner. In particular, we propose to decompose the target spectrum into several segments of different scales, and then allocate the computation of each segment to a certain level of the compressing sequence so that the problem on each level is well-conditioned.

To implement this idea, we first go back to the 1-level compression settings. Suppose that we have accurately obtained the first m essential eigenpairs (μ_i, v_i), $i = 1, \ldots, m$, of $\Theta = \Psi(\Psi^T A \Psi)^{-1}\Psi^T = \Psi A_{st}^{-1}\Psi^T$, and our aim is to compute the following $m_{tar} - m$ eigenpairs (namely, extend to the first m_{tar} eigenpairs) using the Lanczos method. Define $V_m = \text{span}\{v_i : 1 \leq i \leq m\}$ and $V_{m^+} = \text{span}\{v_i : m < i \leq N\} = V_m^{\perp} \cap \text{span}\{\Psi\}$. Then to perform the Lanczos method to compute the next segment of eigenpairs of Θ, we need to repeatedly apply the operator $\Psi A_{st}^{-1}\Psi^T$ to vectors in V_{m^+}, which requires us to compute $A_{st}^{-1}w$ for $w \in W_{m^+} = \Psi^T(V_{m^+})$.

Ideally we want the computation of the following $m_{tar} - m$ eigenpairs to be restricted to a problem with bounded spectrum width that is proportional to $\mu_m/\mu_{m_{tar}}$. This is possible since we assume that we have accurately obtained the span space V_m of the first m eigenvectors, and thus we can consider our problem in the reduced space orthogonal to V_m. In this case, the CG method will be efficient for computing inverse matrix operations.

Definition 4.1. *Let A be an SPD matrix, and V be an invariant subspace of A. We define the condition number of A with respect to V as*

$$\kappa(A, V) = \frac{\lambda_{\max}(A, V)}{\lambda_{\min}(A, V)},$$

where

$$\lambda_{\max}(A,V) = \max_{v\in V, v\neq 0} \frac{v^TAv}{v^Tv}, \qquad \lambda_{\min}(A,V) = \min_{v\in V, v\neq 0} \frac{v^TAv}{v^Tv}.$$

THEOREM 4.2. *Let A be an SPD matrix, and V be an invariant subspace of A. When using the CG method to solve $Ax = b$ with initial guess x_0 such that $r_0 = b - Ax_0 \in V$, we have the following estimate:*

$$\|x_k - x_*\|_A \le 2\left(\frac{\sqrt{\kappa(A,V)}-1}{\sqrt{\kappa(A,V)}+1}\right)^k \|x_0 - x_*\|_A$$

and

$$\|x_k - x_*\|_2 \le 2\sqrt{\kappa(A,V)}\left(\frac{\sqrt{\kappa(A,V)}-1}{\sqrt{\kappa(A,V)}+1}\right)^k \|x_0 - x_*\|_2,$$

where x_* is the exact solution, and $x_k \in x_* + V$ is the solution at the kth step of the CG iteration. Thus it takes $k = O(\kappa(A,V)\cdot\log\frac{1}{\epsilon})$ steps (or $k = O(\kappa(A,V)\cdot(\log\kappa(A,V) + \log\frac{1}{\epsilon}))$ steps) to obtain a solution subject to relative error ϵ in the energy norm (or l_2 norm).

Proof. We only need to notice that the k-order Krylov subspace $\mathcal{K}(A, r_0, k)$ generated by A and r_0 satisfies

$$\mathcal{K}(A, r_0, k) \subset V \quad \forall k \in \mathbb{Z}. \qquad \square$$

Notice that, for any $i = m+1, \dots, N$, though $v_i \in V_{m^+}$ is an eigenvector of $\Theta = \Psi A_{st}^{-1}\Psi^T$, $w_i = \Psi^T v_i$ is not an eigenvector of A_{st}^{-1}(but an eigenvector of $\Psi^T\Psi A_{st}^{-1}$) since we do not require Ψ to be orthonormal. Therefore the space W_{m^+} is not an invariant space of A_{st}, and if we directly use the CG method to solve $A_{st}x = w$, the convergence rate will depend on $\kappa(A_{st}) = \lambda_{\max}(A_{st})\lambda_{\min}(A_{st})^{-1}$, instead of $\lambda_{\max}(A_{st})\mu_m$ as intended. Though we bound $\lambda_{\max}(A_{st})$ from above by $\delta(\mathcal{P})$ and $\lambda_{\min}(A_{st})$ from below by $\lambda_{\min}(A)$ (see Theorem 2.2), $\kappa(A_{st})$ can be still large since we prescribe a bounded compression rate in practice to ensure the efficiency of the compression algorithm.

Therefore, we need to find a proper invariant space, so that we can make use of the knowledge of the space V_m and restrict the computation of $A_{st}^{-1}w$ to a problem of narrower spectrum.

LEMMA 4.3. *Let (μ_i, v_i), $i = 1, \dots, N$, be the essential eigenpairs of $\Theta = \Psi(\Psi^TA\Psi)^{-1}\Psi^T = \Psi A_{st}^{-1}\Psi^T$, such that $\mu_1 \ge \mu_2 \ge \cdots \ge \mu_N > 0$. Let $(\Psi^T\Psi)^{\frac{1}{2}}$ be the square root of the SPD matrix $\Psi^T\Psi$. Then (μ_i, z_i), $i = 1, \dots, N$, are all eigenpairs of $(\Psi^T\Psi)^{\frac{1}{2}}A_{st}^{-1}(\Psi^T\Psi)^{\frac{1}{2}}$, where*

$$z_i = (\Psi^T\Psi)^{-\frac{1}{2}}\Psi^T v_i, \quad 1 \le i \le N.$$

Moreover, for any subset $S \subset \{1, 2, \dots, N\}$, and $Z_S = \operatorname{span}\{z_i : i \in S\}$, we have

$$\mathcal{K}(A_\Psi, z, k) \subset Z_S \quad \forall z \in Z_S \ \forall k \in \mathbb{Z},$$

where $A_\Psi = (\Psi^T\Psi)^{-\frac{1}{2}}A_{st}(\Psi^T\Psi)^{-\frac{1}{2}}$.

LEMMA 4.4. *Let Ψ be given in (7), then we have*

$$\lambda_{\min}(\Psi^T\Psi) \ge 1, \quad \lambda_{\max}(\Psi^T\Psi) \le 1 + \varepsilon(\mathcal{P})\delta(\mathcal{P}),$$

and thus

$$\kappa(\Psi^T\Psi) \le 1 + \varepsilon(\mathcal{P})\delta(\mathcal{P}).$$

Proof. Let U be the orthogonal complement basis of Φ given in (10), so $[\Phi, U]$ is an orthonormal basis of $\mathbb{R}^n$, and we have $\Phi\Phi^T + UU^T = I_n$. Since $\Phi^T\Psi = \Phi^T A^{-1}\Phi(\Phi^T A^{-1}\Phi)^{-1} = I_N$, we have

$$\Psi^T\Psi = \Psi^T\Phi\Phi^T\Psi + \Psi^T UU^T\Psi = I_N + \Psi^T UU^T\Psi.$$

We then immediately obtain $\Psi^T\Psi \succeq I_N$ and, thus, $\lambda_{\min}(\Psi^T\Psi) \geq 1$. To obtain an upper bound of $\lambda_{\max}(\Psi^T\Psi)$, we notice that from the construction of Φ we have

$$\|x - P_\Phi x\|_2^2 \leq \varepsilon(\mathcal{P})x^T Ax \quad \forall x \in \mathbb{R}^n \quad \Longrightarrow \quad (I_n - P_\Phi)^2 \preceq \varepsilon(\mathcal{P})A,$$

where $P_\Phi = \Phi\Phi^T$ denotes the orthogonal projection into $\operatorname{span}\{\Phi\}$. Since $\Phi\Phi^T + UU^T = I_n$, we have

$$UU^T = I_n - \Phi\Phi^T = (I_n - \Phi\Phi^T)^2 \preceq \varepsilon(\mathcal{P})A.$$

Therefore we have

$$\Psi^T\Psi = I_N + \Psi^T UU^T\Psi \preceq I_N + \varepsilon(\mathcal{P})\Psi^T A\Psi = I_N + \varepsilon(\mathcal{P})A_{st},$$

and by Theorem 2.2 we obtain

$$\lambda_{\max}(\Psi^T\Psi) \leq 1 + \varepsilon(\mathcal{P})\lambda_{\max}(A_{st}) \leq 1 + \varepsilon(\mathcal{P})\delta(\mathcal{P}). \qquad \square$$

THEOREM 4.5. *Let A_Ψ and (μ_i, z_i) be defined as in Lemma 4.3. Let $Z_{m^+} = \operatorname{span}\{z_i : m < i \leq N\}$, then Z_{m^+} is an invariant space of A_Ψ, and we have*

$$\kappa(A_\Psi, Z_{m^+}) \leq \mu_{m+1}\delta(\mathcal{P}).$$

Proof. By Lemmas 4.3 and 4.4, we have

$$\begin{aligned}\lambda_{\max}(A_\Psi, Z_{m^+}) \leq \lambda_{\max}(A_\Psi) &= \|(\Psi^T\Psi)^{-\frac{1}{2}}A_{st}(\Psi^T\Psi)^{-\frac{1}{2}}\|_2 \\ &\leq \|A_{st}\|_2\|(\Psi^T\Psi)^{-1}\|_2 \leq \delta(\mathcal{P}).\end{aligned}$$

And by the definition of Z_{m^+}, we have

$$\begin{aligned}\lambda_{\min}(A_\Psi, Z_{m^+}) &= \frac{1}{\lambda_{\max}(A_\Psi^{-1}, Z_{m^+})} \\ &= \frac{1}{\lambda_{\max}((\Psi^T\Psi)^{\frac{1}{2}}A_{st}^{-1}(\Psi^T\Psi)^{\frac{1}{2}}, Z_{m^+})} = \frac{1}{\mu_{m+1}}. \qquad \square\end{aligned}$$

Inspired by Lemma 4.4 and Theorem 4.5, we now consider solving $A_{st}x = w$ efficiently for $w \in W_{m^+} = \Psi^T(V_{m^+}) = (\Psi^T\Psi)^{\frac{1}{2}}(Z_{m^+})$ by making use of the controlled condition number $\kappa(A_\Psi, Z_{m^+})$ and $\kappa(\Psi^T\Psi)$. Theoretically, we can compute $x = A_{st}^{-1}w$ by the following steps:

(i) Compute $b = (\Psi^T\Psi)^{-\frac{1}{2}}w \in Z_{m^+}$;

(ii) Use the CG method to compute $y = A_\Psi^{-1}b$ with initial guess y_0 such that $b - A_\Psi y_0 \in Z_{m^+}$;

(iii) Compute $x = (\Psi^T\Psi)^{-\frac{1}{2}}y$.

Notice that this procedure is exactly solving $A_{st}x = w$ using the preconditioned CG method with preconditioner $\Psi^T\Psi$, which only involves applying A_{st} and $(\Psi^T\Psi)^{-1}$ to vectors, but still enjoys the good conditioning property of A_Ψ restricted to Z_{m^+}. Therefore we have the following estimate.

COROLLARY 4.6. *Consider using the PCG method to solve $A_{st}x = w$ for $w \in W_{m^+}$ with preconditioner $\Psi^T\Psi$ and initial guess x_0 such that $r_0 = w - A_{st}x_0 \in W_{m^+}$. Let x_* be the exact solution, and x_k be the solution at the kth step of the PCG iteration. Then we have*

$$\|x_k - x_*\|_{A_{st}} \le 2\left(\frac{\sqrt{\kappa(A_\Psi, Z_{m^+})}-1}{\sqrt{\kappa(A_\Psi, Z_{m^+})}+1}\right)^k \|x_0 - x_*\|_{A_{st}}$$

and

$$\|x_k - x_*\|_2 \le 2\sqrt{\kappa(\Psi^T\Psi)\kappa(A_\Psi, Z_{m^+})}\left(\frac{\sqrt{\kappa(A_\Psi, Z_{m^+})}-1}{\sqrt{\kappa(A_\Psi, Z_{m^+})}+1}\right)^k \|x_0 - x_*\|_2.$$

Proof. Let $y_k = (\Psi^T\Psi)^{\frac{1}{2}}x_k$ and $y_* = (\Psi^T\Psi)^{\frac{1}{2}}x_*$, then we have

$$\|y_k - y_*\|_2^2 = (x_k - x_*)^T\Psi^T\Psi(x_k - x_*)$$

and

$$\|y_k - y_*\|_{A_\Psi}^2 = (y_k - y_*)^T A_\Psi (y_k - y_*) = \|x_k - x_*\|_{A_{st}}^2.$$

Noticing that $(\Psi^T\Psi)^{-\frac{1}{2}}r_0 \in Z_{m^+}$ and $\mathcal{K}(A_\Psi, (\Psi^T\Psi)^{-\frac{1}{2}}r_0, k) \subset Z_{m^+}$ $\forall k$, the results follow from Theorem 4.2. □

By Corollary 4.6, to compute a solution of $A_{st}x = w$ subject to a relative error ϵ in the A_{st}-norm, the number of needed PCG iterations is

$$O\left(\kappa(A_\Psi, Z_{m^+}) \cdot \log\frac{1}{\epsilon}\right) = O\left(\mu_{m+1}\delta(\mathcal{P}) \cdot \log\frac{1}{\epsilon}\right).$$

This is also an estimate of the number of needed PCG iterations for a relative error ϵ in the l_2-norm, if we assume that $\kappa(\Psi^T\Psi), \kappa(A_\Psi, Z_{m^+}) \le \frac{1}{\epsilon}$.

In what follows we will denote $M = \Psi^T\Psi$. Notice that the nonzero entries of M are due to the overlapping support of column basis vectors of Ψ, while the nonzero entries of $A_{st} = \Psi^T A\Psi$ are results of interactions between column basis vectors of Ψ with respect to A. Thus we can reasonably assume that $nnz(M) \le nnz(A_{st})$. Suppose that in each iteration of the whole PCG procedure, we also use the CG method to compute $M^{-1}b$ for some b subject to a relatively higher precision $\hat{\epsilon}$, which requires a cost of $O(nnz(M) \cdot \kappa(M) \cdot \log\frac{1}{\hat{\epsilon}})$. In practice it is sufficient to take $\hat{\epsilon}$ smaller than but comparable to ϵ (e.g., $\hat{\epsilon} = 0.1\epsilon$), so $\log(\frac{1}{\hat{\epsilon}}) = O(\log\frac{1}{\epsilon})$. By Lemma 4.4 we have $\kappa(M) = O(\varepsilon(\mathcal{P})\delta(\mathcal{P}))$. Then the computational complexity of each single iteration can be bounded by

$$O\big(nnz(A_{st})\big) + O\left(nnz(M)\cdot\kappa(M)\cdot\log\frac{1}{\epsilon}\right) = O\left(nnz(A_{st})\cdot\varepsilon(\mathcal{P})\delta(\mathcal{P})\cdot\log\frac{1}{\epsilon}\right),$$

and the total cost of computing a solution of $A_{st}x = w$ subject to a relative error ϵ is

$$O\left(\mu_{m+1}\delta(\mathcal{P})\cdot nnz(A_{st})\cdot\varepsilon(\mathcal{P})\delta(\mathcal{P})\cdot\left(\log\frac{1}{\epsilon}\right)^2\right). \tag{23}$$

We remark that when the original size of $A \in \mathbb{R}^{n\times n}$ is large, the eigenvectors V are long and dense. It would be expensive to compute inner products with these long vectors over and over again. In fact, in the previous discussions the operator $\Theta = \Psi A_{st}^{-1}\Psi^T$(of the same size as A) and the eigenvectors V are only for the purpose of analysis used to explain the idea of our method. In practice, for a long vector $v = \Psi\hat{v}$, we don't need to keep track of the whole vector, but only need to store its much shorter coefficients $\hat{v}$ of compressed dimension N instead. When we compute $v_2 = \Theta v_1 = \Psi A_{st}^{-1}\Psi^T v_1$, it is equivalent to computing $\hat{v}_2 = A_{st}^{-1}M\hat{v}_1$, where $v_j = \Psi\hat{v}_j,\ j = 1, 2$, and $M = \Psi^T\Psi$. One can check that the analysis presented above still applies. So in the implementation of our method, we only deal with operator $A_{st}^{-1}M$ and short vectors $\widehat{V}$, and the long eigenvectors V and Ψ will not appear until in the very end when we recover $V = \Psi\widehat{V}$. We remark that since the eigenvectors of Θ are orthogonal, their coefficient vectors $\widehat{V}$ are M-orthogonal, i.e., $\widehat{V}^T M\widehat{V} = I$. We use $\|x\|_M$ to denote the norm $\sqrt{x^T Mx}$.

Recall that in the Lanczos method with respect to operator Θ, the upper-Hessenberg matrix T in the Arnoldi relation

$$\Theta V = VT + fe^T$$

is indeed tridiagonal, since Θ is symmetric, and $V^T[V, f] = [I, \mathbf{0}]$. This upper-Hessenberg matrix T being tridiagonal is the reason why the implicit restarting process (Algorithm 2) is efficient. Now since we are actually dealing with the operator $A_{st}^{-1}M$ and the coefficient vectors $\widehat{V}^T = M^{-1}\Psi^T V$, the Arnoldi relation becomes

$$A_{st}^{-1}M\widehat{V} = \widehat{V}T + \hat{f}e^T,$$

where $\hat{f} = M^{-1}\Psi^T f$. So as long as we keep $\widehat{V}$ M-orthogonal and $\hat{f}$ M-orthogonal to $\widehat{V}$, T will still be tridiagonal since

$$T = \widehat{V}^T M\widehat{V}T = \widehat{V}^T M(\widehat{V}T + \hat{f}e^T) = \widehat{V}^T MA_{st}^{-1}M\widehat{V}$$

is symmetric. We therefore modified Algorithm 1 to Algorithm 3 to take M-orthogonality into consideration.

Summarizing the analysis above, we propose Algorithm 4 for extending a given collection of eigenpairs using the Lanczos-type method. The operator $OP(\,\cdot\,; A_{st}, M, \epsilon_{op})$ exploits our key idea that uses $M = \Psi^T\Psi$ as the preconditioner to effectively reduce the number of PCG iterations in every operation of $A_{st}^{-1}M$. For convenience, we will use $x = pcg(A, b, M, x_0, \epsilon)$ to represent the operation of computing $x = A^{-1}b$ using the PCG method with preconditioner M and initial guess x_0, subject to relative error ϵ. $x = pcg(A, b, -, x_0, \epsilon)$ means no preconditioner is used (i.e., the normal CG method), and $x = pcg(A, b, M, -, \epsilon)$ means an all zero vector is used as the initial guess.

Given an existing eigenspace $V_{ini} = \Psi\widehat{V}_{ini}$, Algorithm 4 basically uses the Lanczos method to find the following eigenpairs of Θ in the space $V_{ini}^{\perp}$. Notice that the output $\widehat{V}_{ex}$ gives the coefficients of the desired eigenvectors V_{ex} in the basis Ψ. However, differently from the classical Lanczos method, we do not prescribe a specific number for the output eigenpairs. Instead, we set a threshold μ to bound the last output eigenvalue. As we will develop our idea into a multilevel algorithm that pursues a number of target eigenpairs hierarchically, the output of the current level will be used to generate the initial eigenspace for the higher level. Therefore, the purpose of setting a threshold μ on the current level is to bound the restricted condition number on the higher level, as the initial eigenspace V_{ini} from the lower level helps to bound the restricted condition number on the current level.

Algorithm 3 General Lanczos iteration (p-step extension).

```
Input: $\widehat{V}$, $T$, $\hat{f}$, target operator $op(\cdot)$, $p$, inner product matrix $M$
Output: $\widehat{V}$, $T$, $\hat{f}$
 1: $k$ = column number of $\widehat{V}$;
 2: for $i = 1 : p$ do
 3:     $\beta = \|\hat{f}\|_M$;
 4:     if $\beta < \epsilon$ then
 5:         generate a new random $\hat{f}$, $\beta = \|\hat{f}\|_M$;
 6:     end if
 7:     $T \leftarrow \begin{pmatrix} T \\ \beta e_{k+i-1}^T \end{pmatrix}$,    $\hat{v} = \hat{f}/\beta$,    $\widehat{V} \leftarrow [\widehat{V}, \hat{v}]$;
 8:     $w = op(\hat{v})$;
 9:     $h = \widehat{V}^T M w$,    $T \leftarrow [T, h]$;
10:     $\hat{f} = w - \widehat{V}h$;
11:     Reorthogonalize to adjust $f$(with respect to $M$-orthogonality);
12: end for
```

Function $y =$ **Operator** $OP(x; A_{st}, M, \epsilon_{op})$.

```
1: $w = Mx$;
2: $y = pcg(A_{st}, w, M, -, \epsilon_{op})$;
```

Algorithm 4 Eigenpair extension.

```
Input: $\widehat{V}_{ini}$, $D_{ini}$, $OP(\,\cdot\,; A_{st}, M, \epsilon_{op})$, target number $m_{tar}$,
    prescribed accuracy $\epsilon$, eigenvalue threshold $\mu$, searching step $d$.
Output: $\widehat{V}_{ex}$, $D_{ex}$.
 1: Generate random initial vector $\widehat{V} = \hat{v}$ that is $M$-orthogonal to $\widehat{V}_{ini}$;
 2: repeat
 3:     perform $d$ steps of general Lanczos iteration (Algorithm 3) with operator $OP$
        to extend $\widehat{V}, T$;
 4:     while Lanczos residual $> \epsilon$, do
 5:         Perform $c \cdot d$ steps of shifts to restart Lanczos (Algorithm 2) and renew $\widehat{V}, T$;
 6:     end while
 7:     Find the smallest eigenvalue of $T$ as $\hat{\mu}$;
 8: until $\hat{\mu} < \mu$ or $\dim(\widehat{V}) \geq m_{tar} - \dim(\widehat{V}_{ini})$.
 9: $m_{new} = \dim(\widehat{V})$;
10: while Lanczos residual $> \epsilon$, do
11:     Perform $c \cdot m_{new}$ steps of shifts to restart Lanczos (Algorithm 2) and renew
        $\widehat{V}, T$;
12: end while
13: $PSP^T = T$ (Schur decomposition);
14: $\widehat{V}_{ex} = [\widehat{V}_{ini}, \widehat{V}P]$,    $D_{ex} = \begin{bmatrix} D_{ini} & \\ & S \end{bmatrix}$;
```

The choice of the threshold μ will be discussed in detail after we introduce the refinement procedure. Here, to develop a hierarchical spectrum completion method using the analysis above, we state the hierarchical versions of Lemma 4.4 and Theorem 4.5.

LEMMA 4.7. *Let $\mathbf{\Psi}^{(k)}$ be given in* (14), *and $M^{(k)} = (\mathbf{\Psi}^{(k)})^T\mathbf{\Psi}^{(k)}$. Then we have*

$$\lambda_{\min}(M^{(k)}) \geq 1, \quad \lambda_{\max}(M^{(k)}) \leq 1 + \varepsilon_k\delta_k,$$

and thus

$$\kappa(M^{(k)}) \leq 1 + \varepsilon_k\delta_k.$$

Proof. The proof is similar to the proof of Lemma 4.4. Let $\boldsymbol{U}^{(k)} = (\mathbf{\Phi}^{(k)})^{\perp}$ be the orthogonal complement basis of $\mathbf{\Phi}^{(k)}$. According to Theorem 2.6, we have

$$\|x - P_{\mathbf{\Phi}^{(k)}}x\|_2^2 \leq \varepsilon_k\|x\|_A^2,$$

which implies that

$$\boldsymbol{U}^{(k)}(\boldsymbol{U}^{(k)})^T = (I_n - \mathbf{\Phi}^{(k)}(\mathbf{\Phi}^{(k)})^T) \leq \varepsilon_k A.$$

Noticing that $(\mathbf{\Phi}^{(k)})^T\mathbf{\Psi}^{(k)} = I_{N^{(k)}}$, $\mathbf{\Phi}^{(k)}(\mathbf{\Phi}^{(k)})^T + \boldsymbol{U}^{(k)}(\boldsymbol{U}^{(k)})^T = I_n$, we thus have

$$\begin{aligned} M^{(k)} &= (\mathbf{\Psi}^{(k)})^T\mathbf{\Phi}^{(k)}(\mathbf{\Phi}^{(k)})^T\mathbf{\Psi}^{(k)} + (\mathbf{\Psi}^{(k)})^T\boldsymbol{U}^{(k)}(\boldsymbol{U}^{(k)})^T\mathbf{\Psi}^{(k)} \\ &= I_{N^{(k)}} + (\mathbf{\Psi}^{(k)})^T\boldsymbol{U}^{(k)}(\boldsymbol{U}^{(k)})^T\mathbf{\Psi}^{(k)}, \\ &\Longrightarrow \qquad I_{N^{(k)}} \preceq M^{(k)} \preceq I_{N^{(k)}} + \varepsilon_k(\mathbf{\Psi}^{(k)})^T A\mathbf{\Psi}^{(k)} = I_{N^{(k)}} + \varepsilon_k A^{(k)}. \end{aligned}$$

Therefore we have $\lambda_{\min}(M^{(k)}) \geq 1$, and by Corollary 2.5 we have

$$\lambda_{\max}(M^{(k)}) \leq 1 + \varepsilon_k\lambda_{\max}(A^{(k)}) \leq 1 + \varepsilon_k\delta_k. \qquad \square$$

THEOREM 4.8. *Let $A^{(k)}$ and $\mathbf{\Psi}^{(k)}$ be given in* (14), *and $M^{(k)} = (\mathbf{\Psi}^{(k)})^T\mathbf{\Psi}^{(k)}$. Let $(\mu_i^{(k)}, v_i^{(k)})$, $i = 1, \ldots, N^{(k)}$, be the essential eigenpairs of $\Theta^{(k)} = \mathbf{\Psi}^{(k)}(A^{(k)})^{-1}(\mathbf{\Psi}^{(k)})^T$. Define*

$$z_i^{(k)} = (M^{(k)})^{-\frac{1}{2}}(\mathbf{\Psi}^{(k)})^T v_i^{(k)}, \quad 1 \leq i \leq N^{(k)}.$$

Given an integer m_k, let $Z_{m_k^+}^{(k)} = \operatorname{span}\{z_i^{(k)} : m_k < i \leq N^{(k)}\}$, then $Z_{m_k^+}^{(k)}$ is an invariant space of $A_{\mathbf{\Psi}}^{(k)} = (M^{(k)})^{-\frac{1}{2}}A^{(k)}(M^{(k)})^{-\frac{1}{2}}$, and we have

$$\kappa(A_{\mathbf{\Psi}}^{(k)}, Z_{m^+}^{(k)}) \leq \mu_{m_k+1}^{(k)}\delta_k.$$

Moreover, consider using the PCG method to solve $A^{(k)}x = w$ for $w \in W_{m_k^+}^{(k)}$ with preconditioner $M^{(k)}$ and initial guess x_0 such that $r_0 = w - A^{(k)}x_0 \in W_{m_k^+}^{(k)}$, where

$W^{(k)}_{m_k^+} = \operatorname{span}\{(\boldsymbol{\Psi}^{(k)})^T v_i^{(k)} : m_k < i \le N^{(k)}\}$. *Let x_* be the exact solution, and x_t be the solution at the tth step of the PCG iteration. Then we have*

$$\|x_t - x_*\|_{A^{(k)}} \le 2\left(\frac{\sqrt{\mu^{(k)}_{m_k+1}\delta_k}-1}{\sqrt{\mu^{(k)}_{m_k+1}\delta_k}+1}\right)^t \|x_0 - x_*\|_{A^{(k)}}$$

and

$$\|x_t - x_*\|_2 \le 2\sqrt{\varepsilon_k \mu^{(k)}_{m_k+1}\delta_k^2}\left(\frac{\sqrt{\mu^{(k)}_{m_k+1}\delta_k}-1}{\sqrt{\mu^{(k)}_{m_k+1}\delta_k}+1}\right)^t \|x_0 - x_*\|_2.$$

Recall that we will use the CG method to implement Lanczos iteration on each level k to complete the target spectrum. To ensure the efficiency of the CG method, namely, to bound the restricted condition number $\kappa(A^{(k)}_{\boldsymbol{\Psi}}, Z^{(k)}_{m^+})$ on each level, we need a priori knowledge of the spectrum $\{(\mu_i^{(k)}, v_i^{(k)}) : 1 \le i \le m_k\}$ such that $\mu^{(k)}_{m_k+1}\delta_k$ is uniformly bounded. This given spectrum should be inductively computed on the lower level $k+1$. But notice that there is a compression error between each two neighbor levels, which will compromise the orthogonality and thus the theoretical bound for restricted condition number, if we directly use the spectrum of the lower level as an a priori spectrum of the current level. Therefore we introduce a refinement method in section 5 to overcome this difficulty.

5. Cross-level refinement of eigenspace.

5. Cross-level refinement of eigenspace. In the previous section we have established a 1-level spectrum extension method, given that a partial accurate spectrum is provided. To develop this method into an inductive hierarchical spectrum completion procedure, a natural idea is to use the spectrum computed at the lower level as the initial spectrum to be used in the higher level. However, such an initial spectrum is not actually good enough since there is a compression error between each two neighboring levels. Thus we need to use a compatible refinement technique to refine the initial spectrum.

Now consider the cross-level spectrum refinement between the two consecutive levels, the h-level and the l-level. The two operators are $\Theta^h = \boldsymbol{\Psi}^h((\boldsymbol{\Psi}^h)^T A \boldsymbol{\Psi}^h)^{-1}(\boldsymbol{\Psi}^h)^T$ and $\Theta^l = \boldsymbol{\Psi}^l((\boldsymbol{\Psi}^l)^T A \boldsymbol{\Psi}^l)^{-1}(\boldsymbol{\Psi}^l)^T$, respectively. We have the relations

$$\boldsymbol{\Psi}^l = \boldsymbol{\Psi}^h \Psi^l, \quad \mathcal{U}^l = \boldsymbol{\Psi}^h U^l,$$

$$A^l_{st} = (\boldsymbol{\Psi}^l)^T A \boldsymbol{\Psi}^l = (\Psi^l)^T(\boldsymbol{\Psi}^h)^T A \boldsymbol{\Psi}^h \Psi^l = (\Psi^l)^T A^h_{st} \Psi^l,$$

$$B^l_{st} = (\mathcal{U}^l)^T A \mathcal{U}^l = (U^l)^T(\boldsymbol{\Psi}^h)^T A \boldsymbol{\Psi}^h U^l = (U^l)^T A^h_{st} U^l,$$

$$(A^h_{st})^{-1} = \Psi^l (A^l_{st})^{-1}(\Psi^l)^T + U^l (B^l_{st})^{-1}(U^l)^T,$$

$$\Theta^h = \boldsymbol{\Psi}^h\big(\Psi^l(A^l_{st})^{-1}(\Psi^l)^T + U^l(B^l_{st})^{-1}(U^l)^T\big)(\boldsymbol{\Psi}^h)^T = \Theta^l + \mathcal{U}^l(B^l_{st})^{-1}(\mathcal{U}^l)^T. \tag{24}$$

Now suppose that we have obtained the first m_l essential eigenpairs $(\mu_{l,i}, v_{l,i})$, $i = 1, \dots, m_l$, of Θ^l. We want to use these eigenpairs as an initial guess to obtain the first m_h essential eigenpairs of Θ^h. Recall that we have the estimates

$$|\mu_{h,i} - \mu_{l,i}| \le \varepsilon_l, \quad 1 \le i \le m_l,$$

and

$$\|\Theta^h v_{l,i} - \mu_{h,i} v_{l,i}\|_2 \le 2\varepsilon_l, \quad 1 \le i \le m_l,$$

where ε_l is the compression error bound. These estimates give us confidence that we can obtain $(\mu_{h,i}, v_{h,i})$, $i = 1, \dots, m_h$, efficiently from $(\mu_{l,i}, v_{l,i})$, $i = 1, \dots, m_l$, by using some refinement technique.

Indeed, we will use the orthogonal iteration with Ritz acceleration as our refinement method. Consider an initial guess $Q^{(0)}$ of the first m eigenvectors of an SPD operator Θ. To obtain more accurate eigenvalues and eigenspace, the orthogonal iteration with Ritz acceleration runs as follows:

```
(*)   $Q^{(0)} \in \mathbb{R}^{n\times m}$ given with $(Q^{(0)})^T Q^{(0)} = I_m$
      $F^{(0)} = \Theta Q^{(0)}$
      **for** $k = 1, 2, \dots$
          $Q^{(k)} R^{(k)} = F^{(k-1)}$        (QR factorization)
          $F^{(k)} = \Theta Q^{(k)}$
          $S^{(k)} = (Q^{(k)})^T F^{(k)}$
          $P^{(k)} D^{(k)} (P^{(k)})^T = S^{(k)}$        (Schur decomposition)
          $Q^{(k)} \leftarrow Q^{(k)} P^{(k)}$
          $F^{(k)} \leftarrow F^{(k)} P^{(k)}$
      **end**
```

To state the convergence property of the orthogonal iteration with Ritz acceleration, we first define the distance between two spaces. Let $V_1, V_2 \subset \mathbb{R}^n$ be two linear spaces, and $\boldsymbol{P}_{V_1}, \boldsymbol{P}_{V_2}$ be the orthogonal projections onto V_1, V_2, respectively. We define the distance between V_1 and V_2 as

$$\mathrm{dist}(V_1, V_2) = \|\boldsymbol{P}_{V_1} - \boldsymbol{P}_{V_2}\|_2.$$

We also use the same notation $\mathrm{dist}(V_1, V_2)$ when V_1, V_2 are matrices of column vectors. In this case $\mathrm{dist}(V_1, V_2)$ means $\mathrm{dist}(\mathrm{span}\{V_1\}, \mathrm{span}\{V_2\})$.

Suppose that the diagonal entries $\mu_i^{(k)}$, $i = 1, \dots, m$, of $D^{(k)}$ are in a decreasing order, then $\mu_i^{(k)}$ is a good approximation of the ith eigenvalue of Θ, and $\mathrm{span}\{Q_i^{(k)}\}$ is a good approximation of the eigenspace spanned by the first i eigenvectors of Θ, where $Q_i^{(k)}$ denotes the first i columns of $Q^{(k)}$. We would like to emphasize that the meaning of the superscript (k) of $\mu_i^{(k)}$ is different from those in section 4. More precisely, we have the following convergence estimate.

Theorem 5.1 (Stewart, 1968 [29]). *Let (μ_i, v_i), $i = 1, \dots, N$, be the ordered (essential) eigenpairs of Θ, and let $\mu_i^{(k)}$, $i = 1, \dots, m$, be the ordered eigenvalues of $D^{(k)} = (Q^{(k)})^T \Theta Q^{(k)}$ given in the orthogonal iteration with Ritz acceleration $(*)$. Let $V_m = [v_1, v_2, \dots, v_m]$, and $d^{(0)} = \mathrm{dist}(V_m, Q^{(0)})$. Then we have*

$$|\mu_i - \mu_i^{(k)}| \le O\left(\left(\frac{\mu_{m+1}}{\mu_i}\right)^{2k} \cdot \|\Theta\|_2 \cdot \frac{(d^{(0)})^2}{1 - (d^{(0)})^2}\right), \quad 1 \le i \le m.$$

Moreover, we have

$$\mathrm{dist}(V_m, Q^{(k)}) \le O\left(\left(\frac{\mu_{m+1}}{\mu_m}\right)^{k} \cdot \frac{d^{(0)}}{\sqrt{1 - (d^{(0)})^2}}\right),$$

and for $i = 1, \dots, m-1$, if we further assume that $\alpha_i = \mu_i - \mu_{i+1} > 0$, then we have

$$\operatorname{dist}(V_i, Q_i^{(k)}) \le O\left(\left(\frac{\mu_{m+1}}{\mu_i}\right)^k \cdot \frac{d^{(0)}}{\sqrt{1-(d^{(0)})^2}}\right) + O\left(\frac{\sqrt{i}}{\alpha_i} \cdot \left(\frac{\mu_{m+1}^2}{\mu_m \mu_i}\right)^k \cdot \|\Theta\|_2 \cdot \frac{(d^{(0)})^2}{1-(d^{(0)})^2}\right),$$

where V_i and $Q_i^{(k)}$ are the first i columns of V_m and $Q^{(k)}$, respectively.

Now we go back to our problem, where we have $\Theta = \Theta^h$, $m = m_l$, and $Q^{(0)} = V_{m_l}^l = [v_{l,1}, \dots, v_{l,m_l}]$. We next consider the efficiency of this refinement technique in our problem. As long as the initial distance $d^{(0)} = \operatorname{dist}(V_{m_l}^h, V_{m_l}^l) < 1$, the first m_h eigenvalues and the eigenspace of the first m_h eigenvectors of Θ^h converge exponentially fast at a rate $(\frac{\mu_{h,m_l+1}}{\mu_{h,m_h}})^k$. We can expect that a few iterations of refinement will be sufficient to give an accurate eigenspace for narrowing down the residual spectrum of Θ^h, if we can ensure that the ratio $\frac{\mu_{h,m_l+1}}{\mu_{h,m_h}}$ is small enough. This will be verified in our numerical examples to be presented in section 7. In particular, to refine the first m_h eigenpairs subject to a prescribed accuracy ϵ, we need $K = O(\log(\frac{1}{\epsilon})/\log(\frac{\mu_{h,m_h}}{\mu_{h,m_l+1}}))$ refinement iterations.

The main cost of the refinement procedure comes from the computation of $\Theta^h Q^{(0)}$ and the computation of $\Theta^h Q^{(k)}$ in each iteration. We will reduce the computational cost by using the fact that $Q^{(k)}$ is a good approximation of eigenvectors of Θ^h. We first consider how to compute $\Theta^h Q^{(0)}$ efficiently.

Notice that in our problem, we take $Q^{(0)} = V_{m_l}^l$, whose columns are the first m_l eigenvectors of Θ^l. Therefore by (24), we have

$$\Theta^h Q^{(0)} = \Theta^h V_{m_l}^l = \Theta^l V_{m_l}^l + \mathcal{U}^l (B_{st}^l)^{-1} (\mathcal{U}^l)^T V_{m_l}^l = V_{m_l}^l D_{m_l}^l + \mathcal{U}^l (B_{st}^l)^{-1} (\mathcal{U}^l)^T V_{m_l}^l,$$

where $D_{m_l}^l$ is a diagonal matrix whose diagonal entries are $\mu_{l,1}, \mu_{l,2}, \dots, \mu_{l,m_l}$. Recall that by Lemma 2.3 and Corollary 2.5, $\kappa(B_{st}^l)$ is bounded by $\varepsilon_l \delta_h$ that can be well controlled in the decomposition procedure. Thus it is efficient to solve $(B_{st}^l)^{-1}$ using the CG method. As we have mentioned before, applying $(U^l)^T$ or U^l from the left is performed by doing patchwise Householder transformations that involve only one local Householder vector on each patch, which takes $O(N^h)$ computational cost, where N^h is the compressed dimension on level h or the size of A_{st}^h. Therefore in the CG method, the cost of matrix multiplication of $B_{st}^l = (U^l)^T A_{st}^h U^l$ mainly comes from the number of nonzero entries of A_{st}^h. Then the total computational cost of computing $\Theta^h Q^{(0)}$ subject to a relative error ϵ can be bounded by

$$O\left(m_l \cdot nnz(A_{st}^h) \cdot \varepsilon_l \delta_h \cdot \log\left(\frac{1}{\epsilon}\right)\right).$$

Next, we consider how to compute $\Theta^h Q^{(k)}$. To do so, we first compute $w_i^{(k)} = (\mathbf{\Psi}^h)^T q_i^{(k)}$, where $q_i^{(k)}$ is the ith column of $Q^{(k)}$, then compute $(A_{st}^h)^{-1} w_i^{(k)}$, and apply $\mathbf{\Psi}^h$. Again we will use the PCG method with predictioner $M^h = (\mathbf{\Psi}^h)^T \mathbf{\Psi}^h$ to compute $(A_{st}^h)^{-1} w_i^{(k)}$. As we have discussed in section 4, this is equivalent to using the CG method to compute $(A_{\mathbf{\Psi}}^h)^{-1} z_i^{(k)}$, where $A_{\mathbf{\Psi}}^h = (M^h)^{-\frac{1}{2}} A_{st}^h (M^h)^{-\frac{1}{2}}$, and $z_i^{(k)} = (M^h)^{-\frac{1}{2}} w_i^{(k)} = (M^h)^{-\frac{1}{2}} (\mathbf{\Psi}^h)^T q_i^{(k)}$. Inspired by Corollary 4.6, we seek to provide a good initial guess for the CG method to ensure efficiency. In the orthogonal iteration with Ritz acceleration $(*)$, one can check that $(Q^{(k)})^T(\Theta^h Q^{(k)} - Q^{(k)}D^{(k)}) = \mathbf{0}$, where $D^{(k)}$ is a diagonal matrix with diagonal entries $\mu_1^{(k)}, \mu_2^{(k)}, \ldots, \mu_{m_l}^{(k)}$ and, therefore,

$$\begin{aligned}
&(Z^{(k)})^T\big((A^h_{\mathbf{\Psi}})^{-1}Z^{(k)} - Z^{(k)}D^{(k)}\big)\\
&\quad= (Q^{(k)})^T\mathbf{\Psi}^h(M^h)^{-\frac{1}{2}}\left((A^h_{\mathbf{\Psi}})^{-1}(M^h)^{-\frac{1}{2}}(\mathbf{\Psi}^h)^TQ^{(k)} - (M^h)^{-\frac{1}{2}}(\mathbf{\Psi}^h)^TQ^{(k)}D^{(k)}\right)\\
&\quad= (Q^{(k)})^T\left(\mathbf{\Psi}^h(A^h_{st})^{-1}(\mathbf{\Psi}^h)^TQ^{(k)} - \mathbf{\Psi}^h(M^h)^{-1}(\mathbf{\Psi}^h)^TQ^{(k)}D^{(k)}\right)\\
&\quad= (Q^{(k)})^T\left(\Theta^hQ^{(k)} - Q^{(k)}D^{(k)}\right)\\
&\quad= \mathbf{0},
\end{aligned}$$

where we have used that $Q^{(k)} \in \operatorname{span}\{\mathbf{\Psi}^h\}$ and so $\mathbf{\Psi}^h(M^h)^{-1}(\mathbf{\Psi}^h)^TQ^{(k)} = Q^{(k)}$. This observation implies that if we use $\mu_i^{(k)}z_i^{(k)}$ as the initial guess for computing $(A^h_{\mathbf{\Psi}})^{-1}z_i^{(k)}$ using the CG method, the initial residual $z_i^{(k)} - (A^h_{\mathbf{\Psi}})(\mu_i^{(k)}z_i^{(k)})$ is orthogonal to $(A^h_{\mathbf{\Psi}})^{-1}Z^{(k)}$. Since $Q^{(k)}$ are already good approximate essential eigenvectors of Θ^h, $Z^{(k)}$ are good approximate eigenvectors of $(A^h_{\mathbf{\Psi}})^{-1}$, we can expect that the target eigenspace Z_{m_h}, namely, the eigenspace of the first m_h eigenvectors of $(A^h_{\mathbf{\Psi}})^{-1}$, can be well spanned in $\operatorname{span}\{(A^h_{\mathbf{\Psi}})^{-1}Z^{(k)}\}$. Therefore we can reasonably assume that $z_i^{(k)} - (A^h_{\mathbf{\Psi}})(\mu_i^{(k)}z_i^{(k)}) \in Z_{m_h^+} = Z_{m_h}^{\perp}$, and so again we can benefit from the restricted condition number $\kappa(A^h_{\mathbf{\Psi}}, Z_{m_h^+}) \le \mu_{h,m_h+1}\delta_h$ as introduced in section 4. Moreover, we notice that the spectral residual $\|\Theta^hq_i^{(k)} - \mu_i^{(k)}q_i^{(k)}\|_2$ is bounded by $2\varepsilon_l$ by Lemma 3.3, and we have

(25)
$$\|(A^h_{st})^{-1}w_i^{(k)} - \mu_i^{(k)}(M^h)^{-1}w_i^{(k)}\|_2 \le \|(A^h_{\mathbf{\Psi}})^{-1}z_i^{(k)} - \mu_i^{(k)}z_i^{(k)}\|_2 = \|\Theta^hq_i^{(k)} - \mu_i^{(k)}q_i^{(k)}\|_2,$$

where we have used $\lambda_{\min}(M^h) \ge 1$ (Lemma 4.7). Thus if we use $\mu_i^{(k)}z_i^{(k)}$ as the initial guess, the initial error will be bounded by $2\varepsilon_l$ at most, and the CG procedure will only need

$$O\left(\kappa(A^h_{\mathbf{\Psi}}, Z_{m_h^+}) \cdot \log\left(\frac{\varepsilon_l}{\epsilon}\right)\right) = O\left(\mu_{h,m_h+1}\delta_h \cdot \log\left(\frac{\varepsilon_l}{\epsilon}\right)\right)$$

iterations to achieve a relative accuracy ϵ, instead of $O(\kappa(A^h_{\mathbf{\Psi}}, Z_{m_h^+}) \cdot \log(\frac{1}{\epsilon}))$. Notice that using the initial guess $\mu_i^{(k)}z_i^{(k)}$ for $(A^h_{\mathbf{\Psi}})^{-1}z_i^{(k)}$ is equivalent to using the initial guess $\mu_i^{(k)}(M^h)^{-1}w_i^{(k)}$ for $(A^h_{st})^{-1}w_i^{(k)}$.

Supported by the analysis above, we will compute $(A^h_{st})^{-1}w_i^{(k)}$ using the PCG method with preconditioner M^h and initial guess $\mu_i^{(k)}(M^h)^{-1}w_i^{(k)}$. Again suppose that in each PCG iteration, we also use the CG method to apply $(M^h)^{-1}$ subject to a higher relative accuracy $\hat{\epsilon}$, which takes $O(nnz(M^h) \cdot \kappa(M^h) \cdot \log(\frac{1}{\hat{\epsilon}}))$ computational cost. In practice, it is sufficient to take $\hat{\epsilon}$ comparable to ϵ. Recalling that $nnz(M^h) \le nnz(A^h_{st})$, and $\kappa(M^h) \le O(\varepsilon_h\delta_h)$ (Lemma 4.7), the cost of computing $\Theta^hQ^{(k)}$ subject to a relative error ϵ is then bounded by

$$O\left(m_l \cdot \mu_{h,m_h+1}\delta_h \cdot \log\left(\frac{\varepsilon_l}{\epsilon}\right) \cdot nnz(A^h_{st}) \cdot \varepsilon_h\delta_h \cdot \log\left(\frac{1}{\epsilon}\right)\right).$$

Notice that in each refinement iteration we also need to perform one QR factorization and one Schur decomposition, which together cost $O(N^h \cdot m_l^2)$. However, as we have mentioned in the introduction, we only consider the asymptotic complexity of our method when the original A becomes superlarge. In this case, the number m_{tar} of the target eigenpairs is considered as a fixed constant, and so the term $O(N^h \cdot m_l^2) \leq O(N^h m_{tar}^2)$ is considered to be minor and will be omitted in our complexity analysis. Therefore, the total cost of refining the first m_h eigenpairs subject to a prescribed accuracy ϵ can be bounded by

$$\begin{aligned}
&O\left(m_l \cdot nnz(A_{st}^h) \cdot \varepsilon_l \delta_h \cdot \log\left(\frac{1}{\epsilon}\right)\right) \\
&\quad + O\Bigg(m_l \cdot \mu_{h,m_h+1}\delta_h \cdot \log\left(\frac{\varepsilon_l}{\epsilon}\right) \cdot nnz(A_{st}^h) \\
&\qquad \cdot \varepsilon_h \delta_h \cdot \log\left(\frac{1}{\epsilon}\right) \cdot \log\left(\frac{1}{\epsilon}\right) / \log\left(\frac{\mu_{h,m_h}}{\mu_{h,m_l+1}}\right)\Bigg).
\end{aligned} \tag{26}$$

Again we remark that the operator Θ^h and the long vectors $Q^{(k)}$, $F^{(k)}$, V^l, and V^h are only for analysis use. Operations on long vectors of size n will be very expensive and unnecessary, especially on lower levels where the compression dimension N^h(the size of A_{st}^h) is small. Noticing that all long vectors on the h-level are in span$\{\mathbf{\Psi}^h\}$ as

$$Q^{(k)} = \mathbf{\Psi}^h \widehat{Q}^{(k)}, \quad F^{(k)} = \mathbf{\Psi}^h \widehat{F}^{(k)}, \quad V_{m_l}^l = \mathbf{\Psi}^h \widehat{V}_{m_l}^l, \quad V_{m_h}^h = \mathbf{\Psi}^h \widehat{V}_{m_h}^h,$$

we thus only operate on their coefficients in the basis $\mathbf{\Psi}^h$. Correspondingly, whenever we need to consider orthogonality of long vectors, we replace it by the M^h-orthogonality of their coefficient vectors. One can check that all discussions above still apply. Also another advantage of using the coefficient vectors is that in the previous discussions, the good initial guess $\mu_i^{(k)}(M^h)^{-1}w_i^{(k)} = \mu_i^{(k)}(M^h)^{-1}(\mathbf{\Psi}^h)^T q_i^{(k)} = \mu_i^{(k)}\hat{q}^{(k)}$ is obtained explicitly.

Summarizing the analysis above, we propose the following Algorithm 5 as our refinement method. Since we want the eigenspace spanned by the first m_h eigenvectors of Θ^h to be computed accurately, the refinement stops when dist$(Q_{m_h}^{(k-1)}, Q_{m_h}^{(k)}) < \epsilon$ for some prescribed accuracy ϵ, where $Q_{m_h}^{(k)}$ denotes the first m_h columns of $Q^{(k)}$. Since $Q^{(k)}$ is orthogonal, one can check that

$$\begin{aligned}
\mathrm{dist}(Q_{m_h}^{(k-1)}, Q_{m_h}^{(k)}) &= \|Q_{m_h}^{(k)} - Q_{m_h}^{(k-1)}(Q_{m_h}^{(k-1)})^T Q_{m_h}^{(k)}\|_2 \\
&= \|\widehat{Q}_{m_h}^{(k)} - \widehat{Q}_{m_h}^{(k-1)}(\widehat{Q}_{m_h}^{(k-1)})^T M^h \widehat{Q}_{m_h}^{(k)}\|_{M^h} \\
&\leq \sqrt{\lambda_{\max}(M^h)}\|\widehat{Q}_{m_h}^{(k)} - \widehat{Q}_{m_h}^{(k-1)}(\widehat{Q}_{m_h}^{(k-1)})^T M^h \widehat{Q}_{m_h}^{(k)}\|_2 \\
&\leq \sqrt{1+\varepsilon_h\delta_h}\|\widehat{Q}_{m_h}^{(k)} - \widehat{Q}_{m_h}^{(k-1)}(\widehat{Q}_{m_h}^{(k-1)})^T M^h \widehat{Q}_{m_h}^{(k)}\|_F.
\end{aligned}$$

In practice, we use $\|\widehat{Q}_{m_h}^{(k)} - \widehat{Q}_{m_h}^{(k-1)}(\widehat{Q}_{m_h}^{(k-1)})^T M^h \widehat{Q}_{m_h}^{(k)}\|_F < \frac{\epsilon}{\sqrt{1+\varepsilon_h\delta_h}}$ as the stopping criterion since it is easy to check. We have used Lemma 4.7 to bound $\lambda_{\max}(M^h)$.

6. Overall algorithms.

Combining the refinement method and the extension method, we now propose our overall Algorithm 6 for computing partial eigenpairs of an SPD matrix A. It utilizes the *a priori* multiresolution decomposition of A to compute the first m_{tar} eigenpairs of A^{-1}, by passing approximate eigenpairs from lower levels to higher levels to finally reach a prescribed accuracy. In particular, this algorithm starts with the eigendecomposition of the lowest level (whose dimension is

Algorithm 5 Eigenpair refinement.

```
Input: $\widehat{V}^l_{m_l}$, $D^l_{m_l}$, prescribed accuracy $\epsilon$, target eigenvalue threshold $\mu_h$.
Output: $\widehat{V}^h_{m_h}$, $D^h_{m_h}$.
 1: Set $\widehat{Q}^{(0)} = V^l_{m_l}, \quad D^{(0)} = D^l_{m_l}, \quad k = 0;$
 2: for $i = 1 : m_l$ do
 3:     $g_i = pcg(B^l_{st}, (U^l)^T M^h \hat{q}^{(0)}_i, -, -, \epsilon); \quad (\widehat{Q} = [\hat{q}_1, \ldots, \hat{q}_{m_l}])$
 4: end for
 5: $\widehat{F}^{(0)} = \widehat{Q}^{(0)} D^{(0)} + U^l G; \quad (G = [g_1, \ldots, g_{m_l}])$
 6: repeat
 7:     $k \leftarrow k + 1;$
 8:     $\widehat{Q}^{(k)} R^{(k)} = \widehat{F}^{(k-1)};$ (QR factorization with respect to $M^h$ orthogonality, i.e., $(\widehat{Q}^{(k)})^T M^h \widehat{Q}^{(k)} = I$)
 9:     $W^{(k)} = M^h \widehat{Q}^{(k)};$
10:     for $i = 1 : m_l$ do
11:         $\hat{f}^{(k)}_i = pcg(A^h_{st}, w^{(k)}_i, M^h, \mu^{(k-1)}_i \hat{q}^{(k)}_i, \epsilon); \quad (\widehat{F} = [\hat{f}_1, \ldots, \hat{f}_{m_l}])$
12:     end for
13:     $S^{(k)} = (W^{(k)})^T \widehat{F}^{(k)};$
14:     $P^{(k)} D^{(k)} (P^{k)})^T = S^{(k)}$ (Schur decomposition, diagonals of $D^{(k)}$ in decreasing order);
15:     renew $m_h$ so that $\mu^{(k)}_{m_h} \geq \mu_h > \mu^{(k)}_{m_h+1};$
16:     $\widehat{Q}^{(k)} \leftarrow \widehat{Q}^{(k)} P^{(k)}, \quad \widehat{F}^{(k)} \leftarrow \widehat{F}^{(k)} P^{(k)};$
17: until $\|\widehat{Q}^{(k)}_{m_h} - \widehat{Q}^{(k-1)}_{m_h} (\widehat{Q}^{(k-1)}_{m_h})^T M^h \widehat{Q}^{(k)}_{m_h}\|_F < \epsilon.$
18: $\widehat{V}^h_{m_h} = \widehat{Q}^{(k)}_{m_h}, \quad D^h_{m_h} = D^{(k)}_{m_h}.$ ($D^{(k)}_{m_h}$ denotes the first $m_h$-size block of $D^{(k)}$)
```

small enough), refines and extends the approximate eigenpairs on each level, and stops at the highest level. The overall accuracy is achieved by the prescribed compression error of the highest level. It could be clearer using a flow chart (Figure 1) to illustrate the procedure of our method. If we see the eigenproblem of the original matrix A as a complicated model, our algorithm resolves the model complexity by hierarchically simplifying/coarsening the original model into an inductive sequence of approximate models.

Algorithm 6 Hierarchical eigenpair computation.

```
Input: $K$-level decomposition $\{\Theta^{(k)}\}_{k=1}^K$ of SPD matrix $A$, target number $m_{tar}$, searching step $d$, prescribed multi-level accuracies $\{\epsilon^{(k)}\}$, extension thresholds $\{\mu^{(k)}_{ex}\}_{k=1}^K$, refinement thresholds $\{\mu^{(k)}_{re}\}_{k=1}^K$.
Output: $V$, $D$.
 1: Find the eigen pairs $[\widehat{V}^{(K)}_{ex}, D^{(K)}_{ex}]$ of the eigen problem $(A^{(K)}_{st})^{-1} M^{(K)} x = \mu x;$
 2: for $k = K - 1 : 1$ do
 3:     $\widehat{V}^{(k+1)}_{ex} \leftarrow \Psi^{(k+1)} \widehat{V}^{(k+1)}_{ex}$
 4:     $[\widehat{V}^{(k)}_{ini}, D^{(k)}_{ini}] = \text{Eigen\_Refine}([\widehat{V}^{(k+1)}_{ex}, D^{(k+1)}_{ex}]; \epsilon^{(k)}, \mu^{(k)}_{re});$
 5:     $op = OP(\,\cdot\,; A^{(k)}, M^{(k)}, \epsilon^{(k)});$
 6:     $[\widehat{V}^{(k)}_{ex}, D^{(k)}_{ex}] = \text{Eigen\_Extend}([\widehat{V}^{(k)}_{ini}, D^{(k)}_{ini}]; op, \epsilon^{(k)}, \mu^{(k)}_{ex}, d, m_{tar});$
 7: end for
 8: $V = \Psi^{(1)} \widehat{V}^{(1)}_{ex} \quad D = D^{(1)}_{ex}.$
```

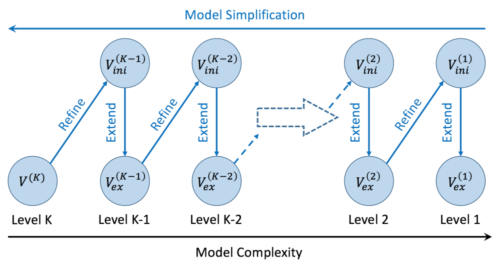


FIG. 1. *Flow chart illustrating the procedure of Algorithm* 6.

Recall that the output $\widehat{V}_{ex}^{(k)}$ of the extension process and the initializing process are the coefficients of $V_{ex}^{(k)}$ in the basis $\mathbf{\Psi}^{(k)}$. When passing these results from level k to level $k-1$, we need to recover the coefficients of $V_{ex}^{(k)}$ in the basis $\mathbf{\Psi}^{(k-1)}$. This can be done by simply reforming $\widehat{V}_{ex}^{(k)} \leftarrow \Psi^{(k)}\widehat{V}_{ex}^{(k)}$(line 3 in Algorithm 6), since $V_{ex}^{(k)} = \mathbf{\Psi}^{(k)}\widehat{V}_{ex}^{(k)} = \mathbf{\Psi}^{(k-1)}\Psi^{(k)}\widehat{V}_{ex}^{(k)}$.

In Algorithm 6, the parameters should be chosen carefully to ensure computational efficiency, by using the analysis in the previous sections. We shall discuss the choice of each parameter separately. To be consistent, we first clarify some notations. Let $\hat{m}_k, m_k$ be the numbers of output eigenpairs of the refinement process and the extension process, respectively, on level k. Ignoring numerical errors, let $(\mu_i^{(k)}, v_i^{(k)})$, $i = 1,\dots,N^{(k)}$, be the essential eigenpairs of the operator $\Theta^{(k)}$ as in section 4. Let $(\mu_i^{(k)}, v_i^{(k)})$, $i = 1,\dots,m_k$, denote the output eigenpairs on level k. Notice that $(\mu_i^{(k)}, v_i^{(k)})$, $i = 1,\dots,\hat{m}_k$, are the outputs of the refinement process, and $(\mu_i^{(k)}, v_i^{(k)})$, $i = \hat{m}_k+1,\dots,m_k$, are the outputs of the extension processs

Choice of multilevel accuracies $\{\boldsymbol{\epsilon^{(k)}}\}$. Notice that there is a compression error ε_k between level k and level $k-1$. That is to say, no matter how accurately we compute the eigenpairs of $\Theta^{(k)}$, they are approximations of eigenpairs of $\Theta^{(k-1)}$ subject to accuracy no better that ε_k. Therefore, on the one hand, the choice of the algorithm accuracy $\epsilon^{(k)}$ for the eigenpairs of $\Theta^{(k)}$ on each level should not compromise the compression error. On the other hand, the accuracy should not be overachieved due to the presence of the compression error. Therefore, we choose $\epsilon^{(k)} = 0.1\times\varepsilon_k$ in practice.

Choice of thresholds $\{(\boldsymbol{\mu_{re}^{(k)}}, \boldsymbol{\mu_{ex}^{(k)}})\}_{\boldsymbol{k=1}}^{\boldsymbol{K}}$. These thresholds provide control on the smallest eigenvalues of output eigenpairs of both the refinement process and the extension process in that

$$\mu_{\hat{m}_k}^{(k)} \geq \mu_{re}^{(k)} > \mu_{\hat{m}_k+1}^{(k)}, \quad \mu_{ex}^{(k)} \geq \mu_{m_k}^{(k)}, \quad k = 1,2,\dots,K.$$

Recall that the outputs of the refinement process are the inputs of the extension process, and the outputs of the extension process are the inputs of the refinement process on the higher level. By Theorem 4.8, to ensure the efficiency of the extension process, we need to uniformly control the restricted condition number

$$\kappa(A_\Psi^{(k)}, Z_{\hat{m}_k^+}^{(k)}) \leq \mu_{\hat{m}_k+1}^{(k)}\delta_k < \mu_{re}^{(k)}\delta_k.$$

Recall that in section 5 the convergence rate of the refinement process is given by $\frac{\mu_{h,m_l+1}}{\mu_{h,m_h}}$, where l corresponds to $k+1$ and h corresponds to k on each level k. Thus to ensure the efficiency of the refinement process we need to uniformly control the ratio

$$\frac{\mu^{(k)}_{m_{k+1}+1}}{\mu^{(k)}_{\hat{m}_k}} \le \frac{\mu^{(k)}_{m_{k+1}}}{\mu^{(k)}_{re}} \le \frac{\mu^{(k+1)}_{m_{k+1}} + \varepsilon_{k+1}}{\mu^{(k)}_{re}} \le \frac{\mu^{(k+1)}_{ex} + \varepsilon_{k+1}}{\mu^{(k)}_{re}},$$

where ε_{k+1} is the compression error between level $k+1$ and level k, and we have used Lemma 3.3. Thus, more precisely, we need to choose thresholds $\{(\mu^{(k)}_{re}, \mu^{(k)}_{ex})\}_{k=1}^K$ so that there exist uniform constants $\kappa > 0, \gamma \in (0,1)$ so that

$$\text{(27)} \qquad \text{(i) } \mu^{(k)}_{re}\delta_k \le \kappa, \qquad \text{(ii) } \frac{\mu^{(k+1)}_{ex} + \varepsilon_{k+1}}{\mu^{(k)}_{re}} \le \gamma.$$

Due to the existence of ε_k, condition (ii) implies that there is no need to choose $\mu^{(k)}_{ex}$ much smaller than ε_k, which suffers from overcomputing but barely improves the efficiency of the refinement process. So one convenient way is to choose

$$\text{(28)} \qquad \mu^{(k)}_{re} = \alpha\varepsilon_{k+1}, \qquad \mu^{(k)}_{ex} = \beta\varepsilon_k$$

for some uniform constants $\alpha, \beta > 0$ such that $\alpha > 1+\beta$. Recall that when constructing the multiresolution decomposition, we impose conditions $\varepsilon_k\delta_k \le c$ and $\varepsilon_k = \eta\varepsilon_{k+1}$ for some uniform constants $c > 0$ and $\eta \in (0,1)$. Thus we have

$$\mu^{(k)}_{re}\delta_k = \frac{\alpha}{\eta}\varepsilon_k\delta_k \le \frac{\alpha c}{\eta} = \kappa, \qquad \frac{\mu^{(k+1)}_{ex} + \varepsilon_{k+1}}{\mu^{(k)}_{re}} = \frac{1+\beta}{\alpha} = \gamma < 1.$$

Choice of searching step *d*. In the first part of the extension algorithm, we explore the number m_k so that $\mu^{(k)}_{m_k} \le \mu^{(k)}_{ex}$, and we do this by setting an exploring step size d and examining the last few eigenvalues every d steps of the Lanczos iteration. The step size d should neither be too large to avoid overcomputing, nor too small to ensure efficiency. In practice, we choose $d = \min\{\lfloor \frac{\dim \mathbf{\Psi}^{(k)}}{10} \rfloor, \lfloor \frac{m_{tar}}{10} \rfloor\}$.

Complexity. Now we summarize the complexity of Algorithm 6 for computing the first m_{tar} largest eigenpairs of A^{-1} for an SPD matrix $A \in \mathbb{R}^{n\times n}$ subject to an error ε. Suppose we are provided a K-level MMD of A with $\varepsilon_k\delta_k \le c$, $\varepsilon_k = \eta\varepsilon_{k+1}$, and $\varepsilon_1 = \varepsilon$. In what follows, we will uniformly estimate $nnz(A^{(k)}_{st}) \le nnz(A)$, $\epsilon^{(k)} \ge \epsilon^{(1)} = 0.1\varepsilon_1$, and $m_k \le m_{tar}$.

We first consider the complexity of all refinement processes. Notice that by our choice $\frac{\varepsilon_{k+1}}{\epsilon^{(k)}} = \frac{\varepsilon_{k+1}}{0.1\epsilon^{(k)}} = \frac{1}{0.1\eta}$, the factor $\log(\frac{\varepsilon_l}{\epsilon})$ in (26), which is now $\log(\frac{\varepsilon_{k+1}}{\epsilon^{(k)}})$, can be estimated as $O(\log(\frac{1}{\eta}))$. Since we can will make sure $\frac{\mu^{(k)}_{m_{k+1}+1}}{\mu^{(k)}_{\hat{m}_k}} \le \gamma$ for some constant $\gamma < 1$, the factor $\log(\frac{\mu_{h,m_h}}{\mu_{h,m_l+1}})$ in (26), which is now $\log(\frac{\mu^{(k)}_{\hat{m}_k}}{\mu^{(k)}_{m_{k+1}+1}})$, can be seen as a constant. Also using estimates $\mu_{h,m_h}\delta_h \le \frac{\alpha c}{\eta} = O(\frac{c}{\eta})$, $\varepsilon_l\delta_h \le \frac{c}{\eta}$, $\varepsilon_h\delta_h \le c$, and $\log\frac{1}{\epsilon} = O(\log\frac{1}{\varepsilon})$, we modify (26) to obtain the complexity of all K-level refinement processes

$$\text{(29)} \qquad O\left(m_{tar}\cdot nnz(A)\cdot\frac{c^2}{\eta}\log\left(\frac{1}{\eta}\right)\cdot\left(\log\frac{1}{\varepsilon}\right)^2\cdot K\right).$$

Next we consider the complexity of all extension process. As we have discussed in section 4, the major cost of the extension process comes from the operation of adding a new vector (the adding operation) to the Lanzcos vectors (line 7 of Algorithm 3 that happens in line 3 of Algorithm 4). Using estimates $\mu_m \delta(\mathcal{P}) \leq \frac{\alpha c}{\eta} = O(\frac{c}{\eta})$, $\varepsilon(\mathcal{P})\delta(\mathcal{P}) \leq c$, $\log \frac{1}{\epsilon} = O(\log \frac{1}{\varepsilon})$, we modify (23) to obtain the cost of every single call of the adding operation as

$$O\left(\frac{c^2}{\eta} \cdot nnz(A) \cdot \left(\log \frac{1}{\varepsilon}\right)^2\right).$$

On every level, the indexes contributing to adding operations go from $\hat{m}_k + 1$ to m_k. Due to the refinement process, we have $\hat{m}_k \leq m_{k+1}$, and so every single index from 1 to m_{tar} may contribute more than one adding operation. But if we reasonably assume that $\mu_{ex}^{(k+1)} > \mu_{re}^{(k-1)}$, namely $\beta > \alpha\eta$ under parameter choice (28), we will have $m^{(k+1)} < \hat{m}^{(k-1)}$, and so every index from 1 to m_{tar} will contribute no more than two adding operations. Therefore the total cost of all extension processes can be estimated as

$$O\left(m_{tar} \cdot \frac{c^2}{\eta} \cdot nnz(A) \cdot \left(\log \frac{1}{\varepsilon}\right)^2\right). \tag{30}$$

We remark that the cost of implicit restarting process is only a constant multiple of (30). Combining (29) and (30), we obtain the total complexity of our method

$$O\left(m_{tar} \cdot nnz(A) \cdot \frac{c^2}{\eta} \log\left(\frac{1}{\eta}\right) \cdot \left(\log \frac{1}{\varepsilon}\right)^2 \cdot K\right). \tag{31}$$

To further simplify (31), we need to use estimates for the MMD given in the previous work [10]. In particular, to preserve sparsity $nnz(A_{st}^{(k)}) \leq nnz(A)$, we need to choose the scale ratio $\eta^{-1} = (\log \frac{1}{\varepsilon} + \log n)^p$ for some constant p. We remark that for graph Laplacian, $p = 1$, the resulting level number is $K = O(\frac{\log n}{\log(\log \frac{1}{\varepsilon} + \log n)})$. The condition bound c can be imposed to be uniformly constant by the algorithm given in [10]. Then the overall complexity of Algorithm 6 can be estimated as

$$\begin{aligned} &O\left(m_{tar} \cdot nnz(A) \cdot \left(\log \frac{1}{\varepsilon} + \log n\right)^p \cdot \left(\log \frac{1}{\varepsilon}\right)^2 \cdot \log n\right) \\ &= O\left(m_{tar} \cdot nnz(A) \cdot \left(\log \frac{1}{\varepsilon} + \log n\right)^{p+3}\right). \end{aligned} \tag{32}$$

7. Numerical examples.

In this section we present several numerical examples for the eigensolver. We will use Algorithm 6 to compute a relative large number of eigenpairs of large matrices subject to prescribed accuracies.

7.1. Dataset description.

The datasets we use are drawn from different physical contexts. They are generated as three-dimensional (3D) point clouds and transformed into graphs by adding edges in the K-nearest neighbors setting.

- The first dataset is the well-known "Stanford Bunny" from the Stanford 3D Scanning Repository.[1] A reconstructed bunny has 35947 vertices that can be embedded into a surface in $\mathbb{R}^3$ with 5 holes in the bottom.

[1]http://graphics.stanford.edu/data/3Dscanrep/

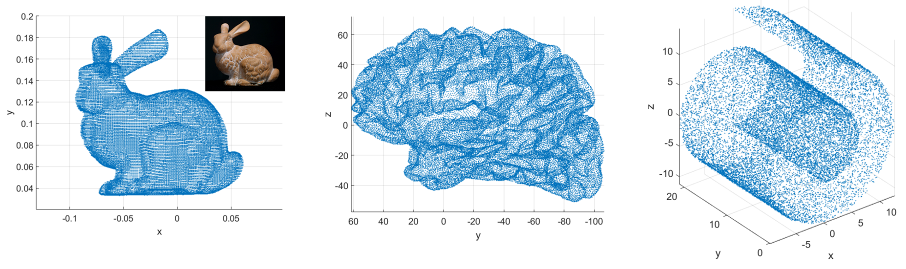


FIG. 2. *Datasets. From left to right:* (1) *bunny (point cloud and sculpture);* (2) *brain;* (3) *Swissroll.*

- The second dataset is the magnetic resonance imaging (MRI) data of the brain from the Open Access Series of Imaging Sciences (OASIS).[2] They use FreeSurfer to reconstruct the surface from an MRI scan and obtain a point cloud with 48463 points.
- The third dataset is a "SwissRoll" model, which is popular in manifold learning. Vertices are generated by

$$(x_i, y_i, z_i) = (t_i cos(t_i), y_i, t_i sin(t_i)) + \boldsymbol{\eta}_i,\ i = 1, 2, \dots, n, \tag{33}$$

  where $t_i \overset{\text{i.i.d}}{\sim} \mathcal{U}[1.5\pi, 4.5\pi]$, $y_i \overset{\text{i.i.d}}{\sim} \mathcal{U}[0, 20]$, and $\eta_i \overset{\text{i.i.d}}{\sim} \mathcal{N}(\mathbf{0}, 0.05I_3)$ (where i.i.d. is independent and identically distributed). It can be viewed as a spiral of one and a half rounds plus random noise. In our examples the roll has $n = 20000$ points.

With point clouds at hand, we apply the k-nearest neighbors to construct graphs with $k_{bunny} = 20$, $k_{brain} = 20$, and $k_{swissroll} = 10$. Each existing edge e_{ij} is weighted as $e^{-r_{i,j}^2/\sigma}$, where $r_{i,j}$ is the Euclidean distance between vertices v_i and v_j, and σ is a parameter. We have $\sigma_{bunny} = 10^{-6}$, $\sigma_{brain} = 10^{-4}$, and $\sigma_{swiss} = 0.1$. Figure 2 shows the point clouds of datasets.

From the graphs given above, we construct their related graph Laplacians L in the general setting:

$$L_{ij} = \begin{cases} \sum_{k\sim i} w_{ik}, & i = j, \\ -w_{ij}, & i \neq j. \end{cases}$$

Further, without loss of generality, we rescale all graph Laplacians and add uniform self-loops of weight 1 to them, so that each of them satisfies (i)$\lambda_1 = 1$, (ii) $\lambda_2 = O(1)$. Under this construction, we obtain three graph Laplacian matrices $L_{bunny}, L_{brain}, L_{swissroll}$. L_{bunny} has size $n = 35947$, sparsity $nnz = 714647$, and condition number $\kappa(L_{bunny}) = 1.86 \times 10^4$; L_{brain} has size $n = 48463$, sparsity $nnz = 1038065$, and condition number $\kappa(L_{bunny}) = 1.14 \times 10^5$; $L_{swissroll}$ has size $n = 20000$, sparsity $nnz = 248010$, and condition number $\kappa(L_{bunny}) = 1.15 \times 10^6$.

7.2. Numerical MMD. Before computing eigenpairs of graph Laplacians from our datasets using Algorithm 6, we need to apply Algorithm 6, proposed in [10], to obtain the MMD, which is the only precomputation step in our proposed algorithm.

[2]http://www.oasis-brains.org/

TABLE 1

Computation time (in seconds) for the 4-level Brain, 3-level SwissRoll, and the 4-level SwissRoll examples using the proposed hierarchical multilevel eigensolver; the IRLM with a CG solver and the IRLM with incomplete Cholesky preconditioned CG solver.

# Eigenpairs	Methods		4-level Brain	4-level SwissRoll	3-level SwissRoll
	Decomposition*		34.589	8.124	9.430
300	Proposed**	Level-4	0.010	0.011	-
		Level-3	0.841	0.560	0.083
		Level-2	29.122	40.796	18.729
		Level-1	61.286	18.846	22.440
	Total (* + **)		125.848	68.337	50.682
	IRLM-CG		174.028	81.005	
	IRLM-ICCG		525.73	289.385	
200	Proposed**	Level-4	0.010	0.011	-
		Level-3	0.826	0.526	0.083
		Level-2	25.560	28.094	11.517
		Level-1	54.951	12.107	18.378
	Total (* + **)		115.936	48.862	39.408
	IRLM-CG		124.871	61.479	
	IRLM-ICCG		417.632	196.217	
100	Proposed**	Level-4	0.010	0.011	-
		Level-3	0.831	0.531	0.083
		Level-2	25.056	22.062	9.883
		Level-1	31.882	8.066	12.029
	Total (* + **)		92.368	38.794	31.425
	IRLM-CG		115.676	48.713	
	IRLM-ICCG		324.648	90.175	

TABLE 2

Matrix decomposition time (in seconds) for different examples.

Data	4-level Brain	4-level SwissRoll	3-level SwissRoll
Time	34.589	8.124	9.430

For each graph Laplacian, we the decomposition with a prescribed condition bound c and a series of multilevel resolutions (compression errors) $\{\varepsilon_k\}_{k=1}^{K}$. Recall that the total complexity for [10, Algorithm 6] is $O(nnz(A) \cdot \log n \cdot (\log \frac{1}{\epsilon} + \log n)^{3d+p})$, where d denotes the intrinsic geometric dimension of the graph. By comparison with the complexity estimate in (32), when $m_{tar} \gg (\log \frac{1}{\epsilon} + \log n)^{3d-2}$, the precomputation time for constructing MMD only takes up a relative small portion of overall time. As illustrated in Table 1, even with the precomputation time taken into account, our proposed algorithm is still faster than other well-established methods.

Note that we perform two decompositions with different multiresolutions for the SwissRoll data. The decomposition time for each example is reported in Table 2. The detailed information of all decompositions which will be used for eigenpair computation are reported in Appendix B.1.

7.3. The coarse level eigenpair approximation. We first use the decompositions given above to compute the first few eigenpairs of graph Laplacians with relatively low accuracies. Numerical results reveal that even on the coarse levels, the compressed (low-dimensional) operators show good spectral approximation properties with regard to the smallest eigenvalues of L (or the largest eigenvalues of L^{-1}). A quantitative error report is presented in Appendix B.2. We also qualitatively test the accuracy of the approximate eigenvectors of the compressed operators, by comparing their behaviors in image segmentation to those of the true eigenvectors of the original Laplacian operators.

7.4. The multilevel eigenpair computation. To test the efficacy of the hierarchical structure in our approach, we use our main Algorithm 6 to compute a relatively large number of eigenpairs of Laplacian matrices subject to the prescribed accuracy. In particular, we compute the first 500 eigenpairs of the graph Laplacian from both the Brain data and the SwissRoll data, subject to some prescribed accuracy ϵ. The numerical results are reported in Appendix B.3. Our results show the efficiency of our algorithm. We can see that both the iteration number of each refinement process and the iteration number of each CG/PCG operation in every level are well-bounded uniformly, which is due to the proper strategy of parameter selection discussed in section 6.

8. Comparison with the IRLM.

Owing to the observation in [16] that the IRLM is still one of the best performing and well-known algorithms for finding a large portion of smallest eigenpairs, in this section, we compare the computation complexity of our proposed algorithm with the IRLM.

To quantitatively compare the two methods, we record the computation time and the number of CG iterations as the benchmarks. The reasons for doing this are as follows:

- In a large-scale setting, a direct method for solving the sparse matrix A^{-1} is general, but not practical since large memory storage is required. Instead, iterative methods, especially the CG method (as A is SPD in our case) is employed.
- In both the IRLM and our proposed algorithm, the dominating complexity comes from the operation of computing $A^{-1}b$ for some b.

Remark 8.1. For small-scale problems, a direct solver (such as sparse Cholesky factorization) for A^{-1} is preferred in the IRLM. In this way, only one factorization step for A is required prior to the IRLM. Moreover, solving for A^{-1} in each iteration is replaced by solving two lower triangular matrix systems. This will bring a significant speedup for the IRLM. However, recall that we are aiming at understanding the asymptotic behavior and performance of these methods. Therefore, the IRLM discussed in this section employs the iterative solver instead of a direct solver.

To be consistent, all the experiments are performed on a single machine equipped with Intel(R) Core(TM) i5-4460 CPU with 3.2 GHz and 8 GB DDR3 1600 MHz RAM. Both the proposed algorithm and the IRLM are implemented using C++ with the Eigen Library for fairness. In particular, the built-in (P)CG solvers are used in the IRLM implementation, instead of implementing on our own.

Table 1 shows the overall computation time for computing the leftmost (i) 300, (ii) 200, and (iii) 100 eigenpairs using (i) our proposed algorithm; (ii) the IRLM with incomplete Cholesky PCG (IRLM-ICCG); and (iii) the IRLM with a classical CG method (IRLM-CG). In this numerical example, the error tolerance of the eigenvalues in all three cases are set to 10^{-5}. Since the error for IRLM cannot be obtained a priori, we fine-tune the relative error tolerance for the (P)CG solver such that eigenvalues error are of order $O(10^{-6})$. For the proposed algorithm, the time required for levelwise eigenpair computation is recorded. In the bottom level (level-4 or level-3 in these cases), we have used the built-in eigensolver function in the Eigen Library to obtain the full eigenpairs (corresponding to line 1 in Algorithm 6). As the problem size is small, the time complexity is insignificant for all three examples.

The total runtime of our proposed algorithm in each example is computed by summing up all levels' computation time, plus the MMD time (which is the second row in Table 1). For all these examples, our proposed algorithm outperforms the IRLM. Although both the size of the matrices and their corresponding condition numbers are not extremely large, the numerical experiments already show a observable improvement. From the theoretical analysis discussed in the previous sections, this improvement will even be magnified if the SPD matrices are of larger scales and more ill-conditioned. Indeed, we assert that our proposed algorithm cannot be fully utilized in these illustrations. Therefore, one of the main future works is to perform detailed numerical experiments in these cases. For instance, by considering the 3-level and 4-level SwissRoll examples, we observe that a 3-level decomposition is indeed sufficient for the SwissRoll graph Laplacian, where we recall the corresponding condition number is $\|A\|_2 = 1.15 \times 10^6$. Therefore, using a 3-level decomposition, the overall runtime reduction goes up to approximately 37% when 300 eigenpairs are required.

Notice that the time required for the IRLM-ICCG is notably much more than that of the IRLM-CG, which contradicts our usual experience regarding preconditioning. In fact, such a phenomenon can be explained as follows: In the early stage of the IRLM, preconditioning with incomplete Cholesky factorization helps in reducing the iteration number of the CG. However, when the eigensubspace is gradually projected away throughout the IRLM process, the spectrum of the remaining subspace reduces and therefore CG iteration numbers also drop significantly. On the contrary, preconditioning with incomplete Cholesky ignores such an update in the spectrum and, therefore, the CG iteration number is uniform throughout the whole Lanczos iteration. Hence, the classical CG method is preferred if a large number of leftmost eigenpairs are required. Figure 3(a) shows the CG iteration numbers in the IRLM-ICCG, IRLM-CG, and, respectively, our proposed hierarchical eigensolver versus the Lanczos iteration. More precisely, if we call V_k in (20) to be the *Lanczos vector*, the horizontal axis in the figure then corresponds to the first time we generate the ith column of the Lanczos vector. For IRLM methods, it is equivalent to the extension procedure for the ith column of the Lanczos vector, which corresponds to lines 6–8 in Algorithm 1. In particular, the CG iteration number recorded in this figure corresponds to the operation *op* in Line 8 of Algorithm 1. For our proposed algorithm, there are three separate sections, each section's CG iteration numbers correspond to the formation of Lanczos vectors in the 3rd-, 2nd-, and 1st-levels, respectively. Since we may also update some of these Lanczos vectors during the refinement process, some overlaps in the recording of CG iteration numbers corresponding to those Lanczos vectors are observed. With the spectrum-preserving hierarchical preconditioner M introduced in our algorithm, the CG iteration number for computing $A^{-1}b$ for some b is tremendously reduced. In contrast, the CG iteration number for IRLM-CG is the largest at the beginning but decreases exponentially and asymptotically converges to our proposed result. For IRLM-ICCG, the incomplete Cholesky factorization does not capture the spectrum update and therefore the iteration numbers are uniform throughout the computation. This observation is also consistent with the time complexity as shown in Table 1. Figure 3(b) shows the corresponding normalized plot, where the iteration number is normalized by $\log(\frac{1}{\epsilon})$.

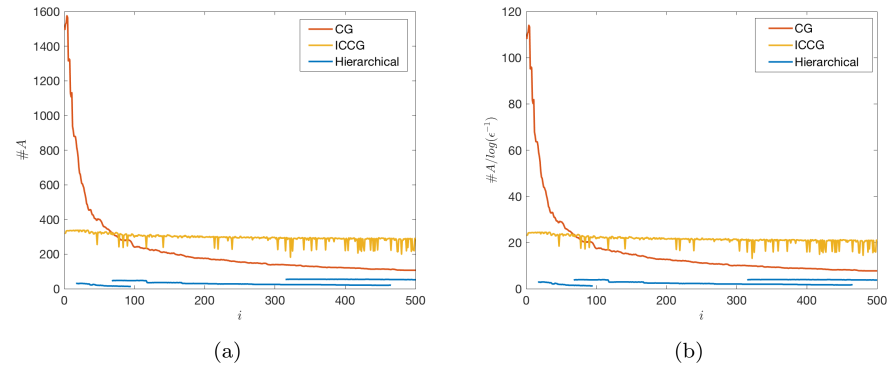


Fig. 3. (a) *The PCG iteration number in the* 4*-level SwissRoll example. The IRLM-ICCG methods exhibit a uniform iteration number, while the IRLM-ID has an exponentially decaying iteration number. For our proposed algorithm, since the spectrum-preserving hierarchical preconditioner M is employed, the CG iteration number is minimum. This is also consistent with the time complexity shown in Table* 1. (b) *The corresponding normalized plot, where the iteration number is normalized by* $\log(\epsilon)$.

Similar results can also be plotted for the 4-level Brain and the 3-level SwissRoll examples. We therefore skip those plots to avoid repetition.

9. Conclusion and future works. In this work, we proposed a spectrum preserving preconditioned hierarchical eigensolver to compute a large number of leftmost eigenpairs of a sparse SPD matrix. This eigensolver exploits the well-conditioned property of the decomposition components obtained through the MMD, the nice spectral property Lanczos procedure, and also the preconditioning characteristics of the CG method. In particular, we proposed an extension-refinement iterative scheme, in which eigenpairs are hierarchically extended and refined from the ones obtained from the previous level up to the desired amount. Moreover, we introduced a specially designed spectrum-preserving preconditioner for the PCG method to compute $x = A^{-1}b$ for some b during the iterations. We also presented a theoretical analysis on the runtime complexity and the asymptotic behavior of our proposed algorithm is reported. Finally, we conducted quantitative numerical experiments and a comparison with the IRLM, which demonstrated the efficiency and effectiveness of our proposed algorithm.

We would like to remark that the proposed algorithm and its implementation are still in the early stage as the main purpose of this work is to explore the possibility of integrating the multiresolution operator compression framework with the Krylov-type iterative eigensolver. Therefore, one of the future topics is to conduct comprehensive numerical studies of our algorithm with various large-scale, real data such as graph Laplacians of real network data, or stiffness matrices stemmed from the discretization of high-contrasted elliptic PDEs. These studies will help numerically confirm the asymptotic behavior of the relative condition numbers of M and A_{st}, especially when we need to compute a large number of leftmost eigenpairs from large-scale operators. Another possible research direction is to investigate the parallelization of this algorithm. This is important when we solve a large-scale eigenvalue problem.

Appendix A. In this section, we compare our method for compressed eigenproblems and the method proposed by Ozoliņš et al. [22]. We start with the straightforward compression directly using the eigenvectors corresponding to the smallest eigenvalues, which can be obtained by solving the following optimization problem:

$$\begin{aligned} \Psi = \arg\min_{\widehat{\Psi}} & \ \textstyle\sum_{i=1}^N \hat{\psi}_i^T A \hat{\psi}_i \\ \text{s.t.} & \ \ \hat{\psi}_i^T \hat{\psi}_j = \delta_{ij},\ i,j = 1,2,\dots,N. \end{aligned} \tag{34}$$

The compression using eigenvectors well known as the PCA method is optimal in the 2-norm sense for fixed compressed dimension N. However, computing a large number of eigenvectors is a hard problem itself, not to mention that we actually intend to approximate eigenpairs using the compressed operator. Also the spatially extended profiles of exact eigenvectors make them less favorable in many fields of research. Then as modification, Ozoliš et al. [22] added an L_1 regularization term to impose the desired locality on Ψ. They modified the optimization problem (34) as

$$\begin{aligned} \Psi = \arg\min_{\widehat{\Psi}} & \ \textstyle\sum_{i=1}^N \left(\hat{\psi}_i^T A \hat{\psi}_i + \frac{1}{\mu}\|\hat{\psi}_i\|_1 \right) \\ \text{s.t.} & \ \ \hat{\psi}_i^T \hat{\psi}_j = \delta_{ij},\ i,j = 1,2,\dots,N. \end{aligned} \tag{35}$$

The L_1 regularization, as widely used in many optimization problems for sparsity pursuit, effectively ensures each output ψ_i to have spatially compact support, at the cost of compromising the approximation accuracy compared to PCA. The factor μ controls the locality of Ψ. A smaller μ gives more localized profiles of Ψ, which, however, result in a larger compression error for a fixed N. The loss of approximation accuracy can be compensated for by increasing, yet not significantly, the basis number N. An algorithm based on the split Bregman iteration was also proposed in [22] to effectively solve the problem (35). In summary, their work provides an effective method to find a bunch of localized basis functions that can approximately span the eigenspace of smallest eigenvalues of A.

Although our approach to operator compression is originally developed from a different perspective based on the finite element method (FEM), it can be reformulated as an optimization problem similar to (34). In fact, to obtain the basis Ψ used in our method, we can simply replace the nonlinear constraints $\psi_i^T\psi_j = \delta_{ij},\ i,j = 1,2,\dots,N$, by linear constraints $\psi_i^T\phi_j = \delta_{ij},\ i,j = 1,2,\dots,N$, to get

$$\begin{aligned} \Psi = \arg\min_{\widehat{\Psi}} & \ \textstyle\sum_{i=1}^N \hat{\psi}_i^T A \hat{\psi}_i \\ \text{s.t.} & \ \ \hat{\psi}_i^T \phi_j = \delta_{ij},\ i,j = 1,2,\dots,N. \end{aligned} \tag{36}$$

Here $\Phi = [\phi_1, \phi_2, \dots, \phi_N]$ is a dual basis that we construct ahead of Ψ to provide an a priori compression error estimate as stated in Theorem 2.1. As the constraints become linear, problem (36) can be solved explicitly by $\Psi = A^{-1}\Phi(\Phi^T A^{-1}\Phi)^{-1}$ as mentioned in (7). Instead of imposing locality by adding L_1 regularization as in (35), we obtain the exponential decaying feature of Ψ by constructing each dual basis function ϕ_i locally. That is the locality of Φ and the strong correlation $\Psi^T\Phi = I$ automatically give us the locality of Ψ under the energy minimizing property. The optimization form (36) was derived by Owhadi in [20] where Ψ was used as the FEM basis to solve second order elliptic equations with rough coefficients. This methodology was then generalized to problems on higher order elliptic equations [11], general Banach spaces [21], and general sparse SPD matrices [10]. In all previous works the nice spectral property of Ψ has been observed and in particular the eigenspace corresponding to the smallest M eigenvalues of A can be well approximately spanned by Ψ of a relatively larger dimension $N = O(M)$.

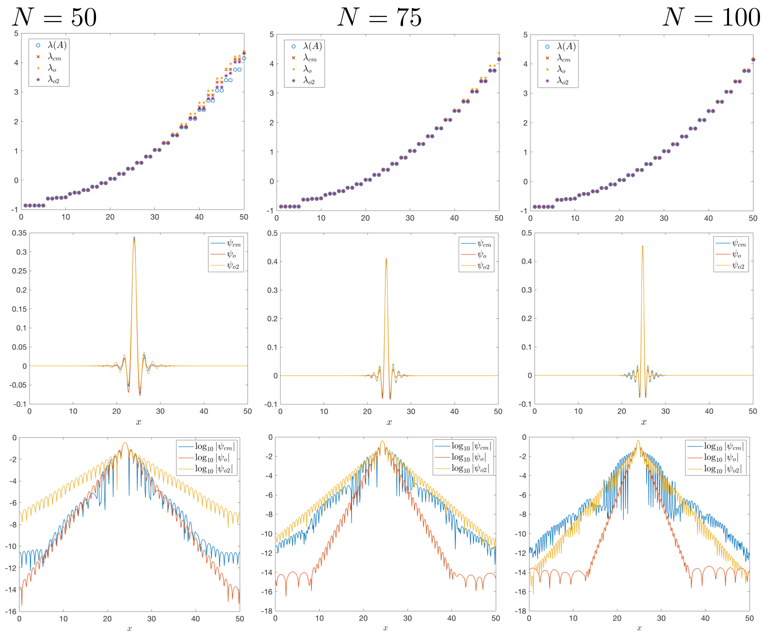


FIG. 4. *Results of problems* (35) *and* (36) *for* $N = 50$ *(first column),* $N = 75$ *(second column), and* $N = 100$ *(third column). First row: the first* 50 *eigenvalues of* A *and those of the compressed problems. Second row: examples of local basis functions. Third row: examples of local basis functions in log scale.*

To further compare the problems (35) and (36), we test both of them on the one-dimensional Kronig–Penney model studied in [22] with rectangular potential wells replaced by inverted Gaussian potentials. In this example, the matrix A comes from discretization of the PDE operator $-\frac{1}{2}\Delta + V(x)$ defined on the domain Ω with periodic boundary condition. In particular, $\Omega = [0, 50]$ and $V(x) = -V_0 \sum_{j=1}^{N_{el}} \exp(-\frac{(x-x_j)^2}{2\delta^2})$. As in [22], we discretize Ω with 512 equally spaced nodes, and we choose $N_{el} = 5$, $V_0 = 1$, $\delta = 3$, and $x_j = 10j - 5$ (instead of $x_j = 10j$ in [22], which essentially changes nothing).

For problem (36), we divide Ω into N equal-length intervals $\{\Omega_i\}_{i=1}^N$, and choose the dual basis $\Phi = [\phi_1, \phi_2, \ldots, \phi_N]$ such that ϕ_i is the discretization of the indicator function $\mathbf{1}(\Omega_i)(\mathbf{1}(\Omega_i)(x) = 1$ for $x \in \Omega_i$, otherwise, $\mathbf{1}(\Omega_i)(x) = 0$. We use Ψ_o to denote the exact result of problem (36), namely, $\Psi_o = A^{-1}\Phi(\Phi^T A^{-1}\Phi)^{-1}$. Since Ψ_o is not orthogonal, we should compute the eigenvalues from the general eigenvalue problem $\Psi_o^T A \Psi_o v = \lambda \Psi_o^T \Psi_o v$ (Lemma 3.2) as approximations of the eigenvalues of A. We use λ_o to denote these approximate eigenvalues.

For problem (35), we use Algorithm 1 and exactly the same parameters provided in [22], which means we are simply reproducing their results, except that we use a finer discretization (512 rather than 128) and we shift the potential $V(x)$. We have used normalized Φ as the initial guess for Algorithm 1 in [22], and choose $\mu = 10$. We use Ψ_{cm} to denote the result of problem (35). We use λ_{cm} to denote the eigenvalues of $\Psi_{cm}^T A \Psi_{cm}$.

We compare the approximate eigenvalues to the first 50 eigenvalues of A. The first row of Figure 4 shows that both methods give very good approximations of $\lambda(A)$. And when N increases, the approximations become better. But relatively, the results λ_{cm}

from Figure 4 are closer to the ground truth than our results λ_o from (36). To improve our results, we simply solve problem (36) again, but this time using the previous result Ψ_o as the dual basis. That is we compute $\Psi_{o2} = A^{-1}\Psi_o(\Psi_o^T A^{-1}\Psi_o)$, and compute eigenvalues λ_{o2} from the general eigenvalue problem $\Psi_{o2}^T A\Psi_{o2}v = \lambda\Psi_{o2}^T\Psi_{o2}v$. We can see that the approximate eigenvalues λ_{o2} are even closer to the ground truth. An interpretation of this improvement is that if we see $\Psi_o = A^{-1}\Phi(\Phi^T A^{-1}\Phi)^{-1}$ as a transformation from Φ to Ψ_o, then the part $A^{-1}\Phi$ is equivalent to applying the inverse power method to make Ψ_o more aligned to the eigenspace of the smallest eigenvalues, while the part $(\Phi^T A^{-1}\Phi)^{-1}$ is to force $\Psi_o^T\Phi = I$ so Ψ_o inherits some weakened locality from Φ. So if we apply this transformation to Ψ_o again to obtain Ψ_{o2}, Ψ_{o2} will approximate the eigenspace of the smallest eigenvalues better, but with more loss of locality.

In the second and third rows of Figure 4, we show some examples of the local basis functions ψ_{cm}, ψ_o, and ψ_{o2} (all are normalized to have unit l_2 norm). Interestingly, these basis functions are not just localized as expected, but indeed they have very similar profiles. One can see that for $N = 75$, the basis functions ψ_{cm} and ψ_o are almost identical. So it seems that in spite of how we impose locality (either the L_1 minimization approach, or the construction of the dual basis Φ), the local behaviors of the basis functions are determined by the operator A itself. We believe that this "coincidence" is governed by some intrinsic property of A, which may be worth further exploration and study. If we can understand a higher level, unified mechanism that results in the locality of the basis, we may be able to extend these methods to a more general class of operators. We also observed that as N goes large, ψ_o and ψ_{o2} become more and more localized since the support of the dual basis functions are smaller and smaller. However the locality of ψ_{cm} doesn't change much as N increases, since we use the same penalty parameter $\mu = 10$ for (35) in this experiment.

We would like to remark that, though these two problems result in local basis functions with similar profiles, problem (35) requires us to use the split Bregman iteration to obtain the N basis functions simultaneously. In our problem (36), since the constraints are linear and separable, the basis functions can be obtained separately and directly without iteration. Furthermore, thanks to the exponential decay of the basis functions, each subproblem for obtaining one basis function can be restricted to a local domain without significant loss of accuracy, and the resulting local problem can be solved very efficiently. For definitions and detailed properties of these local problems for obtaining a localized basis; please refer to [10, section 3]. Therefore the algorithm for solving problem (36) can be highly localized and embarrassingly parallel.

Appendix B.

B.1. Numerical details for subsection 7.2.

Tables 3 and 4 give the detailed information of all decompositions we will use for eigenpair computation. In Table 3, K is the number of levels, ε_1 is the finest (prescribed) accuracy, η is the ratio $\varepsilon_k/\varepsilon_{k+1}$, and c is the condition bound such that $\varepsilon_k\delta_k \le c$. By Lemma 4.7, the condition number of $M^{(k)}$ is bounded as $\kappa(M^{(k)}) \le 1 + \varepsilon_k\delta_k \approx c$, and by Corollary 2.5, the condition number of $B^{(k)}$ is bounded as $\kappa(B^{(k)}) \le \varepsilon_k\delta_{k-1} \le c/\eta$. We can see in Table 4 that these bounds are well satisfied. Recall that the bounded condition number of $M^{(k)}$ is essential for the efficiency of Algorithm 4, and the bounded condition number of $B^{(k)}$ is essential for the efficiency of Algorithm 5.

Table 4 also shows the detailed information for all four decompositions. The 2-norm of $A^{(k)}$, namely, $\lambda_{\max}(A^{(k)})$, decreases as k increases, and is well-bounded as

Table 3
Decomposition information.

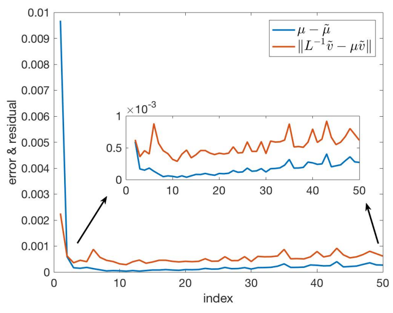
Data	K	ε_1	η	c	Bound on $\kappa(M^{(k)})$	Bound on $\kappa(B^{(k)})$
Bunny	2	10^{-3}	0.1	20	20	200
Brain	4	10^{-4}	0.2	20	20	100
SwissRoll	3	10^{-5}	0.1	20	20	200
SwissRoll	4	10^{-5}	0.2	20	20	100

Table 4
Decomposition information of (i) *Bunny (2-level)* (ii) *Brain (4-level), and* (iii) *SwissRoll (*3*, 4-level) data.* $m \triangleq nnz(A^{(0)})$.

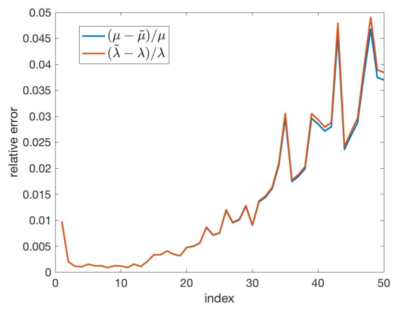
Level k	ε_k	Size of $A^{(k)}$	$nnz(A^{(k)})$	$\|A^{(k)}\|_2$	$nnz(M^{(k)})$	$\kappa(M^{(k)})$	$\kappa(B^{(k)})$
The 2-level decomposition of Bunny data.							
0	-	35947	$714647 = m$	1.86×10^4	-	-	-
1	10^{-3}	2641	$613571 \approx 0.86m$	1.05×10^4	$203445 \approx 0.28m$	1.45	5.58
2	10^{-2}	198	$27774 \approx 0.04m$	1.37×10^3	$10808 \approx 0.02m$	2.05	45.03
The 4-level decomposition of Brain data.							
0	-	48463	$1038065 = m$	1.14×10^5	-	-	-
1	10^{-4}	11622	$2546246 \approx 2.45m$	7.82×10^4	$725328 \approx 0.70m$	1.29	5.80
2	5×10^{-4}	1713	$431269 \approx 0.42m$	2.01×10^4	$189051 \approx 0.18m$	1.84	18.34
3	2.5×10^{-3}	252	$37230 \approx 0.04m$	3.33×10^3	$20126 \approx 0.02m$	2.19	28.23
4	1.25×10^{-2}	35	$1217 < 0.01m$	4.53×10^2	$1093 < 0.01m$	2.02	35.08
The 3-level decomposition of SwissRoll data.							
0	-	20000	$248010 = m$	1.15×10^6	-	-	-
1	10^{-5}	5528	$689032 \approx 2.78m$	4.31×10^5	$197020 \approx 0.79m$	1.45	10.06
2	10^{-4}	723	$108887 \approx 0.44m$	7.44×10^4	$35213 \approx 0.14m$	2.30	67.47
3	10^{-3}	55	$2215 < 0.01m$	5.45×10^3	$1365 < 0.01m$	3.92	185.93
The 4-level decomposition of SwissRoll data.							
0	-	20000	$248010 = m$	1.15×10^6	-	-	-
1	10^{-5}	5528	$689032 \approx 2.78m$	4.31×10^5	$197020 \approx 0.79m$	1.45	10.06
2	5×10^{-5}	1347	$215811 \approx 0.87m$	9.36×10^4	$65169 \approx 0.26m$	1.90	26.52
3	2.5×10^{-4}	203	$18849 \approx 0.08m$	1.89×10^4	$9063 \approx 0.04m$	3.06	98.87
4	1.25×10^{-3}	53	$1939 < 0.01m$	3.72×10^3	$1193 < 0.01m$	3.36	51.14

$\|A^{(k)}\|_2 \le \delta_k \le c\varepsilon_k^{-1}$ as expected by Corollary 2.5 (we have normalized $\mu_1 = \|L^{-1}\|_2$ to 1). And the sparsities of $A^{(k)}$ and $M^{(k)}$ are of the same order as the sparsity of $A^{(0)} = L$, i.e., $nnz(A^{(k)}), nnz(M^{(k)}) = O(nnz(A^{(0)}))$ as we mentioned at the end of subsection 2.1.

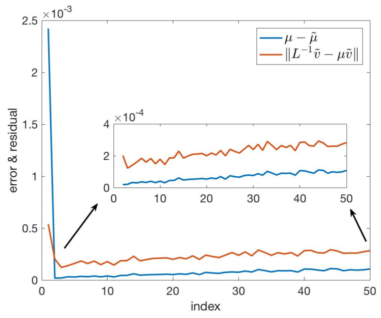
B.2. Numerical details of subsection 7.3. We take the bunny data and the brain data as examples. For the bunny data, we use the lowest level $k = 2$ with compression error $\varepsilon_2 = 0.01$; for the brain data, we use level $k = 3$ with compression



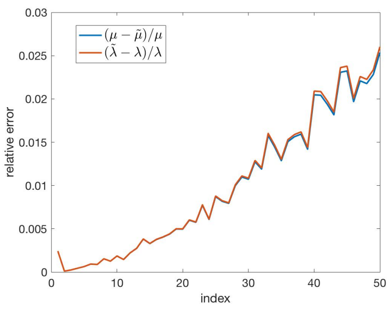
(a) error $\mu_i - \tilde{\mu}_i$ and residual $\|L^{-1}\tilde{v}_i - \mu_i\tilde{v}_i\|_2$



(b) relative error of μ_i and λ_i



(c) error $\mu_i - \tilde{\mu}_i$ and residual $\|L^{-1}\tilde{v}_i - \mu_i\tilde{v}_i\|_2$



(d) relative error of μ_i and λ_i

FIG. 5. *The error, the residual, and the relative error. Top: Bunny data; bottom: Brain data.*

error $\varepsilon_3 = 0.0025$. We compute the first 50 eigenpairs $\{\tilde{v}_i, \tilde{\lambda}_i\}$ of the compressed operator by directly solving the general eigenproblem (Lemma 3.2)

$$A^{(k)} z_i = \tilde{\lambda}_i M^{(k)} z_i, \quad \tilde{v}_i = \mathbf{\Psi}^{(k)} z_i, \quad i = 1, \ldots, 50.$$

The computation of the coarse level eigenproblem is much more efficient due to the compressed dimension. To show the error of the approximate eigenvalues, the ground truth is obtained by using the Eigen C++ library.[3] Figure 5 shows the absolute and relative errors of these eigenvalues. In both cases μ_i is the ith largest eigenvalue of L^{-1} and $\lambda_i = 1/\mu_i$; $\tilde{\mu}_i$ is the ith largest eigenvalue of the compressed problem $\Theta^{(k)}$ and $\tilde{\lambda}_i = 1/\tilde{\mu}_i$. By Lemma 3.3, $|\mu_i - \tilde{\mu}_i|$ is bounded by ε_k and $\|L^{-1}\tilde{v}_i - \mu_i\tilde{v}_i\|_2$ is bounded by $2\varepsilon_k$. We can see in Figure 5 that both estimates are well satisfied. In particular, the error of the first eigenvalue is close to the bound of ε_k. However, the first eigenpair is already known. Therefore, we are only interested in the 2nd up to 50th eigenvalues and we embed the subplot of these eigenvalue errors as shown in Figures 5(a) and 5(c), respectively.

Next, we qualitatively examine the accuracy of the approximate eigenvectors of the compressed operators by comparing their behaviors in image segmentation to those of the true eigenvectors of the original Laplacian operators. In the image segmentation, the eigenvectors of the graph Laplacian provide a solution to the graph

[3]Eigen C++ Library is available at http://eigen.tuxfamily.org/index.php?title=Main_Page.

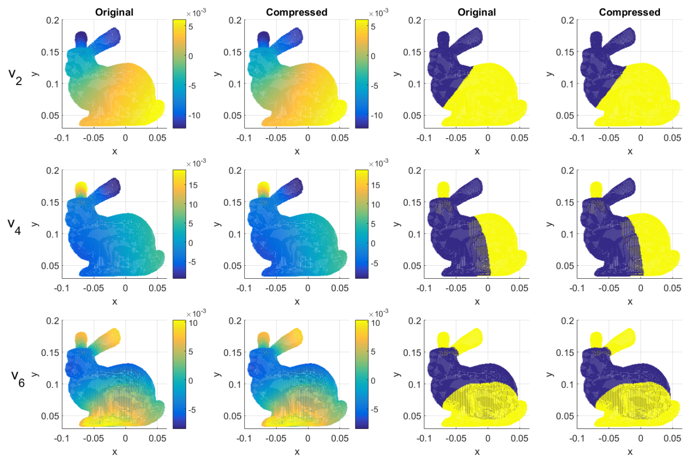


FIG. 6. *Colormap (left) and partition (right) using the* 2nd, 4th, *and* 6th *eigenvectors of the original/compressed operator.*

partitioning problem. Namely, for a partition (A, B) that satisfies $A \cup B = V$ and $A \cap B = \emptyset$, a measure of their disassociation called the normalized cut $(Ncut)$ is defined as [25]

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}, \tag{37}$$

where

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v), \qquad assoc(A, V) = \sum_{u \in A, t \in V} w(u, t).$$

Shi and Malik [25] shows that, for a connected graph, minimizing $Ncut$ can be rephrased as finding the eigenvector v_2 that corresponds to the second smallest eigenvalue λ_2 of the graph Laplacian (since we always have $\lambda_1 = 0$ and v_1 a uniform vector). Taking $sign(v_2)$ transforms it into a binary vector which gives a satisfactory cut. Moreover, the next few eigenvectors provide further cuts of the previously partitioned fractions. Therefore, our eigensolver may serve as a powerful tool for graph partitioning, as well as its applications including image segmentation and manifold learning.

We test graph partitioning on bunny and brain datasets using the eigenvectors of both original and compressed operators. Figures 6 and 7 shows the colormap and the partition generated by some selected eigenvectors. From the pictures we can see that the original and the compressed operators give very similar results when it comes to graph partitioning. The compressed operator is not only easier to compute, but also gives a satisfactory partition in practical settings.

Figure 8 gives an example of refining the partition with more eigenvectors. In the brain data, a fraction that is left intact in the first 5 eigenvectors (the light green part on the left) is divided into a lot more fractions when eigenvectors pile up to 15.

B.3. Numerical details of subsection 7.4. For both the Brain data and the SwissRoll data, we compute the first 500 eigenpairs of the graph Laplacian subject to prescribed accuracy $|\lambda_i^{-1} - \tilde{\lambda}_i^{-1}| = |\mu_i - \tilde{\mu}_i| \leq \epsilon = \varepsilon_1$. The three decompositions

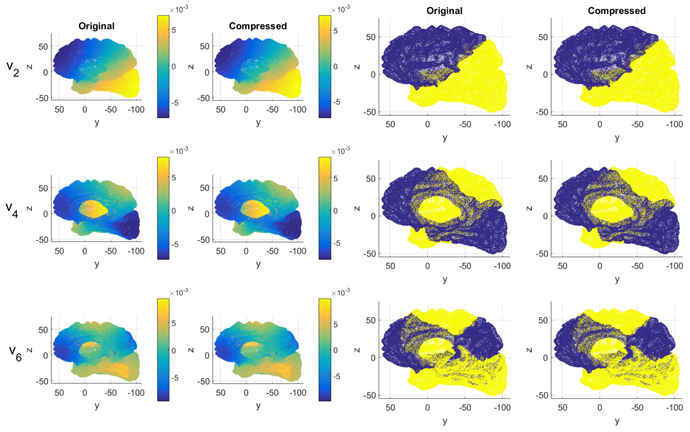


FIG. 7. *Colormap (left) and partition (right) using the* 2nd*,* 4th*, and* 6th *eigenvectors of the original/compressed operator.*

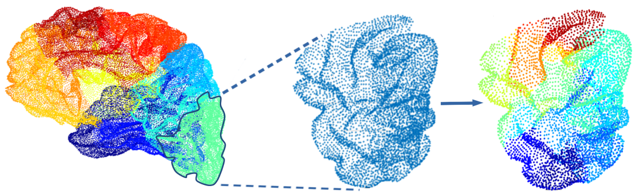


FIG. 8. *Heaping up more eigenvectors leads to finer partition. Left: partition using the first* 5 *eigenvectors. Middle: a uniform fraction from the previous partition. Right: further partition using the next* 10 *eigenvectors.*

TABLE 5
Computation information. $m \triangleq nnz(A^{(0)})$.

Data	Decomposition	(α,β)	(η,c)	κ	γ	Total #Iter	Total main cost
Brain	4-level	$(5,2)$	$(0.2,20)$	500	3/5	12	$4.37\times10^5\cdot m$
	4-level	$(3,1)$	$(0.2,20)$	300	2/3	15	$4.13\times10^5\cdot m$
SwissRoll	3-level	$(5,2)$	$(0.1,20)$	1000	3/5	13	$7.56\times10^5\cdot m$
	3-level	$(3,1)$	$(0.1,20)$	600	2/3	16	$7.17\times10^5\cdot m$
SwissRoll	4-level	$(5,2)$	$(0.2,20)$	500	3/5	19	$7.00\times10^5\cdot m$
	4-level	$(3,1)$	$(0.2,20)$	300	2/3	28	$5.86\times10^5\cdot m$

of these two datasets are used in this section. For each decomposition, we apply Algorithm 6 with two sets of parameters, $(\alpha,\beta)=(5,2)$ and $(\alpha,\beta)=(3,1)$. The details of the results that are obtained using Algorithm 6 are summarized in Tables 5–8. In Table 5, parameters $\alpha,\beta,\kappa,\gamma$ are defined in Section 6. In Tables 6–8, we collect numerical results that reflect the efficiency of each single process (refinement or extension). Here we give a detailed description of the notations in these tables:

- #I and #O denote the number of input and output eigenpairs. To be consistent with the notations defined in section 6, we use (#I,#O) $=(m_{k+1},\hat{m}_k)$

TABLE 6
4-level eigenpairs computation of Brain data with $(\eta, c) = (0.2, 20)$, $m \triangleq nnz(A^{(0)})$.

$(\alpha,\beta)=(5,2)$									
Refinement	Level k	(#I,#O)	#Iter	$\#_{\mathrm{cg}}(B^{(k+1)})$	$\overline{\#}(B^{(k+1)})$	$\#_{\mathrm{pcg}}(A^{(k)})$	$\overline{\#}(A^{(k)})$	$\overline{\#}(M^{(k)})$	Main cost
	3	$(7,4)$	4	7	24.43	28	10.97	6.10	$5.66 \times 10^1 \cdot m$
	2	$(41,17)$	4	41	25.90	164	16.26	6.12	$4.50 \times 10^3 \cdot m$
	1	$(207,84)$	4	207	23.44	828	19.17	4.64	$1.02 \times 10^5 \cdot m$
Extension	Level k	(#I,#O)	$\widehat{\#}(A^{(k)})$	$\epsilon^{(k)}$	$\frac{\widehat{\#}(A^{(k)})}{log(1/\epsilon^{(k)})}$	$\#_{\mathrm{pcg}}(A^{(k)})$	$\overline{\#}(A^{(k)})$	$\overline{\#}(M^{(k)})$	Main cost
	3	$(4,41)$	43	2.5×10^{-4}	5.18	175	16.93	5.39	$4.37 \times 10^2 \cdot m$
	2	$(17,207)$	75	5.0×10^{-5}	7.57	500	32.27	5.47	$2.27 \times 10^4 \cdot m$
	1	$(84,500)$	82	10^{-5}	7.12	1248	44.23	4.45	$3.07 \times 10^5 \cdot m$

$(\alpha,\beta)=(3,1)$									
Refinement	Level k	(#I,#O)	#Iter	$\#_{\mathrm{cg}}(B^{(k+1)})$	$\overline{\#}(B^{(k+1)})$	$\#_{\mathrm{pcg}}(A^{(k)})$	$\overline{\#}(A^{(k)})$	$\overline{\#}(M^{(k)})$	Main cost
	3	$(15,6)$	5	15	24.54	75	7.74	6.07	$1.08 \times 10^2 \cdot m$
	2	$(78,28)$	5	78	25.85	390	11.17	6.01	$7.39 \times 10^3 \cdot m$
	1	$(276,140)$	5	276	23.43	1380	14.28	4.67	$1.29 \times 10^5 \cdot m$
Extension	Level k	(#I,#O)	$\widehat{\#}(A^{(k)})$	$\epsilon^{(k)}$	$\frac{\widehat{\#}(A^{(k)})}{log(1/\epsilon^{(k)})}$	$\#_{\mathrm{pcg}}(A^{(k)})$	$\overline{\#}(A^{(k)})$	$\overline{\#}(M^{(k)})$	Main cost
	3	$(6,78)$	37	2.5×10^{-4}	4.46	225	14.12	5.41	$4.70 \times 10^2 \cdot m$
	2	$(28,276)$	57	5.0×10^{-5}	5.75	600	27.91	5.43	$2.34 \times 10^4 \cdot m$
	1	$(140,500)$	63	10^{-5}	5.47	1080	42.09	4.46	$2.53 \times 10^5 \cdot m$

for refinement processes on level k, and (#I,#O)= $(\hat{m}_k, m_k)$ for extension processes on level k.

- #Iter denotes the number of orthogonal iterations in the refinement process. Note that this number is controlled by the ratio γ.
- $\#_{\mathrm{cg}}(B^{(k)})$ denotes the number of CG calls concerning $B^{(k)}$ in the refinement process; $\#_{\mathrm{pcg}}(A^{(k)})$ denotes the number of PCG calls concerning $A^{(k)}$ in the refinement process and the extension process. $\overline{\#}(B^{(k)})$ and $\overline{\#}(A^{(k)})$ denote the average numbers of matrix-vector multiplications concerning $B^{(k)}, A^{(k)}$, respectively, namely, the average numbers of iterations, in one single call of CG or PCG. Note that $\overline{\#}(B^{(k)})$ is controlled by $\log(1/\epsilon^{(k)})\kappa(B^{(k)}) \leq \log(1/\epsilon^{(k)})c/\eta$, and $\overline{\#}(A^{(k)})$ by $\log(1/\epsilon^{(k)})\kappa(A_{\mathbf{\Psi}}^{(k)}, Z_{\hat{m}_k^+}^{(k)}) \leq \log(1/\epsilon^{(k)})\alpha c/\eta$.
- As the extension process proceeds, the target spectrum to be computed on this level shrinks even more, and so does the restricted condition number of the operator. Thus the number of iterations in each PCG call get much smaller than its expected control $\log(1/\epsilon^{(k)})\alpha c/\eta$, which is a good thing in practice. So to study how the theoretical bound $\log(1/\epsilon^{(k)})\alpha c/\eta$ really affects

TABLE 7
3-level eigenpairs computation of SwissRoll data with $(\eta, c) = (0.1, 20)$, $m \triangleq nnz(A^{(0)})$.

	$(\alpha,\beta)=(5,2)$								
Refinement	Level k	(#I,#O)	#Iter	$\#_{\text{cg}}(B^{(k+1)})$	$\overline{\#}(B^{(k+1)})$	$\#_{\text{pcg}}(A^{(k)})$	$\overline{\#}(A^{(k)})$	$\overline{\#}(M^{(k)})$	Main cost
	2	$(21, 12)$	7	21	52.14	147	17.61	6.33	$3.91 \times 10^3 \cdot m$
	1	$(232, 100)$	6	232	47.23	1392	16.08	5.29	$1.86 \times 10^5 \cdot m$
Extension	Level k	(#I,#O)	$\widehat{\#}(A^{(k)})$	$\epsilon^{(k)}$	$\frac{\widehat{\#}(A^{(k)})}{log(1/\epsilon^{(k)})}$	$\#_{\text{pcg}}(A^{(k)})$	$\overline{\#}(A^{(k)})$	$\overline{\#}(M^{(k)})$	Main cost
	2	$(12, 232)$	94	10^{-5}	8.16	650	28.20	7.25	$2.67 \times 10^4 \cdot m$
	1	$(100, 500)$	101	10^{-6}	7.31	1200	59.44	6.10	$5.42 \times 10^5 \cdot m$

	$(\alpha,\beta)=(3,1)$								
Refinement	Level k	(#I,#O)	#Iter	$\#_{\text{cg}}(B^{(k+1)})$	$\overline{\#}(B^{(k+1)})$	$\#_{\text{pcg}}(A^{(k)})$	$\overline{\#}(A^{(k)})$	$\overline{\#}(M^{(k)})$	Main cost
	2	$(35, 19)$	8	35	51.89	280	13.13	6.45	$5.74 \times 10^3 \cdot m$
	1	$(315, 165)$	8	315	46.85	2520	12.73	5.37	$2.66 \times 10^5 \cdot m$
Extension	Level k	(#I,#O)	$\widehat{\#}(A^{(k)})$	$\epsilon^{(k)}$	$\frac{\widehat{\#}(A^{(k)})}{log(1/\epsilon^{(k)})}$	$\#_{\text{pcg}}(A^{(k)})$	$\overline{\#}(A^{(k)})$	$\overline{\#}(M^{(k)})$	Main cost
	2	$(19, 315)$	69	10^{-5}	5.99	700	25.10	7.29	$2.57 \times 10^4 \cdot m$
	1	$(165, 500)$	78	10^{-6}	5.65	1005	54.91	6.11	$4.20 \times 10^5 \cdot m$

the efficiency of PCG calls, it is more reasonable to investigate the maximal number of iterations in one PCG call on each level. We use $\widehat{\#}(A^{(k)})$ to denote the largest number of iterations in one single PCG call on level k.

- $\overline{\#}(M^{(k)})$ denotes the average number of matrix-vector multiplications concerning $M^{(k)}$ in one single CG call concerning $M^{(k)}$. Such CG calls occur in the PCG calls concerning $A^{(k)}$, where $M^{(k)}$ acts as the preconditioner. Note that $\overline{\#}(M^{(k)})$ is controlled by $\log(1/\epsilon^{(k)})\kappa(M^{(k)}) \le \log(1/\epsilon^{(k)})(1+c)$.
- "Main cost" denotes the main computational cost contributed by matrix-vector multiplication flops. In the refinement process we have

$$\begin{aligned}\text{main cost} =& \#_{\text{cg}}(B^{(k+1)}) \cdot \overline{\#}(B^{(k+1)}) \cdot nnz(A^{(k)}) \\ &+ \#_{\text{pcg}}(A^{(k)}) \cdot \overline{\#}(A^{(k)}) \cdot \big(nnz(A^{(k)}) + \overline{\#}(M^{(k)}) \cdot nnz(M^{(k)})\big),\end{aligned}$$

while in the extension process we have

$$\text{main cost} = \#_{\text{pcg}}(A^{(k)}) \cdot \overline{\#}(A^{(k)}) \cdot \big(nnz(A^{(k)}) + \overline{\#}(M^{(k)}) \cdot nnz(M^{(k)})\big).$$

Tables 6–8 show the efficiency of our algorithm. We can see that $\overline{\#}(B^{(k)})$ and $\overline{\#}(M^{(k)})$ are well-bounded as expected, due to the artificial imposition of the condition bound c. $\widehat{\#}(A^{(k)})$ and the numerical condition number $\widehat{\#}(A^{(k)})/\log(1/\epsilon^{(k)})$ are also well-controlled by choosing α properly to bound $\kappa = \alpha c/\eta$. It is worth mentioning that $\widehat{\#}(A^{(k)})/\log(1/\epsilon^{(k)})$ appears to be uniformly bounded for all levels, actually much smaller than κ, which reflects our uniform control on efficiency. #Iter is well-bounded due to the proper choice of β for bounding $\gamma = (1+\beta)/\alpha$.

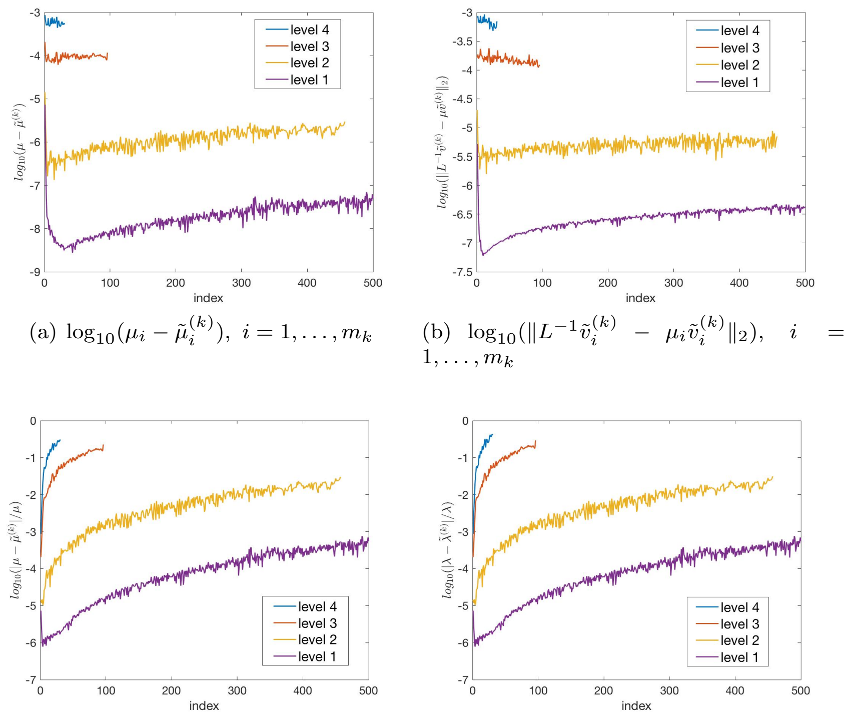
Table 8
4-level eigenpairs computation of SwissRoll data: $(\eta, c) = (0.2, 20)$*,* $m \triangleq nnz(A^{(0)})$*.*

$(\alpha,\beta)=(5,2)$									
Refinement	Level k	(#I,#O)	#Iter	$\#_{\mathrm{cg}}(B^{(k+1)})$	$\overline{\#}(B^{(k+1)})$	$\#_{\mathrm{pcg}}(A^{(k)})$	$\overline{\#}(A^{(k)})$	$\overline{\#}(M^{(k)})$	Main cost
	3	$(18, 10)$	6	18	22.61	108	7.19	7.87	$3.39 \times 10^2 \cdot m$
	2	$(84, 44)$	8	84	43.45	672	10.42	6.49	$2.11 \times 10^4 \cdot m$
	1	$(390, 195)$	5	390	28.85	1950	11.68	5.42	$1.92 \times 10^5 \cdot m$
Extension	Level k	(#I,#O)	$\widehat{\#}(A^{(k)})$	$\epsilon^{(k)}$	$\frac{\widehat{\#}(A^{(k)})}{log(1/\epsilon^{(k)})}$	$\#_{\mathrm{pcg}}(A^{(k)})$	$\overline{\#}(A^{(k)})$	$\overline{\#}(M^{(k)})$	Main cost
	3	$(10, 84)$	42	2.5×10^{-5}	3.96	200	18.32	8.43	$1.53 \times 10^3 \cdot m$
	2	$(44, 390)$	63	5×10^{-6}	5.16	1050	29.30	7.24	$8.47 \times 10^4 \cdot m$
	1	$(195, 50)$	71	10^{-6}	5.13	915	57.47	6.10	$4.00 \times 10^5 \cdot m$

$(\alpha,\beta)=(3,1)$									
Refinement	Level k	(#I,#O)	#Iter	$\#_{\mathrm{cg}}(B^{(k+1)})$	$\overline{\#}(B^{(k+1)})$	$\#_{\mathrm{pcg}}(A^{(k)})$	$\overline{\#}(A^{(k)})$	$\overline{\#}(M^{(k)})$	Main cost
	3	$(31, 16)$	7	31	22.45	217	6.09	8.09	$5.89 \times 10^2 \cdot m$
	2	$(95, 67)$	12	95	43.44	1140	7.66	6.66	$2.63 \times 10^4 \cdot m$
	1	$(459, 314)$	7	459	28.75	3656	8.71	5.56	$2.65 \times 10^5 \cdot m$
Extension	Level k	(#I,#O)	$\widehat{\#}(A^{(k)})$	$\epsilon^{(k)}$	$\frac{\widehat{\#}(A^{(k)})}{log(1/\epsilon^{(k)})}$	$\#_{\mathrm{pcg}}(A^{(k)})$	$\overline{\#}(A^{(k)})$	$\overline{\#}(M^{(k)})$	Main cost
	3	$(16, 95)$	31	2.5×10^{-5}	2.92	200	16.61	8.48	$1.39 \times 10^3 \cdot m$
	2	$(67, 459)$	49	5×10^{-6}	4.01	1100	25.66	7.27	$7.79 \times 10^4 \cdot m$
	1	$(314, 500)$	55	10^{-6}	3.98	558	50.61	6.12	$2.15 \times 10^5 \cdot m$

We may also compare the results for the same decomposition but from two different sets of parameters (α, β). For all three decompositions, the experiments with $(\alpha, \beta) = (5, 2)$ have a smaller $\gamma = \frac{3}{5}$, and thus is more efficient in the refinement process (fewer #Iter and less refinement main cost). While the experiments with $(\alpha, \beta) = (3, 1)$ have a smaller κ that leads to better efficiency in the extension process (smaller $\widehat{\#}(A^{(k)})/\log(1/\epsilon^{(k)})$ and less extension main cost). But since the dominant cost of the whole process comes from the extension process, then the experiments with $(\alpha, \beta) = (3, 1)$ have a smaller total main cost.

We remark that the choice of (α, β) not only determines (κ, γ) that will affect the algorithm efficiency, but also determines the segmentation of the target spectrum and its allocation towards different levels of the decomposition. Smaller values of α and β mean more eigenpairs being computed on coarser levels (larger k), which relieves the burden of the extension process for finer levels, but also increases the load of the refinement process. There could be an optimal choice of (α, β) that minimizes the total main cost, balancing between the refinement and the extension processes. However, without a priori knowledge of the distribution of the eigenvalues, which is the case in practice, a safe choice of (α, β) would be $\alpha, \beta = O(1)$.



(a) $\log_{10}(\mu_i - \tilde{\mu}_i^{(k)}),\ i = 1, \dots, m_k$

(b) $\log_{10}(\|L^{-1}\tilde{v}_i^{(k)} - \mu_i\tilde{v}_i^{(k)}\|_2),\ i = 1, \dots, m_k$

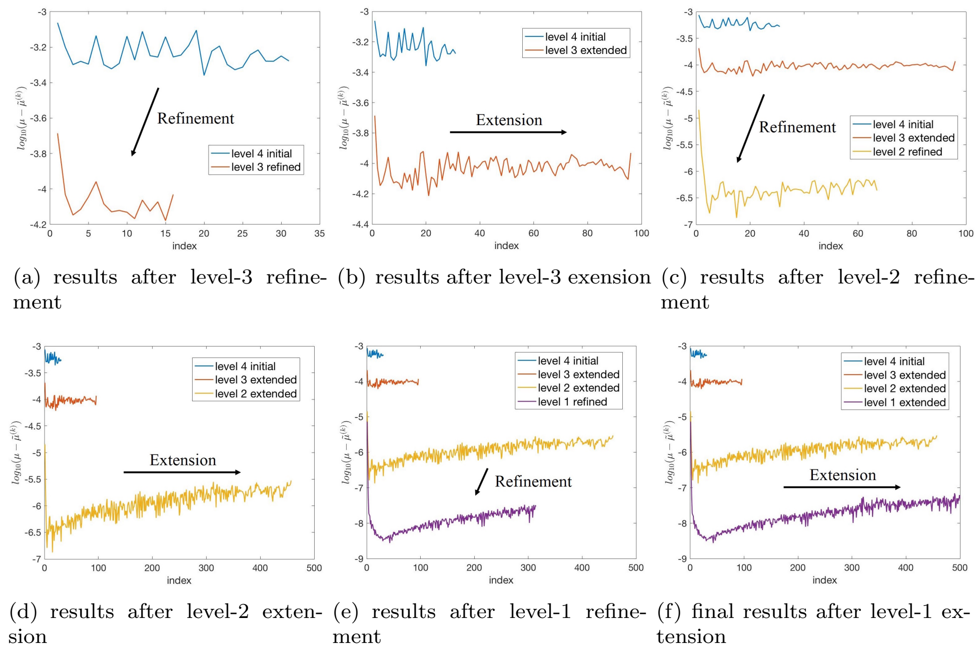
(c) $\log_{10}((\mu_i - \tilde{\mu}_i^{(k)})/\mu_i),\ i = 1, \dots, m_k$

(d) $\log_{10}((\lambda_i - \tilde{\lambda}_i^{(k)})/\lambda_i),\ i = 1, \dots, m_k$

FIG. 9. *Convergence of computed spectrum in different errors.*

To further investigate the behavior of our algorithm, we focus on numerical experiments carried out on the 4-level decomposition of the SwissRoll data. Figure 9 shows the convergence of the computed spectrum in different errors. Figure 10 shows the completion and the convergence processes of the target spectrum in the case of $(\alpha, \beta) = (3, 1)$ (corresponding to Table 8). We use a log-scale plot to illustrate the error $|\mu_i - \tilde{\mu}^{(k)}|$ after we complete the refinement process and the extension process, respectively, on each level k. As we can see, each application of the refinement process improves the accuracy of the first $\hat{m}_k$ eigenvalues at least by a factor of $\eta = \frac{\varepsilon_k}{\varepsilon_{k+1}}$, but at the price of discarding the last $m_{k+1} - \hat{m}_k$ computed eigenvalues. So the computation of the last $m_{k+1} - \hat{m}_k$ computed eigenvalues on the coarser level $k+1$ actually serves as preconditioning to ensure the efficiency of the refinement process on level k. Then the extension process extends the spectrum to m_k that is determined by the threshold $\mu_{ex}^{(k)}$. The whole computation is an iterative process that improves the accuracy of the eigenvalues by applying the hierarchical Lanczos method to each eigenvalue at most twice.

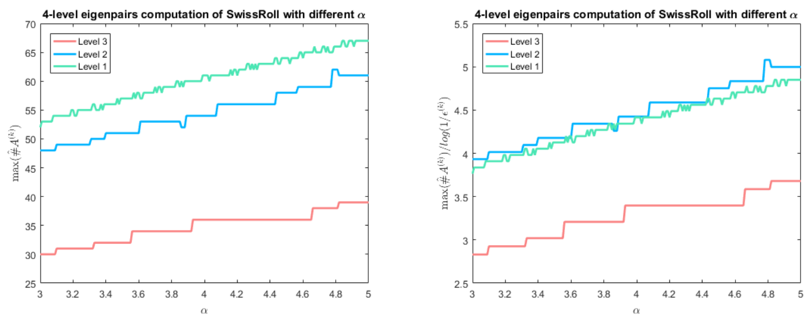
We also further verify our critical control on the restricted condition number $\kappa(A_{\Psi}^{(k)}, Z_{\hat{m}_k^+}^{(k)})$ by $\kappa = \alpha c/\eta$, by showing the dependence of $\widehat{\#}(A^{(k)})$(or $\widehat{\#}(A^{(k)})/\log(1/\epsilon^{(k)})$) on κ. Recall that $\widehat{\#}(A^{(k)})$ denotes the largest number of iterations in one single PCG call concerning $A^{(k)}$ on level k. Using the 4-level decomposition of the SwissRoll data with $(\eta, c) = (0.2, 20)$, we perform Algorithm 6 with fixed $\beta = 1$ but different $\alpha \in [3, 5]$. Figure 11 shows $\widehat{\#}(A^{(k)})$ versus α for all three levels. By



(a) results after level-3 refinement (b) results after level-3 exension (c) results after level-2 refinement

(d) results after level-2 extension (e) results after level-1 refinement (f) final results after level-1 extension

FIG. 10. *The completion and convergence process of the target spectrum. The refinement process retains part of the spectrum subject to threshold $\mu_{re}^{(k)}$ with improved accuracy, and the extension process extends the spectrum subject to threshold $\mu_{ex}^{(k)}$. The whole process is an iterative procedure that aims at improving the accuracy of the eigenvalue solver.*



(a) $\hat{\#}A^{(k)}$ versus α (b) $\hat{\#}A^{(k)}/\log(\epsilon_{pcg}^{-1})$ versus α

FIG. 11. $\hat{\#}A^{(k)}$ *versus* α *in the* 4*-level SwissRoll example.*

Theorem 4.8, we expect that $\widehat{\#}(A^{(k)}) \propto \kappa \cdot \log(1/\epsilon^{(k)}) \propto \alpha \cdot \log(1/\epsilon^{(k)})$. This linear dependence is confirmed in Figure 11. It is also important to note that the curve (green) corresponding to level-1 is below the curve (blue) corresponding to level-2 in Figure 11(b), which again implies that $\widehat{\#}(A^{(k)})/\log(1/\epsilon^{(k)})$ is uniformly bounded for all levels.

Acknowledgments. Ziyun Zhang would like to acknowledge ACM, Caltech and SMS, PKU for supporting her research visit to Caltech in 2017 summer. She would also like to thank ACM's staff for their hospitality during her visit.

REFERENCES

[1] L. Bergamaschi and E. Bozzo, *Computing the smallest eigenpairs of the graph Laplacian*, SeMA J., 75 (2018), pp. 1–16.

[2] L. Bergamaschi, G. Gambolati, and G. Pini, *Asymptotic convergence of conjugate gradient methods for the partial symmetric eigenproblem*, Numer. Linear Algebra Appl., 4 (1997), pp. 69–84.

[3] E. Bozzo and M. Franceschet, *Effective and Efficient Approximations of the Generalized Inverse of the Graph Laplacian Matrix with an Application to Current-Flow Betweenness Centrality*, preprint, arXiv:1205.4894, 2012.

[4] E. Bozzo and M. Franceschet, *Resistance distance, closeness, and betweenness*, Soc. Netw., 35 (2013), pp. 460–469.

[5] D. Calvetti, L. Reichel, and D. C. Sorensen, *An implicitly restarted Lanczos method for large symmetric eigenvalue problems*, Electron. Trans. Numer. Anal., 2 (1994), pp. 1–21.

[6] F. R. Chung, *Spectral Graph Theory*, CBMS Reg. Conf. Ser. Math. 92, American Mathematical Society, Providence, RI, 1997.

[7] S. Cocco, R. Monasson, and M. Weigt, *From principal component to direct coupling analysis of coevolution in proteins: Low-eigenvalue modes are needed for structure prediction*, PLoS Comput. Biol., 9 (2013), e1003176.

[8] J. Francis, *The transformation: A unitary analogue to the transformation.* I, Comput. J., 4 (1961), pp. 265–271.

[9] S. Goedecker, *Low complexity algorithms for electronic structure calculations*, J. Comput. Phys., 118 (1995), pp. 261–268.

[10] T. Hou, D. Huang, K. Lam, and P. Zhang, *An adaptive fast solver for a general class of positive definite matrices via energy decomposition*, Multiscale Model. Simul., 16 (2018), pp. 615–678.

[11] Y. T. Hou and P. Zhang, *Sparse operator compression of higher-order elliptic operators with rough coefficients*, Res. Math. Sci., 4, (2017), 24.

[12] R. B. Lehoucq and D. C. Sorensen, *Deflation techniques for an implicitly restarted Arnoldi iteration*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 789–821.

[13] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998.

[14] Q. Lin and H. Xie, *A multi-level correction scheme for eigenvalue problems*, Math. Comp., 84 (2015), pp. 71–88.

[15] A. Målqvist and D. Peterseim, *Localization of elliptic multiscale problems*, Math. Comp., 83 (2014), pp. 2583–2603.

[16] Á. Martínez, *Tuned preconditioners for the eigensolution of large SPD matrices arising in engineering problems*, Numer. Linear Algebra Appl., 23 (2016), pp. 427–443.

[17] L. Meirovitch, *Elements of Vibration Analysis*, McGraw-Hill, New York, 1975.

[18] M. Newman, *Networks: An Introduction*, Oxford University Press, Oxford, 2010.

[19] A. Y. Ng, M. I. Jordan, and Y. Weiss, *On spectral clustering: Analysis and an algorithm*, in 14th NIPS'01, MIT Press, Cambridge, MA, 2001, pp. 849–856.

[20] H. Owhadi, *Multigrid with rough coefficients and multiresolution operator decomposition from hierarchical information games*, SIAM Rev., 59 (2017), pp. 99–149.

[21] H. Owhadi and C. Scovel, *Universal Scalable Robust Solvers from Computational Information Games and Fast Eigenspace Adapted Multiresolution Analysis*, preprint, arXiv:1703.10761, 2017.

[22] V. Ozoliņš, R. Lai, R. Caflisch, and S. Osher, *Compressed modes for variational problems in mathematics and physics*, Proc. Natl. Acad. Sci. USA, 110 (2013), pp. 18368–18373.

[23] E. Romero, M. B. Cruz, J. E. Roman, and P. B. Vasconcelos, *A parallel implementation of the Jacobi-Davidson eigensolver for unsymmetric matrices*, in VECPAR 2010, Springer, Berlin, 2010, pp. 380–393.

[24] F. Schäfer, T. Sullivan, and H. Owhadi, *Compression, Inversion, and Approximate PCA of Dense Kernel Matrices at Near-Linear Computational Complexity*, preprint, arXiv:1706.02205, 2017.

[25] J. Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE Trans. Pattern Anal. Mach. Intell., 22 (2000), pp. 888–905.

[26] G. L. Sleijpen and H. A. Van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM Rev., 42 (2000), pp. 267–293.

[27] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.

[28] D. C. Sorensen, *Implicitly restarted Arnoldi/Lanczos methods for large scale eigenvalue calculations*, in Parallel Numerical Algorithms, Springer, Dordrecht, The Netherlands, 1997, pp. 119–165.

[29] G. Stewart, *Accelerating the orthogonal iteration for the eigenvectors of a Hermitian matrix*, Numer. Math., 13 (1969), pp. 362–376.

[30] H. Xie, *A multigrid method for eigenvalue problem*, J. Comput. Phys., 274 (2014), pp. 550–561.

[31] H. Xie, L. Zhang, and H. Owhadi, *Fast Eigenpairs Computation with Operator Adapted Wavelets and Hierarchical Subspace Correction*, preprint, arXiv:1806.00565, 2018.