A Shift Selection Strategy for Parallel Shift-invert Spectrum Slicing in Symmetric Self-consistent Eigenvalue Computation

DAVID B. WILLIAMS-YOUNG, Lawrence Berkeley National Laboratory
PAUL G. BECKMAN, University of Chicago
CHAO YANG, Lawrence Berkeley National Laboratory

The central importance of large-scale eigenvalue problems in scientific computation necessitates the development of massively parallel algorithms for their solution. Recent advances in dense numerical linear algebra have enabled the routine treatment of eigenvalue problems with dimensions on the order of hundreds of thousands on the world's largest supercomputers. In cases where dense treatments are not feasible, Krylov subspace methods offer an attractive alternative due to the fact that they do not require storage of the problem matrices. However, demonstration of scalability of either of these classes of eigenvalue algorithms on computing architectures capable of expressing massive parallelism is non-trivial due to communication requirements and serial bottlenecks, respectively. In this work, we introduce the SISLICE method: a parallel shift-invert algorithm for the solution of the symmetric self-consistent field (SCF) eigenvalue problem. The SISLICE method drastically reduces the communication requirement of current parallel shift-invert eigenvalue algorithms through various shift selection and migration techniques based on density of states estimation and k-means clustering, respectively. This work demonstrates the robustness and parallel performance of the SISLICE method on a representative set of SCF eigenvalue problems and outlines research directions that will be explored in future work.

CCS Concepts: • **Mathematics of computing → Mathematical software performance**; • **Applied computing → Mathematics and statistics**;

Additional Key Words and Phrases: Eigenvalues, parallel eigenvalue algorithms, self-consistent field, shift-invert spectrum slicing

ACM Reference format:
David B. Williams-Young, Paul G. Beckman, and Chao Yang. 2020. A Shift Selection Strategy for Parallel Shift-invert Spectrum Slicing in Symmetric Self-consistent Eigenvalue Computation. *ACM Trans. Math. Softw.* 46, 4, Article 35 (October 2020), 31 pages.
https://doi.org/10.1145/3409571

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program (D. B. W. and C. Y.), and by the Berkeley Lab Undergraduate Research (BLUR) Program, managed by Workforce Development & Education at Berkeley Lab (P. B.). This research was also partially supported by the Exascale Computing Project (No. 17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.
Authors' addresses: D. B. Williams-Young and C. Yang, Lawrence Berkeley National Laboratory, 1 Cyclotron Rd, MS 50A-3111, Berkeley, CA 94720 US; emails: {dbwy, cyang}@lbl.gov; P. G. Beckman, University of Chicago, 5801 S Ellis Ave, Chicago, IL 60637; email: pbeckman@uchicago.edu.

0098-3500/2020/10-ART35 $15.00
https://doi.org/10.1145/3409571

1 INTRODUCTION

Large-scale symmetric eigenvalue problems arise in many types of scientific computation [Yang 2005]. In the case of electronic structure calculations based on the Hartree–Fock approximation or Kohn-Sham density functional theory, a large symmetric nonlinear eigenvalue problem must be iteratively solved through what is known as the self-consistent field (SCF) procedure. Typical methods to solve the so-called SCF eigenvalue problem require the partial diagonalization of a sequence of matrix pencils where each pencil of the sequence is generated using a subset of the eigenvectors of the previous pencil. The SCF problem is considered solved when convergence of the sequence is achieved, i.e., the change in the matrix pencil (or equivalently, the desired eigenvectors of said pencil) between two subsequent iterations of the SCF procedure falls below a specified threshold. One of the hallmarks of the SCF procedure is that the desired eigenpairs need not be computed to full accuracy before convergence is reached. However, as the sequence progresses, an increasing level of accuracy in the desired eigenpairs is needed to ensure convergence to the proper SCF solution. Although the spectrum of the matrix pencils may change quite a bit in the first few iterations of the SCF procedure, this change becomes progressively smaller as convergence is reached. This feature makes the design and implementation of algorithms for solving the SCF eigenvalue problem somewhat different from traditional algorithms for solving an eigenvalue problem of a fixed matrix. The repeated diagonalization required by the SCF procedure is often the computational bottleneck in large-scale electronic structure calculations [Banerjee et al. 2018; Jay et al. 1999; Shepard 1993], especially in cases where a large number of computational resources are available. As such, methods must be developed to efficiently solve this class of nonlinear eigenvalue problem on modern, massively-parallel computing architectures.

In this work, we consider the partial diagonalization of n_e eigenpairs of a converging sequence of symmetric matrix pencils, $(A^{(i)}, B)$, of dimension N,

$$A^{(i)}X^{(i)} = BX^{(i)}\Lambda^{(i)}, \tag{1}$$

where $i \in \mathbb{Z}^+$ is a sequence index, $A^{(i)} \in \mathbb{R}^{N\times N}$ is symmetric and $B \in \mathbb{R}^{N\times N}$ is symmetric positive definite (SPD). $X^{(i)} \in \mathbb{R}^{N\times n_e}$ and $\Lambda^{(i)} \in \mathbb{R}^{n_e\times n_e}$ are the eigenvectors and the diagonal matrix of eigenvalues corresponding to the desired eigenpairs of $(A^{(i)}, B)$, respectively. We denote the eigenvalues as $\Lambda^{(i)}_{pq} = \delta_{pq}\lambda^{(i)}_p$ and will refer to the increment of i as an *SCF iteration*. Further, we will make the following assumptions about the sequence of matrix pencils:

- We assume that $A^{(i+1)}$ depends in some (possibly non-linear) way on $(X^{(i)}, \Lambda^{(i)})$.
- As the SCF iterations progress, we assume $A^{(i)}$ converges toward a matrix A, but are not concerned with how this convergence is achieved other than the requirement that the convergence is not chaotic and the desired eigenpairs of $(A^{(i)}, B)$ must be computed to progressively higher accuracy as this convergence occurs.
- We assume that the desired eigenpairs of each matrix pencil in the sequence are contiguous within the desired spectral region bounded by $\lambda^{(i)}_{\text{min}}$ and $\lambda^{(i)}_{\text{max}}$. Remark that these bounds need not correspond to the spectral bounds of $(A^{(i)}, B)$.

In cases where n_e is relatively small ($O(<1{,}000)$) compared to N, or when $(A^{(i)}, B)$ is sparse or structured, iterative algorithms such as the implicitly restarted Lanczos algorithm [Lehoucq et al. 1998], the Jacobi-Davidson algorithm [Sleijpen and Van der Vorst 2000; Stathopoulos and McCombs 2007], and the locally optimal block preconditioned conjugate gradient (LOBPCG) algorithm [Knyazev 2001] are often very effective. In cases where n_e is a considerable fraction of N or when n_e is larger than thousands or tens of thousands, iterative algorithms become less efficient

partly due to the need to solve a projected dense eigenvalue problem as a part of the Rayleigh-Ritz procedure via some dense eigensolver.

Dense eigensolvers, such as those available in the LAPACK [Anderson et al. 1999], ScaLAPACK [Blackford et al. 1997] and ELPA [Marek et al. 2014] libraries, are also often used to perform a full diagonalization of each $(A^{(i)}, B)$. Recent advances in dense numerical linear algebra have made it possible to perform full diagonalizations for matrices of dimension $O(10{,}000) - O(100{,}000)$ in a few wall clock minutes using thousands to tens of thousands computational cores. However, making further improvements when even more computational resources (e.g., GPU accelerators) become available appears to be difficult due to the communication requirement of existing parallel algorithms.

In this report, we present the SISLICE method: a parallel symmetric eigensolver based on shift-invert spectrum slicing for the solution of the SCF eigenvalue problem described in Equation (1). In spectrum slicing methods, the eigenspectrum of the problem of interest is divided into several subintervals (spectral slices) such that the eigenvalues within each slice may be computed simultaneously. This approach eliminates the Rayleigh-Ritz bottleneck and increases the potential for concurrency in a parallel implementation. The notion of spectrum slicing is well documented in the literature for many classes of eigenvalue problems [Bai et al. 2000; Saad 2011]. While the basic idea behind spectrum slicing is relatively simple, its practical implementation on large computational resources is non-trivial. The SISLICE method has been developed to address the following practical issues of spectrum slicing for the SCF eigenvalue problem:

- Efficiently partitioning a spectral region of interest into spectral slices given minimal *a priori* knowledge of the eigenvalue distribution,
- Tracking and updating spectral slice locations using emerging knowledge of the dynamic (but convergent) eigenvalue distribution as it changes throughout the SCF procedure, and
- Accelerating convergence of the desired eigenpairs using knowledge of the approximate eigenvectors obtained from previous SCF iterations.

Fundamental to the development of spectrum slicing methods is the choice of method for computing the approximate eigenpairs within each spectral slice. For this purpose, the SISLICE method employs the shift-invert subspace iteration. Despite its simplicity, the shift-invert subspace iteration is particularly attractive, because the convergence of the method is sufficiently fast for eigenpairs in the spectral neighborhood of a target shift. From a practical perspective, it is robust and relatively simple to implement on contemporary high-performance computing architectures. There are a few alternatives to the shift-invert subspace iteration, which include:

- The shift-invert Lanczos method (SI-Lanczos). Historically, this has been the method of choice for spectrum slicing methods [Aktulga et al. 2014; Campos and Roman 2012; Grimes et al. 1994; Keçeli et al. 2016, 2018; Zhang et al. 2007]. However, the dependency among the sequence of linear systems the method must solve makes it difficult to achieve good parallel scalability.
- Polynomial filtering-based methods, such as those implemented in the Eigenvalue Slicing Library (EVSL) [Li et al. 2019], apply either the subspace [Banerjee et al. 2016, 2018] or Lanczos iteration [Li et al. 2016] to a bandpass matrix polynomial filter, which amplifies the spectral components associated with eigenvalues within a particular spectral slice. However, in cases when the spectral slice of interest is not well separated from adjacent slices or when the size of the slice is small, polynomial filtering methods often require the construction of extremely high degree polynomials to ensure convergence. In practice, this can

lead to significant performance degradation and load imbalance in spectrum slicing based on polynomial filtering.

- Contour integral-based methods [Sakurai and Sugiura 2003] such as the widely adopted (P)FEAST method [Kestyn et al. 2016; Polizzi 2008; Tang and Polizzi 2014] require the solution of a number of complex shifted linear systems over a quadrature discretization of the spectral projection operator in some spectral region of interest. However, choosing contours and quadrature discretizations to balance convergence and computational work for arbitrary spectral domains is non-trivial [Aktulga et al. 2014].

Another advantage of using a shift-invert subspace iteration is that the method can use the approximate eigenvectors obtained from previous SCF iterations as the initial guess to drastically improve the convergence rate of the subspace iteration for the current SCF iteration. We note that this feature is shared by other methods such as FEAST [Polizzi 2008] and polynomial filtering methods based on the subspace iteration [Banerjee et al. 2016].

Over the years, several schemes have been developed for parallel shift-invert spectrum slicing. The SIPs method of Zhang et al. [2007] utilizes a scheme that concurrently processes overlapping spectral subdomains through dynamically selecting shifts by tracking the convergence of the SI-Lanczos iterations within each subdomain. The dynamic shift selection within each subdomain is similar to the sequential shift selection scheme presented by Grimes et al. [1994]. More recently, the SIPs method has been extended to a dynamically scheduled two-level parallel scheme targeting the SCF eigenvalue problem (SIESTA-SIPs [Keçeli et al. 2018]), which processes non-overlapping spectral subdomains with an improved dynamic shift selection within each domain as developed in the SLEPc library [Campos and Roman 2012]. While robust, this shift selection strategy does not maximize the potential for concurrency viz independent slices as the dynamic shift selection within a spectral subdomain still occurs sequentially.

To address this, SISLICE adopts a strategy that allows for selection of all of the shifts that partition a spectral region of interest at the same time. We utilize a spectral density estimation (also known as the density of states or DOS) to approximate the distribution of eigenvalues within the domain of interest as discussed in Li et al. [2016], Lin et al. [2016], Saad [2011], and Xi et al. [2018]. In addition to allowing for the concurrent processing of all shifts simultaneously, DOS-based shift selection also allows for a more optimal initial shift selection, as we are able to place more shifts in spectral regions that have many tightly spaced eigenvalues that are not well separated from the rest of the spectrum and fewer shifts in regions that have isolated clusters with small radii.

To address the eigenvalue migration of the SCF eigenvalue problem, the SISLICE method employs an eigenvalue clustering strategy to refine shift placement. SIESTA-SIPs also adopts a similar clustering strategy for this purpose, though there are a number of important differences with the SISLICE scheme, which will be discussed in Section 3. Because the matrix sequence in Equation (1) can change significantly in early SCF iterations when it is far from converged, some of the selected shifts resulting from the analysis of approximate eigenvalues in the previous SCF iteration may not be optimal. Consequently, some of these shifts may need to be deleted and new shifts may need to be inserted to ensure no eigenvalue is missed and all eigenvalues within the spectral region of interest can be computed efficiently by the shift-invert subspace iteration. We also discuss how this can be implemented in conjunction with a spectral slice validation scheme in Section 3.

The SISLICE method is designed to minimize communication overhead and thus improve parallel scalability at the expense of performing more local calculations. This strategy takes into account the recent trend in high performance computing platforms in which floating point operations have become cheaper due to the emergence of multicore processors and accelerators while data movement remains costly. In SISLICE, we compute more approximate eigenpairs than the number of

Fig. 1. Partitioning the spectrum of interest into several slices or subintervals, which may be computed simultaneously.

eigenvalues within a spectral slice. This redundancy does not necessarily increase time to solution if there is an abundance of computational resources that can accommodate such redundancy. However, it makes the validation of eigenpairs easier and more efficient to implement. In particular, we will show that in SISLICE it is not necessary to check mutual orthogonality of approximate eigenvectors obtained in different spectral slices. As a result, our validation scheme does not require moving vectors across different nodes or processor groups, which is often costly. This key feature enables SISLICE to scale to very large processor counts.

This article is organized as follows. Section 2 briefly reviews the salient aspects of shift-invert spectrum slicing and the spectral slice validation scheme used by the SISLICE method. Sections 3 and 4 examine the practical issues of spectrum slicing, such as shift selection, parallel load balance, and so on, and how the SISLICE method aims to resolve them. Section 5 provides a series of numerical experiments that exhibit the performance and robustness of the proposed SISLICE method, and some additional improvements to the SISLICE method, which we will implement in the future, are discussed in Section 6.

2 SHIFT-INVERT SPECTRUM SLICING

Algorithm 1 outlines the major steps of the SISLICE method for solving the SCF eigenvalue problem. At each SCF iteration, SISLICE partitions the spectral region of interest of a matrix pencil $(A^{(i)}, B)$ into subintervals that may be treated independently. These spectral subintervals will be referred to as *spectral slices* in this work. The partition is done by selecting a set of n_s points $\{\sigma_j \mid \sigma_j \in \mathbb{R}\}_{j=1}^{n_s}$ as shown in Figure 1. A particular σ_j will be referred to as a *spectral shift*. These shift yields $n_s + 1$ spectral slices. Each spectral slice is bounded on either side by either a spectral shift or $\lambda_{\min}$ ($\lambda_{\max}$) for slice 1 ($n_s + 1$), respectively. As such, the problem of computing the eigenpairs within a particular spectral slice amounts to computing approximate eigenpairs in the neighborhood of the shifts that form its boundary and validating those eigenpairs against some well defined criteria. We note here that the SISLICE method treats n_s as a static quantity throughout the SCF procedure. This constraint is adopted primarily for load balance considerations in a distributed parallel computing environment (see Section 4 for details). In this work, we use the shift-invert subspace iteration to compute eigenpairs near a spectral shift. The approximate eigenvectors obtained from a particular SCF iteration are used as a *best guess* approximation (initial guess) for the subsequent iteration. The validation of eigenpairs takes into account the shifted matrix inertia as well as the accuracy of the computed eigenpairs. Specific details regarding the selection of spectral shifts are given in Section 3. In this section, we review the salient aspects of the shift-invert subspace iteration and slice validation scheme used by the SISLICE method given a set of spectral shifts.

We should note that the algorithm outlined in Algorithm 1 can be used to compute desired eigenpairs of a fixed matrix pencil also. When the matrix pencil (A, B) is fixed, we obviously do not need to perform the update in Step 5; however, we may improve the efficiency of the shift-invert subspace iteration by repartitioning the spectral region of interest (Step 1) using previously computed eigenvalue approximations as the reference. The application of Algorithm 1 to a fixed eigenvalue problem is particularly attractive for large, sparse matrix pencils with $N > O(100{,}000)$

where sparse matrix factorizations are possible, but direct eigenvalue decomposition is impractical on currently available computer hardware. See Section 5.5 for such an example.

ALGORITHM 1: Shift-invert spectrum slicing for computing n_e eigenpairs of a sequence of matrix pencils $(A^{(i)}, B)$ as they converge to a pencil (A, B).

Input: $(A^{(0)}, B)$, number of desired eigenpairs n_e, number of slices K

Output: $X \in \mathbb{R}^{N \times n_e}$ and diagonal matrix $\Lambda \in \mathbb{R}^{n_e \times n_e}$, which describe the desired n_e eigenpairs of the converged (A, B).

for $i = 0, 1, 2, \ldots$ **do**
1. Partition the spectral region of interest for $(A^{(i)}, B)$ into K slices by selecting $n_s = K - 1$ spectral shifts;
2. Choose starting guesses for eigenvectors within each slice;
3. Use shift-invert subspace iteration to obtain approximateeigenpairs within each slice;
4. Validate the computed eigenvalues $\rightarrow (X, \Lambda)$;
5. Compute the matrix $A^{(i+1)}$ using (X, Λ);
   if $A^{(i+1)}$ *converged* **then**
6. **return** (X, Λ);
   end

end

2.1 The Shift–Invert Subspace Iteration

Approximate eigenpairs of (A, B) corresponding to eigenvalues in the neighborhood of a shift, σ, may be obtained through the shift-invert subspace iteration [Bai et al. 2000; Saad 2011]. For the purposes of this work, we assume that σ is distinct from any eigenvalue of (A, B). One possible implementation of the shift-invert subspace iteration is given in Algorithm 2. The major cost of this implementation is in the Bunch-Kaufman (LDL^T) factorization of $A - \sigma B$. When A and B are dense, we employ the LDL^T factorization implemented in LAPACK or ScaLAPACK. When A and B are sparse, we may use a symmetric sparse solver such as those implemented in MUMPS [Amestoy et al. 2001, 2006], PARDISO [Schenk and Gärtner 2002, 2006; Schenk et al. 2000], symPACK [Bachan et al. 2017, 2019], and so on. The diagonal factor D produced by this factorization may be further used in the validation of eigenpairs within a spectral slice (see Section 2.2 for details). We use the Cholesky QR algorithm to B-orthonormalize the basis vector V of the subspace produced by the subspace iteration. This procedure requires computing the metric Gramian matrix $G = V^T BV$, performing a Cholesky factorization $G = RR^T$ and using back substitution to update V via $V \leftarrow VR^{-T}$. Upon the completion of the subspace iteration, we use the Rayleigh Ritz procedure to retrieve approximate eigenvalues and eigenvectors of (A, B) from the subspace V. This procedure requires solving the projected eigenvalue problem $(V^T AV)Q = Q\Omega$ and updating V by $V \leftarrow VQ$.

Remark that the algorithm outlined in Algorithm 2 is the simplest version of the shift-invert subspace iteration. Several modifications can be made to improve the performance and robustness of the algorithm, especially in the context of a convergent sequence of matrix pencils, which must be treated in Equation (1). For example, the number of columns of $V_{(m)}$ needs only be *at least* k to obtain approximations for k eigenpairs. In practice, the convergence rate of the subspace iteration in obtaining the k *desired* eigenpairs may be drastically improved by choosing a trial vector space that is several times larger than k (see Section 5.4 for examples). Further, we may

exploit the fact that $(X^{(i)}, \Lambda^{(i)})$ need not be computed to full accuracy until the SCF iterations of Equation (1) are nearly converged. The rate at which $V_{(m)}$ approaches X in Algorithm 2 largely depends on the choice of initial guess $V_{(0)}$. If the distance between $V_{(0)}$ and X (as measured in terms of subspace angle) is sufficiently small, then convergence may be achieved in only a few subspace iterations. As $A^{(i)}$ converges to A, the change in the eigensystem between $(A^{(i)}, B)$ and $(A^{(i+1)}, B)$ becomes sufficiently small such that the distance between $X^{(i)}$ and $X^{(i+1)}$ is also small. Thus, the subspace iteration may be seeded with $X^{(i)}$ to obtain $X^{(i+1)}$ to enable faster convergence. This assumption is typically most valid in the last few SCF iterations, though this seeding procedure will be demonstrated to be effective throughout the SCF procedure for the problems considered in Section 5.

ALGORITHM 2: The Shift-invert Subspace Iteration: SISubIt$(A, B, V_{(0)}, \sigma, k, M)$

Input: Symmetric matrices $A, B \in \mathbb{R}^{N\times N}$ with B being SPD, a target shift $\sigma \in \mathbb{R}$, the number of eigenpairs to be computed k, an initial guess of the eigenvectors $V_{(0)} \in \mathbb{R}^{N\times k}$, and a number of subspace iterations M

Output: (X, Λ), which approximates k eigenpairs of (A, B) in the spectral neighborhood of σ.

$V_{(0)} \leftarrow$ CholeskyQR$(V_{(0)}, B)$;
$(L, D) \leftarrow LDL^T$ factorization of $A - \sigma B$;
for m = 1,2, . . . , M **do**
  $V_{(m)} \leftarrow L^{-T}D^{-1}L^{-1}BV_{(m-1)}$;
  $V_{(m)} \leftarrow$ CholeskyQR$(V_{(m)}, B)$;
end
return $(X, \Lambda) \leftarrow$ RayleighRitz$(A, B, V_{(M)})$;

Once a Rayleigh-Ritz procedure has been performed, the convergence of the subspace iteration can be assessed by computing a set of residuals, $R_j = AX_j - BX_j\Lambda_j \in \mathbb{R}^{N\times k}$, and evaluating the 2-norm of each column of R_j. The tuple $(\sigma_j, \Lambda_j, X_j, R_j)$ will be referred to as the jth spectral probe throughout the remainder of this work and will be occasionally denoted SP(σ_j). We note for clarity that one need not consider both V_j and X_j simultaneously due to the fact that they admit identical linear spans (X_j is simply a rotation of V_j). As X_j contains more useful information related to the eigensystem of (A, B) than V_j, V_j is typically discarded in favor of X_j for eigenvalue calculations.

2.2 Validation of Spectral Slices

In the SISLICE method, the approximate eigenpairs associated with a particular slice (σ_j, σ_{j+1}) are obtained by analyzing the Ritz pairs that are computed from the spectral probes defined by the spectral shifts σ_j and σ_{j+1}, which we denote by SP(σ_j) and SP(σ_{j+1}), respectively. The Ritz values obtained from SP(σ_j) can potentially overlap with those obtained from SP(σ_{j+1}). It is also possible that σ_j and σ_{j+1} are too far apart such that a number of desired eigenvalues are not captured by either SP(σ_j) or SP(σ_{j+1}). Thus, a key aspect in the development of a robust spectrum slicing method is to provide a mechanism to select approximate eigenpairs within a spectral slice from the Ritz pairs of its associated probes as to avoid double counting and detect any missing or spurious eigenpairs, if present. Such selected eigenpairs will be referred to as being *validated*.

To select candidates for validation within the spectral slice (σ_j, σ_{j+1}), we examine the Ritz values computed from the spectral probes SP(σ_j) and SP(σ_{j+1}) that are within (σ_j, σ_{j+1}). We choose a point τ between σ_j and σ_{j+1}, e.g., $\tau = (\sigma_j + \sigma_{j+1})/2$ and select all Ritz values obtained from SP(σ_j) that

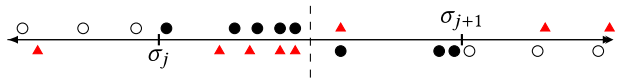


Fig. 2. Scheme for selection of validation candidates for the spectral slice bounded by (σ_j, σ_{j+1}). The points with markers above the axis are Ritz values computed from the SP(σ_j), and those below the axis are those computed from SP(σ_{j+1}). The vertical dashed line denotes the midpoint of the spectral slice (τ). The filled black circles represent the validation candidates for the slice, while the open circles represent Ritz values that may belong to other spectral slices depending on the placement of σ_{j-1} and σ_{j+2}. The red triangles represent Ritz values that are not considered for validation for the spectral slice.

are in (σ_j, τ), and those from SP(σ_{j+1}) that are in (τ, σ_{j+1}) as validation candidates. A graphical representation of this candidate selection process is depicted in Figure 2. For the spectral slices at both ends of the desired spectral region of interest, validation candidates are selected as those Ritz values that are in $[\lambda_{\min}, \sigma_1)$ and $(\sigma_{n_s+1}, \lambda_{\max}]$, respectively. This partitioning strategy follows from the assumption that Ritz values that approximate eigenvalues closer to a spectral shift tend to converge faster. That is, if θ_j and θ_{j+1} are both approximations to the same eigenvalue λ that lies in (σ_j, τ), but are obtained from two different spectral probes SP(σ_j) and SP(σ_{j+1}), the residual norm associated with θ_j is likely to be smaller, because λ is closer to σ_j than to σ_{j+1}. This feature is demonstrated numerically for each of the numerical experiments examined in Section 5. As a result, this partitioning strategy is responsible for removing the majority of eigenpair duplication between adjacent probes. We denote the number of validation candidates for the slice bounded by SP(σ_j) and SP(σ_{j+1}) as n_j^{cand} in the following.

The duplication of eigenvalues can also be checked by measuring the mutual orthogonality of the corresponding eigenvectors. However, such a scheme would require comparing Ritz vectors computed by different spectral probes. In a parallel implementation in which different spectral probes are mapped to different processor groups, this scheme would require excessive data communication. We choose to check duplication or missing eigenvalues by simply comparing the number of validation candidates with the exact eigenvalue count that can be obtained from the factorization $L_j D_j L_j^T = A^{(i)} - \sigma_j B$ for each spectral shift. By making use of Sylvester's inertia theorem [Sylvester 1852], we are able to ascertain the exact number of eigenvalues within the slice bounded by (σ_j, σ_{j+1}) by taking the difference between the number of negative diagonal elements of D_{j+1} and D_j, respectively. We denote this exact count as n_j^{exact}.

Given n_j^{exact} and n_j^{cand} for a particular spectral slice, there are three limiting cases that allow us to detect missing or spurious eigenpairs within that slice:

- If $n_j^{exact} = n_j^{cand}$, then we view each of the candidates as a reasonable approximation to a true eigenpair and consider it to be validated.
- If $n_j^{exact} > n_j^{cand}$, then there are true eigenpairs within this spectral slice that are not captured by either SP(σ_j) or SP(σ_{j+1}). Thus, either more shift-invert subspace iterations need to performed or a spectral shift must be added somewhere within (σ_j, σ_{j+1}) to ensure that all desired eigenpairs are accounted for. We examine the specifics of this shift addition in Section 3.3.
- If $n_j^{exact} < n_j^{cand}$, then a number of the validation candidates are either spurious eigenpairs or very poor duplicate approximations to eigenpairs that have true eigenvalues in another spectral slice. In either of these cases, the residual norm for these unwanted eigenpairs will be larger than the desired ones. As a result, they may be removed by selecting the n_j^{exact} eigenpairs in the slice from the bounding spectral probes with the smallest residual norms to be the validated eigenpairs.

3 SHIFT SELECTION AND MIGRATION

In this section, we examine the selection of the set of spectral shifts $\{\sigma_j\}_{j=1}^{n_s}$ which partition the spectral region of interest into $n_s + 1$ spectral slices. We first utilize a Lanczos approximation for the so-called density of states (DOS) to select an initial set of shifts for the first SCF iteration. In subsequent SCF iterations, we use a clustering algorithm to select shifts based on approximate eigenvalues obtained in the previous SCF iteration.

3.1 Shift Selection via Lanczos-based Density of States Estimation

Consider the eigenvalue decomposition of the pencil (A, B) in Equation (1). The exact density of states (DOS) for (A, B) is given by

$$\phi(\omega) = \sum_{j=1}^{N} \delta(\omega - \lambda_j), \tag{2}$$

where $\delta(\cdot)$ is the Dirac delta distribution. The cumulative density of states (CDOS) which returns the number of eigenvalues of (A, B) below a certain value ω is defined as

$$\Phi(\omega) \equiv \int_{-\infty}^{\omega} \phi(\omega')\mathrm{d}\omega'. \tag{3}$$

It follows from Equation (3) that $\gamma(a, b) = \Phi(b) - \Phi(a)$ provides an eigenvalue count for (A, B) on $[a, b]$. Clearly, the construction of the exact DOS and CDOS requires a full diagonalization of (A, B), which is something we are trying to avoid. One way to obtain an approximate DOS is to use the Lanczos algorithm as explained in Lin et al. [2016], Saad [2011], and Xi et al. [2018]. For a generalized eigenvalue problem, one can either run the Lanczos algorithm on $L^{-1}AL^{-T}$, where L is the Cholesky factor of B, i.e., $B = LL^T$, or run a B-orthonormal Lanczos on $B^{-1}A$. The latter requires the initial vector of the Lanczos iteration to be generated to have an appropriate random distribution. In this work, we select the elements of the initial starting guess using the normal distribution $\mathcal{N}(0, 1)$, though we note that more robust schemes exist for generalized eigenvalue problems [Xi et al. 2018].

Let $\{(\theta_j, g_j)\}_{j=1}^{k}$ be the Ritz pairs produced from a k-step Lanczos iteration associated with a fixed random starting guess. An estimated DOS may be written as

$$\phi(\omega) = \frac{N}{\sqrt{2\pi}} \sum_{j=1}^{k} \frac{\zeta_j^2}{\nu_j} \exp\left(-\kappa_j(\omega)^2\right), \qquad \zeta_j = e_1^T g_j, \quad \kappa_j(\omega) = \frac{\omega - \theta_j}{\nu_j\sqrt{2}}. \tag{4}$$

ν_j is a length parameter which determines the width of the Gaussian. For the purposes of this work, we choose ν_j so that each Gaussian factor in the sum nearly vanishes some distance d_j away from θ_j. The parameter d_j is chosen to be either the maximum or average of $\theta_j - \theta_{j-1}$ and $\theta_{j+1} - \theta_j$. Some safeguard is used to prevent d_j from becoming too small when eigenvalues are tightly clustered. We may obtain a closed form expression for the corresponding CDOS as

$$\Phi(\omega) = \frac{N}{2} \sum_{j=1}^{k} \zeta_j^2 \left[\mathrm{erf}(\kappa_j(\omega)) + 1\right], \tag{5}$$

where $\mathrm{erf}(\cdot)$ is the error function. To get a more accurate estimate of the DOS and CDOS, multiple Lanczos runs with different random starting guesses may need to be used. We refer readers to Lin et al. [2016] and Xi et al. [2018] for more details. The remainder of this subsection will be dedicated to how to partition a spectral region of interest into reasonable subintervals and select shifts for the spectral probes based on the DOS and CDOS of (A, B) produced by the Lanczos algorithm. As such, we expect that these methods are able to produce a specified number of intervals, n_s. Without loss

of generality, we consider an arbitrary spectral domain $[a, b]$ with approximate eigenvalue count $C(a, b) \equiv \lceil \gamma(a, b) \rceil$.

In the case where the CDOS increases gradually in a nearly continuous fashion (e.g., Figure 7), we may partition the spectral region of interest into $n_s = \frac{C(a,b)}{K}$ intervals containing roughly K eigenvalues by determining the roots of the CDOS at evenly spaced intervals,

$$\Phi(\omega) - (a + Kj) = 0, \qquad j \in [0, n_s]\,. \tag{6}$$

As $\Phi(\cdot)$ does not admit an analytic inverse, these roots may be found either through bisection or Newton's method. Letting l_j and u_j be the $(j-1)$-st and jth root of Equation (6), the spectral intervals produced by this method may be given as $I = \{(l_j, u_j)\}_{j=1}^{n_s}$. This simple strategy is guaranteed to produce exactly a specified number of intervals but is sub-optimal when the spectral region of interest is highly clustered with large gaps between clusters. These clusters may be identified by sharp peaks in the DOS (e.g., Figure 6). The previously described method will often miss these isolated clusters due to the sampling nature of the methods used to solve Equation (6). In this scenario, we introduce an alternative method that aims to find these spectral clusters via identification of the local maximizers of the DOS itself.

ALGORITHM 3: Obtain intervals containing eigenvalue clusters: DOSCluster$(T, [a, b], n_\omega)$

Input: Ritz pairs $T = \{(\theta_j), g_j\}_{j=1}^{k}$ from a k-step Lanczos iteration,
Search interval $[a, b]$ and number of discretization points n_ω.
Output: Eigenvalue cluster intervals I.

1 Let $S = \{\omega_i\}_{i=1}^{n_\omega}$ with $\omega_i = a + \dfrac{b-a}{n_\omega - 1}(i-1)$;

2 Identify all local maximizers $\{\hat{\omega}_j\}_{j=1}^{n_c}$ of $\phi(\cdot)$ restricted to S;

3 **if** $n_c = 1$ **then return** $I = \{(a, b)\}$;

4 Identify local minimizers between maximizers, $\{\mu_j\}_{j=0}^{n_c}$ with $\mu_0 = a$, $\mu_{n_c} = b$ and

$$\mu_j = \underset{\omega \in (\hat{\omega}_j, \hat{\omega}_{j+1}) \subset S}{\arg\min} \phi(\omega), \quad j \in [1, n_c);$$

5 Let $I = \{(\mu_{j-1}, \mu_j)\}_{j=1}^{n_c}$;

6 Remove intervals in I that do not contain a Ritz value;

return I;

Algorithm 3 depicts the method used to partition the DOS into intervals containing eigenvalue clusters. However, due to the highly non-linear nature of the Gaussian approximation in Equation (4) and the possibility of spurious Ritz pairs in the k-step Lanczos procedure, care must be taken in identification of the local maximizers. In particular, local maximizers must be identified using an appropriate spectral resolution. Evaluating DOS on a very fine spectral grid may produce too many "artificial" local maximizers that are introduced by the inexact nature of the estimated DOS. Evaluating DOS on a very coarse grid may result in missing an important local maximizer (hence, a cluster, see, e.g., Figures 3(a) and 3(b)). To complicate matters further, there is no simple way of determining the "right" discretization *a priori* to overcome these issues. To address this, we have adopted a refinement strategy to minimize the chance of missing important spectral information contained in the DOS approximation:

(1) Obtain an initial set of cluster intervals (I) from Algorithm 3 on a coarse discretization of $[a, b]$. We have found that $n_\omega = 10k$ suffices for this purpose in most cases.

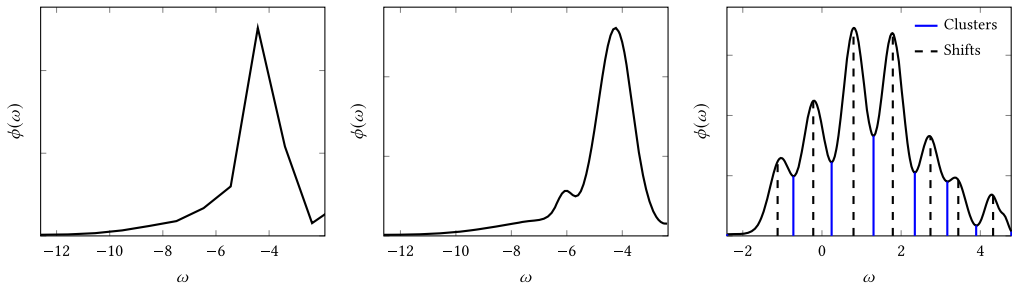


Fig. 3. Refining the resolution of the DOS (a) in the third cluster shown in Figure 6(a) reveals an additional cluster near −6.0 (b). The refinement also allows us to set tighter bounds for refined clusters. (c) depicts the interval definitions and shift placement for the upper end of Figure 6(a).

(2) If $\gamma(l_j, u_j)$ for some $(l_j, u_j) \in I$ is less than some specified tolerance, e.g., 2, then merge this interval into the adjacent interval with the fewest estimated eigenvalues if it exists (i.e., $(l_{j\pm1}, u_{j\pm1})$ if $u_{j-1} = l_j$ or $u_j = l_{j+1}$, respectively). If there is no adjacent interval (i.e., the interval represents a cluster isolated from the rest of the spectrum), then the interval must remain unaltered to ensure that it is captured in the shift-invert subspace iteration.
(3) Rather than refine the search for clusters across all of the intervals in I, it is best to limit this refined search only to intervals with large eigenvalue counts. Thus if $\gamma(l_j, u_j)$ is larger than some specified tolerance, e.g., 50, we rerun Algorithm 3 with (l_j, u_j) as the search interval with a finer discretization. We have found that doubling the resolution within the spectral interval, i.e., $n'_\omega = \frac{2C(l_j,u_j)}{C(a,b)} n_\omega$, suffices in most cases.
(4) If the refined search produces two or more intervals, then these intervals replace (l_j, u_j) in I. Steps (2) and (3) are then repeated until no new intervals are produced.

Once the set of spectral intervals have been produced by either of the aforementioned schemes, a shift may be selected for each interval by one of several schemes. The simplest scheme takes the shift to be the midpoint of the interval, e.g., $\sigma_j = \frac{u_j - l_j}{2}$. However, in cases when the eigenvalues are irregularly distributed throughout the interval, we may take the expected value of ω with respect to the DOS over the interval to be the shift,

$$\sigma_j = \frac{1}{\gamma(l_j, u_j)} \int_{l_j}^{u_j} \omega\phi(\omega)\,\mathrm{d}\omega = \frac{N}{\Phi(u_j) - \Phi(l_j)} \sum_{i=1}^{k} \zeta_i^2 \left(\psi_i(u_j) - \psi_i(l_j)\right), \tag{7}$$

where

$$\psi_i(\omega) = \frac{\theta_i}{2} \mathrm{erf}\left(\kappa_i(\omega)\right) - \frac{\nu_i}{\sqrt{2\pi}} \exp\left(-\kappa_i(\omega)^2\right). \tag{8}$$

An example of this shift selection scheme combined with the identification of clusters though iterative refinement of the DOS is shown in Figure 3(c).

Unlike the CDOS root finding method, Algorithm 3 is not guaranteed to produce a specified number of intervals even after refinement. In fact, this method produces something akin to the natural clustering of the eigenvalue distribution. A number of safeguards have to be put in place to ensure that we are able to produce exactly a specified number of intervals

- If $|I| < n_s$, then we subdivide the intervals with the largest number of approximate eigenvalues. This process is repeated until $|I| = n_s$.

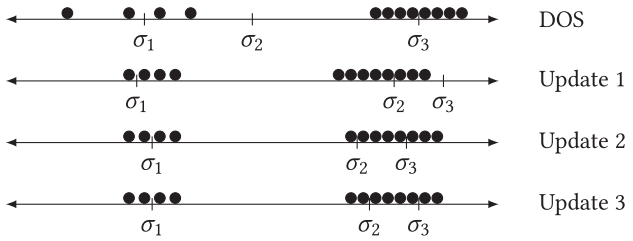


Fig. 4. A graphical representation of the shift migration process throughout the SCF procedure. The SCF iterations progress from top to bottom with the filled circles representing the validated Ritz values at that iteration. σ_1, σ_2, σ_3 represent the spectral shifts used to obtain the Ritz values at each iteration. The shifts for the first SCF iteration are representative of a typical DOS-shift selection scheme where shifts are chosen both in regions without eigenvalues as well as regions with a disproportionately large number of eigenvalues due to inaccuracy in the DOS approximation. At each subsequent SCF iteration, new shifts are chosen via k-means clustering of the Ritz pairs obtained from the previous iteration.

- $|I| > n_s$, we merge intervals with only a few number of eigenvalues into adjacent intervals until $|I| = n_s$. Care must be taken in this procedure as merging well separated clusters (even if adjacent) will likely lead to poor convergence in the shift-invert subspace iterations.

3.2 Shift Refinement and Eigenvalue Clustering

Due to the limited number of Ritz values that can be extracted from the Lanczos method in the spectral interior, it is possible that the initial selection of spectral shifts produced by a Lanczos DOS estimation procedure is far from optimal. In particular, shifts may be placed in spectral regions devoid of eigenvalues. Another possible scenario is that an insufficient number of shifts are placed in regions that contain a disproportionately large number of eigenvalues. An illustration of this issue is given on the axis labeled "DOS" in Figure 4. However, shift misplacement can be incrementally corrected in subsequent SCF iterations by using a clustering algorithm to partition previously computed eigenvalue approximations and refine the shift selection.

For each SCF iteration, we obtain a set of eigenpair approximations for $(A^{(i)}, B)$. Thus, for $i > 0$, we have available to us a set of approximate eigenvalues for $(A^{(i-1)}, B)$. In the early SCF iterations, when the change in eigensystem between two subsequent iterations is relatively large, it is possible that the shifts selected for $(A^{(i-1)}, B)$ would not be appropriate for the slicing of the spectrum for $(A^{(i)}, B)$. We discuss a strategy to determine this (dis)similarity and strategies for subsequent shift selection in Section 3.3. However, if the eigenvalues of $(A^{(i)}, B)$ are sufficiently close to those of $(A^{(i-1)}, B)$, then we may use the approximated eigenvalues of $(A^{(i-1)}, B)$ as a reference to determine the spectral shift placement for the spectrum slicing of $(A^{(i)}, B)$. Due to the localized nature of the shift-invert spectral transformation, rapid convergence of the shift-invert subspace iteration is achieved when spectral shifts are placed centrally in clusters of eigenvalues. Thus, we may determine more optimal shift placement by identifying spectral clusters from the computed eigenvalues $(A^{(i-1)}, B)$ and placing shifts in the centroids of these clusters for determination of the eigenpairs of $(A^{(i)}, B)$.

To identify spectral clusters, we employ the k-means clustering algorithm [Lloyd 1982]. At the ith SCF iteration for $i > 0$, we use k-means to identify n_s clusters from the validated eigenpairs obtained from the $(i-1)$-st iteration. The centroids of the clusters may then be used in the generation of the ith set of spectral shifts. As the SCF procedure converges, the centroids of the clusters will also converge to a particular set of spectral shifts. An illustration of this convergence behavior is given in Figure 4.

Although the k-means clustering problem is generally NP-hard, we do not necessarily need to obtain a globally optimal solution to the clustering problem to identify appropriate spectral shifts. Our objectives are to identify eigenvalue clusters and to partition nearly uniformly distributed eigenvalues into slices of roughly equal size. In general, determination of n_s clusters is a drastic over clustering of the Ritz values. However, k-means clustering usually results in equal sized clusters, even in the case of over clustering. Due to the fact that n_s is relatively small, obtaining clusters from these data using k-means may be achieved with negligible cost.

The k-means algorithm is an iterative procedure initialized with a set of guesses to cluster centroids. The choice of these initial guesses can have a significant effect on the convergence of the algorithm and the quality of cluster centroids it produces. In the SISLICE method, these guesses are usually taken to be the spectral shifts used in the previous SCF iteration. However, if the previous spectral shifts are generated from the DOS shift selection strategy, it is possible that the k-means algorithm can converge to a sub-optimal solution if the centroids are initialized with these shifts. To address this issue, we employ the k-means++ [Arthur and Vassilvitskii 2007] cluster initialization strategy to improve initial guesses of the centroids prior to the k-means clustering process. Rather than select all guess centroids at random with uniform probability for all eigenvalues, k-means selects the first centroid with a uniform probability and then selects all subsequent centroids from the remaining eigenvalues according to a probability distribution that is quadratic in the distance of each eigenvalue to its nearest existing guess centroid. This process is repeated until n_s centroids have been selected. The result is a set of guess centroids that are well separated and are selected such that there is a provable upper bound to the k-means objective function. For more details, we refer the reader to Arthur and Vassilvitskii [2007].

Assuming that the eigenvalue distribution does not change drastically throughout the SCF procedure, k-means also allows the SISLICE method to track eigenvalue changes and migrate shifts between SCF iterations as shown in the axes labeled "Update" in Figure 4. Until convergence is reached, shift migration is performed by k-means using the validated eigenpairs of the previous SCF iteration as the initial guess centroids. As k-means converges to the local optimum nearest to the initial guess, this choice of guess allows for the k-means shift migration strategy to converge to a single set of shifts as the SCF converges. It is important that the clustering is performed on validated eigenpairs to avoid oversampling of spectral regions for which the Ritz pairs of adjacent spectral probes overlap or contain spurious Ritz values. We demonstrate the efficacy of this migration scheme in Section 5.

Once a clustering of the validated Ritz values has been obtained, we may use the centroids of those clusters to generate the set of spectral shifts for the next SCF iteration. Instead of creating new spectral probes, we would like to reuse the Ritz vectors produced by the existing spectral probes as the initial guesses to the desired eigenvectors in the subsequent shift-invert subspace iteration to improve convergence. Therefore, in the SISLICE method, we update the shifts of the existing spectral probes based on the clustering information rather than starting completely from scratch. To update the shift for each spectral probe, we form a mapping between the eigenvalue clusters and spectral probes such that each cluster is mapped to the probe with which it has maximal overlap.

In the case that this map is bijective, the spectral shift associated with the spectral probe is taken to be the centroid of its associated cluster. However, it is possible, especially in the early SCF iterations when the eigenspectrum undergoes considerable change, that this map is not bijective. As a result, there is some ambiguity as to how to best update the probes that have no preimage under this mapping.

The case of non-bijective maps between clusters and spectral probes is typically a symptom of poor shift selection in the previous SCF iteration resulting in some spectral probes picking up

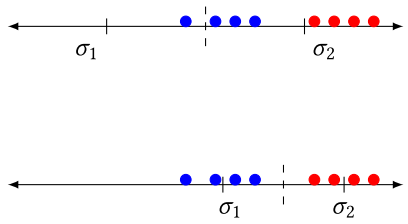


Fig. 5. A schematic illustration of how multiple clusters may be mapped to the same probe, and how an old probe may be deleted and a new probe can be inserted. The blue and red dots are two clusters of approximate eigenvalues that are both mapped to the same spectral probe centered at σ_2. After the mapping between clusters and the previous spectral probes is established, the probe centered at σ_1 is deleted, because no cluster is mapped to it. A new problem centered at a new shift σ_1 is inserted, and σ_2 is also moved to the right.

only a few validated Ritz values while others capturing a disproportionate number of validated Ritz values. As the validated Ritz values are separated by the k-means algorithm into different clusters, validated Ritz values retrieved from one spectral probe may be separated into several clusters resulting in several clusters being mapped to the same spectral probe (see the schematic illustration in Figure 5). In the meantime, the few validated Ritz values obtained by a poorly placed spectral probe SP(σ_j) may be placed into a cluster that gets mapped to a different spectral probe SP(σ_j), leaving SP(σ_j) without any cluster to map to.

When several clusters are mapped to the same spectral probe, we merge these clusters into a single cluster. The resulting centroid of the merged Ritz values is taken to be the spectral shift of the associated spectral probe. In the case that the mapped cluster contains too many eigenvalues, a probe may be inserted to ensure proper load balance.

If a spectral probe is not mapped to any cluster, then it is simply deleted. However, since the number of spectral shifts (and thus the number of spectral probes) is fixed throughout the SCF procedure, when a probe is removed, another probe must be added to maintain this fixed number of shifts. We choose to add a probe to a cluster that contains the largest number of Ritz values. To add such a spectral probe to such a cluster, we need to break up the cluster first into two clusters. In practice, this may be achieved by performing a 2-means clustering of the Ritz pairs in the largest cluster. One of the clusters is mapped to the original spectral probe mapped to the cluster before it was broken up. The shift associated with that cluster is replaced with the centroid of the new smaller cluster. The other cluster is mapped to the newly added spectral probe. In addition to setting the shift of the probe to the centroid of the new cluster, we also need to copy (or send in a distributed memory implementation) the Ritz vectors associated with the Ritz values in this cluster to the added spectral probe. This process of breaking up a large cluster and adding a new spectral probe is repeated until the desired number of spectral shifts and probes is obtained. The specifics regarding probe insertion in a distributed infrastructure are discussed Section 4.

The SIESTA-SIPs method of [Keçeli et al. 2018] adopts a similar k-means strategy for tracking the eigenvalue migration throughout the SCF procedure to avoid costly methods to perform shift selection every SCF iteration. The SIESTA-SIPs method is not as sensitive to changes in the definitions of spectral slice intervals between SCF iterations as long as they are distributed such to allow for similar convergence rates between slices due to their use of SI-Lanczos. This also inhibits their ability to leverage the knowledge of the convergent nature of the eigenvectors to accelerate their eigenpair convergence as the SCF iterations progress. This leads to two main differences in the utilization of k-means between SISLICE and SIESTA-SIPs:

(1) Whereas SISLICE aims to minimize the change in centroid location between SCF iterations by initializing the centroids as the shifts from the previous SCF iteration, SIESTA-SIPs obtains a new and possibly different set of centroids every iteration based on the convergence of k-means using a uniform initial guess.
(2) The fact that they do no utilize previous eigenvector information means that they need not map the newly obtained cluster centroids to existing sets of spectral probes as this information is not generally passed between SCF iterations in their prescription.

3.3 Missing Eigenvalues

Because $A^{(i)}$ can change significantly from $A^{(i-1)}$ in early SCF iterations, a shift selection scheme based on the clustering of approximate eigenvalues of $A^{(i-1)}$ may not be optimal for computing eigenpairs of $A^{(i)}$. In particular, it is possible that spectral probes constructed from the sub-optimal selection of target shifts miss some eigenvalues. The spectral slices in which these missing eigenvalues reside can be identified in the validation process described in Section 2.2.

When missing eigenvalues are detected, we perform a new DOS estimation on $(A^{(i)}, B)$ with an appropriate resolution to place new shifts in spectral slices that contain missing eigenvalues. New spectral probes are created to recompute approximate eigenvalues within these newly created spectral slices. This is a costly step, because the next SCF cycle cannot start until all missing eigenvalues are accounted for. An example of this state of affairs is demonstrated in Section 5.3.

To reduce the likelihood of missing eigenvalues resulting from shift misplacement, we can monitor the convergence of SCF for drastic changes in the spectrum by comparing the partial traces of the system matrices within the subspace spanned by the previously validated eigenvectors. For example, given the metric

$$\eta(V, A) = \mathrm{Tr}\left(V^T A V\right), \tag{9}$$

if the difference between $\eta(X^{(i-1)}, A^{(i-1)})$ and $\eta(X^{(i-1)}, A^{(i)})$ is larger than some specified threshold, then the spectra of the matrices may be deemed to be sufficiently dissimilar. If this is found to be the case, then it would be beneficial to use the DOS shift selection strategy discussed in Section 3.1. Although this strategy does not completely eliminate the possibility of missing eigenvalues (because individual eigenvalues can move around without affecting the trace of the $A^{(i)}$), it may help reduce that possibility and the cost associated with generating new probes to seek the missing eigenvalues.

4 PARALLEL IMPLEMENTATION

The algorithmic subtasks described in the previous sections have been constructed in such a way as to allow for maximal concurrency in the slicing of the spectral region of interest: Each of the spectral probes may be constructed independently of any other spectral probe. As the construction of the spectral probes through shift-invert subspace iterations constitutes the bulk of the work in the SISLICE method, this task independence should lead to scalable performance. The slice validation scheme outlined in Section 2.2 would require some level of synchronization between independent computing units. Further, the shift insertion and deletion schemes outlined in Section 3.2 would require some data to be copied from some processors/nodes to others. However, these communication and synchronization overheads are generally small as we will see in the next section. In this section, we outline the salient aspects of the parallel implementation of the SISLICE method.

We note that the parallelization discussed here focuses exclusively on the parallel execution of spectral probes. An additional level of finer grain parallelism exists within each spectral probe. If matrices A and B can be replicated and stored on each single many-core compute node, then a hybrid-parallelism scheme utilizing both shared-memory and message passing parallelism may

be achieved through exploitation of optimized implementations of threaded BLAS and LAPACK (such as those found in Intel(R) MKL, IBM(R) ESSL, OpenBLAS, BLIS, ATLAS, cuBLAS, etc.) within a particular MPI rank. While we do not treat this level of parallelism explicitly in this section, its leverage is trivial on modern computing architectures and is implied for the numerical experiments in Section 5. If A and B are too large to be stored on a single compute node, then the factorizations and linear system solves required for each probe may be performed using ScaLAPACK in the case of dense matrices, or a distributed sparse solver such as symPACK, MUMPS, or PARDISO in the case of sparse matrices. We should note that the parallel scalability of LDL^T factorization and back substitutions for solving triangular systems with multiple right hand sides is generally much better than what can be achieved in a dense eigensolver. These distributed calculations for each spectral probe may take place on a subset of the total number of MPI ranks, allowing leverage of massive parallelism on large computing clusters. We do not treat this particular parallelism scheme in this work, but it has been discussed at length in other related work [Keçeli et al. 2016, 2018; Zhang et al. 2007].

4.1 Spectral Probe Distribution and Synchronization

The SISLICE method is designed for taking advantage of computer systems that have a large amount of computational resources in terms of compute nodes and cores within each node. In an ideal scenario, the number of spectral probes should match the number of nodes (or groups of nodes) so that all probes can be executed simultaneously. The optimal number of nodes corresponding to the natural clustering of the eigenvalues could be determined by the DOS-based shift selection in Section 3.1 and queried prior to running the SISLICE solver. However, we note that this is not a requirement of the implementation discussed in this work.

When the number of computational nodes is less than the number of spectral probes, a round-robin distribution of probes to nodes can be used, i.e., $\mathrm{SP}(\sigma_j)$ may be mapped to the $(j \bmod n_r)$th MPI rank, where n_r is the number of MPI ranks. In this case, the computation is not load balanced if n_r does not divide n_s.

Once the spectral probes have been constructed, each $\mathrm{SP}(\sigma_j)$ contains a set of Ritz pairs that approximate the eigenpairs in the neighborhood of σ_j. However, the slice validation scheme described in Section 2.2 requires knowledge of Ritz pair information from adjacent probes. If each of the adjacent spectral probes has been constructed on a different MPI rank (or group of MPI ranks), then the validation scheme requires some level of communication / synchronization of Ritz pair information between the MPI ranks. However, the validation scheme only requires knowledge of the Ritz values and associated residual norms to validate the spectral slices. The Ritz vectors are not explicitly required.

If the Ritz values and residual norms were only to be used in the slice validation scheme, then their synchronization could be further limited to only the neighboring ranks of the owner of a particular spectral probe. However, because the entire set of validated Ritz values is used in updating spectral shifts through k-means clustering (as described in Section 3.2), it is useful to synchronize this information across all of the MPI ranks. While a distributed implementation of k-means clustering is possible, the fact that each spectral probe only accounts for a relatively small number of validated Ritz pairs would require excessive communication to perform the clustering. Because the storage requirement of the Ritz values and residual norms is negligible relative to the Ritz vectors, this synchronization scheme poses no storage overhead. In the SISLICE method, the synchronization of Ritz values and residual norms for each spectral probe may be implemented as `MPI_Allgather`, and poses only minimal communication overhead relative to the computational cost of the shift-invert subspace iterations. As the Ritz values and residual norms are replicated across each MPI rank, the tasks of slice validation and shift updates may also be replicated to avoid

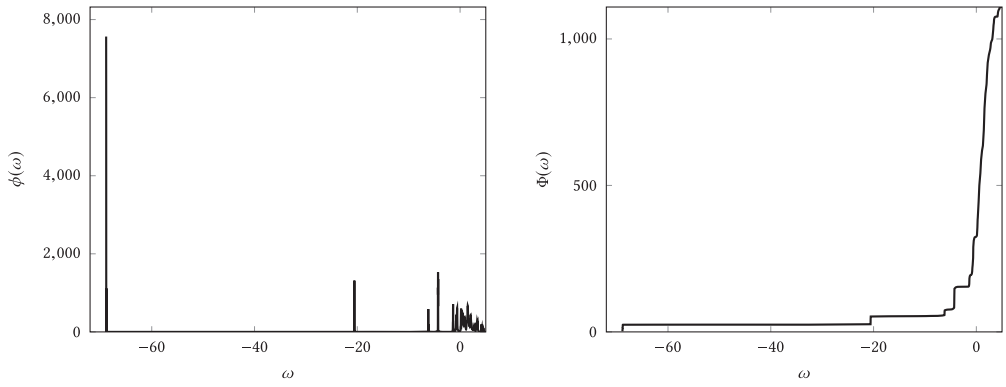
communication. We note for clarity that in the case of random initialization of the clustering problem through, e.g., k-means++, the Ritz value clustering may still be replicated through the use of pseudo random number generation using the same seed value. The scalability of this distribution and synchronization scheme is demonstrated in Section 5.5.

4.2 Spectral Probe Insertion and Removal

As was discussed in Section 3.2, occasionally shift selection and migration schemes employed by the SISLICE method yield sub-optimal shift placement leading to redundant spectral probes and probes that are responsible for a disproportionate number of validated eigenpairs. This is typically the case in the early SCF iterations due to the crude DOS approximation by the Lanczos procedure described in Section 3.1. The presence of redundancies in the spectral probes leads to a load imbalance that should be avoided to ensure scalability on large, distributed computing systems. For the purposes of this section, the term "load balance" should be thought of as balance of *useful* work. Technically speaking, even in the case of redundancies in the spectral probes, the computational work performed for each spectral probe will always be roughly the same given that the number of subspace iterations performed and subspace dimensions are uniform across all probes. Thus this work is always "balanced." However, we want to ensure that each rank is performing a roughly equal amount of useful work (in the sense of yielding a roughly equal number of validated Ritz pairs) rather than wasting valuable computational resources in spectral regions where it is not needed.

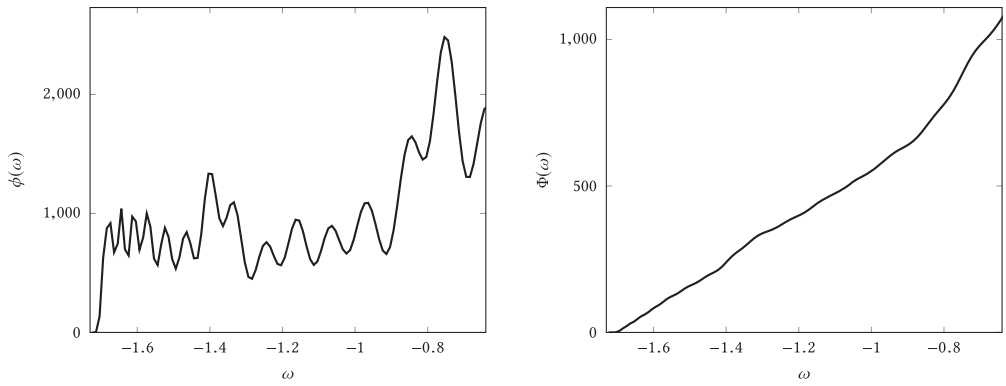
As the number of spectral probes is fixed in the SISLICE method, removal of a spectral probe necessitates the insertion of a spectral probe to balance the work in another spectral region. This probe removal necessarily leads to a load imbalance if the work was balanced in the previous SCF iteration. As discussed in Section 3.2, probes are inserted to break up large clusters of validated Ritz values such that they effectively span multiple probes after the subsequent shift-invert subspace iterations. In a distributed computing environment, care must be taken to ensure probe insertion is performed in such a way as to balance the work between independent computing ranks while avoiding a large communication overhead. As the k-means clustering is replicated on each rank, the decision of where to insert new spectral probes may also be effectively replicated with only minimal communication. For each probe to be inserted, the determination of the two new spectral shifts is replicated on each rank. The new probe that is to be inserted is assigned to the rank with the least amount of work (thus ensuring load balance). Once this decision has been made, the probe whose shift has been moved through this procedure communicates its Ritz vector data to the newly inserted probe to allow its reuse in the subsequent SCF iteration. The cost of this point-to-point communication is relatively small in practice and may be overlapped with the determination and communication of other probe insertions.

In the case when shifts must be inserted due to missing eigenvalues within a particular spectral slice as described in Section 3.3, we may leverage the fact that the computation is done in parallel to our advantage. It may be the case that the DOS shift insertion strategy yielded several shifts in the spectral region that contains the missing eigenvalues. Rather than have processors or processor groups sit idle while the missing eigenpairs are obtained sequentially, the newly inserted probes may be distributed in the same manner as the initial probe distribution. Due to the fact that missing eigenvalues are typically a symptom of poorly placed shifts, not of too few shifts, inserting probes will not yield more useful probes than n_s, i.e., if a probe had to be inserted to resolve missing eigenpairs, then it is typically the case that some probes did not produce validated eigenpairs. However, even if each of the probes from the first round of subspace iterations produced validated eigenpairs, the mapping scheme between eigenvalue clusters and spectral probes will preclude the possibility of yielding more than n_s probes for the subsequent SCF iteration. This is due to the fact



(a) (b)

Fig. 6. Lanczos DOS (a) and CDOS (b) for the entire spectrum of Silane ($N = 1{,}109$). Silane exhibits a number of isolated eigenvalue clusters (spikes in the DOS) lower in the spectrum and a more uniform distribution at larger eigenvalues. The DOS and CDOS calculations were performed using 100 Lanczos iterations with the converged matrix pencil.



(a) (b)

Fig. 7. Lanczos DOS (a) and CDOS (b) for the lowest 1,000 eigenvalues of Graphene ($N = 9{,}360$). Graphene exhibits a nearly uniform eigenvalue distribution throughout its entire spectrum. The DOS and CDOS calculations were performed using 100 Lanczos iterations with the converged matrix pencil.

that the SISLICE method obtains n_s clusters regardless of the number of probes used to produce the validated eigenpairs in the previous SCF iteration. Thus, even in the case of probe insertion in the previous SCF iteration, the SISLICE method ensures load balance is maintained in subsequent SCF iterations.

5 NUMERICAL EXPERIMENTS

In this section, we report a set of numerical experiments that demonstrate the effectiveness of the proposed shift selection technique for computing all or a subset of eigenvalues of a matrix pencil or a sequence of matrix pencils. We examine two limiting cases of eigenvalue distribution shown in Figures 6 and 7.

The Silane test case (Figure 6, $N = 1{,}109$) is an all-electron density functional theory calculation using a Gaussian basis set. Its spectrum exhibits a number of isolated eigenvalue clusters at lower eigenvalues and a more uniform distribution at larger eigenvalues. The isolated eigenvalue clusters at low eigenvalues are a common feature in all-electron density functional calculations. All matrices related to the Silane test case in this work were obtained using the NWChemEx software package [Kowalski et al. n.d.].

The Graphene test case (Figure 7, $N = 9{,}360$) is a density functional theory calculation using pseudo-potentials for the core electrons. As such, its spectrum does not contain isolated clusters. The more uniform nature of the spectrum is a common feature in nearly all pseudo-potential-based density functional calculations. All matrices related to the Graphene test case in this work were obtained using the SIESTA software package [Soler et al. 2002].

5.1 Shift Selection for a Fixed Matrix Pencil

To demonstrate how the shift selection strategy enables rapid convergence of the SISLICE method as the SCF procedure approaches convergence, i.e., when the matrix pencils change very little between SCF iterations, we examine the convergence of eigenpairs for fixed matrix pencils in this section. This experiment allows us to gauge the effectiveness of the shift selection strategy when more accurate estimation of the desired eigenvalues becomes available in successive SCF iterations.

To simplify our exposition, we examine a set of representative spectral windows for the aforementioned test cases using the converged matrix pencil (A, B) as the representative eigenvalue problem. Even though the matrix pencil does not change, an artificial SCF procedure is carried out and a new set of shifts may be chosen after a fixed number of subspace iterations have been performed. This procedure can be viewed as a generalized (block) Rayleigh quotient iteration.

All calculations in this section were performed using a probe basis dimension of $k = 100$ and 4 shift-invert subspace iterations per SCF iteration. The SCF iteration is considered converged if the maximum of the residual norms associated with all validated approximate eigenpairs is below the threshold of 10^{-13}. In both of the presented cases, we observe the expected monotonic convergence of the eigenvalues within individual spectral slices once shift migration has been performed.

Silane. For the case of Silane, SISLICE was applied to perform a full diagonalization using 100 shifts so that $n_e/n_s \approx 11$. We examine two representative spectral windows for this test case, $C_1 = [-20.59, -20.55]$ (Figure 8) and $C_2 = [-0.9, -0.39]$ (Figure 9). The C_1 window represents a dense, isolated cluster of eigenvalues, while the eigenvalues in C_2 are embedded in a dense region of eigenvalues. Due to the different distribution characteristics of these two spectral windows, the convergence behavior of the eigenpairs within these windows are different. However, because these windows are not treated separately in the sense of the larger eigenvalue calculation, SCF iterations are performed until convergence is reached across the spectrum. Further, in this test case, DOS-based shift selection yielded 25 useless probes that were not well placed, i.e., probes that did not produce any validated eigenvalues after the validation scheme outlined in Section 2.2 was applied. These probes were redistributed in the subsequent iterations via the method outlined in Section 3.2.

Because eigenvalues in C_1 are well separated from the rest of the spectrum, the convergence of the subspace iteration is rapid. Using the DOS-based shift partitioning, a single shift is placed just below $\lambda = -20.57$ to account for the 37 eigenvalues in the immediate vicinity. In the first SCF iteration, the eigenvalues near the selected shift converged much more rapidly than those further away. After the first SCF iteration, k-means eigenvalue clustering yielded 3 clusters of ~12 eigenvalues with centroids shown in Figure 8(a). Convergence for this spectral window is achieved within two SCF iterations both with and without the k-means shift update, with all eigenvalues

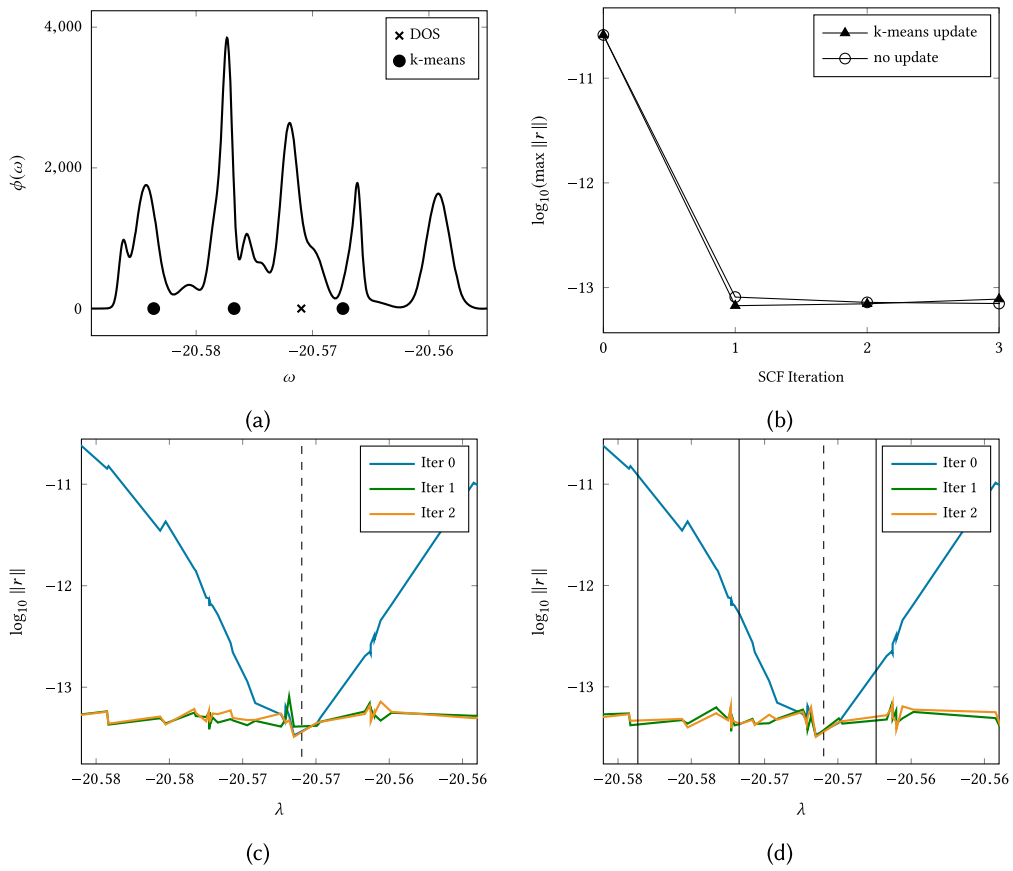


Fig. 8. Isolated cluster of Silane eigenvalues $[-20.59, -20.55]$ (37 eigenvalues). (a) Initial Lanczos DOS along with DOS shift placement and k-means update. (b) Convergence behavior of the largest residual norm in the spectral window both with and without the k-means shift update. Overall residual convergence for the SISLICE method within the spectral window with (d) and without (c) k-means shift update. Converges in two SCF iterations both with and without k-means shift update.

converging at roughly the same rate notwithstanding their distance to the nearest shift. We can also see that for the case of this isolated cluster, k-means clustering yielded no noticeable effects on residual convergence.

In contrast, the convergence of approximate eigenvalues in C_2 is less rapid due to the fact that there exist eigenvalues both immediately below and above the eigenvalues in this spectral window. DOS-based shift selection and spectrum partitioning placed 8 evenly spaced shifts to account for the 141 eigenvalues within this window. After the first SCF iteration, k-means clustering revealed a non-uniform distribution of eigenvalues within this window, yielding 12 clusters of ∼11 eigenvalues. Convergence rates for the eigenvalues in this window vary considerably based on their distance to their nearest shift. Convergence across the entire spectral window is achieved within four SCF iterations with the k-means update and six SCF iterations without the update. Thus, for this cluster, the k-means shift update yielded a discernible improvement in the residual convergence of the approximate eigenpairs.

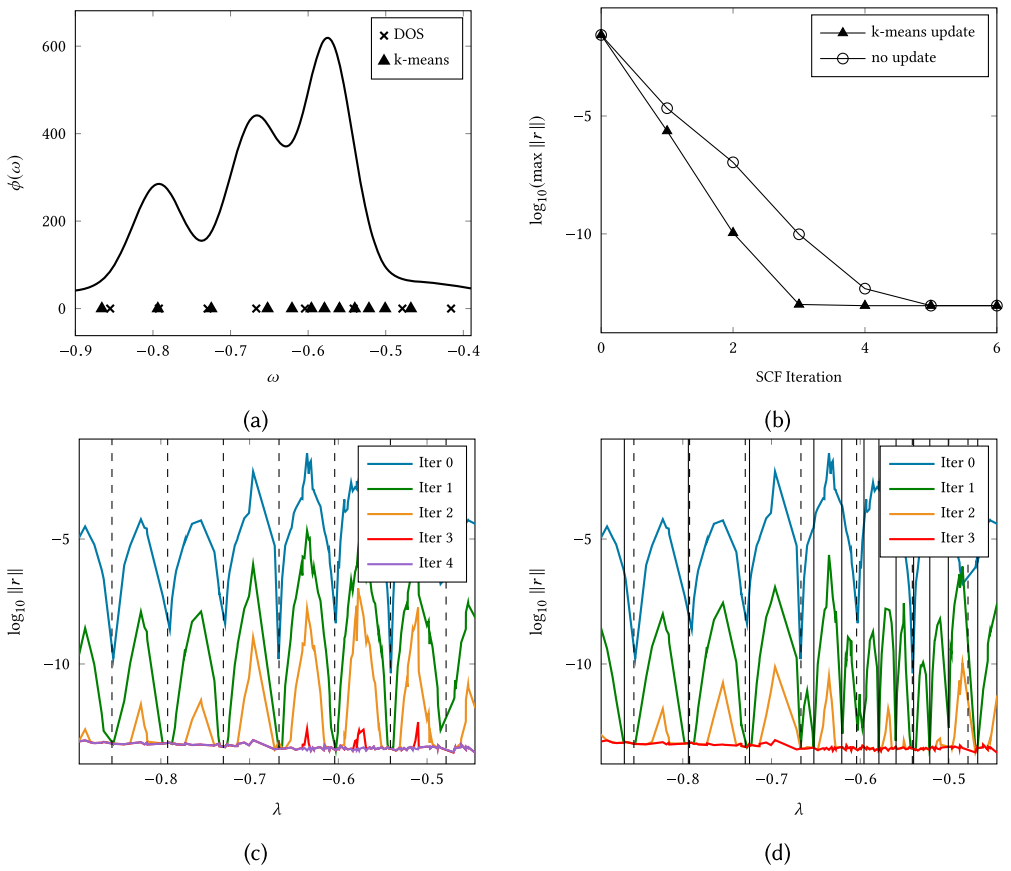


Fig. 9. Embedded cluster of Silane eigenvalues $[-0.9, -0.39]$ (141 eigenvalues) (a) Initial Lanczos DOS along with DOS shift placement and k-means update. (b) Convergence behavior of the largest residual norm in the spectral window both with and without the k-means shift update. Overall residual convergence for the SISLICE method within the spectral window with (d) and without (c) k-means shift update. Converges in four SCF iterations with k-means shift update and six iterations without shift update.

Graphene. For the case of Graphene, SISLICE was applied to perform a partial diagonalization of the lowest 1,000 eigenvalues using 100 shifts to obtain $n_e/n_s \approx 10$. As can be seen in Figure 7(a), the eigenvalue distribution for Graphene is approximately uniform. As such, the DOS-based shift selection produced uniformly distributed shifts along the entire spectral window, yielding no useless probes. We examine the eigenvalue interval $C = [-1.4, -1.3]$ (Figure 10) as a representative example of the convergence behavior for this test case.

Within C, the Graphene test case admits 93 eigenvalues in a roughly uniform distribution. As such, the DOS-based shift partitioning places eight evenly spaced shifts in this spectral window so that $n_e/n_s \approx 11$. Within each spectral probe, convergence is more rapid near the shifts than further away. The k-means shift update simply migrates the shifts without any appreciable changes to the shift spacing, i.e., the k-means result yields eight clusters of ~11 Ritz values with centroids of roughly equal spacing. Convergence for this spectral window is achieved within three SCF iterations both with and without the k-means shift update. We note that in the second SCF iteration, the convergence is slightly worse with the k-means selected shift than the DOS selected shifts.

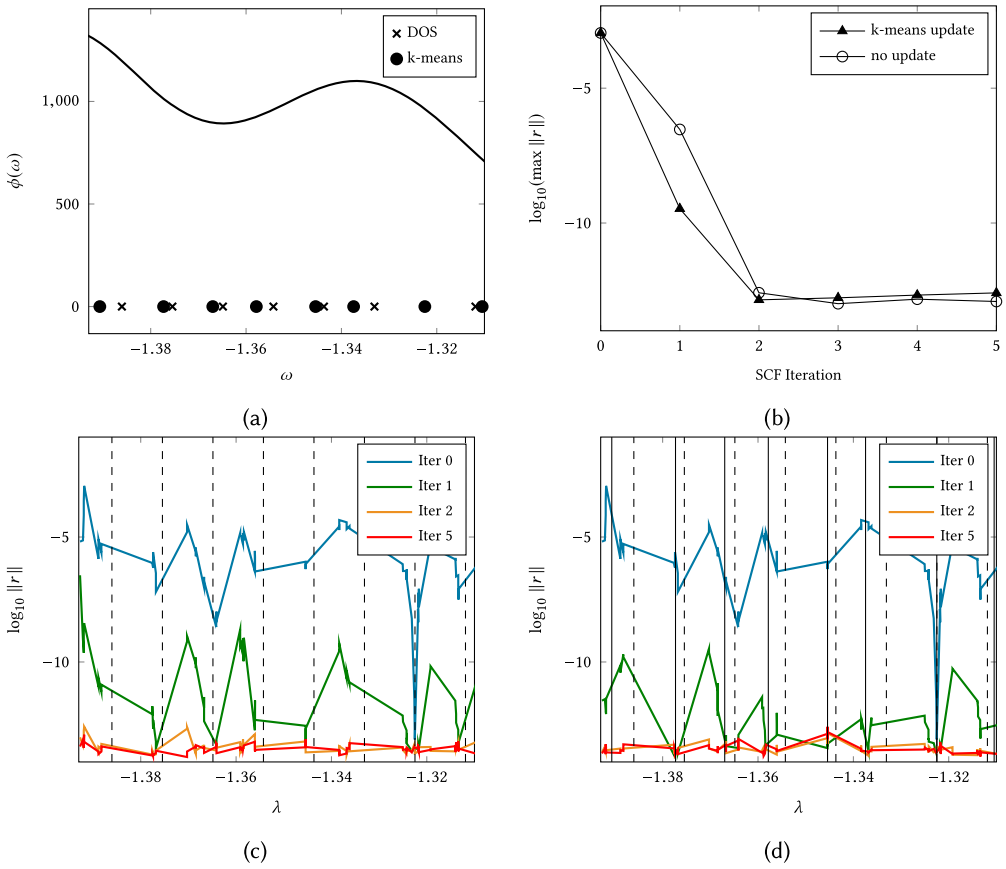
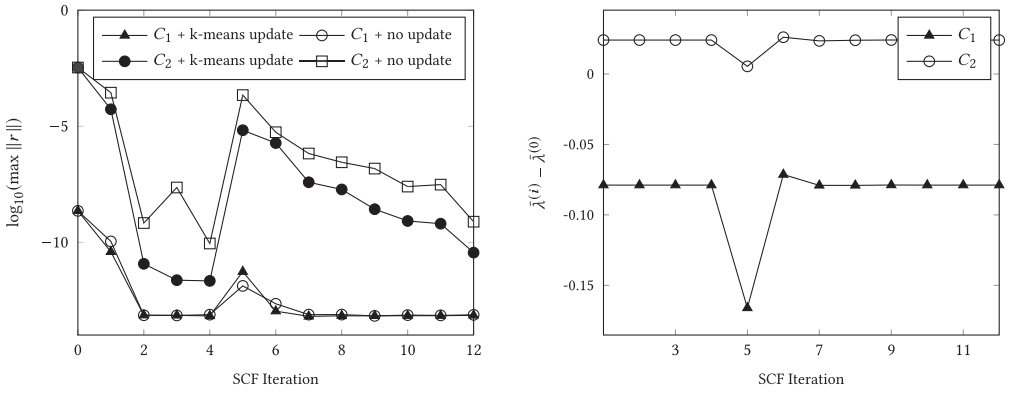


Fig. 10. Graphene eigenvalue cluster $[-1.4, -1.3]$ (93 eigenvalues). (a) Initial Lanczos DOS along with DOS shift placement and k-means update. (b) Convergence behavior of the largest residual norm in the spectral window both with and without the k-means shift update. Overall residual convergence for the SISLICE method within the spectral window with (d) and without (c) k-means shift update. Converges in three SCF iterations both with and without k-means shift update.

However, as both methods yield convergence in the same number of SCF iterations overall, we do not believe this discrepancy to be problematic in practice.

5.2 Shift Selection for a Converging Matrix Pencil Sequence

In this section, we examine how our shift selection strategy enables the SISLICE method to efficiently compute eigenpairs of pregenerated, convergent sequences of matrix pencils obtained from a true SCF procedure. Such a test allows us to gauge the ability of the SISLICE method to solve true SCF eigenvalue problems. To determine the efficacy of the shift selection and migration strategy, we examine both the convergence of the residuals produced by the SISLICE method and the change of the true eigenvalues throughout the SCF procedure itself. The latter is possible, because these matrices are pregenerated, thus we have access to the exact eigenvalues of these matrices as a reference to compare the convergence of the SISLICE method. Further, as was examined in the previous section, we perform analogous comparisons of the SISLICE method both with and without k-means updates to the spectral shifts throughout the SCF procedure.



(a) (b)

Fig. 11. Convergence of the SISLICE method applied to the Silane SCF procedure for two representative spectral windows. (a) Comparison of the convergence behavior of the largest residual norm in the respective spectral windows both with and without k-means shift updates throughout the SCF procedure. (b) The change in average eigenvalue from the initial average of the two spectral windows, respectively.

Silane. The Silane SCF procedure converged within 13 iterations in the NWChemEx software package. It is the nature of this particular test case (and is typical of all-electron density functional theory calculation) that the spectrum is separated into well defined clusters throughout the entire SCF procedure. For this reason, we are able to examine the same eigenvalue clusters as discussed in the previous section for this test case. The convergence behavior of the SISLICE method applied to this test case is given in Figure 11.

Much like the results presented in Section 5.1, we see a significant difference in the convergence behavior between the two clusters. Due to the isolated nature of C_1, convergence of the subspace iteration is rapid. Despite changes in the eigenvalues resulting from the changes in the matrix pencil in early SCF iterations, the shift selection strategy we developed is able to track this change, and construct and move spectral probes to obtain eigenvalues within C_1 at convergence. The convergence for C_2 is much less rapid, as was also the case in the previous experiments due to the lack of a large separation between eigenvalues within C_2 and the rest of the spectrum. Furthermore, the change in the eigenvectors in C_1 is much less than those in C_2, thus they provide excellent initial guesses for subsequent SCF iterations. The eigenvectors in C_2 undergo a much more drastic change, but it can be seen in Figure 11(a) that this change becomes less as the SCF converges.

Note that the convergence of the SCF for the Silane test case is not smooth; there is a large change in the average eigenvalue for the two examined clusters at the fifth SCF iteration. This is not an uncommon feature in the SCF procedure for density functional calculations. There is an analogous change in the residual norms for the SISLICE method at the same SCF iteration. The reason for this is twofold. In the case where k-means clustering is used to migrate the shifts between SCF iterations, the fact that the clustering is performed using the validated eigenpairs from the *previous* SCF iteration yielded a non-optimal placement for the fifth iteration. However, because the change in residual norms is present also for the experiment without k-means shift updates, the shift migration is not the only reason for this change. The large change in average eigenvalue for this SCF iteration is also accompanied by a change in corresponding eigenvectors within these spectral windows. Thus, the validated eigenvectors from the previous SCF iteration are also not optimal choices for the initial guess to seed the subspace sequence at this iteration. Remark that the increase in residual norm is in fact less for the C_2 cluster with k-means shift

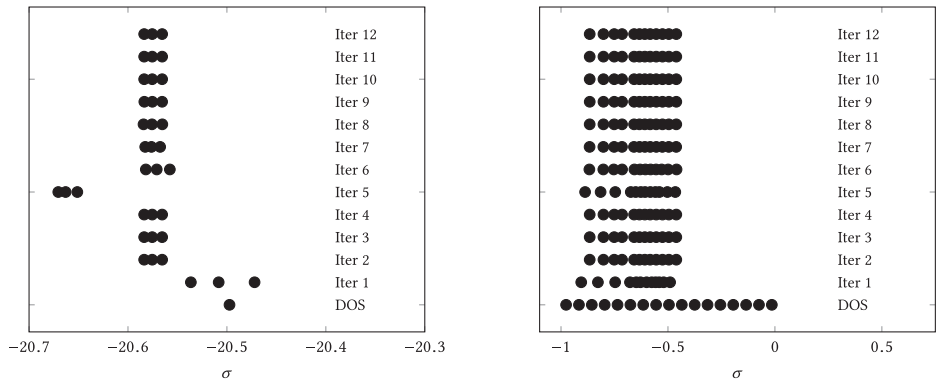


Fig. 12. Shift migration for the SISLICE method in the Silane SCF procedure for the C_1 (a) and C_2 (b) eigenvalue clusters.

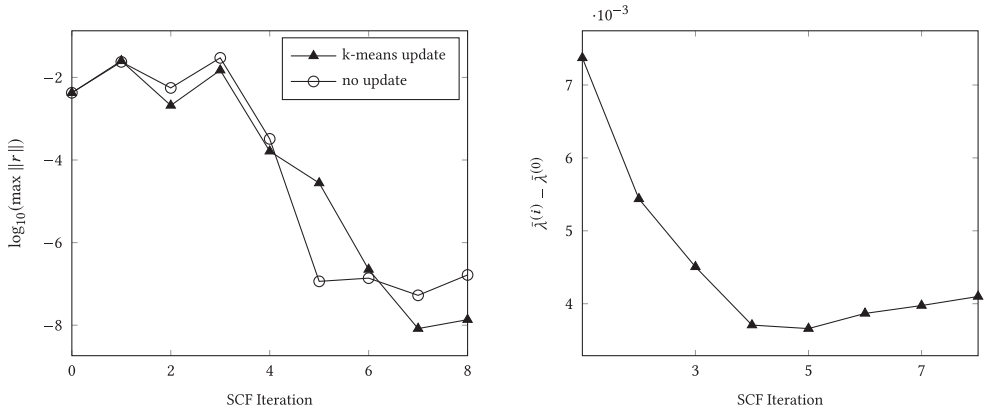
updates, indicating that the shift migration strategy is beneficial for this cluster even when the shifts are placed non-optimally.

Due to the fact that Silane admits well-defined (and trackable) clusters in its spectrum, we are also able to examine the shift migration within these clusters in Figure 12. The largest change in shift placement occurred between the first and second SCF iterations, the former of which was produced by the DOS strategy. Because the characteristic of these clusters is largely unchanging throughout the SCF procedure, we are able to see that the k-means shift update remains visually unchanging with the exception of the fifth SCF iteration. Remark that the k-means shift selection strategy was able to track the change in eigenvalues in this iteration and subsequently recover to a reasonably static set of shifts in the following SCF iterations.

Graphene. The Graphene SCF procedure converged within ten iterations in the SIESTA software package. Convergence results for the SISLICE method applied to this problem may be found in Figure 13. Unlike the Silane test case, the homogeneity of the eigenvectors for the Graphene test case makes tracking eigenvalue clusters throughout the SCF procedure impractical. The eigenpairs in a particular spectral interval at one SCF iteration are not likely to be of the same character in the subsequent iterations until convergence is reached. As such, we examine the convergence globally across all of the 1,000 eigenpairs desired for this test case.

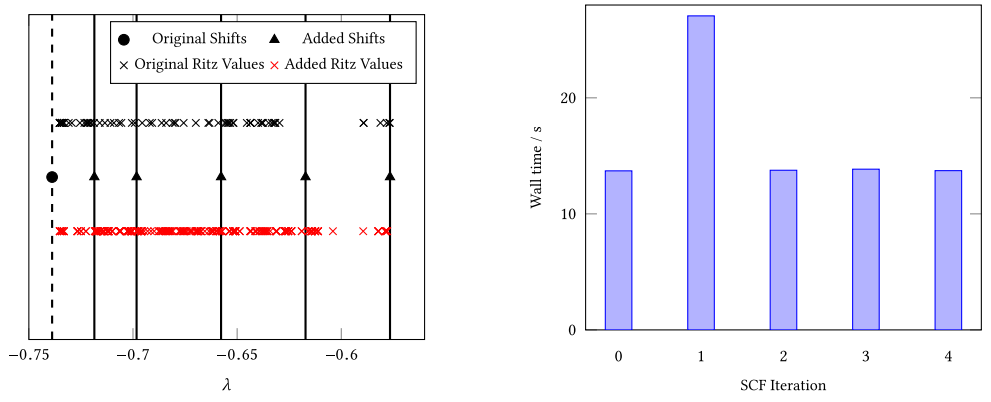
Unlike the Silane test case, the SCF convergence for Graphene is smooth. This smooth SCF convergence is mirrored in the monotonic convergence of the SISLICE method as the SCF approaches convergence. When the SCF procedure yielded large changes in the underlying spectrum, i.e., the first three iterations, the error produced by the SISLICE method was larger as the bases from the previous SCF iteration were not as good of an initial guess as they were in the later iterations. After the fourth SCF iteration, the spectrum is only undergoing small changes and the SISLICE method exhibits rapid convergence. As was the case for the previous numerical experiments with Graphene, no discernible difference between DOS and k-means shifts is exhibited. For example, at the fifth SCF iteration, the DOS shifts produced more accurate results whereas at the seventh, the k-means shifts produced more accurate results.

Due to the fact that the SCF underwent large spectral changes in the early SCF iterations, the extent to which the shifts were able to be usefully updated using the spectrum of the previous matrix pencils was limited. As such, shifts needed to be inserted per the prescription in Section 4.2. In the following subsection, we examine an example of this insertion for the Graphene test case.



(a) (b)

Fig. 13. Convergence of the SISLICE method applied to the Graphene SCF procedure for the lowest 1,000 eigenvalues. (a) Comparison of the convergence behavior of the largest residual norm both with and without k-means shift updates throughout the SCF procedure. (b) The change in average eigenvalue from the initial average of the lowest 1,000 eigenvalues.



(a) (b)

Fig. 14. Example shift insertion for the Graphene test case. The last slice of this test case was determined to have missing eigenvalues per the slice validation scheme at SCF iteration 1. Panel (a) shows the positions of the inserted shifts and the new slices and eigenvalues produced by this insertion. Panel (b) shows the computation required to perform the first five SCF iterations with this insertion.

5.3 Missing Eigenvalues and Probe Insertion

For the Graphene example, we found that some eigenvalues were missed in the second SCF iteration due to a poorly placed shift produced by the k-means clustering of approximate eigenvalues obtained in the first SCF iteration. The red crosses in Figure 14(a) show all eigenvalues within the spectral slice $[-0.75, -0.56]$ that contains the missing eigenvalues. The black crosses mark the locations of the approximate eigenvalues that were found by the spectral probe associated with the poorly placed target shift to the left of this interval (marked by a solid black circle).

In this case, after performing a DOS estimate of the matrix pencil updated in the second SCF iteration as suggested in Section 3.3, we constructed five new probes whose target shifts were

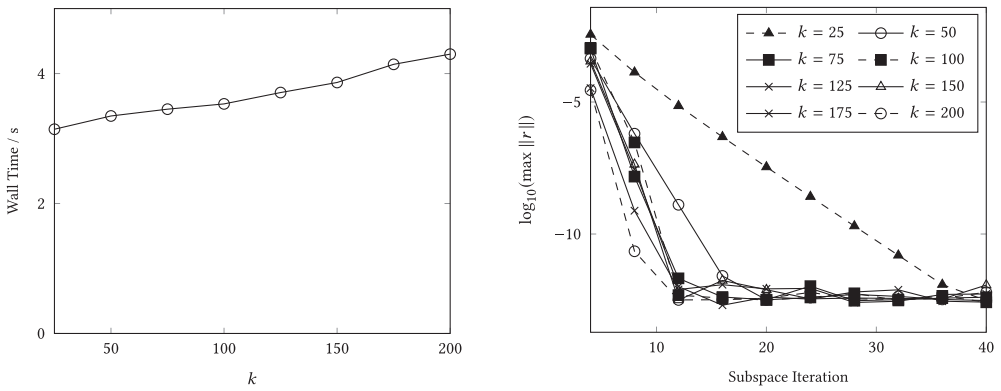


Fig. 15. The effects of the probe basis dimension on timings and convergence in the SISLICE method. Results were obtained using the Graphene test case with 100 spectral shifts. Timings (a) were obtained with Intel(R) MKL on 32 Intel(R) Haswell threads. The convergence (b) is tracked as the largest residual norm for the first spectral slice as a function of subspace iteration.

placed at the positions marked by the black vertical lines (with arrows) in Figure 14(a). After new target shifts were selected from the k-means clustering of the computed eigenvalues, some of the inserted probes were mapped to the new shifts and some of them were deleted in the third SCF iterations. In all subsequent SCF iterations, no missing eigenvalue was detected, and SCF convergence was achieved in ten iterations.

While the inserted new probes captured all the missing eigenvalues, there is an associated cost/penalty for this insertion as shown in Figure 14(b). Because the insertion of new probes essentially amounts to a recalculation of part of the spectrum in the second SCF iteration, the wall clock time required to complete that iteration was doubled. However, we should point out that this type of probe insertion is rare in our experiments. Because it only occurs in early SCF iterations, the extra cost is typically amortized over the remainder of the SCF procedure.

5.4 Probe Basis Dimension

As was discussed in Section 2.1, the dimension of the basis used for the shift-invert subspace iterations need only be *at least* the number of desired eigenpairs in the neighborhood of a particular spectral shifts. In practice, the basis dimension should be chosen to be slightly larger to enable faster convergence. However, as the basis dimension increases, the computational time required to perform the shift-invert subspace iterations also increases due to the need to solve linear systems with a larger number of right-hand sides. In Figure 15, we examine the effects of basis dimension on the convergence of the subspace iterations as well as on the computational time required to perform the subspace iterations for the Graphene test case.

Figure 15(a) shows the increase in computational time required to perform the subspace iterations as a function of basis dimension. All timing results were obtained using the Haswell partition of the Cori Supercomputer (2×16 Intel(R) Xeon(TM) Processor E5-2698 v3 at 2.3 GHz) using Intel(R) MKL to solve the linear systems and are representative of a single set of 4 subspace iterations. It is clear that even with a tenfold increase in the basis dimension, the effect on overall timing for the subspace iterations is negligible.

Figure 15(b) tracks the convergence of a particular spectral slice as a function of the number of subspace iterations. Due to the nearly uniform distribution of the eigenvalues within the

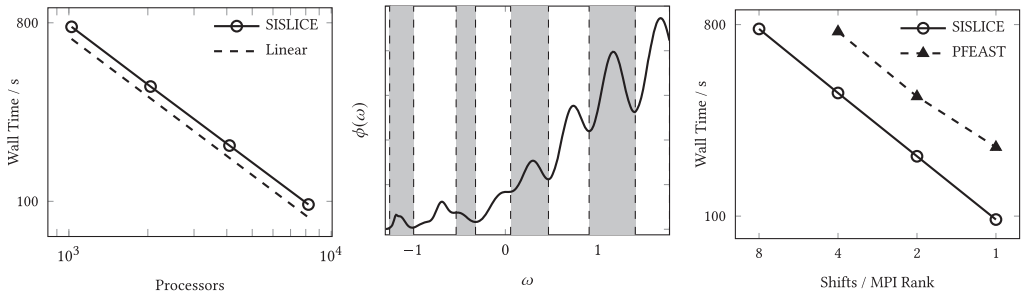


Fig. 16. Scaling of the SISLICE and PFEAST methods. All SISLICE calculations were performed to obtain the lowest 1,280 eigenvalues of the Ga10As10H30. (a) Strong scaling of SISLICE in comparison with linear scaling. (b) Slice intervals for PFEAST. (c) Strong scaling comparison of SISLICE with PFEAST with respect to the number of shifted inversions performed on a single MPI rank.

Graphene's spectrum, this slice is representative of the entire spectrum. The shift placement was chosen such that each spectral probe is responsible for ∼10 validated eigenpairs. Unlike the effects on timing, the convergence of the examined spectral slice is sensitive to the basis dimension; with small basis dimensions (e.g., 25) yielding very sub-optimal convergence results and large basis dimensions (e.g., 100–200) yielding much faster convergence. In practice, we have found that choosing $k \approx 10n_e/n_s$ yields sufficiently fast convergence in most cases.

5.5 Parallel Scalability

In this section, we examine the parallel scaling behavior of the proposed SISLICE method and detail a performance comparison with the PFEAST method of Kestyn et al. [2016]. All timing results were obtained using the Knight's Landing (KNL) partition of the Cori Supercomputer (Intel(R) Xeon Phi(TM) CPU 7250 @ 1.40 GHz) and all shifted linear systems (factorizations and backsubstitutions) were performed using the sparse symmetric solver PARDISO as distributed with Intel(R) MKL. All shifted inversions were performed on a single KNL node using 64 threads, and all MPI communication is internode. Although both SISLICE and (P)FEAST use the shift-invert subspace iteration, we will refer to the application of the contour spectral projector to the basis in the FEAST method as a contour subspace iteration do differentiate the methods. We note for clarity that this includes both the complex-shifted linear system solves as well as the integration reduction required by the FEAST method, i.e., each contour subspace iteration requires communication of the subspace to some subset of the MPI ranks.

Figure 16 depicts several scaling comparisons between SISLICE and PFEAST for the Ga10As10 matrix from the SuiteSparse collection [Davis and Hu 2011] (N = 113,081, NNZ = 6,115,633). Fill reducing reordering for this matrix was performed using the METIS software [Karypis and Kumar 2009] as packaged with PARDISO implementation in Intel(R) MKL. Both methods were used to obtain the lowest 1,280 eigenvalues of Ga10As10H30. The SISLICE calculations were performed using 128 shifts with k = 100 and 4 shift-invert subspace iterations per shift. The shifts were selected according to the DOS-based refinement strategy in Section 3.1. The PFEAST calculations were performed with 8 slices using 8 half-contour quadrature points per slice (a total of 64 total shifted factorizations) and k = 400. The default elliptic contour was used for each slice. As PFEAST does not admit its own scheme to partition the spectrum of interest into slices, we used the same DOS partitioning scheme as SISLICE to obtain the intervals for the 8 slices (see Figure 16(b)).

8 slices were chosen as to make a fair comparison with SISLICE by limiting the contours to reasonably localized eigenvalue clusters as to avoid excessive contour subspace iterations for embedded spectral regions. PFEAST was run to a trace convergence of 10^{-5} to ensure all eigenpairs were accounted for. All slices converged within 3 contour subspace iterations. We note for clarity that the 8 half-contour quadrature points and $k = 400$ basis dimension used per slice is not required for each of the slices, especially those at the lower end of the spectrum. However, as the scaling limit of PFEAST will be limited to the computation time required by the slowest executing contour integral, this does not impact the validity of this comparison. The basis dimension of 400 was chosen as to allow for convergence of the larger eigenvalue clusters at the upper end of the considered spectral domain. However FEAST offers a fallback mechanism to limit the number of vectors to which the contour subspace iteration is applied for contours containing only a few eigenvalues. This fallback was triggered for the first two slices in the FEAST calculations, which produced 21 and 69 vectors after the first contour subspace iteration, respectively.

The timing given in Figures 16(a) and 16(c) are indicative all all operations that must take place within a single SCF iteration for the two methods. This includes the triangular factorizations, backsolves, and synchronization / integration (as required by FEAST). SISLICE additionally includes the time required to gather the non-uniformly distributed set of eigenvectors to a single MPI rank (implemented with `MPI_Gatherv`). Remark that both SISLICE and PFEAST scale linearly out to their strong scaling limit (1 shift/MPI rank). This is due to the fact that their communication requirements are minimal and thus pose little to no overhead in execution time. However, due to the fact that PFEAST uses complex arithmetic, its overall computational cost per shift is ∼2× more than SISLICE. We note for clarity that due to the differing number of shifts used for SISLICE and PFEAST, the number of nodes that are represented for the data points in Figure 16(c) differ by a factor of 2.

6 CONCLUSION

In this work, we have developed the SISLICE method: a robust and efficient parallel shift-invert spectrum slicing strategy for self-consistent symmetric eigenvalue computation. The novelty of the SISLICE method is in its shift selection and migration strategies, which allow for only minimal communication requirements in its distributed-memory parallel implementation. Like all spectrum slicing methods, the SISLICE method partitions a spectral region of interest into intervals that are then treated independently. However, unlike previous and contemporary slicing methods, which rely on effectively sequential shift placement to partition the spectral region of interest, the SISLICE method utilizes DOS estimates to form the entire set of spectral slices at once. This strategy allows for maximal concurrency with minimal communication overhead. As the desired eigenvalues of the considered matrix pencils are dynamic throughout the SCF procedure, the SISLICE method employs a shift migration strategy based on k-means clustering that allows for tracking of the relevant eigenvalues throughout the SCF procedure without the need to recompute the costly DOS estimation at each iteration.

We have demonstrated the robustness and parallel efficiency of the SISLICE method for a representative set of SCF eigenvalue problems commonly encountered in electronic structure theory in Section 5. In particular, we have demonstrated that the k-means shift migration yielded noticeable convergence improvements in spectral regions with a highly irregular distribution of eigenvalues (such as the one exhibited for the Silane test case). Further, even in the cases where it did not yield discernible improvements (e.g., Graphene), its was shown that the k-means migration did not yield convergence degradation either.

From the perspective of performance, the SISLICE method was demonstrated to exhibit linear strong scaling for medium to large problem dimensions to large processor counts. Further, we

have demonstrated that the main communication requirement, the synchronization of Ritz values and residual norms across the distributed network, does not yield a noticeable change in overall scaling. We have also compared the performance of SISLICE to the PFEAST method due to their algorithmic similarity. Both methods scale linearly out to the strong scaling limit of 1 shift / MPI rank, but the use of complex arithmetic in PFEAST leads to an overall performance gain of 2× by the SISLICE method. Further, due to the non-trivial optimal definitions of contours for arbitrary spectral regions, SISLICE may be treated as a "black-box," whereas PFEAST requires considerable tuning to achieve optimal results.

A number of important aspects of the SISLICE method for the SCF eigenvalue problem, such as the DOS spectral partitioning scheme, k-means migration strategies and eigenvector seeding for subspace methods, could be extended to spectrum slicing methods methods such as (P)FEAST, SIPs, and polynomial filtering in cases where their constituent eigenvalue methods would be better performing that the shift-invert subspace iteration. This is of particular interest for spectra that exhibit similar characteristics as the all-electron Silane test case, which admits several isolated eigenvalue clusters in the lower region of the spectrum. Due to the isolated nature of these clusters, they would likely be better addressed by polynomial filtering, whereas the larger "clusters" of eigenvalues higher in the spectrum are well addressed by SISLICE. This topic will be addressed in future work.

Despite the demonstrated success of the proposed SISLICE method, there are several topics that were not addressed in this work that should be addressed to fully demonstrate the effectiveness of the method in real applications. The first is the integration of the SISLICE method into an actual SCF code such as NWChemEx, SIESTA, and so on. While our results have demonstrated the usefulness of the SISLICE method for pregenerated matrix sequences, the accuracy of the eigenvectors at any particular SCF iteration will influence the overall convergence of the SCF. This topic will also be addressed in future work.

ACKNOWLEDGMENTS

The authors thank the National Energy Research Scientific Computing (NERSC) center for making computational resources available to them. They also thank Wei Hu for providing the Graphene test matrices generated from the SIESTA software and Ajay Panyala for helping generate the matrix pencils for the Silane example in the NWChemEx software, as well as Murat Keçeli for helpful discussions regarding the development of the SIPs and SIESTA-SIPs software.

REFERENCES

H. M. Aktulga, L. Lin, C. Haine, E. G. Ng, and C. Yang. 2014. Parallel eigenvalue calculation based on multiple shift-invert lanczos and contour integral based spectral projection method. *Parallel Comput.* 40, 7 (2014), 195.

P. R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent. 2001. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM J. Matrix Anal. Appl.* 23, 1 (2001), 15–41.

P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet. 2006. Hybrid scheduling for the parallel solution of linear systems. *Parallel Comput.* 32, 2 (2006), 136–156.

E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. 1999. *LAPACK Users' Guide* (3rd ed.). Society for Industrial and Applied Mathematics, Philadelphia, PA. DOI: https://doi.org/10.1137/1.9780898719604

David Arthur and Sergei Vassilvitskii. 2007. K-means++: The advantages of careful seeding. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'07)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1027–1035. Retrieved from http://dl.acm.org/citation.cfm?id=1283383.1283494.

John Bachan, Scott B. Baden, Steven Hofmeyr, Mathias Jacquelin, Amir Kamil, Dan Bonachea, Paul H. Hargrove, and Hadia Ahmed. 2019. UPC++: A high-performance communication framework for asynchronous computation. In *Proceedings of the 33rd IEEE International Parallel & Distributed Processing Symposium (to appear)*.

John Bachan, Dan Bonachea, Paul H. Hargrove, Steve Hofmeyr, Mathias Jacquelin, Amir Kamil, Brian van Straalen, and Scott B. Baden. 2017. The UPC++ PGAS library for exascale computing. In *Proceedings of the 2nd Annual PGAS Applications Workshop*. ACM, 7.

Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, and Henk van der Vorst. 2000. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM.

Amartya S. Banerjee, Lin Lin, Wei Hu, Chao Yang, and John E. Pask. 2016. Chebyshev polynomial filtered subspace iteration in the discontinuous Galerkin method for large-scale electronic structure calculations. *J. Chem. Phys.* 145, 15 (2016), 154101. DOI: https://doi.org/10.1063/1.4964861

Amartya S. Banerjee, Lin Lin, Phanish Suryanarayana, Chao Yang, and John E. Pask. 2018. Two-level chebyshev filter based complementary subspace method: Pushing the envelope of large-scale electronic structure calculations. *J. Chem. Theory Comput.* 14, 6 (2018), 2930–2946. DOI: https://doi.org/10.1021/acs.jctc.7b01243 PMID: 29660292.

L. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. Whaley. 1997. *ScaLAPACK Users' Guide*. Society for Industrial and Applied Mathematics. DOI: https://doi.org/10.1137/1.9780898719642

Carmen Campos and Jose E. Roman. 2012. Strategies for spectrum slicing based on restarted Lanczos methods. *Numer. Algor.* 60, 2 (2012), 279–295.

Timothy A. Davis and Yifan Hu. 2011. The University of Florida sparse matrix collection. *ACM Trans. Math. Softw.* 38, 1, Article 1 (Dec. 2011), 25 pages. DOI: https://doi.org/10.1145/2049662.2049663

Roger G. Grimes, John G. Lewis, and Horst D. Simon. 1994. A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems. *SIAM J. Matrix Anal. Appl.* 15, 1 (1994), 228–272.

Laurent O. Jay, Hanchul Kim, Yousef Saad, and James R. Chelikowsky. 1999. Electronic structure calculations for plane-wave codes without diagonalization. *Comput. Phys. Commun.* 118, 1 (1999), 21–30. DOI: https://doi.org/10.1016/S0010-4655(98)00192-1

George Karypis and Vipin Kumar. 2009. MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0. Retrieved from http://www.cs.umn.edu/metis.

Murat Keçeli, Fabiano Corsetti, Carmen Campos, Jose E. Roman, Hong Zhang, Álvaro Vázquez-Mayagoitia, Peter Zapol, and Albert F. Wagner. 2018. SIESTA-SIPs: Massively parallel spectrum-slicing eigensolver for an ab initio molecular dynamics package. *J. Comput. Chem.* 39, 22 (2018), 1806–1814. DOI: https://doi.org/10.1002/jcc.25350

Murat Keçeli, Hong Zhang, Peter Zapol, David A. Dixon, and Albert F. Wagner. 2016. Shift-and-invert parallel spectral transformation eigensolver: Massively parallel performance for density-functional based tight-binding. *J. Comput. Chem.* 37, 4 (2016), 448–459. DOI: https://doi.org/10.1002/jcc.24254

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

A. Knyazev. 2001. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM J. Sci. Comput.* 23, 2 (2001), 517–541. DOI: https://doi.org/10.1137/S1064827500366124

Karol Kowalski, Edoardo Apra, Ray Bair, Colleen Bertoni, Jeffery S. Boschen, Eric J. Bylaska, Wibe A. de Jong, Jr., Thom Dunning, Niri Govind, Robert J. Harrison, Kris Keipert, Sriram Krishnamoorthy, Erdal Mutlu, Ajay Panyala, Ryan M. Richard, T. P. Straatsma, Edward F. Valeev, Hubertus J. J. van Dam, Álvaro Vázquez-Mayagoitia, David B. Williams-Young, Chao Yang, and Theresa L. Windus. [n.d.]. NWChemEx—Computational chemistry for the exascale era. *Chem. Rev.* ([n.d.]), in preparation.

Richard B. Lehoucq, Danny C. Sorensen, and Chao Yang. 1998. *ARPACK Users' Guide: Solution of Large-scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. Vol. 6. SIAM, Philadelphia, PA.

Ruipeng Li, Yuanzhe Xi, Lucas Erlandson, and Yousef Saad. 2019. The eigenvalues slicing library (EVSL): Algorithms, implementation, and software. *SIAM J. Sci. Comput.* 41, 4 (2019), C393–C415. DOI: https://doi.org/10.1137/18M1170935

Ruipeng Li, Yuanzhe Xi, Eugene Vecharynski, Chao Yang, and Yousef Saad. 2016. A thick-restart Lanczos algorithm with polynomial filtering for Hermitian eigenvalue problems. *SIAM J. Sci. Comput.* 38, 4 (2016), A2512–A2534.

Lin Lin, Yousef Saad, and Chao Yang. 2016. Approximating spectral densities of large matrices. *SIAM Rev.* 58, 1 (2016), 34–65.

Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE Trans. Info. Theory* 28, 2 (1982), 129–137.

Andreas Marek, Volker Blum, Rainer Johanni, Ville Havu, Bruno Lang, Thomas Auckenthaler, Alexander Heinecke, Hans-Joachim Bungartz, and Hermann Lederer. 2014. The ELPA library: Scalable parallel eigenvalue solutions for electronic structure theory and computational science. *J. Phys.: Condensed Matter* 26, 21 (2014), 213201.

E. Polizzi. 2008. Density-matrix-based algorithm for solving eigenvalue problems. *Phys. Rev. B* 79 (2008), 115112.

Yousef Saad. 2011. *Numerical Methods for Large Eigenvalue Problems: Revised Edition*. Vol. 66. SIAM, Philadelphia, PA.

T. Sakurai and H. Sugiura. 2003. A projection method for generalized eigenvalue problems using numerical integration. *J. Comput. Appl. Math.* 159 (2003), 119–128.

Olaf Schenk and Klaus Gärtner. 2002. Two-level dynamic scheduling in PARDISO: Improved scalability on shared memory multiprocessing systems. *Parallel Comput.* 28, 2 (2002), 187–197.

Olaf Schenk and Klaus Gärtner. 2006. On fast factorization pivoting methods for sparse symmetric indefinite systems. *Electr. Trans. Numer. Anal.* 23, 1 (2006), 158–179.

Olaf Schenk, Klaus Gärtner, and Wolfgang Fichtner. 2000. Efficient sparse LU factorization with left-right looking strategy on shared memory multiprocessors. *BIT Numer. Math.* 40, 1 (2000), 158–176.

Ron Shepard. 1993. Elimination of the diagonalization bottleneck in parallel direct-SCF methods. *Theoretica Chimica Acta* 84, 4 (Jan. 1993), 343–351. DOI: https://doi.org/10.1007/BF01113273

Gerard L. G. Sleijpen and Henk A. Van der Vorst. 2000. A Jacobi–Davidson iteration method for linear eigenvalue problems. *SIAM Rev.* 42, 2 (2000), 267–293.

José M. Soler, Emilio Artacho, Julian D. Gale, Alberto García, Javier Junquera, Pablo Ordejón, and Daniel Sánchez-Portal. 2002. The SIESTA method for ab initio order-N materials simulation. *J. Phys.: Condensed Matter* 14, 11 (2002), 2745.

A. Stathopoulos and J. R. McCombs. 2007. Nearly optimal preconditioned methods for Hermitian eigenproblems under limited memory. Part II: Seeking many eigenvalues. *SIAM J. Sci. Comput.* 29, 5 (2007), 2162–2188.

James Joseph Sylvester. 1852. XIX. A demonstration of the theorem that every homogeneous quadratic polynomial is reducible by real orthogonal substitutions to the form of a sum of positive and negative squares. *London, Edinburgh, Dublin Philos. Mag. J. Sci.* 4, 23 (1852), 138–142.

P. T. P. Tang and E. Polizzi. 2014. FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection. *SIAM J. Matrix Anal. Appl.* 35 (2014), 354–390.

Yuanzhe Xi, Ruipeng Li, and Yousef Saad. 2018. Fast computation of spectral densities for generalized eigenvalue problems. *SIAM J. Sci. Comput.* 40, 4 (2018), A2749–A2773. DOI: https://doi.org/10.1137/17M1135542

Chao Yang. 2005. Solving large-scale eigenvalue problems in SciDAC applications. *J. Phys.: Conf. Series* 16 (Jan. 2005), 425–434. DOI: https://doi.org/10.1088/1742-6596/16/1/058

Hong Zhang, Barry Smith, Michael Sternberg, and Peter Zapol. 2007. SIPs: Shift-and-invert parallel spectral transformations. *ACM Trans. Math. Softw.* 33, 2 (2007), 9.

Received August 2019; revised May 2020; accepted July 2020