PHIDE: A Parallel Hybrid Direct–Iterative Eigensolver for Hermitian Eigenvalue Problems

Shengguo Li, Xinzhe Wu, Jose E. Roman, Ziyang Yuan, Ruibo Wang, Tiejun Li, Yi Xie, Bo Yang, and Xuguang Chen

***Abstract*—In this paper, we propose a Parallel Hybrid Direct–Iterative Eigensolver for Hermitian Eigenvalue Problems without tridiagonalization, denoted by PHIDE, which combines direct and iterative methods. PHIDE first reduces a Hermitian matrix to banded form, then applies a spectrum slicing algorithm to the banded matrix, and finally computes the eigenvectors of the original matrix via backtransformation. Compared with conventional direct eigensolvers, PHIDE avoids tridiagonalization, which involves many memory-bound operations. In PHIDE, the banded eigenvalue problem is solved using the contour integral method implemented in FEAST, which may yield slightly lower accuracy than tridiagonalization-based approaches. For sequences of correlated Hermitian eigenvalue problems arising in density functional theory (DFT), PHIDE achieves an average speedup of $1.22\times$ over the state-of-the-art direct solver in ELPA when using 1024 processes. Numerical experiments are conducted on dense Hermitian matrices from real applications as well as large sparse matrices from the SuiteSparse and ELSES collections.**

***Index Terms*—Eigenvalues, spectrum-slicing algorithms, banded matrices, direct eigenvalue methods.**

I. Introduction

LARGE-SCALE Hermitian eigenvalue problems are prevalent in numerous scientific computing applications, such as condensed matter physics [1] and thermoacoustics [2]. For instance, in electronic structure calculations based on Kohn-Sham density functional theory, a large nonlinear eigenvalue problem is iteratively solved using the self-consistent field (SCF) procedure, requiring the solution of a sequence of correlated linear eigenvalue problems. Depending on the discretization method and basis used, the matrices involved can be sparse, dense, or banded. In this work, we focus on the dense case. Dense eigenvalue problems are typically solved using direct methods, with well-known packages such as ScaLAPACK [3], ELPA [4], and EigenExa [5]. However, when addressing sequences of correlated eigenvalue problems, these eigensolvers fail to exploit the inherent properties of the problem sequence, instead treating each problem independently and solving it in isolation.

Compared to direct methods, one advantage of iterative methods is the ability to utilize approximate eigenvalues and eigenvectors from previous SCF iterations [6]. Based on subspace iteration (SI) accelerated by Chebyshev polynomials, an efficient eigensolver called ChASE has been proposed for sequences of dense Hermitian eigenvalue problems [7]. Compared to Elemental [8], ChASE is faster when computing a small portion of the extremal spectrum. However, ChASE becomes slower when computing a large number of eigenpairs [9]. Even with further parallel optimizations [10], ChASE struggles to achieve good scalability on distributed GPU systems when seeking a large number of eigenpairs. This is also a common problem for other iterative methods such as the implicitly restarted Lanczos algorithm [11] and LOBPCG [12], as they must solve a projected dense eigenvalue problem as part of the Rayleigh–Ritz procedure.

When computing a large number of eigenpairs with iterative algorithms, one widely used approach is the spectrum slicing technique [13], [14], that tries to compute the eigenvalues by chunks. Spectrum slicing methods include the shift-invert spectral transformation [15], [16], [17], [18], [19], [20], polynomial filtering [7], [21], [22], [23], and contour integral methods [24], [25], [26], [27].

Iterative methods are seldom used for dense eigenvalue problems because both dense matrix-vector multiplication and solving dense linear systems are computationally expensive. When computing a large number of eigenpairs, the complexity of iterative methods for dense matrices can easily approach or even exceed $O(n^3)$, the cost of direct methods [28]. Typically, direct methods are employed for dense eigenvalue problems, especially when many eigenpairs are required. Fig. 1 illustrates the main procedure, which consists of *three steps* [4], [29]. In Step (II), the eigendecomposition of a banded matrix is computed. The classical approach reduces it to tridiagonal form using orthogonal transformations, which involve memory-bound operations and exhibit poor scalability.

Received 30 January 2025; revised 22 September 2025; accepted 6 October 2025. Date of publication 20 October 2025; date of current version 4 December 2025. This work was supported in part by National Key R&D Program of China under Grant 2023YFB3001700, in part by National Natural Science Foundation of China under Grant U1611261, Grant 61872392, and Grant U1811461. The work of J. E. Roman was supported in part by MICIU/AEI/10.13039/501100011033 under Grant PID2022-139568NB-I00, and in part by ERDF/EU under Grant RED2022-134176-T. Recommended for acceptance by B. Ucar. *(Corresponding author: Xuguang Chen.)*

Shengguo Li, Ruibo Wang, Tiejun Li, Yi Xie, Bo Yang, and Xuguang Chen are with the College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China (e-mail: nudtlsg@nudt.edu.cn; ruibo@nudt.edu.cn; tjli@nudt.edu.cn; xieyi00@nudt.edu.cn; yangbo@nudt.edu.cn; chenxuguang@nudt.edu.cn).

Xinzhe Wu is with Jülich Supercomputing Centre, Forschungszentrum, 52428 Jülich, Germany (e-mail: xin.wu@fz-juelich.de).

Jose E. Roman is with D. Sistemes Informàtics i Computació, Universitat Politècnica de València, 46022 València, Spain (e-mail: jroman@dsic.upv.es).

Ziyang Yuan is with the Academy of Military Sciences, Beijing 100097, China (e-mail: yuanziyang11@nudt.edu.cn).

Digital Object Identifier 10.1109/TPDS.2025.3623188

1045-9219 © 2025 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: National Univ of Defense Tech. Downloaded on February 23,2026 at 02:36:26 UTC from IEEE Xplore. Restrictions apply.

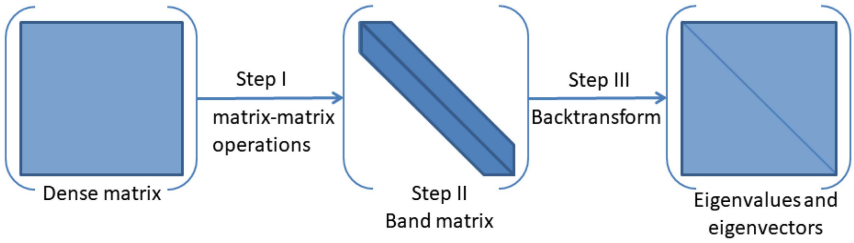


Fig. 1. The main steps of direct methods when using two-step symmetric banded reduction.

Moreover, Step (II) does not take advantage of the correlation present in the sequence of eigenvalue problems. In this paper, we introduce a novel approach that combines iterative methods with direct methods. The core idea is to apply iterative methods to a banded matrix with a small bandwidth (usually less than 64) instead of the original dense matrix. This allows iterative methods to efficiently compute a partial or full eigendecomposition of a dense Hermitian or symmetric matrix. This new framework for solving dense Hermitian eigenvalue problems is denoted as PHIDE, short for *Parallel Hybrid Direct-Iterative Eigensolver for Hermitian Eigenvalue Problems without Tridiagonalization*.

To combine direct methods with iterative methods, efficient data redistribution routines are essential. This is because direct methods, such as those in ScaLAPACK and ELPA, store matrices using a 2D block cyclic data distribution (BCDD). Iterative methods, such as those in Hypre [30], PETSc [31], Trilinos [32], and FEAST [26], use one-dimensional distributions. Therefore, an efficient data redistribution algorithm is needed to ensure consistent data formatting. This context is introduced in Section III-B, and our newly proposed approach can be significantly faster than the approach provided in ELPA, see also [33].

We evaluate PHIDE using two complex dense Hermitian matrices from an SCF application, along with several large-scale real sparse matrices from the SuiteSparse and ELSES collections [34], [35]. For these large matrices, using random initial vectors can also make PHIDE outperform ELPA. It is worth mentioning that the sparse matrices are stored in dense format to assess the performance of PHIDE on large-scale matrices. We benchmark PHIDE against ELPA on a supercomputer, utilizing up to 1024 MPI processes. Numerical results demonstrate that our approach can greatly surpass ELPA by leveraging the correlation within sequences of eigenvalue problems.

In summary, the main contributions of this work are as follows:

1) We propose a new framework for solving sequences of Hermitian or real symmetric eigenvalue problems by combining direct and iterative methods.
2) For self-consistent field eigenvalue problems with correlations, PHIDE achieves a speedup of up to $1.38\times$ over ELPA.
3) We conducted extensive experiments to test PHIDE, using dense Hermitian matrices from real applications and large sparse matrices from SuiteSparse and ELSES.

II. Preliminaries

In this section, we provide a brief introduction to direct methods and contour integral-based iterative methods for solving eigenvalue problems.

A. Direct Methods

In this section, we introduce the main steps of direct methods for solving a standard eigenvalue problem,

$$AX = X\Lambda, \tag{1}$$

where A is an $n \times n$ Hermitian or real symmetric matrix, X is the $n \times m$ eigenvector matrix and the diagonal matrix Λ contains eigenvalues λ_i, $i = 1, \ldots, m$.

We mainly follow the description in [4] and introduce the *three* steps of a direct method [36], [37].

I) Reduce A to band form,

$$D = U^H A U, \tag{2}$$

where U is a unitary matrix.

II) Compute the eigendecomposition of the banded matrix $D\widehat{X} = \widehat{X}\Lambda$ via the following three steps:

III) Reduce the banded matrix to tridiagonal form: $T = VDV^H$, where V is a unitary matrix.

IV) Compute m eigenpairs of the tridiagonal matrix: $T\widehat{Y} = \widehat{Y}\Lambda$.

V) Backtransform to get the eigenvectors of the banded matrix, $\widehat{X} = V^H\widehat{Y}$.

VI) Backtransform to get the eigenvectors X of matrix A,

$$X = U^H\widehat{X}. \tag{3}$$

Fig. 1 shows the main steps of direct methods. Step (II) is traditionally based on tridiagonalization via symmetric bandwidth reduction (SBR) [38] and solved by the Divide-and-Conquer (DC) [39] or MRRR [40] algorithms. The starting point of this work is that Step (II) has poor scalability and it takes a large portion of total time. See Table II in Section IV for more details.

B. The FEAST Algorithm

The FEAST algorithm utilizes spectral projection and subspace iteration to obtain selected interior eigenpairs [26]. A Rayleigh–Ritz procedure is employed to project the original matrix A onto a reduced search subspace, resulting in the matrix

$$\widetilde{A} = Q^H A Q, \tag{4}$$

Authorized licensed use limited to: National Univ of Defense Tech. Downloaded on February 23,2026 at 02:36:26 UTC from IEEE Xplore. Restrictions apply.

TABLE I
THE TEST PLATFORM AND ENVIRONMENT OF ONE NODE

Items	Values
CPU	Phytium 2000+@2.2GHz, 64 cores
Memory size	128 GB (DDR4)
Operating System	Linux 5.4.0-65-generic
Compiler	gcc 9.3.0
Optimization	−02

where $A \in \mathbb{C}^{n\times n}$ is a symmetric (real) or Hermitian (complex) matrix, and $Q \in \mathbb{C}^{n\times m}$ is a matrix with orthonormal columns. The projected matrix $\widetilde{A} \in \mathbb{C}^{m\times m}$ is of significantly smaller dimension, with $m \ll n$, where m denotes the number of eigenpairs to be computed.

The approximate eigenvalues $\widehat{\Lambda}$ and eigenvectors $\widehat{X}$ of the original matrix A can be obtained by solving the much smaller projected eigenvalue problem:

$$\widetilde{A}W = W\widetilde{\Lambda}, \tag{5}$$

as

$$\widehat{X} = QW, \quad \widehat{\Lambda} = \widetilde{\Lambda}. \tag{6}$$

In theory, the ideal filter for the Hermitian problem would act as a projection operator $X_m X_m^H$ onto the subspace spanned by the eigenvector basis, which can be expressed via the Cauchy integral formula

$$X_m X_m^H = \frac{1}{2\pi\sqrt{-1}} \oint_\Gamma (zI - A)^{-1} dz, \tag{7}$$

where the eigenvalues associated with the orthonormal eigenvectors X_m are located within an interval delimited by the closed curve Γ.

In practice, the spectral projector is approximated by a quadrature rule using n_e integration nodes and weights $\{(z_j, w_j)\}_{j=1,\ldots,n_e}$,

$$Q_{m_0} = \sum_{j=1}^{n_e} w_j (z_j I - A)^{-1} \widehat{X}_{m_0} \triangleq r_{n_e}(A)\widehat{X}_{m_0}, \tag{8}$$

where z_j and w_j are complex numbers, the search subspace has dimension $m_0 \geq m$, and $r_{n_e}(x) = \sum_{j=1}^{n_e} \frac{w_j}{z_j - x}$ is a rational function of order $(n_e - 1, n_e)$. In typical usage, FEAST employs $n_e = 8$ or 16 integration nodes for each contour Γ. As FEAST is essentially a subspace iteration method, the subspace dimension m_0 must exceed m in order to reliably capture the m desired eigenvalues. A common choice is $m_0 = 1.5$ m. The initial matrix $\widehat{X}_{m_0}$ is usually populated with random vectors.

Note that the computation of Q_{m_0} amounts to solving a set of n_e independent complex shifted linear systems with multiple right-hand sides,

$$(z_j I - A)Q_{m_0}^{(j)} = \widehat{X}_{m_0}, \qquad \text{for } j = 1, \ldots, n_e, \tag{9}$$

followed by the weighted summation,

$$Q_{m_0} = \sum_{j=1}^{n_e} w_j Q_{m_0}^{(j)},$$

Algorithm 1: The FEAST Algorithm.

Input: The input matrix A, initial vectors $\widehat{X}_{m_0}$, the relative residual tolerance ϵ, integration nodes and weights $\{(z_j, w_j)\}_{j=1,\ldots,n_e}$.
Output: The approximated eigenvalues and eigenvectors: $\widehat{X}_m$, $\widehat{\Lambda}_m$.
while $(\|A\widehat{X}_m - \widehat{X}_m\Lambda_m\| \geq \epsilon)$ **do**
  Compute $Q_{m_0} = r_{n_e}(A)\widehat{X}_{m_0}$
  Construct $\widetilde{A} = Q_{m_0}^H A Q_{m_0}$, and solve $\widetilde{A}W = W\widetilde{\Lambda}$
  Compute $\widehat{X}_{m_0} = Q_{m_0}W, \widehat{\Lambda} = \widetilde{\Lambda}$
end

which would have orthogonal columns [25]. The resulting matrix Q_{m_0} is then used as the Rayleigh–Ritz projector to construct the reduced matrix $\widetilde{A}$. Even when A is real symmetric, the FEAST algorithm performs complex arithmetic due to the use of complex integration nodes, resulting in a higher computational cost compared to shift-and-invert spectrum slicing methods that use real-axis shifts [15]. However, since FEAST employs a rational function to approximate the spectral projector, it typically converges faster and requires fewer iterations than polynomial filtering methods, making it a competitive and efficient approach in practice.

The general outline of FEAST is shown in Algorithm 1 for computing m eigenpairs in a given closed curve. The initial vectors $\widehat{X}_{m_0}$ are usually a set of random vectors, and good initial vectors can accelerate the convergence [26]. It is mathematically equivalent to subspace iteration applied with a rational matrix function $r_{n_e}(A)$, see [26]. The rational function can be constructed from numerical approximation of the integral (7) by using Gauss quadrature [25] or Trapezoid rule [27]. An improved rational approximation based on the work of Zolotarev is proposed in [41], which improves both convergence robustness and load balancing when FEAST runs on multiple search intervals in parallel. In Algorithm 1, the residual norm $\|A\widehat{X} - \widehat{X}\Lambda\|$ is defined as the relative 1-norm [24], [26], i.e.,

$$\|A\widehat{X} - \widehat{X}\Lambda\| = \frac{\|A\widehat{X} - \widehat{X}\Lambda\|_1}{\|A\|_2\|\widehat{X}\|_1},$$

where $\|\cdot\|_1$ and $\|\cdot\|_2$ are the 1-norm and 2-norm of matrix, respectively.

III. PROPOSED ALGORITHM

In this section, we introduce our new algorithm PHIDE, which can be seen as a hybrid algorithm, that combines the techniques of direct and iterative methods.

To reduce computational complexity, the direct methods first reduce a dense matrix A to a canonical form such as tridiagonal, bidiagonal or Hessenberg, and then apply QR, DC or MRRR to compute the eigenvalue or singular value decomposition. PHIDE also follows this approach. It first reduces a dense Hermitian matrix to its banded form via a sequence of unitary transformations. In contrast to classical direct algorithms, PHIDE uses spectrum slicing methods to compute the eigendecomposition of

Authorized licensed use limited to: National Univ of Defense Tech. Downloaded on February 23,2026 at 02:36:26 UTC from IEEE Xplore. Restrictions apply.

TABLE II
THE EXECUTION TIMES (IN SECONDS) OF EACH STEP OF ELPA WHEN CHOOSING DIFFERENT nev

np	nev = 2000			nev = 5000			nev = 10000		
	Step (I)	Step (II)	Step (III)	Step (I)	Step (II)	Step (III)	Step (I)	Step (II)	Step (III)
64	10.82	7.32	2.88	10.76	11.03	5.30	10.69	17.20	9.45
256	6.00	3.89	1.72	6.05	4.71	2.53	6.02	6.51	4.10
1024	4.45	2.52	1.32	5.02	2.89	1.66	4.58	3.20	2.24

Algorithm 2: PHIDE for Hermitian Eigenvalue Problems.

Input: Hermitian matrix $A^{(\ell)}$, number of eigenpairs to be computed nev, and approximated eigenvectors $X^{(\ell-1)}$, where ℓ denotes the ℓth related eigenvalue problem.

Output: The computed eigenpairs (X, Λ).

Step (I): Banded reduction

- Reduce $A^{(\ell)}$ to a banded matrix,

$$D^{(\ell)} = UA^{(\ell)}U^H, \quad \text{see (2).}$$

Step (II): Spectrum slicing algorithm

- Redistribute the banded matrix to each process and redundantly store it in compact form;
- Apply a spectrum slicing algorithm to matrix $D^{(\ell)}$ with $X^{(\ell-1)}$ as initial vectors, and compute nev eigenpairs in parallel,

$$D^{(\ell)}\widehat{X} = \widehat{X}\Lambda.$$

- Redistribute the computed eigenvectors $\widehat{X}$ to 2D BCDD form;

Step (III): Backtransform

- Compute the eigenvectors X of matrix $A^{(\ell)}$ via backtransform,

$$X = U^H\widehat{X}, \quad \text{see (3).}$$

the banded matrix. After obtaining the eigenpairs of the banded matrix, the eigenvectors of A are computed via backtransforms. The advantage is that it gets around the SBR process, which consists of memory-bound operations and has poor scalability just as shown in Example 1 in Section IV.

The procedure of PHIDE is shown in Algorithm 2. The structure of PHIDE looks like a *sandwich:* the two outer layers remain unchanged as in the classical direct methods, but the inner kernel is replaced by an iterative method. PHIDE is more like a framework, and we do not specify which spectrum slicing algorithm to use. Step (II) of PHIDE can use any type of spectrum slicing algorithms mentioned in Section I. In Algorithm 2, the approximate eigenvalues are obtained from the banded matrix reduced from previous (dense) Hermitian matrix $A^{(\ell-1)}$, which are used to partition the spectrum interval.

In this work, the banded eigenvalue problem is solved by the contour integral based method implemented in FEAST [24]. We currently focus on the Hermitian standard eigenvalue problem. For the generalized eigenvalue problem, we assume it has been reduced to the standard form. FEAST can be seen as a subspace iteration eigensolver accelerated by approximate spectral projection [26], and good starting vectors $X^{(\ell-1)}$ can speed up the convergence rate. However, Algorithm 2 can also use randomized vectors.

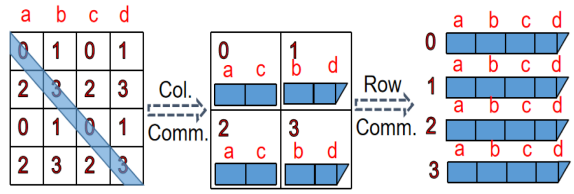


Fig. 2. Redistribute a banded matrix to 1D compact form.

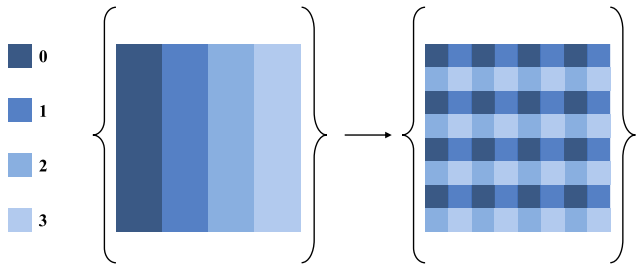


Fig. 3. From 1D irregular block to 2D BCDD form.

We use the ELPA framework to reduce our matrix to band form. The reduced banded matrix in ELPA is stored in 2D BCDD form. In contrast, iterative methods use 1D block data distribution (BDD) to store matrices. Since the semibandwidth of the intermediate banded matrix D is small, usually $n_{bw} = 64$, we store the whole banded matrix D locally on *each* process. Therefore, we first need a data redistribution to obtain D by collecting data from all processes. The process is shown in Fig. 2, and then D is stored compactly as in LAPACK.

When eigenvectors are computed by FEAST without utilizing L3 parallelism [24], they are redundantly stored in each process within the same group. We then convert the eigenvectors in each group to a 1D BDD format. Finally, we redistribute them across all process groups into a 2D BCDD format. This process will be introduced in Section III-B and shown in Fig. 3.

A. Spectrum Partition

In spectrum slicing algorithms, the spectrum is divided into multiple slices, with the eigenvalues in each slice computed by a dedicated group of processes. This corresponds to the first level of parallelism, denoted as **L1** parallelism. A classical method for spectrum partitioning is based on Sylvester's law of inertia, which involves computing the LDL^H factorization of $A - s_iI$ [36]. Alternatively, a more modern approach estimates

Authorized licensed use limited to: National Univ of Defense Tech. Downloaded on February 23,2026 at 02:36:26 UTC from IEEE Xplore. Restrictions apply.

the *density of states* (DOS) [42], [43] of A, relying on matrix-vector multiplications. In our problem, the intermediate band matrix has a narrow semibandwidth of $n_{bw} = 64$. Consequently, the cost of LDL^H factorizations is relatively low, allowing for spectrum partitioning using Sylvester's law of inertia.

When prior knowledge of the eigenvalue distribution is available from the initial SCF iterations, a clustering algorithm can be employed to divide the spectrum into different slices. A well-known clustering method is the *k*-means algorithm [44]. The primary objective is to group eigenvalues into distinct slices such that the slice boundaries do not intersect any eigenvalue cluster. This approach generally performs well for most matrices. However, a potential drawback is the risk of imbalanced slices, where some slices may contain a large number of eigenvalues while others have very few, particularly when partitioning into a large number of slices.

A good clustering should meet the following objectives, as discussed in [20]:

1) Eigenvalues that are very close (the gap within 10^{-7}) should be grouped into the same slice.
2) The number of eigenvalues in each slice should be approximately equal.
3) The eigenvalue range within each slice should be minimized.

Objective 1 is crucial for maintaining the orthogonality of the computed eigenvectors and is therefore given the highest priority. Objectives 2 and 3 address load imbalance issues and are treated as secondary considerations in this paper.

The classical *k*-means algorithm is designed for multi-dimensional arrays. In our case, the problem is much simpler: we have a one-dimensional set of points $\mathcal{C} = \{d_1, d_2, \ldots, d_n\}$, ordered increasingly, and we aim to partition $\mathcal{C}$ into K groups.

To address this, we propose a simplified and efficient one-dimensional *k*-means algorithm with the following advantages compared to the traditional *k*-means algorithm:

1) The proposed method has lower computational complexity, requiring only $n + K$ floating-point operations (flops), where n is the number of points and K is the number of groups. In contrast, the classical *k*-means algorithm requires approximately $(K + 1)n$ flops.
2) This method typically produces partitions with smaller variance in the number of points across all slices, resulting in more balanced group sizes.

Our algorithm solves the same optimization problem as the classical *k*-means method:

$$J = \min_{\mu_j} \sum_{i=1}^{n} \sum_{j=1}^{K} |d_i - \mu_j|, \tag{10}$$

where μ_j $(j = 1, 2, \ldots, K)$ are the centroids of the groups $G_1, G_2, \ldots, G_K$. The entire clustering process is outlined in Algorithm 3. In our implementation, we use the euclidean norm, and the centroid of group G_i is calculated as $\mu_i = (d_{l_i} + d_{r_i})/2$, where d_{l_i} and d_{r_i} are the leftmost and rightmost points in G_i, respectively.

The spectrum is partitioned by recording the number of eigenvalues, m_i, in each slice. The boundary set $\mathcal{B}$ is then updated in the subsequent SCF iteration using the formula $b_i = \frac{d_{r_i} + d_{l_{i+1}}}{2}$, where d_{r_i} is the largest value in G_i and $d_{l_{i+1}}$ is the smallest value in G_{i+1}. Since Algorithm 3 operates in linear time, *k*-means clustering is performed at every SCF iteration. During the final few SCF iterations before convergence, the spectrum partitions computed by Algorithm 3 remain unchanged.

Algorithm 3: A Simplified 1D *k*-Means Method.

Input: A sorted dataset $\mathcal{C} = \{d_1, d_2, \ldots, d_n\}$ and the number of clusters K.
Output: The boundary points $\mathcal{B} = \{b_0, b_1, \ldots, b_K\}$ and the point count set $\mathcal{M} = \{m_1, m_2, \ldots, m_K\}$.
Partition $\mathcal{C}$ into K groups $\{G_i\}_{i=1}^{K}$ such that $(n - \lfloor n/K \rfloor \cdot K)$ groups have $(\lfloor n/K \rfloor + 1)$ points and the other have $\lfloor n/K \rfloor$ points;
Construct $\mathcal{B} = \{b_i\}_{i=0}^{K}$, where $b_i = \frac{dr_i + d_{l_{i+1}}}{2}$. Here, d_{r_i} is the largest value in G_i, and $d_{l_{i+1}}$ is the smallest value in G_{i+1} for $i = 1, \cdots, K-1$. The values b_0 and b_K represent the lower and upper bounds of $\mathcal{C}$, respectively;
Let $\mathcal{B}_0 = \{\}$;
while $(\mathcal{B} \neq \mathcal{B}_0)$ **do**
  $\mathcal{B}_0 = \mathcal{B}$;
  for $i = 1, \cdots, K-1$ **do**
    $\ell = \mathbf{argmin}_{d_\ell \in \mathcal{C}} \{d_\ell > \frac{\text{centroid}(G_i) + \text{centroid}(G_{i+1})}{2}\}$,
    where *centroid*$(\cdot)$ calculates the centroid;
    $b_i = (d_\ell + d_{\ell-1})/2$;
    update the boundary between G_i and G_{i+1}, and record the number m_i of points in G_i;
  end
  while $(\exists b_p, b_q \in \mathcal{B}$, s.t. $b_p = b_q)$ **do**
    $\mathcal{B} = \mathcal{B} \backslash b_q$, $K = K - 1$;
  end
end

B. Data Redistributions

Classical dense numerical linear algebra packages store matrices in 2D BCDD form. Standard iterative methods usually store matrices in 1D BDD form. In order to combine direct methods with iterative methods, we should be able to efficiently redistribute matrices into the consistent format.

After Step (I) of direct methods, the banded matrix is stored in BCDD among all processes. Before applying the spectrum slicing algorithm, we first need to gather all blocks together and store the banded matrix in compact form. We propose a *two-step data redistribution* method, which first communicates along the process columns and then along the process rows. Finally, all the processes obtain the whole banded matrix and store it in compact form. The procedure is shown as a cartoon in Fig. 2. Every process uses a temporary workspace to store the matrix blocks obtained from the processes in the same process column, which can be implemented by using `MPI_Alltoallv` among the processes in the same column. After this, all the processes in the same column have the same data. Then, communications among processes in the same process rows are performed similarly, after

Authorized licensed use limited to: National Univ of Defense Tech. Downloaded on February 23,2026 at 02:36:26 UTC from IEEE Xplore. Restrictions apply.

which all the processes obtain the banded matrix, just as shown in the rightmost diagram of Fig. 2.

After computing the eigendecomposition of the banded matrix by using spectrum slicing methods, the eigenvectors are usually stored locally in 1D block data distribution along the column direction. We need to perform a data redistribution to transform a matrix from BDD to BCDD, to be consistent with ELPA. Due to the load imbalance problem, the number of local eigenvectors in different processes may be different, and we say the eigenvectors are stored in an *irregular* way. For this task, we used the data redistribution algorithm that is proposed in [33].

C. Validation and Other Details

The contour integral based algorithm can compute all the wanted eigenvalues enclosed in a given interval. To guarantee that nev eigenpairs can be computed, the lower bound b_0 and upper bound b_K must enclose all the wanted nev eigenvalues.[1] First, the upper bound b_K should be larger than the nevth eigenvalue of the current matrix. Since we know the previous eigenvalues, b_K can be computed by $b_K = (1+\alpha)d_{\mathrm{nev}}$, where α is a small real number and d_{nev} is the nevth previous eigenvalue. Note that b_K should not be too large. Otherwise, $[b_0, b_K]$ will enclose too many eigenvalues. b_K can be checked by computing the inertia of $(D^{(\ell)} - b_k I)$. If b_K is not large enough, we iteratively enlarge it by ratio α, $b_K = (1+\alpha)b_K$, and check it again until there are more than nev eigenvalues less than b_K. Algorithm 4 illustrates the process of computing the upper bound b_K. Since LAPACK does not support LDL^H factorization for banded symmetric matrices, we implemented a simplified version of the retraction algorithm [45], which is used to only compute the inertia without recording the matrix L, and the storage and computational cost are further reduced, becoming very small.

Another approach is to enlarge b_K as $b_K = b_K + \hat{m}_1\delta$, where $\hat{m}_1$ is an integer and $\delta = d_{\mathrm{nev}} - d_{\mathrm{nev}-1}$, the distance between the nevth and $(\mathrm{nev}-1)$th previous eigenvalues. When b_k is not large enough, it will be enlarged by another $\hat{m}_1\delta$. Its computation is similar to the first approach. Since we are solving sequences of eigenvalue problems, the parameters α and $\hat{m}_1$ can be updated during the SCF iterations. In our implementation, we let $\alpha = \pm 10^{-3}$ and $\hat{m}_1 = 6$. For the first fewer SCF iterations, Algorithm 4 needs to iterate at most 2–3 times to obtain the upper bound point. Since $D^{(\ell)}$ is banded with small bandwidth, the computation of these inertias is very cheap.

The lower bound b_0 can be computed similarly, $b_0 = (1+\beta)d_1$, and make sure $b_0 < d_1$ is a lower bound. When computing the smallest few eigenvalues, the parameter β is relatively easy to choose since any lower bound works. Similarly, b_0 can also be updated by $b_0 = b_0 + \hat{m}_2\delta_2$, where $\hat{m}_2$ is an integer and $\delta_2 = d_1 - d_2$, negative. When computing interior eigenvalues, β should be chosen carefully in case b_0 is too small to enclose too many eigenvalues in $[b_0, b_K]$. Since the process is similar to Algorithm 4, it is not included.

[1] For simplicity, we are assuming that the smallest nev eigenvalues are required. The cases for the largest or interior nev eigenvalues are similar. The users should provide how many (nev) eigenvalues are required.

Algorithm 4: Compute the Upper Bound b_K.

Input: The nev-th previous eigenvalue d_{nev} and integer nev, parameters α, $\hat{m}_1$ and δ which equals $d_{\mathrm{nev}} - d_{\mathrm{nev}-1}$, and matrix $D^{(\ell)}$.
Output: The upper bound b_K.
Compute $b_K = d_{\mathrm{nev}} + \max(\alpha \cdot d_{\mathrm{nev}}, \hat{m}_1\delta)$;
Let flag=true and $it = 1$;
while (flag) **do**
  Compute the inertia of $(D^{(\ell)} - b_K I)$, (n_-, n_0, n_+);
  if $(n_- \leq nev)$ **then**
    flag=false;
  else
    $b_K = b_K + \max(\alpha \cdot d_{\mathrm{nev}}, \hat{m}_1\delta)$;
    it = it + 1;
end

The spectrum is divided into K slices and one process group works on one slice. Note that only the first process group needs to compute the lower bound b_0, and only the Kth process group needs to compute the upper bound b_K. The ith process group computes the eigenvalues in the interval (b_{i-1}, b_i), and every process can compute the exact number m_i of eigenvalues in the current interval by computing the inertia of matrices $(D^{(\ell)} - b_{i-1}I)$ and $(D^{(\ell)} - b_i I)$. Therefore, every process group can check whether the number of computed eigenvalues is as expected or not. If some eigenvalues are missed, the corresponding process group should rerun FEAST by using more starting vectors.

In our problem, the value m_i is the exact number of eigenvalues in the ith slice, which is used to check that the number of computed eigenpairs is as expected. For stability, we further enlarge the number of starting vectors in FEAST, $m_i^{(0)} = \max(1.3 \times m_i, m_i + 10)$[2]. In our experiments, FEAST always computes more than nev eigenvalues and no eigenvalues are missed.

IV. Evaluation

A. Setup

All experimental results were obtained on a supercomputer equipped with 64-core Phytium 2000+ CPUs. Details of the hardware platform are provided in Table I. The codes were compiled using GNU compiler collection (GCC) version 9.3.0 and MPICH version 3.4.2, linked with OpenBLAS version 0.3.12. The −O2 optimization option was applied to each node. Plain MPI was used for all numerical experiments, i.e., with multi-threading disabled.

The experiments focused on using FEAST to accelerate the computation of Step (II) of ELPA. The ELPA code used in this work is from the latest version 2024.03.001. For complex Hermitian matrices, we used the kernel COMPLEX_NEON_ARCH64_BLOCK2 and for real symmetric

[2] The number 10 is chosen arbitrarily in order to deal with the case that m_i is too small and $m_i^{(0)}$ equals $1.3 \times m_i$.

Authorized licensed use limited to: National Univ of Defense Tech. Downloaded on February 23,2026 at 02:36:26 UTC from IEEE Xplore. Restrictions apply.

matrices, we used the kernel `REAL_NEON_ARCH64_BLOCK6`. We denote the number of MPI processes used by `np`.

B. Results for Real and Complex Matrices

In this subsection, we first use one random matrix to show the time cost for each step of ELPA, and then use two complex matrices from real applications and three real matrices from the SuiteSparse and ELSES collections [34], [35] to compare `PHIDE` with ELPA. Note that for sparse matrices, we do not exploit their sparsity during the banded reduction stage; instead, we treat them as dense matrices.

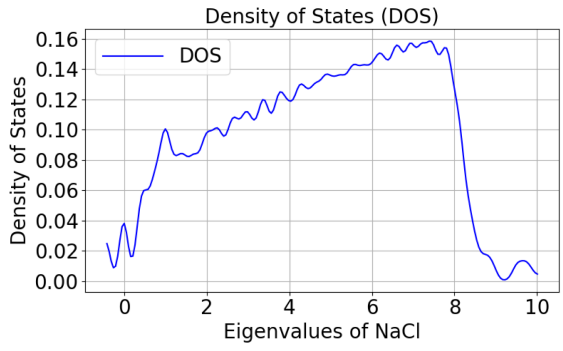
Example 1: In this example, we use a randomly generated real dense symmetric matrix with dimension $n = 10{,}000$. We apply ELPA to compute both partial and full eigendecompositions of the matrix, performing a strong scaling test, with the aim of demonstrating the high time ratio of Step (II). The performance of ELPA is sensitive to the parameter n_b, which defines the block size used for storing matrices in the 2D BCDD format. Many works suggest that $n_b = 128$, 64, 32 are usually good choices, while smaller values of n_b such as less than 16 will degrade the performance, see [46] and [47] for more details. All the performance results for ELPA in this paper are obtained using $n_b = 64$.

The results for computing the smallest $\texttt{nev} = 2000$, 5000 and 10,000 eigenpairs are shown in Table II when using 64, 256, and 1024 processes. We measure the times of Step (I)-(III). From the results, we can see that although the computational cost of Step (II) is small compared to the other two steps, it still accounts for approximately 30% of the total time, especially when `np` is large. Therefore, it is better to replace Step (II) with a more efficient and faster solver. In this paper, we use a spectrum slicing approach for this step. The next example shows that this approach can reduce the times of Step (II) by half, and our final algorithm (`PHIDE`) can be up to 33% times faster than ELPA.

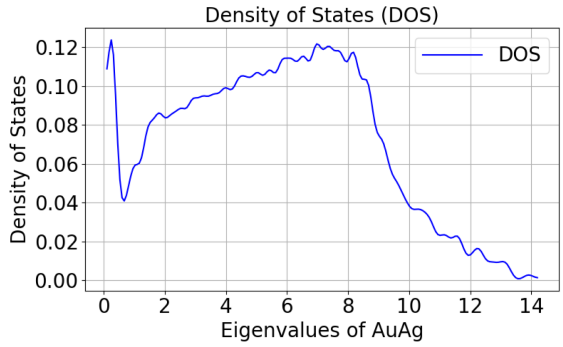
Example 2: In this example, we use two sequences of Hermitian eigenvalue problems. The first is the `NaCl` sequence of size 9273, used in [7]. There are 16 such matrices. We use `PHIDE` to compute the partial eigendecomposition of the 15th matrix by using the eigenvalues of the 14th matrix. The second is the `AuAg` with dimension $n = 13{,}379$, which originates from simulations using FLAPW [48] and has been used in [7]. There are 25 such matrices, and we compute the eigendecomposition of the 24th matrix by using the information from the 23 rd matrix. These two kinds of matrices have different density of states, essentially a histogram of the eigenvalues of a matrix, which are shown in Fig. 4.

The purpose of this example is to show that `PHIDE` works for highly related eigenvalue problems. Since the SCF loop is nearly converged, eigenvalues of the current and previous matrices are similar. Therefore, we can use the eigenvalues of the previous matrix to partition the slices and use its eigenvectors as good starting vectors. In this example, we compute the smallest $\texttt{nev} = 1000$, 2000 and 5000 eigenpairs and compare `PHIDE` with the latest ELPA.

Since Step (I) and Step (III) of `PHIDE` are identical to those in ELPA, we first compare `PHIDE` with Step (II) of ELPA. The



(a) NaCl



(b) AuAg

Fig. 4. DOS for matrices `NaCl` and `AuAg`.

timing results of `NaCl` and `AuAg` are presented in Table III. The column labeled `With` reports the execution time of FEAST when using eigenvectors from the previous matrix as starting vectors, while the `WO` column corresponds to runs with random starting vectors. Both timings include the data redistribution overhead, detailed in the `Redist` column. The results indicate that data redistribution incurs minimal overhead, accounting for no more than 6.91% of the total execution time of Step (II). The reason FEAST can be faster than Step (II) of ELPA is that it solves a banded eigenvalue problem with a small bandwidth, $n_{bw} = 64$. The timing breakdown of FEAST is reported in Table IV. The column labeled `Factor` corresponds to the LU factorization time of SPIKE [49], the `Solve` column shows the time spent solving the banded linear systems, and the `Iter.` column gives the number of subspace iterations performed by FEAST. The results demonstrate that both factorization and solution of banded systems are highly efficient.

The speedups over Step (II) of ELPA are shown in Fig. 5(a) when using different processes (64, 256, and 1024) to compute 1000, 2000 and 5000 eigenpairs, respectively. The speedup can be up to 1.72 and 2.23 when using 256 and 1024 processes, respectively. The total speedups of `PHIDE` over ELPA are shown in Fig. 5(b). `PHIDE` can be more than 22% faster than ELPA on average when using 1024 MPI processes.

The parameters used in our experiments are detailed below. When computing a large number of eigenpairs, such as $\texttt{nev} = 2000$ or 5000, we used 64 slices for $\texttt{np} = 64$ and 256, and 128 slices for $\texttt{np} = 1024$. The number of processes for **L2** parallelism was set to one, four, and eight, respectively. Note

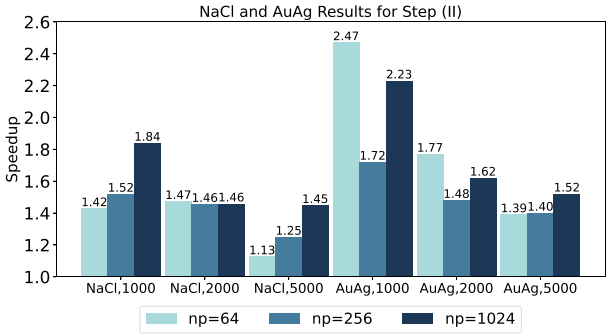
Authorized licensed use limited to: National Univ of Defense Tech. Downloaded on February 23,2026 at 02:36:26 UTC from IEEE Xplore. Restrictions apply.

TABLE III
THE RUNTIME (IN SECONDS) OF STEP (II) IN PHIDE WITH AND WITHOUT USING PREVIOUS EIGENVECTORS AS INITIAL VECTORS

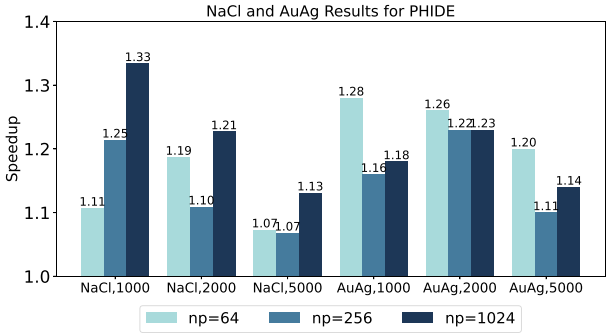
	np	nev = 1000				nev = 2000				nev = 5000			
		WO	**With**	Redist	ELPA	WO	**With**	Redist	ELPA	WO	**With**	Redist	ELPA
NaCl	64	6.74	5.21	1.49e-1	7.44	8.01	7.48	1.52e-1	11.02	23.17	20.32	2.15e-1	22.88
	256	3.10	2.24	9.47e-2	3.41	3.94	3.00	1.40e-1	4.36	6.84	5.84	1.60e-1	7.29
	1024	2.51	1.64	1.13e-1	3.01	2.98	2.16	1.24e-1	3.15	3.11	2.89	1.66e-1	4.19
AuAg	64	11.06	6.22	4.83e-2	15.38	13.03	12.63	9.45e-2	22.41	45.88	32.02	2.28e-1	44.62
	256	3.98	2.91	2.63e-2	5.01	6.65	4.84	4.20e-2	7.15	12.96	9.31	1.59e-1	12.99
	1024	1.58	1.58	3.89e-2	3.52	3.56	2.21	5.53e-2	3.58	4.92	4.02	1.32e-1	6.13

TABLE IV
THE RUNTIME BREAKDOWN (IN SECONDS) OF FEAST WHEN USING PREVIOUS EIGENVECTORS AS INITIAL VECTORS

	np	nev = 1000			nev = 2000			nev = 5000		
		Factor	Solve	Iter.	Factor	Solve	Iter.	Factor	Solve	Iter.
NaCl	64	3.50e-1	1.75	5	7.94e-1	4.20	7	7.16e-1	8.56	6
	256	1.07e-1	5.18e-1	5	1.78e-1	1.06	7	1.84e-1	2.22	4
	1024	9.40e-2	7.43e-1	7	9.10e-2	7.32e-1	13	9.00e-2	8.66e-1	8
AuAg	64	5.25e-1	3.57	4	1.01	6.68	7	1.02	13.16	7
	256	1.27e-1	9.47e-1	5	2.52e-1	2.76	7	2.63e-1	5.83	7
	1024	1.71e-1	3.80e-1	5	1.28e-1	6.87e-1	12	1.29e-2	1.11e-1	8



(a) The speedup over ELPA for Step (II).



(b) The total speedup of PHIDE over ELPA.

Fig. 5. The speedup of PHIDE over ELPA.

that **L2** parallelism means each linear system in (8) can be computed independently [24]. Since **L3** parallelism is not used in our experiment, each linear system is handled by a single MPI process, with the banded matrix redundantly stored on all processes. For a smaller number of computed eigenvalues, using too many slices may hinder convergence. For instance, when nev = 1000 with 128 slices, each slice contains only eight eigenvalues, potentially splitting a single eigenvalue cluster into multiple parts. To mitigate this issue, we used 64 slices and limited the banded eigenvalue solver to 512 when nev = 1000 and np = 1024, and the computed eigenvalue and eigenvectors were then redistributed across all processes.

1) Orthogonality: In this paper, the orthogonality and residual of the computed eigenpairs are measured as follows:

$$\text{Orth} = \|X^H X - I\|_F, \tag{11}$$

$$\text{Resid} = \|AX - X\Lambda\|_F, \tag{12}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, A is the input matrix, I is the identity matrix of size nev, and X and Λ represent the computed eigenvectors and eigenvalues, respectively.

Note that the orthogonality of the eigenvectors computed by FEAST is slightly less accurate than that achieved by direct methods such as ELPA, which generally reaches machine precision. Nevertheless, the computed eigenpairs are sufficiently accurate for self-consistent field (SCF) iterations. To further improve the orthogonality, a QR factorization can be applied to the eigenvectors. The resulting orthogonality, shown in Fig. 6, reaches a level comparable to ELPA, around 10^{-13}. Although QR orthogonalization introduces some computational overhead, it accounts for no more than 3% of the total runtime of PHIDE. Before reorthogonalization, the orthogonality is around 10^{-9}, which is less than those computed by direct methods, but the residuals are similar to these in Fig. 7.

The residuals of the computed eigenpairs are all approximately 10^{-11} or smaller when computing fewer than 5000 eigenpairs, consistent with the relative residual tolerance $\epsilon = 10^{-11}$ used by FEAST in this study. This is a reasonable value, and it is smaller than the residual tolerance of 10^{-10} used in the previous work [15]. Since FEAST solves linear systems as part of its procedure, the resulting eigenpairs are generally

Authorized licensed use limited to: National Univ of Defense Tech. Downloaded on February 23,2026 at 02:36:26 UTC from IEEE Xplore. Restrictions apply.

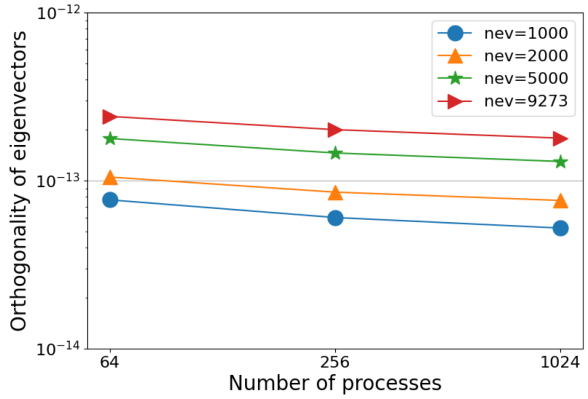


Fig. 6. Orthogonality of the computed eigenvectors for the NaCl matrix.

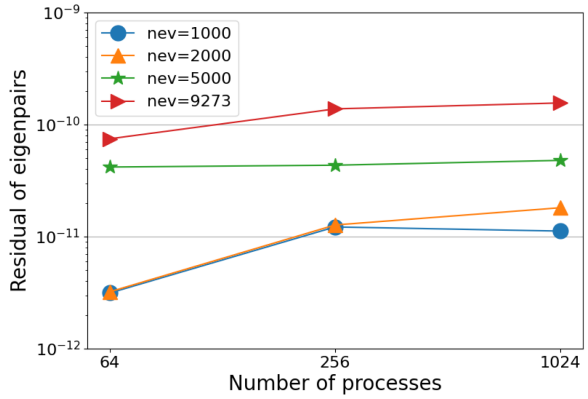


Fig. 7. Residuals of the computed eigenpairs for the NaCl matrix.

less accurate than those obtained by direct solvers or methods based on polynomial filtering [50]. The corresponding residual results are shown in Fig. 7, which are slightly larger than those of ELPA. Note that Figs. 6 and 7 also present the results of applying PHIDE to compute the full eigenvalue decomposition of the NaCl matrix. While the results remain satisfactory, the computed residuals are slightly larger compared to those obtained when computing a smaller subset of eigenpairs. Therefore, we do not recommend using PHIDE for full eigenvalue decomposition or in scenarios where high accuracy is required.

For matrices arising in the early stages of SCF iterations–such as the fifth matrix–the "good" estimated eigenvalues provide limited benefit. However, at these early stages, the eigenvalue problems only need to be solved approximately. Therefore, we set the residual tolerance to $\epsilon = 10^{-9}$ and use random initial vectors. Under these settings, we observe that Step (II) of PHIDE can still outperform ELPA in terms of speed, with results similar to those shown in Fig. 5(a).

During the early stages, it is also more difficult to reliably partition the eigenvalue interval into a large number of slices–e.g., more than 128. As a result, PHIDE is more suitable for the later stages of SCF iterations, where eigenvalue estimates are more reliable and higher accuracy is required.

Example 3: In this example, we use several relatively large matrices to evaluate the performance of our algorithm. Two matrices are selected from the SuiteSparse collection [34]: Si5H12, with $n = 19{,}896$ and $\texttt{nnz} = 379{,}247$, and H2O, with $n = 67{,}024$ and $\texttt{nnz} = 2{,}216{,}736$. The third matrix, CLIQ55296std, is taken from the ELSES matrix collection [35], with $n = 55{,}296$ and $\texttt{nnz} = 853{,}824$. All three matrices are real, symmetric, and sparse. The DOS of these three matrices are shown in Fig. 8.

The DOS of Si5H12 and H2O are similar, but that of CLIQ55296std is quite different.

Since there is only one matrix of each type, we generate three additional matrices by applying a small random symmetric perturbation to the nonzero entries, preserving the original sparsity structure and producing similar eigenvalue spectra. Each nonzero entry $a_{i,j}$ is perturbed according to

$$a_{i,j} = a_{i,j}(1 + \tau\eta_{i,j}), \tag{13}$$

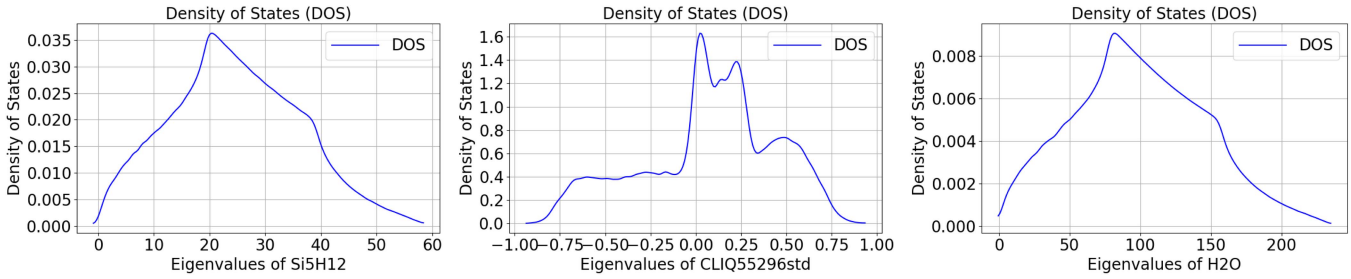
where $\tau = 10^{-4}$ and $\eta_{i,j}$ is a uniformly distributed random number in the interval $[0, 1]$. In this example, we compute the smallest 5000 eigenpairs and use 64, 256 and 1024 processes, respectively. For these experiments, the convergence relative residual tolerance is also $\epsilon = 10^{-11}$, and the dimension of the subspace is $m_i^{(0)} = \max(1.3 \times m_i, m_i + 10)$, where m_i is the estimated number of eigenvalues in the current slice. We note that $m_i^{(0)}$ can significantly affect performance. When $m_i^{(0)}$ is small, it makes linear systems with fewer right-hand sides much easier to solve. When $m_i^{(0)}$ is too small, it may prevent FEAST from converging. There is a trade-off between speed and robustness.

The speedups of FEAST over Step (II) of ELPA are reported in Table V. The results indicate that predicting the performance advantage of spectrum slicing algorithms over Step (II) of ELPA is challenging, primarily because the quality of the spectrum partitioning significantly impacts the efficiency of these algorithms. In particular, determining the optimal number of slices is nontrivial, as it depends on both the eigenvalue distribution and the specific spectrum slicing method employed. Among various factors, we observe that achieving good eigenvalue separation is more critical for accuracy than merely balancing the computational load. If a partition boundary lies within a dense cluster of eigenvalues, it can lead to convergence issues and poor orthogonality. To address this, we adopt a simplified *k*-means algorithm to partition the spectrum while also considering load balancing. For the three large matrices studied–each requiring the computation of thousands of eigenvalues–we find that previously computed eigenvectors offer limited benefit. In fact, using random initial vectors can sometimes result in faster performance than Step (II) of ELPA.

The total speedups of PHIDE over ELPA are shown in Fig. 9, which are always larger than 1.17 and up to 1.23 when using 1024 processes. Note that to improv the orthogonality of computed eigenvectors, we applied the QR factorization to the computed eigenvectors, and the times are included in PHIDE. The wall times of FEAST used to compute Step (II) are shown in Table V, and the speedup can be up to $7.05\times$. The speedups of FEAST over Step (II) of ELPA are shown in Fig. 10.

In Fig. 9, the observed speedup tends to decrease as the number of processes np increases, particularly when $\texttt{np} = 1024$.

Authorized licensed use limited to: National Univ of Defense Tech. Downloaded on February 23,2026 at 02:36:26 UTC from IEEE Xplore. Restrictions apply.



(c) Si5H12 (d) CLIQ55296std (d) H2O

Fig. 8. DOS for the large matrices.

TABLE V
THE EXECUTION TIME (IN SECONDS) OF STEP (II) FOR LARGE MATRICES

np	Si5H12				CLIQ55296std				H2O			
	WO	**With**	Redist	ELPA	WO	**With**	Redist	ELPA	WO	**With**	Redist	ELPA
64	32.94	27.17	1.68e-01	49.82	77.17	65.83	6.04e-01	445.0	108.67	95.03	6.55e-01	669.50
256	9.58	9.31	6.48e-02	14.78	26.79	25.62	1.81e-01	117.18	50.43	41.50	2.40e-01	179.99
1024	5.71	5.49	3.48e-02	8.30	9.17	8.27	7.61e-02	39.95	19.73	13.57	8.64e-02	57.45

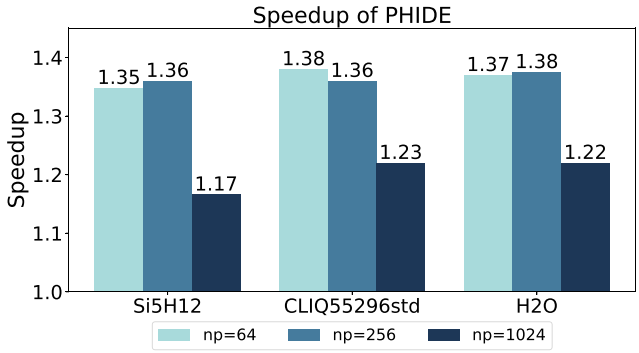


Fig. 9. Total speedups of PHIDE over ELPA for large matrices.

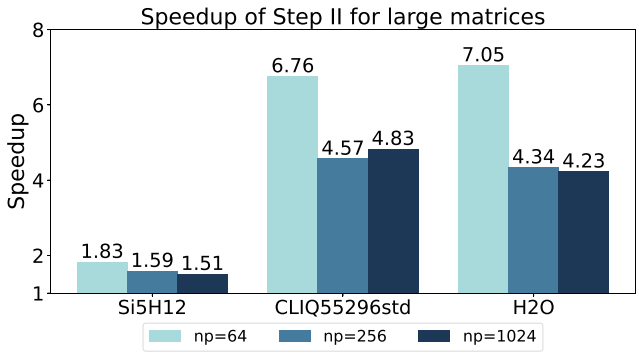


Fig. 10. Speedup of FEAST over Step (II) of ELPA for large matrices.

We conjecture that for these large matrices, ELPA continues to exhibit good scalability at this level of parallelism. As a result, the relative cost of Step (II) in ELPA decreases with increasing np, as shown in Table VI. However, when np increases further and ELPA approaches its scalability bottleneck, the performance advantage of PHIDE over ELPA may become more pronounced. This trend is consistent with the results shown in Fig. 5(b), where

TABLE VI
TIME RATIO OF STEP (II) IN ELPA FOR LARGE MATRICES

np	Si5H12	CLIQ55296std	H2O
64	42.49%	56.26%	35.52%
256	36.64%	52.17%	31.61%
1024	31.03%	41.04%	26.24%

PHIDE achieves higher speedups for matrices with smaller dimensions when using $np = 1024$ processes.

V. CONCLUSIONS AND FUTURE WORKS

In this paper, we propose a new framework for computing a partial eigendecomposition of dense Hermitian or real symmetric matrices, which can be easily extended to generalized eigenvalue problems. This new framework is denoted by PHIDE, which combines classical direct methods with spectrum slicing algorithms. In this work, we combine ELPA [46] with FEAST [24] as both packages are implemented in Fortran, making their integration straightforward. PHIDE can also be combined with other spectrum slicing algorithms such as those implemented in SLEPc [19] or EVSL [23]. This will be a topic for future work.

In PHIDE, a dense matrix is first reduced to a banded form, after which spectrum slicing algorithms are employed to compute its partial eigendecomposition–bypassing the traditional reduction to tridiagonal form. As a result, the symmetric bandwidth reduction (SBR) process, which involves memory-bound operations, is avoided. In this work, the resulting banded linear systems are solved using the SPIKE algorithm [49]. In PHIDE, the banded eigenvalue problem is solved using the contour integral method implemented in FEAST, which may yield slightly lower accuracy than tridiagonalization-based approaches. Numerical experiments were conducted on a domestic supercomputer based on ARM architecture developed in China, utilizing up to 1024

Authorized licensed use limited to: National Univ of Defense Tech. Downloaded on February 23,2026 at 02:36:26 UTC from IEEE Xplore. Restrictions apply.

processes. The test matrices include dense Hermitian matrices from real-world applications, as well as large-scale real symmetric sparse matrices from the SuiteSparse [34] and ELSES [35] collections. For thses large matrices, using random initial vectors can also make PHIDE outperform ELPA. For sequences of correlated eigenvalue problems arising from self-consistent field (SCF) calculations, PHIDE achieves a speedup of up to $1.38\times$ compared to ELPA.

The primary objective of this work is to demonstrate that the newly proposed PHIDE framework effectively solves sequences of Hermitian eigenvalue problems and, in some cases, outperforms ELPA in terms of computational efficiency. Several directions are worth exploring in future work. One is the investigation of alternative spectrum partitioning strategies beyond *k*-means clustering. Another is the enhancement of eigenvector orthogonality across different slices–a known challenge in spectrum slicing methods [51]. In this work, we simply use QR factorization. Moreover, integrating PHIDE into the ELSI infrastructure [52] or into electronic-structure packages such as QUANTUM ESPRESSO [53] would facilitate its application in large-scale simulations. In addition, we plan to extend our algorithm to GPU architectures. However, this requires the development of an efficient GPU-based spectrum slicing algorithm, which is beyond the scope of this paper.

REFERENCES

[1] W. Kohn, "Nobel lecture: Electronic structure of matter—wave functions and density functionals," *Rev. Modern Phys.*, vol. 71, no. 5, 1999, Art. no. 1253.

[2] P. Salas, L. Giraud, Y. Saad, and S. Moreau, "Spectral recycling strategies for the solution of nonlinear eigenproblems in thermoacoustics," *Numer. Linear Algebra Appl.*, vol. 22, no. 6, pp. 1039–1058, 2015.

[3] J. S. Blackford et al., *ScaLAPACK Users' Guide*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1997.

[4] A. Marek et al., "The ELPA library: Scalable parallel eigenvalue solutions for electronic structure theory and computational science," *J. Phys.: Condens. Matter*, vol. 26, pp. 1–15, 2014.

[5] T. Imamura, S. Yamada, and M. Yoshida, "Development of a high-performance eigensolver on a peta-scale next-generation supercomputer system," *Prog. Nucl. Sci. Technol.*, vol. 2, pp. 643–650, 2011.

[6] E. D. Napoli and M. Berljafa, "Block iterative eigensolvers for sequences of correlated eigenvalue problems," *Comput. Phys. Commun.*, vol. 184, no. 11, pp. 2478–2488, 2013.

[7] J. Winkelmann, P. Springer, and E. D. Napoli, "ChASE: Chebyshev accelerated subspace iteration eigensolver for sequences of hermitian eigenvalue problems," *ACM Trans. Math. Softw.*, vol. 45, no. 2, pp. 1–34, 2019.

[8] J. Poulson, B. Marker, R. A. van de Geijn, J. R. Hammond, and N. A. Romero, "Elemental: A new framework for distributed memory dense matrix computations," *ACM Trans. Math. Softw.*, vol. 39, no. 2, pp. 13:1–13:24, 2013.

[9] X. Wu, D. Davidović, S. Achilles, and E. Di Napoli, "ChASE: A distributed hybrid CPU-GPU eigensolver for large-scale hermitian eigenvalue problems," in *Proc. Platform Adv. Sci. Comput. Conf.*, 2022, pp. 1–12.

[10] X. Wu and E. Di Napoli, "Advancing the distributed multi-GPU ChASE library through algorithm optimization and NCCL library," in *Proc. Workshops Int. Conf. High Perform. Comput., Netw., Storage, Anal.*, 2023, pp. 1688–1696.

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

[12] A. V. Knyazev, "Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method," *SIAM J. Sci. Comput.*, vol. 23, no. 2, pp. 517–541, 2001.

[13] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van derVorst, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. Philadelphia, PA, USA: SIAM, 2000.

[14] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*. Philadelphia, PA, USA: SIAM, 2011.

[15] H. M. Aktulga, L. Lin, C. Haine, E. G. Ng, and C. Yang, "Parallel eigenvalue calculation based on multiple shift–invert Lanczos and contour integral based spectral projection method," *Parallel Comput.*, vol. 40, no. 7, pp. 195–212, 2014.

[16] R. G. Grimes, J. G. Lewis, and H. D. Simon, "A shifted block lanczos algorithm for solving sparse symmetric generalized eigenproblems," *SIAM J. Matrix Anal. Appl.*, vol. 15, no. 1, pp. 228–272, 1994.

[17] D. B. Williams-Young, P. G. Beckman, and C. Yang, "A shift selection strategy for parallel shift-invert spectrum slicing in symmetric self-consistent eigenvalue computation," *ACM Trans. Math. Softw.*, vol. 46, no. 4, pp. 1–31, 2020.

[18] H. Zhang, B. Smith, M. Sternberg, and P. Zapol, "SIPs: Shift-and-invert parallel spectral transformations," *ACM Trans. Math. Softw.*, vol. 33, no. 2, pp. 1–19, 2007.

[19] C. Campos and J. E. Roman, "Strategies for spectrum slicing based on restarted lanczos methods," *Numer. Algorithms*, vol. 60, no. 2, pp. 279–295, 2012.

[20] M. Keçeli et al., "SIESTA-SIPs: Massively parallel spectrum-slicing eigensolver for an *ab initio* molecular dynamics package," *J. Comput. Chem.*, vol. 39, no. 22, pp. 1806–1814, 2018.

[21] A. S. Banerjee, L. Lin, W. Hu, C. Yang, and J. E. Pask, "Chebyshev polynomial filtered subspace iteration in the discontinuous galerkin method for large-scale electronic structure calculations," *J. Chem. Phys.*, vol. 145, no. 15, 2016, Art. no. 154101.

[22] R. Li, Y. Xi, E. Vecharynski, C. Yang, and Y. Saad, "A thick-restart lanczos algorithm with polynomial filtering for hermitian eigenvalue problems," *SIAM J. Sci. Comput.*, vol. 38, no. 4, pp. A2512–A2534, 2016.

[23] R. Li, Y. Xi, L. Erlandson, and Y. Saad, "The eigenvalues slicing library (EVSL): Algorithms, implementation, and software," *SIAM J. Sci. Comput.*, vol. 41, no. 4, pp. C393–C415, 2019.

[24] J. Kestyn, V. Kalantzis, E. Polizzi, and Y. Saad, "PFEAST: A high performance sparse eigenvalue solver using distributed-memory linear solvers," in *Proc. IEEE Int. Conf. High Perform. Comput., Netw., Storage Anal.*, 2016, pp. 178–189.

[25] E. Polizzi, "Density-matrix-based algorithm for solving eigenvalue problems," *Phys. Rev. B*, vol. 79, no. 11, 2009, Art. no. 115112.

[26] P. T. Peter Tang and E. Polizzi, "FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection," *SIAM J. Matrix Anal. Appl.*, vol. 35, no. 2, pp. 354–390, 2014.

[27] T. Sakurai and H. Sugiura, "A projection method for generalized eigenvalue problems using numerical integration," *J. Comput. Appl. Math.*, vol. 159, pp. 119–128, 2003.

[28] J. Demmel, *Applied Numerical Linear Algebra*. Philadelphia, PA, USA: SIAM, 1997.

[29] A. Haidar, P. Luszczek, J. Kurzak, and J. Dongarra, "An improved parallel singular value algorithm and its implementation for multicore hardware," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Denver, CO, USA, 2013, pp. 90:1–90:14.

[30] R. D. Falgout and U. M. Yang, "Hypre: A library of high performance preconditioners," in *Proc. Int. Conf. Comput. Sci.*, 2002, pp. 632–641.

[31] S. Balay et al., PETSc/TAO users manual Revision 3.19," 2023.

[32] M. A. Heroux et al., "An overview of the trilinos project," *ACM Trans. Math. Softw.*, vol. 31, no. 3, pp. 397–423, 2005.

[33] S. Li et al., "Efficient data redistribution algorithms from irregular to block cyclic data distribution," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 3667–3677, Dec. 2022.

[34] T. A. Davis and Y. Hu, "The university of florida sparse matrix collection," *ACM Trans. Math. Softw.*, vol. 38, no. 1, pp. 1–25, 2011.

[35] T. Hoshi et al., "Extremely scalable algorithm for 108-atom quantum material simulation on the full system of the k computer," in *Proc. 7th IEEE Workshop Latest Adv. Scalable Algorithms Large-Scale Syst.*, 2016, pp. 33–40.

[36] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed.Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[37] C. Bischof, B. Lang, and X. Sun, "Parallel tridiagonalization through two-step band reduction," in *Proc. IEEE Scalable High- Perform. Comput. Conf.*, Los Alamitos, CA, 1994, pp. 23–27.

[38] C. H. Bischof, B. Lang, and X. B. Sun, "Algorithm 807: The SBR toolbox-software for successive band reduction," *ACM Trans. Math. Softw.*, vol. 26, no. 4, pp. 602–616, 2000.

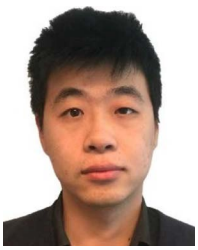
[39] M. Gu and S. C. Eisenstat, "A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem," *SIAM J. Matrix Anal. Appl.*, vol. 16, pp. 172–191, 1995.

Authorized licensed use limited to: National Univ of Defense Tech. Downloaded on February 23,2026 at 02:36:26 UTC from IEEE Xplore. Restrictions apply.

[40] I. S. Dhillon, B. N. Parlett, and C. Vömel, "The design and implementation of the MRRR algorithm," *ACM Trans. Math. Softw.*, vol. 32, no. 4, pp. 533–560, 2006.
[41] S. Guttel, E. Polizzi, P. T. P. Tang, and G. Viaud, "Zolotarev quadrature rules and load balancing for the FEAST eigensolver," *SIAM J. Sci. Comput.*, vol. 37, no. 4, pp. A2100–A2122, 2015.
[42] L. Lin, Y. Saad, and C. Yang, "Approximating spectral densities of large matrices," *SIAM Rev.*, vol. 58, no. 1, pp. 34–65, 2016.
[43] Y. Xi, R. Li, and Y. Saad, "Fast computation of spectral densities for generalized eigenvalue problems," *SIAM J. Sci. Comput.*, vol. 40, no. 4, pp. A2749–A2773, 2018.
[44] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.
[45] L. Kaufman, "The retraction algorithm for factoring banded symmetric matrices," *Numer. Linear Algebra Appl.*, vol. 14, no. 3, pp. 237–254, 2007.
[46] T. Auckenthaler et al., "Parallel solution of partial symmetric eigenvalue problems from electronic structure calculations," *Parallel Comput.*, vol. 37, no. 12, pp. 783–794, 2011.
[47] S. Li, X. Liao, Y. Lu, J. E. Roman, and X. Yue, "A parallel structured banded DC algorithm for symmetric eigenvalue problems," *CCF Trans. High Perform. Comput.*, vol. 5, pp. 116–128, 2023.
[48] H. Jansen and A. J. Freeman, "Total-energy full-potential linearized augmented-plane-wave method for bulk solids: Electronic and structural properties of tungsten," *Phys. Rev. B*, vol. 30, no. 2, 1984, Art. no. 561.
[49] E. Polizzi and A. Sameh, "SPIKE: A parallel environment for solving banded linear systems," *Comput. Fluids*, vol. 36, no. 1, pp. 113–120, 2007.
[50] Z. Jia and K. Zhang, "A FEAST SVDsolver based on Chebyshev–Jackson series for computing partial singular triplets of large matrices," *J. Sci. Comput.*, vol. 97, no. 1, 2023, Art. no. 21.
[51] G. Schofield, J. R. Chelikowsky, and Y. Saad, "A spectrum slicing method for the Kohn–Sham problem," *Comput. Phys. Commun.*, vol. 183, no. 3, pp. 497–505, 2012.
[52] V. W. Yu et al., "ELSI–an open infrastructure for electronic structure solvers," *Comput. Phys. Commun.*, vol. 256, 2020, Art. no. 107459.
[53] P. Giannozzi et al., "QUANTUM ESPRESSO: A modular and open-source software project for quantum simulations of materials," *J. Phys.: Condens. Matter*, vol. 21, no. 39, 2009, Art. no. 395502.



Shengguo Li received the BS, MS and PhD degrees in computational mathematics from National University and Defense Technology (NUDT), Changsha, China. He is currently an associate professor with the College of Computer Science and Technology, NUDT. His research interests include numerical linear algebra, high performance computing, and machine learning.



Xinzhe Wu received the PhD degree in computer science from Université de Lille and Maison de la Simulation, France, in 2019. He is currently a scientific researcher with the Jülich Supercomputing Centre, Forschungszentrum Jülich, Germany. His research interests include numerical linear algebra, iterative methods for large-scale linear systems and eigenvalue problems, high-performance computing on heterogeneous architectures with a focus on GPUs, and software design for scalable numerical linear algebra libraries.



Jose E. Roman received the MSc and PhD degrees in computer science from Universitat Politecnica de Valencia (UPV), Spain. He is currently a professor with the School of Computer Science with UPV. His research interests focus mainly on software engineering for large-scale scientific computing, especially in the field of numerical solution of large-scale eigenvalue problems. He is the lead developer of SLEPc, a parallel software library for eigenvalue computations. Other topics of interest are numerical linear algebra, matrix functions, matrix equations, and high performance computing.



Ziyang Yuan received the MS and PhD degrees in mathematics from the National University of Defense Technology (NUDT), Changsha, China, in 2017 and 2021, respectively. He is currently an assistant professor with the Academy of Military Sciences. His research interests include information theory, high-dimensional probability, optimization theory, inverse problems, intelligent signal processing, and social computation.



Ruibo Wang received the PhD degree in computer science from the National University of Defense Technology (NUDT), Changsha, China, in 2011. He is currently a professor with the College of Computer Science and Technology, NUDT. As a key member of the research and development team, he has made significant contributions to the development of multiple generations of Tianhe supercomputers. His research interests include operating systems, file systems and resource scheduling in the field of high-performance computing.



Tiejun Li received the PhD degree in computer science from the National University of Defense Technology (NUDT), Changsha, China. He is currently a chair professor with the College of Computer Science and Technology, NUDT and the director of a series of research projects. His research interests include high performance computing and microprocessor architecture.



Yi Xie received the MS degree in computer science from the National University of Defense Technology (NUDT), Changsha, China. His research interests align closely with the fields of numerical linear algebra, high performance computing, and machine learning.



Bo Yang received the MS and PhD degrees in computer science and technology from National University and Defense Technology (NUDT), Changsha, China. He is currently an associate professor in the College of Computer Science and Technology, NUDT. His research interests include performance model, communication optimization, and high performance computing.



Xuguang Chen received the BS degree from the College of Science, National University of Defense Technology (NUDT), Changsha, China. He is an assistant professor with the College of Computer Science and Technology, NUDT. His research interests include high performance computing and structural finite element simulation.

Authorized licensed use limited to: National Univ of Defense Tech. Downloaded on February 23,2026 at 02:36:26 UTC from IEEE Xplore. Restrictions apply.