Solving Millions of Eigenvectors in Large-Scale Quantum-Many-Body-Theory Computations

Alexey Tal
VASP Software GmbH
Vienna, Austria
alexey.tal@vasp.at

Martijn Marsman
VASP Software GmbH
Vienna, Austria
martijn.marsman@univie.ac.at

Georg Kresse
University of Vienna
Vienna, Austria
georg.kresse@univie.ac.at

Anton Anders
NVIDIA
Santa Clara, CA, USA
antona@nvidia.com

Samuel Rodriguez
NVIDIA
Santa Clara, CA, USA
srodriguezbe@nvidia.com

Kyungjoo Kim
NVIDIA
Albuquerque, NM, USA
kyungjook@nvidia.com

Alexander Kalinkin
NVIDIA
Santa Clara, CA, USA
akalinkin@nvidia.com

Alexey Romanenko
NVIDIA
Yerevan, Armenia
aromanenko@nvidia.com

Matthias Noack
NVIDIA
Berlin, Germany
mnoack@nvidia.com

Patrick Atkinson
NVIDIA
Bristol, UK
patkinson@nvidia.com

Stefan Maintz
NVIDIA
Zurich, Switzerland
smaintz@nvidia.com

***Abstract*—We present large-scale simulations of photovoltaics materials in the Vienna Ab initio Simulation Package [1] (VASP) that are only possible by pushing the boundaries of practically solvable eigenproblems. Thus, we enable analyses for semiconductors that were inaccessible within state-of-the-art predictive methods before. To achieve this, we a) implemented a distributed eigensolver in cuSOLVERMp that fully runs on the GPU, b) adapted VASP's Bethe-Salpeter Equation (BSE) algorithm for GPUs and to employ cuSOLVERMp, and c) dramatically improved the BSE workload distribution to yield near perfect load-balancing at scale. With a size of two million, we solve one of the largest dense, complex eigenproblems reported so far in under 5 hours sustaining 7.8 PFLOPS using 34.5 GJ on 4096 GPUs of NVIDIA's Selene supercomputer. Our results facilitate new breakthroughs in material science. These improvements in compute and energy efficiency apply to other domains relying on solving large eigenproblems, as well.**

***Index Terms*—High Performance Computing, Accelerator Architectures, Linear Algebra, Distributed Eigensolvers, Quantum-Many-Body-Theory Simulations, Quantum Chemistry, Bethe-Salpeter Equation, VASP, Photovoltaics**

I. Introduction

Breaking the ground for new results in scientific high performance computing requires joint innovations on multiple levels, for instance, at the domain-specific application codes implementing specific mathematical models, and at the highly optimized numerical solver libraries that make sure hardware such as GPUs is fully utilized. In this work, we present advances in implementing a distributed GPU-only eigensolver, integrate it into VASP's implementation of the Bethe-Salpeter-Equation (BSE) method, and use it to compute properties of complex photovoltaic materials inaccessible to this method before. This involves solving one of the largest dense, complex eigenproblems reported so far with a dimension of two million within 5 hours at 7.8 PFLOPS on the Selene supercomputer using 4096 GPUs.

This paper is structured as follows: The remainder of this section introduces the various problems we address. Thereafter, we provide an overview of related work (Section II), describe the innovations realized on the different layers (Section III), and present the insights gained for the simulated materials (Section IV). Sections V and VI then provide the benchmark methodology and detailed performance analysis. The paper concludes with a summary discussing the implications of our results (Section VII).

A. Predicting Optical Properties

The ongoing search for materials capable of efficiently converting solar energy into electricity has become one of the main priorities in the collaborative effort to reduce carbon dioxide emissions and transition to green technologies. Solar panels have shown a steep increase in efficiency reaching 25% in just a few years [2]. The properties of materials that make them efficient for photovoltaic (PV) applications are grounded in their ability to absorb solar radiation, which is determined by the energy separating the electronic valence states from the conduction states, i.e., the band gap. Another crucial property of these materials is the energy required to move apart an electron excited by incoming light and a positive charge formed in-place of the excited electron called "hole" as shown schematically in Fig. 1. This binary system of an excited electron and a hole is known as exciton and their binding strength is called the exciton binding energy. The band gap and the exciton binding energy are the fundamental properties that define the efficiency of the material for PV.

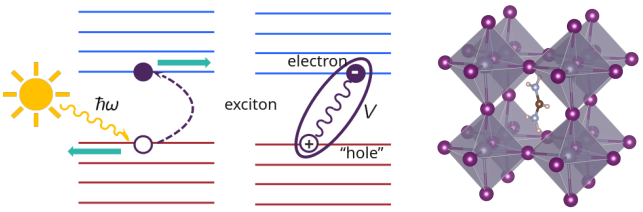

Fig. 1. Left: Schematic representation of the exciton formation in the process of photoabsorption. Right: The atomic structure of $FASnI_3$. The coloring scheme is the following: I: magenta, Sn: purple, N: blue, C: brown, H: white.

Among the most promising materials for such applications is a class of systems with the *perovskite* atomic lattice structure. The perovskite structure offers a great variety of semiconducting materials, which exhibit suitable band gaps and a small exciton binding energy which is ultimately required for an efficient PV system. Furthermore, perovskite materials are cheap and easy to produce, stable at elevated temperatures, and non-toxic, which overall makes them a very good candidate for high-efficiency solar panels. The formula OMX_3 shows that cubic perovskite materials consist of three components: an organic cation O, a metal M, and a halide X. This leads to a vast number of combinations that can potentially be explored [3].

Computational modeling has sped up the process to discover and synthesize materials in PV research significantly, while the search for the ideal perosvkite system remains a very challenging task. Most theoretical modeling is driven by calculations based on the density-functional theory (DFT), which is incredibly successful and warranted a Noble Prize to its authors Kohn and Pople [4]. However, while DFT can inexpensively treat materials consisting of thousands of atoms on HPC hardware, it is fundamentally a ground-state theory [5]. The excitation effects required in PV research cannot be described within DFT. The Many-Body Perturbation Theory (MBPT) provides the means to accurately describe the excitations. Thus, the GW approximation within MBPT is considered state-of-the-art for the band-gap predictions in solids. However, the increased computational cost of the approach typically limits simulations to materials composed of several hundreds of atoms. As shown by Ben *et al.* [6], for a system of 2,742 atoms, the GW approach can greatly benefit from running the calculations on GPUs. Nevertheless, the GW approximation does not account for the interaction between excited particles, which makes it inadequate for describing the photoabsorption process in semiconducting materials.

Here, we will focus entirely on the approach capable of accurately describing the excitation of electrons due to an incoming photon in crystalline materials. This is done by solving the Bethe-Salpeter Equation (BSE) [7]. The BSE is an integro-differential equation dependent on four spatial and time coordinates. It was derived from quantum-many-body-theory (QMBT) for describing a two-particle bound state, such as an exciton. The standard approach for solving this equation is to map it onto an eigenvalue problem. Furthermore, by neglecting the coupling between excitations and de-excitations in the Tamm-Dancoff approximation (TDA) [8], the problem is reduced further to the Hermitian eigenvalue problem

$$HX = X\Omega, \tag{1}$$

where H is a two-particle Hamiltonian describing the propagation of the interacting electron-hole pairs, Ω represents the eigenvalues corresponding to the two-particle bound state eigenenergy, and X contains the eigenvectors describing the two-particle quantum state. In PV research, the detailed analysis of the excitations often requires all the eigenvectors as well as the eigenvalues.

The dimension of the Hamiltonian scales as N^3 with the size of the system. Thus, the overall problem of finding the solution of Eq. 1 scales as N^6 with state-of-the-art eigensolvers. The scaling of the method has hitherto limited it to materials that can be represented by not more than a few atoms.

B. The Eigenvalue Problem

A general procedure to solve the eigenproblem described in Eq. 1 consists of three phases: 1) apply the orthogonal transformation to reduce the given Hermitian matrix H to a tridiagonal form $Q^H HQ = T$, 2) compute all eigenpairs of $T = VDV^H$, and 3) backtransform the eigenvectors $H = QVD(QV)^H$. Thus, D is a diagonal matrix including eigenvalues and QV is the eigenvectors of the Hermitian matrix H. Implementing these functions efficiently on a distributed GPU platform is a challenging task, as it requires achieving a high degree of concurrency for all building blocks, which have diverse communication patterns and computational intensity. For instance, the reduction to a tridiagonal form and backtransformation of the eigenvectors use block Householder transformations [9] to exploit efficient matrix-matrix multiplication, which can be leveraged by using cuBLAS. On the other hand, our implementation for the eigensolution of the tridiagonal form, based on Cuppen's divide-and-conquer algorithm [10], benefits little from cuBLAS and cuSOLVER. Different kinds of parallelism need to be exploited traversing the resulting binary tree: at the bottom level, there are many small independent subproblems, while the problem sizes increase towards the root. To achieve efficient parallelism for all levels, we need new kernels, e.g., a batched version of the QR algorithm and root-finding kernels for the secular equations. Another important aspect of the parallel implementation on a distributed GPU cluster is to use fast and non-blocking communications among GPUs enabling kernels to run asynchronously. To support various communication libraries and cutting-edge interconnects, cuSOLVERMp introduces the Communication Abstraction Library (CAL). This allows cuSOLVERMp to use the best available transport layers across a variety of systems, e.g., UCX or NCCL for different communication sizes and collectives. Developing an efficient and scalable eigensolver for GPU supercomputers required solving the above problems and harnessing all the components together.

II. Related Work

A. Solving the Bethe-Salpeter Equation

The calculations for systems with a BSE Hamiltonian matrix of ranks up to 200,000 are performed routinely on modern hardware. Beside the VASP [1], [11], [12] implementation, BerkeleyGW [13] and BSEPACK [14] address solving the BSE. To our best knowledge, the BSE method is GPU accelerated in BerkeleyGW [13], but we are not aware of reported scientific applications at a comparable scale. BSEPACK [14] which goes beyond the TDA even has been combined with ELPA (Eigenvalue soLvers for Petaflop Applications) [15] for GPU support [16], with the largest matrix rank reported being 125,000. Hence, our order-of-magnitude increase of the matrix rank n for typical BSE jobs leads to a 1,000x increase of the problem size in light of the $O(n^3)$ complexity.

This improvement is crucial for complex materials such as perovskites, where a large number of k-points is required to sample the material on a three-dimensional grid in reciprocal space. At the same time, the number of wavefunctions grows very quickly with the number of atoms in the material. The BSE matrix is of rank $N_k \times N_c \times N_v$, where N_k is the number of k-points, N_c the number of the treated conduction wavefunctions and N_v the number of valence wavefunctions. Hence, the rank of the BSE matrix quickly reaches one million and more for these complex materials, which hitherto has been impossible to solve without crude approximations.

The BSE Hamiltonian in VASP [1], [11], [12] is represented in the basis of electron-hole pairs and consists of three terms:

$$\begin{aligned} H_{vc\mathbf{k}}^{v'c'\mathbf{k}'} =& (\varepsilon_{c\mathbf{k}} - \varepsilon_{v\mathbf{k}})\delta_{vv'}\delta_{cc'}\delta_{\mathbf{k}\mathbf{k}'} \\ &+ \langle c\mathbf{k}, v'\mathbf{k}'|V|v\mathbf{k}, c'\mathbf{k}'\rangle \\ &- \langle c\mathbf{k}, v'\mathbf{k}'|W|c'\mathbf{k}', v\mathbf{k}\rangle. \end{aligned} \tag{2}$$

The computation of the last two terms of Eq. 2 poses the main challenge as it requires computing the integrals

$$\begin{aligned} W_{vc\mathbf{k}}^{v'c'\mathbf{k}'} &= \langle c\mathbf{k}, v'\mathbf{k}'|W|c'\mathbf{k}', v\mathbf{k}\rangle \\ &= \int d\mathbf{r}d\mathbf{r}'\phi^*_{c\mathbf{k}}(\mathbf{r})\phi^*_{v'\mathbf{k}'}(\mathbf{r}')W(\mathbf{r},\mathbf{r}')\phi_{c'\mathbf{k}'}(\mathbf{r})\phi_{v\mathbf{k}}(\mathbf{r}'), \end{aligned} \tag{3}$$

and similar integrals for V. Thus, in VASP, the matrix elements of the Hamiltonian are computed with the following algorithm:

Algorithm 1 Algorithm for computing matrix elements of W

for $k \leftarrow 1$ to N_k **do**
  for $k' \leftarrow k$ to N_k **do**
    $\rho_{c\mathbf{k},c'\mathbf{k}'}(\mathbf{r}) = \phi^*_{c\mathbf{k}}(\mathbf{r})\phi_{c'\mathbf{k}'}(\mathbf{r})$
    $\rho_{c\mathbf{k},c'\mathbf{k}'}(\mathbf{G}) = \mathrm{FFT}\{\rho_{c\mathbf{k},c'\mathbf{k}'}(\mathbf{r})\}$
    $\tilde{\rho}_{c\mathbf{k},c'\mathbf{k}'}(\mathbf{G}) = \mathrm{ZGEMM}\{W(\mathbf{G},\mathbf{G}'), \rho_{c\mathbf{k},c'\mathbf{k}'}(\mathbf{G}')\}$

    $\rho_{v\mathbf{k},v'\mathbf{k}'}(\mathbf{r}') = \phi^*_{v'\mathbf{k}'}(\mathbf{r}')\phi_{v\mathbf{k}}(\mathbf{r}')$
    $\rho_{v\mathbf{k},v'\mathbf{k}'}(\mathbf{G}') = \mathrm{FFT}\{\rho_{v\mathbf{k},v'\mathbf{k}'}(\mathbf{r}')\}$

    $W_{vc\mathbf{k}}^{v'c'\mathbf{k}'} = \mathrm{ZGEMM}\{\tilde{\rho}_{c\mathbf{k},c'\mathbf{k}'}(\mathbf{G}), \rho_{v\mathbf{k},v'\mathbf{k}'}(\mathbf{G}')\}$
  end for
end for

First, the product of the wavefunctions in real space is computed. The calculated density is then transformed into the reciprocal space via the Fourier transform. Finally, the screened potential is applied and the two overlap densities are multiplied. The matrix elements $V_{vc\mathbf{k}}^{v'c'\mathbf{k}'}$ are computed in the same loop for the corresponding indices. The next step after that is solving the eigenproblem via a distributed eigensolver for the energies and quantum states of the excitations. In this work, we contribute a) a scalable way to set up the matrices, and b) a distributed GPU-only eigensolver.

B. Distributed Parallel Eigensolvers

ScaLAPACK is widely recognized as the standard library for distributed dense linear algebra operations [17], [18]. It offers a vast range of solvers and is designed for CPU-only systems, with most of its performance coming from underlying calls to BLAS and LAPACK libraries [19], [20]. ScaLAPACK includes routines for solving eigenproblems for real-valued (e.g. `PDSYEVD`) and complex-valued (e.g. `PZHEEVD`) matrices. The eigensolver in ScaLAPACK only supports one-stage tridiagonalization and solves a tridiagonal eigenproblem using the divide-and-conquer algorithm [21].

Another project for solving large-scale eigenproblems is ELPA. It contains two distinct methods of solving dense eigenproblems: a one-stage and a two-stage solver, these are commonly referred to as ELPA1 and ELPA2, respectively. Both of these solvers have GPU support [22]. ELPA's one-stage solver performs the tridiagonalization in a single step, employing a similar method to the ScaLAPACK routines `PDSYEVD` and `PZHEEVD`. Whilst the two-stage solver performs the tridiagonalization by first reducing a full matrix to a banded matrix, and then reducing the banded matrix to a tridiagonal matrix. Prior to version 2023.11, all MPI communications were done via the host. This reduced overall performance data as communication required transferring data from GPU to the host, sending via MPI, and transferring back to the GPU. ELPA version 2023.11 can now perform some – but not all – communications via NCCL. This eliminates many host-device transfers and has led to significant performance improvements in ELPA (see Section VI-C).

Recent results [23] in Fermionic Quantum Turbulence utilized ELPA's one-stage solver for dense, complex eigenproblems of similar sizes as shown in this work on up to 4096 GPUs of the LUMI supercomputer.

SLATE serves as the successor to the ScaLAPACK library [24]. It aims to improve performance over ScaLAPACK by introducing asynchrony and GPU support. Like ELPA, it also aims to implement both the one-stage and two-stage algorithms for solving eigenproblems. The addition of eigensolver routines is relatively recent [25] and results have not yet been published for any large-scale eigenproblems running on distributed GPU systems. SLATE could be a target for future studies.

A prior record eigenproblem with a dimension of one million was computed on 82,944 nodes of the K-computer [26]. This experiment was repeated on 4096 nodes of the Fukagu

system [27], achieving similar runtimes but half the efficiency due to a decreased ratio of network to node performance. The reported eigenproblem is a synthetic benchmark of a dense, real input matrix, while QMBT computations typically require solving dense, complex eigenproblems. It was solved using the one-stage tridiagonal solver of the EigenExa software, which does not support GPUs.

III. Implementation

In order to successfully tackle the challenges described in Section I at scale, both, VASP and cuSOLVERMp, require algorithmic innovations, which are described in this section.

A. Construction of the Two-particle Hamiltonian

The presented implementation of the BSE achieves the best performance by exploiting the parallelism over k-points, which reduces the necessary communication and minimizes redundant calculations. As the BSE Hamiltonian is a Hermitian matrix, only the upper triangle of the matrix is calculated explicitly. The computational load is efficiently distributed over both k-point indices, i.e., k and k'. This way, the matrix elements $M_{\mathbf{k},\mathbf{k}'} = V_{vc\mathbf{k}}^{v'c'\mathbf{k}'} - W_{vc\mathbf{k}}^{v'c'\mathbf{k}'}$ for each pair of $\mathbf{k}$ and $\mathbf{k}'$ are computed independently.

All significant mathematical operations on the GPU are performed by using cuBLAS for linear algebra and cuFFT for Fourier transforms. To avoid the data movement between CPU and GPU, all the data, such as single-particle wavefunctions, grids, and matrix elements of the BSE matrix, are copied to the device as early as possible. To minimize the memory footprint, we only store the wavefunctions in reciprocal space for locally treated k-points and clear them from the GPU before calling the distributed eigensolver.

1) Communication and math libraries: The communication between the ranks is required in three parts of the code: a) when relevant wavefunctions are gathered from the round-robin distribution over wavefunctions; b) in `PZTRMR2D`, and c) in cuSOLVERMp. For (a) we employ grouped `ncclSend/Recv` calls for the relevant wavefunctions which allows NCCL to optimize communication paths and hence hide the latency of the plethora of individual messages.

The major part of the matrix element computation consists of FFTs. The FFTs of the wavefunctions and charge densities involve rather small grids, that do not saturate a modern GPU. Thus, FFT batching is used throughout the computation to increase throughput on each single GPU. Here, the FFTs from reciprocal to real space are done in one batch for all N_c and N_v wavefunctions, maximizing FFT efficiency.

The asynchronous execution of the FFT kernels allows us to efficiently overlap them with the aforementioned communications and to further optimize the Fourier transform part in the calculation of the integral as in Eq. 3.

2) Data Distribution: Most distributed eigensolvers, including cuSOLVERMp, use a 2D block-cyclic distribution of the matrix and the elements of the locally computed matrix must be copied into the distributed matrix. This poses the main challenge for setting up the BSE Hamiltonian on the GPUs efficiently. In the original VASP CPU-implementation the communication in the data distribution scheme was based on broadcasting, which became a bottleneck when the calculation of the local matrices was ported to GPUs.

In order to distribute the matrix computed locally into the block-cyclic matrix, we created a GPU implementation of the ScaLAPACK `PZTRMR2D` routine. The GPU version of `PZTRMR2D` performs the packing and unpacking of the matrix data to and from the communication buffers in parallel on the GPU and the data is communicated using CUDA-aware MPI. The efficiency of such a scheme increases with the size of the matrix, which means that the distribution of the matrix by parts must be avoided. This imposes the requirement that a large submatrix is computed before it can be distributed. To be able to use such a scheme for distributing a triangular matrix and avoid load imbalance, we have to iterate the k-points in a specific order that allows to copy large matrices in as few `PZTRMR2D` calls as possible.

In Fig. 2, we illustrate the distribution scheme of the BSE matrix with 8 k-points computed on a grid of 20 ranks in an 8×5 grid. The colored numbers show the k- and k'-point indices and the superscript is the process number in the grid. A 256×256 matrix $M_{\mathbf{k},\mathbf{k}'}$ corresponds to a pair of k-points and is represented by a single cell in the table. In this scheme, the upper half of the triangular matrix is computed in the natural order and copied in a single `PZTRMR2D` call. The k indices for the bottom half of the triangular matrix are calculated in the reverse order. As shown in Fig. 2, by reversing the order of processes in the grid (Fig. 2.b) and swapping the local matrices (Fig. 2.c), the bottom half of the matrix is transformed into the upper triangular matrix which can be copied in a single `PZTRMR2D` call. Thus, the full matrix can be copied in only two `PZTRMR2D` calls. Minimizing the number of `PZTRMR2D` calls increases the end-to-end bandwidth by avoiding overheads from posting individual communication operations to the CPU.

B. Distributed GPU-only Eigensolver

We developed a GPU-only distributed parallel eigensolver in cuSOLVERMp. A conventional approach in a parallel eigensolver on GPUs is to split the computation and use both CPU and GPU wherever it is appropriate. For instance, the GPU can be used effectively for the reduction to a tridiagonal form and backtransformation phases while the CPU can be used for the eigensolutions of the reduced tridiagonal system. The hybrid approach can gain certain performance benefits [28]; however, it also includes inherent inefficiency and complexity compared to the developed GPU-only implementation. First, the hybrid implementation essentially requires data transfers between CPU and GPU, and each such transfer could become a synchronization point. When the transfer is not carefully managed, it can also waste significant time on both CPU and GPU waiting for the other. Even if the overhead can be mitigated by using efficient pipelining that can overlap computations such as double buffering, this requires additional tuning efforts for various heterogeneous computing platforms.

a)

$1,1^{1}$	$1,2^{1}$	$1,3^{2}$	$1,4^{2}$	$1,5^{3}$	$1,6^{3}$	$1,7^{4}$	$1,8^{4}$
$8,8^{5}$	$2,2^{5}$	$2,3^{6}$	$2,4^{6}$	$2,5^{7}$	$2,6^{7}$	$2,7^{8}$	$2,8^{8}$
$7,8^{9}$	$7,7^{9}$	$3,3^{10}$	$3,4^{10}$	$3,5^{11}$	$3,6^{11}$	$3,7^{12}$	$3,8^{12}$
$6,8^{13}$	$6,7^{13}$	$6,6^{14}$	$4,4^{14}$	$4,5^{15}$	$4,6^{15}$	$4,7^{16}$	$4,8^{16}$
$5,8^{17}$	$5,7^{17}$	$5,6^{18}$	$5,5^{18}$	$\_^{19}$	$\_^{19}$	$\_^{20}$	$\_^{20}$

b)

$\_^{20}$	$\_^{20}$	$\_^{19}$	$\_^{19}$	$5,6^{18}$	$5,5^{18}$	$5,8^{17}$	$5,7^{17}$
$4,7^{16}$	$4,8^{16}$	$4,5^{15}$	$4,6^{15}$	$6,6^{14}$	$4,4^{14}$	$6,8^{13}$	$6,7^{13}$
$3,7^{12}$	$3,8^{12}$	$3,5^{11}$	$3,6^{11}$	$3,3^{10}$	$3,4^{10}$	$7,8^{9}$	$7,7^{9}$
$2,7^{8}$	$2,8^{8}$	$2,5^{7}$	$2,6^{7}$	$2,3^{6}$	$2,4^{6}$	$8,8^{5}$	$2,2^{5}$
$1,7^{4}$	$1,8^{4}$	$1,5^{3}$	$1,6^{3}$	$1,3^{2}$	$1,4^{2}$	$1,1^{1}$	$1,2^{1}$

c)

$\_^{20}$	$\_^{20}$	$\_^{19}$	$\_^{19}$	$5,5^{18}$	$5,6^{18}$	$5,7^{17}$	$5,8^{17}$
$4,8^{16}$	$4,7^{16}$	$4,6^{15}$	$4,5^{15}$	$4,4^{14}$	$6,6^{14}$	$6,7^{13}$	$6,8^{13}$
$3,8^{12}$	$3,7^{12}$	$3,6^{11}$	$3,5^{11}$	$3,4^{10}$	$3,3^{10}$	$7,7^{9}$	$7,8^{9}$
$2,8^{8}$	$2,7^{8}$	$2,6^{7}$	$2,5^{7}$	$2,4^{6}$	$2,3^{6}$	$2,2^{5}$	$8,8^{5}$
$1,8^{4}$	$1,7^{4}$	$1,6^{3}$	$1,5^{3}$	$1,4^{2}$	$1,3^{2}$	$1,2^{1}$	$1,1^{1}$

Fig. 2. a) The table showing the order of k-points calculated on a rectangular grid of processes 8×5. The processor numbers are given in superscript and the k-point indices are given in blue and green fonts for upper and lower part of the matrix, respectively. Each cell of this table denotes a matrix $M_{\mathbf{k},\mathbf{k}'}$ with the corresponding k-point indices. The bold lines separate data corresponding to different MPI ranks. b) Reverse the processors grid. c) Reverse the order of local matrices.

Second, our GPU-only implementation can effectively leverage the fast interconnect technologies and advanced software infrastructure that are dedicated for GPUs. CUDA-aware MPI can transfer data directly among GPUs and enable fast intra-node communication via NVLink. We also adopted CAL to support multiple transport layers such as UCC and NCCL. The communication library exposes stream semantics allowing asynchronous communication and computation.

On the other hand, the main challenge of the GPU-only parallel eigensolver is to implement all computational kernels to run well on GPUs. In particular, we implemented an efficient GPU-only parallel divide-and-conquer algorithm to compute eigensolutions of a symmetric tridiagonal system, `STEDC` in LAPACK notation, which is usually computed on the CPU in the conventional hybrid approach. A general procedure of the tridiagonal eigensolver can be briefly described as follows: A tridiagonal system is recursively partitioned into two subproblems forming a binary tree until it reaches the minimum problem size. The independent subproblems at the leaf-level of the tree can be solved using the QR algorithm. Combining the two eigensolutions along the tree, a next level subproblem is formed and eigenvalues are computed by solving the secular equations. Then, the eigenvectors are updated via matrix-matrix multiplication. This process is recursively performed until it reaches the root. The algorithm first requires solving many small subproblems at the leaf-level and the problem sizes become bigger with traversing the tree. Thus, we need all the building blocks of the algorithm considering a wide range of problem sizes and different parallel implementations are necessary in all levels, i.e., thread-block, single GPU, and multi-GPU level. To solve the many small subproblems at leaf-level, we developed a batched version of the QR algorithm (`STEQR`). The parallel implementation of the merging step (`LAED2,4`) including the deflation and the solutions of the secular equations should correspond to increasing problem sizes as traversing the binary tree. When subproblems are scoped within a single GPU, we use multiple instances of merging kernels with different thread-block sizes such that a bigger thread-block is used to solve bigger problems. For the case that the subproblem dimension spans over multiple GPUs, we use another variant of the implementation that handles communication. The same goes for the updates of the eigenvectors. For small problems, we use a batched version and multiple subproblems are updated simultaneously. For bigger problems, eigenvectors are updated using the highly efficient cuBLAS `GEMM`.

Putting all building blocks together, we demonstrate a scalable and efficient GPU-only distributed eigensolver in this paper.

TABLE I
REQUIREMENTS FOR ACCURATE BSE CALCULATION OF FASNI

Variable	Description	Value
N_v	Number of valence states	16
N_c	Number of conduction states	16
N_k	Number of k-points	16^3
N_G	Number of G-vectors	5044

IV. Application to Photovoltaics

In this section, we apply the adapted VASP BSE implementation using cuSOLVERMp to study the properties of a perovskite material relevant for photovoltaics.

A. Material under Study

Formamidinium tin iodide ($FASnI_3$) is a very important example of a system whose excitonic properties are very difficult to describe without making crude approximations. The crystal structure of $FASnI_3$ is shown in Fig. 1. This material has a strong dependency of the single-particle energies on the momentum in reciprocal space, which means that the BSE calculation has to be performed for a dense k-point mesh [29]. In the previous attempt to study the excitonic properties of this material, the authors could only account for 2 valence and 2 conduction wavefunctions, i.e., covering only 1/10 of the valence band, when computing the exciton binding energy for a $20 \times 20 \times 20$ k-points grid. For that reason, and to the best of our knowledge, the converged optical absorption spectrum of this material has never been demonstrated.

By using cuSOLVERMp and the GPU implementation of the BSE algorithm, we are able to perform the BSE calculation of $FASnI_3$ with the explicit grid of $16 \times 16 \times 16$ k-points, 16

valence, and 16 conduction wavefunctions. In the performed calculations, the spin-orbit coupling is explicitly treated. The screening of the Coulomb potential W is approximated by a model dielectric function described in [29]. The DFT electronic structure is corrected by the scissor operator to match the GW band gap. The parameters of the calculation are summarized in Tab. I.

B. Computational Results

We found an exciton binding energy of 51 meV. The good agreement with the results reported in Ref. [29] indicates that in $FASnI_3$ the exciton is strongly delocalized and is dominated by the two states at the band edges.

By treating explicitly the states beyond the band edges in the BSE, we yielded the optical absorption spectrum up to 6 eV with the convergence error of 20 meV, which shows an excellent agreement with the experiment in Fig. 3. Discrepancies at higher energies are attributed to the fact that the scissor operator was used to match the band edges of GW and is therefore less accurate for transitions at higher energies. In addition, we provide the so-called "fatband" plot in Fig. 4, which includes many states beyond the band edges. Fig. 4 shows that the two highest states of the valence band and the two lowest states of the conduction band minimum give the dominant contribution for the first optically active exciton, while the contribution of the other states is much smaller.

The accuracy of 20 meV is sufficient for a converged optical absorption spectrum. However, in order to find the exciton binding energy with a higher accuracy denser k-points grids should be used. For such calculations we consider the $18\times18\times18$ and $20\times20\times20$ k-points grids, for which we provide the performance analysis of the Eigensolver call in Section VI-A. The Hamiltonian for the largest calculation has the rank of 2,048,000.

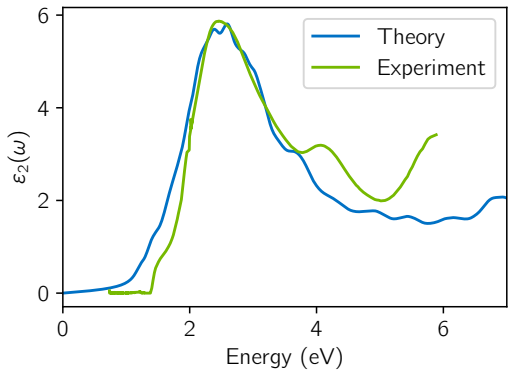

Fig. 3. The imaginary dielectric function of $FASnI_3$ calculated for $16\times16\times16$ k-points grid with 16 occupied and 16 unoccupied states compared to the experimental spectrum from [30]. A Lorentzian broadening of 0.1 eV and the scaling of 1.4 is applied to the calculated spectrum to facilitate the comparison with experiment.

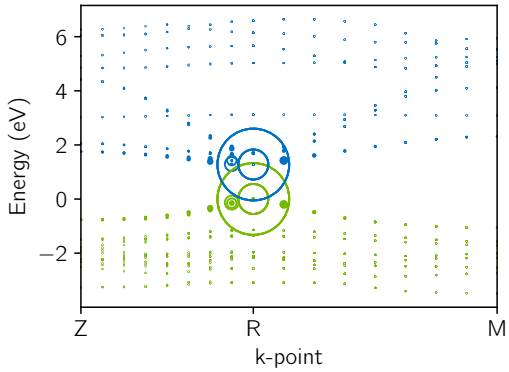

Fig. 4. The fatband plot for the lowest exciton in $FASnI_3$ calculated for $16\times16\times16$ k-points grid with 16 occupied and 16 unoccupied states.

V. Performance Measurements

A. Environment and Application Profiling

Our runs were performed on NVIDIA's Selene supercomputer [31] which has the following configuration:

- up to 560x DGX A100 nodes
- 8x A100 SXM4 80GB GPUs (connected via NVSwitch) per node
- 2x AMD EPYC 7742 (64-core) CPUs per node (NPS4 mode)
- 8x HDR200 InfiniBand connections per node (1 per GPU)

Selene is currently ranked the thirteenth fastest supercomputer in the Top500 List [32]. The full system entered the list on rank five in November 2020.

We used the following software environment in a container using the overhead-free Enroot runtime:

- VASP (development branch based on VASP 6.3)
- NVIDIA HPC SDK 23.1
- HPCX-2.13
- cuSOLVERMp (development branch)

VASP provides internal timers to measure the elapsed runtime for different parts of the code. It can generate a detailed profile at the lowest possible overheads as the profiling markers are called only at appropriate places. Hence, they interfere as little as possible with the execution of the application, especially in contrast to attaching sampling-based profilers. This feature needs to be enabled in VASP at compile time. The `OUTCAR` file then contains the timing information we used. No additional software or tools are required to measure time-to-solution numbers.

The Selene system allows collecting various metrics which are aggregated using Grafana [33]. Its API allows querying power use over time for individual CPUs, GPUs, and PSUs, either for the whole job, or a given time frame. We use this data with timestamps from the VASP output to quantify the energy used for the eigensolver and complete run. The reported numbers for energy consumption do not take into account the network infrastructure, which is negligible for large jobs.

For performance measurements, we use the same $FASnI_3$ compound, but varied the k-point grid to control the dimension of the Hamiltonian and hence the problem size to solve. This allowed us to run similar jobs at different scales. For each input size, we selected a specific launch configuration for running on the Selene supercomputer. The optimal configuration depends on a number of constraints: Since VASP is optimized well for GPUs, a single rank per device is sufficient to best utilize them. Moreover, GPU oversubscription is not allowed by the NCCL library, which is used in VASP to pass data asynchronously between GPUs to overlap GPU computation and communication. Thus, each MPI rank is assigned to a single GPU. The NUMA and PCIe topology of the nodes requires some consideration for the ideal process placement. Given that there are 8 NUMA nodes, but only 4 of them have a direct physical link to a GPU and NIC, every second rank is placed on a NUMA node that has one extra, intra-socket hop to the ideal GPU and NIC affinity. Here, this is the best compromise between not sharing CPU resources and device affinities. The second limitation is the available GPU memory. To solve the eigenvalue problem, each GPU needs to store its part of the input matrix, the eigenvectors, the eigenvalues, and a workspace for intermediate data. Given the available memory per device, we deduce the minimal number of nodes needed to fit the problem into GPU memory and achieve a balanced domain decomposition. Tab. II and III summarize the problem sizes and the resulting node and GPU counts.

B. Matrix Diagonalization Efficiency

The diagonalization of the BSE matrix based on cuSOLVERMp's `SYEVD` implementation consists of three main subroutines: `SYTRD`, `STEDC`, and `ORMTR`. The peak limiting hardware bound for each routine differs, therefore we quantify their efficiencies separately. They are calculated as a percentage of either peak compute throughput or peak memory bandwidth.

The `STEDC` and `ORMTR` routines are both compute-bound, bottle-necked by the large `GEMM` operations they must perform. The FLOP counts of the complex valued `ORMTR` are taken from the LAPACK working note [34] and would be $8n^3$ ignoring lower order terms where n is the size of the matrix. The FLOP count of `STEDC` is $\frac{8}{3}n^3$, as given by Haidar et. al [35]. To calculate efficiency for `STEDC` and `ORMTR`, the FLOP rate is calculated from the FLOP count and the runtime, and then expressed as a percentage of the peak FLOPS of all GPUs. The peak FLOPS used is the FP64 tensor-core rate of the NVIDIA A100 GPU, which is 19.5 TFLOPS.

Unlike the `STEDC` and `ORMTR`, the `SYTRD` is memory-bandwidth bound. The algorithm implemented in `SYTRD` is denoted as two-sided factorization, where the orthogonal transformations are applied from both left and right sides. Although a blocked version of the algorithm [9], [36] is used for the reduction casting, the trailing updates with level-3 BLAS operations, each panel factorization still requires to perform level-2 BLAS operations, i.e., `SYMV` with the entire trailing matrix. This requires a total number of matrix elements read from memory of $\frac{n^3}{3}$ [37]. Using this equation, we can estimate the total amount of data transferred from main memory, and therefore calculate the efficiency as a percentage of the peak memory bandwidth. The peak memory bandwidth of a single A100 SXM4 80GB GPU is 2.039 TB/s.

The overall `SYEVD` efficiency is then calculated as the ratio of the theoretical minimum runtime achievable given the algorithmic workload and peak capabilities of the hardware, and the actually measured runtime.

VI. Benchmark Results

A. Timing, Energy, and Peak Performance

In this section, we present the performance numbers measured for VASP and cuSOLVERMp on the Selene supercomputer. For the problem at hand, we ran VASP with a BSE matrix rank of up to 1,048,576. We further benchmarked cuSOLVERMp's `SYEVD` in a standalone benchmark for matrices up to rank 2,048,000. With 512 nodes, the largest jobs used almost the entire Selene supercomputer and ran for several hours. Given that the algorithms used in `SYEVD` are non-iterative and the execution path is independent of the actual values in the matrix, this is equivalent in terms of the performance that would be observed in VASP's BSE matrix diagonalization step when computing inputs resulting in such a problem size.

TABLE II
VASP: Runtime and energy consumption

k-points	10^3	16^3
nodes	16	128
GPUs	128	1,024
matrix rank	256,000	1,048,576
elapsed time [sec]	1,129.1	8,739.6
- BSE setup [sec]	137.8	1,049.1
- BSE diag [sec]	906.0	7,258.0
total energy [MJ]	74.8	4,351.3
- GPU [MJ]	33.3	2,060.3
- CPU [MJ]	3.8	231.1

Tab. II and III list the measured times, energy consumption, and sustained PFLOPS. We show VASP's internal timer for the total time to solution (*Elapsed time*), the time for setting up the BSE matrix (*BSE setup*), and for running the `SYEVD` solver (*BSE diag*). The *Total energy* is based on the PSU power metric of the system and thus contains CPU and GPU power, as well as all other node components, such as NICs, DRAM, and local storage.

To our best knowledge, with a rank of 2,048,000, the solved dense, complex eigenproblem is on par with only one other recently reported result [23]. The solver took 17,727.8 seconds, i.e. under 5 hours, and achieved 7.8 PFLOPS. The energy efficiency for the eigensolver with 4.0 GFLOPS/W is roughly 17% of Selene's LINPACK efficiency reported in the Green500 List [38]. While the calculations with different matrix sizes and different node counts are not directly comparable, the sweet spot in terms of energy efficiency is the 128 node job solving a matrix with a rank of 1,048,576 with an achieved overall 4.9 GFLOPS/W.

TABLE III
CUSOLVERMP SYEVD: RUNTIME AND ENERGY CONSUMPTION

nodes	16	128	384	512
GPUs	128	1,024	3,072	4,096
matrix rank	256,000	1,048,576	1,492,992	2,048,000
runtime [sec]	906.0	7,258.0	9927.2	17,905.1
total energy [MJ]	60.1	3,776.4	14,015.3	34,532.9
- GPU [MJ]	28.4	1,891.7	6,224.4	15,890.0
- CPU [MJ]	2.9	187.7	764.4	1,819.0
perf [PFLOPS]	0.3	2.6	5.4	7.8
perf/GPU [TFLOPS]	2.3	2.5	1.8	1.9
eff. [GFLOPS/W]	4.5	4.9	3.8	4.0

While the methods and algorithms outlined in this paper are by no means limited to the NVIDIA platform, any system equipped with NVIDIA GPUs of an appropriate size can be readily used with our implementation.

B. Efficiency and Scalability

The 'perf/GPU' row of Table III shows that a low percentage of peak performance is achieved per GPU—the peak is 19.5 TFLOPS whilst SYEVD only achieves between 1.8 and 2.5 TFLOPS per GPU. This result is explained by examining Table IV, which shows the runtime of SYEVD broken down by each of its subroutines. The forward transformation (SYTRD) constitutes by far the largest portion of the runtime. Combined with the fact the routine is memory-bandwidth bound, this explains why SYEVD overall achieves a low percentage of peak FLOPS.

TABLE IV
PERCENTAGE OF RUNTIME SPENT IN EACH OF OF CUSOLVERMP'S SYEVD SUBROUTINES.

N	SYTRD	STEDC	ORMTR
256000	84.41%	3.48%	12.11%
1048576	85.70%	3.47%	10.80%
1492992	83.13%	7.00%	9.87%
2048000	83.04%	7.09%	9.87%

To better assess how close the eigensolver implementation is to the maximally achievable performance, we show the efficiencies of the three major eigensolver subroutines across the different problem sizes, and aggregate them into an overall SYEVD efficiency in Tab. V. Hence, we employ the term efficiency as the achieved percentage of the theoretical optimum on the given compute hardware. As only large problem instances are of scientific interest here, we did not focus on dedicated scalability experiments. However, we performed a small 16-node run of a system with 10^3 k-points to compare efficiency with. The resulting efficiencies are with the exception of STEDC very similar across all node counts and the individual parts of the solver range between 22% and 61% of the respective peak hardware capability.

The efficiency for the memory-bandwidth bound SYTRD varies between 35% and 47%. During the routine, the updates to the local matrices gradually decrease in size such that the communication overhead is more easily exposed. This effect becomes stronger with fewer local matrix blocks on each GPU.

TABLE V
ACHIEVED EFFICIENCIES OF CUSOLVERMP'S SYEVD SUBROUTINES, AS A PERCENTAGE OF THE PEAK OF THEIR LIMITING HARDWARE BOUND.

subroutine bound by	SYTRD memory	STEDC compute	ORMTR compute	SYEVD combined
16 nodes (256k)	45%	57%	49%	46%
128 nodes (1M)	47%	61%	59%	59%
384 nodes (1.5M)	35%	22%	46%	35%
512 nodes (2.0M)	37%	23%	49%	37%

We see a very similar effect in ORMTR as well where efficiencies range from 46% to even 59% on 128 nodes. However, relative to its surrounding routines STEDC is much more exposed to such influences: From a satisfying 61% of the available compute performance for the 128 node problem, its efficiency drops by almost a factor of 3 down to 22% on 384 nodes with a matrix rank of 1,492,992. This is due to the creation of subcommunicators on every step of the divide and conquer recursion. The underlying expensive splitting operation in NCCL takes up a significant portion of the STEDC runtime. This is expected to be solved in future release of NCCL. While the eigensolver presented in this work shows impressive performance and enables computations of groundbreaking problem sizes with good efficiency, there is still room for improvement. Optimizing both, VASP and cuSOLVERMp, is an ongoing process.

C. Comparisons to Other Dense Linear Algebra Libraries

We compared the performance of cuSOLVERMp to ELPA (2023.11 release) for other large symmetric double-complex eigenproblems. Fig. 5 shows the performance of ELPA1 with and without NCCL (NVIDIA Collective Communications Library), and cuSOLVERMp for a 294k $\times$ 294k matrix on 128, 256, and 512 GPUs, and a 576k $\times$ 576k matrix running on 1000 GPUs. The GPU-only approach of cuSOLVERMp delivers significant speed-ups over the fastest ELPA1 reference: 1.12x, 1.25x, and 1.11x for 128, 256, and 512 GPUs, respectively, on the 294k matrix, and 1.33x for the 576k matrix on 1000 GPUs. We used a single process per GPU, with a matrix block size of 256—the same configuration as used for the VASP BSE results. For this configuration, ELPA2 lags significantly behind ELPA1 and cuSOLVERMp as can be seen in Fig. 6.

ELPA2 requires oversubscribing MPI ranks to GPUs to perform well for two reasons. Firstly, the banded-to-tridiagonal step of the method is computed entirely on the CPU, hence one needs to make use of the available CPU cores. Secondly, oversubscription is required to improve throughput of the GPU operations that are part of the tridiagonal-to-banded step [22]. When oversubscription is necessary, NVIDIA Multi-Process Service (MPS) is also required for good performance. As discussed in the Performance Measurements section, oversubscribing ranks to GPUs is in conflict with the optimal VASP configuration which uses only one rank per GPU.

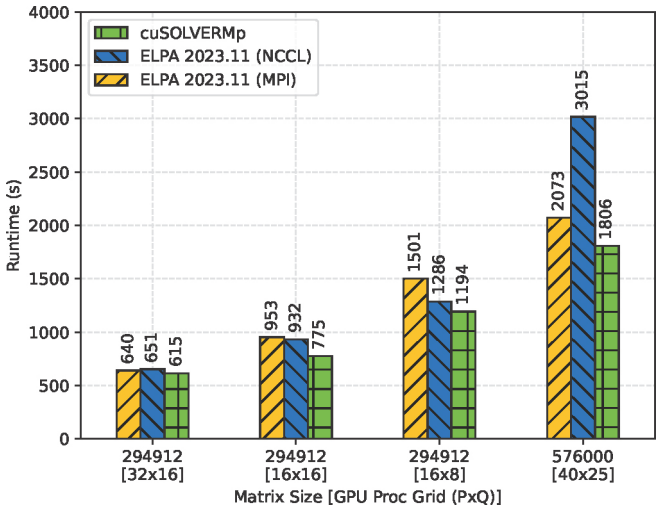

Fig. 5. Runtime comparison of the matrix diagonalization step in cuSOLVERMp and ELPA (one-stage) as used in VASP, i.e. with one process per GPU.

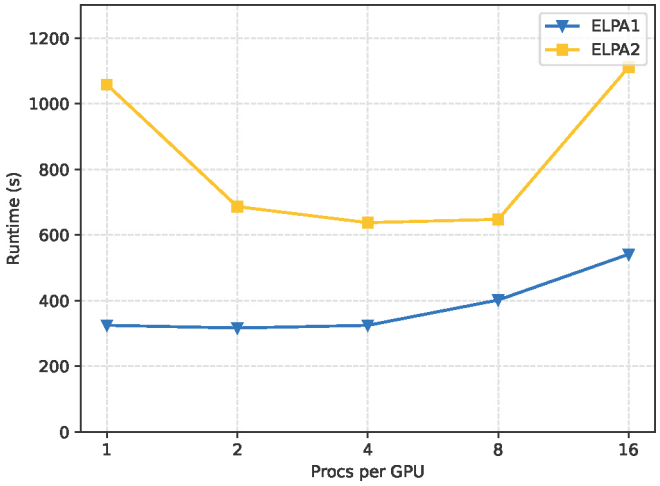

Fig. 6. Runtime impact of oversubscribing MPI ranks (processes) to GPUs for a 128K double complex eigenproblem running on 64 GPUs for both ELPA1 and ELPA2.

In comparison to the two-stage solver as shown in Fig. 6, the one-stage solver in ELPA performs well when using a single rank per GPU. Another difference is that the performance of the two-stage solver can be affected by the matrix block size. Fig. 7 shows that performance degrades for ELPA2 with increasing matrix block size. In comparison, cuSOLVERMp performs well with block sizes over 64, and ELPA1 is mostly unaffected by any choice of block size.

Currently, cuSOLVERMp is optimized for problems of the largest scales where, e.g., the combined memory of a large amount of GPUs is required to fit the problem. For small matrices, using ELPA or even a non-distributed eigensolver (e.g. cuSolver) can be preferable.

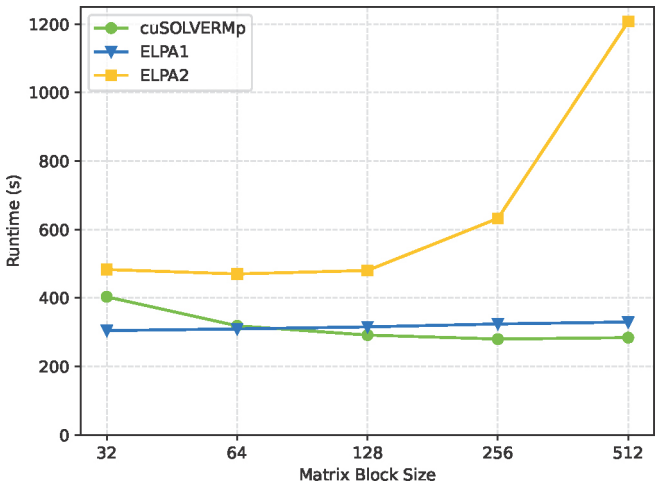

Fig. 7. Runtime of solving a 128K double complex eigenproblem on 64 GPUs using cuSOLVERMp, ELPA1, ELPA2 with varying block sizes.

VII. Summary

We have demonstrated the results of an efficient implementation of an eigensolver and the BSE algorithm for the GPU architecture. In particular, we have shown that by performing the BSE calculations on GPUs, the feasible problem size for the computationally demanding BSE method increases by 1,000x. To achieve this, we solved one of the largest reported dense, complex eigenproblems.

By solving the BSE for $FASnI_3$ with the $16 \times 16 \times 16$ k-points grid and including 32 single-particle states explicitly, we have obtained the converged optical absorption spectrum, which can be directly compared to the experiment and shows an excellent agreement, which was inaccessible before. The analysis of the states beyond the band edges based on the fatband plot suggests that the contribution of the strongly delocalized wavefunctions of the bands maxima gives the dominant contribution for the exciton binding energies.

The results of this work have implications for material science in general, and photovoltaics research in particular. Our method lays the foundation for high-throughput search of complex materials for solar cells with efficiency as high as 45% [39]. The achieved performance bridges the BSE method to complex systems, such as double-perovskites [40], two-dimensional heterostructures [41], metal-organic frameworks [42], and other systems which can require tens or hundreds of atoms to be treated explicitly for accurate modeling. Having shown that such accurate calculations are possible with modern supercomputers, we believe that this level of accuracy will find its way into common practice amongst material researchers.

While we have applied the accelerated BSE calculations to a material to help with sustainable energies, the applied methodology also facilitates predictions of rather mundane properties like the color of a material, important for dyes. Beyond that, it can extend into high-impact research areas, such as quantum computing and superconductivity, and also any other area, where excitonic effects play a major role. More

generally, higher-order diagrams in the many-body perturbation theory which require solving large eigenvalue problems become accessible with the help of cuSOLVERMp, improving the accuracy of currently used approximations.

The scientific domain of quantum-chemistry is surely one of the easily relatable heavy users for eigensolvers, given that it is fundamentally based on the Schrödinger equation, which implicitly connects it to this mathematical discipline. However, there are other domains outside our own area of expertise that also employ eigensolvers for their computational predictions like geosciences, mechanical, and civil engineering. Given the fast increases in compute capabilities, what is groundbreaking supercomputing today will likely become mainstream methods in just a few years—with an unpredictable impact on material science and other domains.

ACKNOWLEDGMENTS

We would like to thank the ELPA authors, especially Andreas Marek and Peter Karpov, for their quick support with benchmarking.

REFERENCES

[1] G. Kresse and J. Hafner, "Ab initio molecular dynamics for liquid metals," *Phys. Rev. B*, vol. 47, pp. 558–561, Jan 1993. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.47.558

[2] J. Jeong, M. Kim, J. Seo, H. Lu, P. Ahlawat, A. Mishra, Y. Yang, M. A. Hope, F. T. Eickemeyer, M. Kim, Y. J. Yoon, I. W. Choi, B. P. Darwich, S. J. Choi, Y. Jo, J. H. Lee, B. Walker, S. M. Zakeeruddin, L. Emsley, U. Rothlisberger, A. Hagfeldt, D. S. Kim, M. Grätzel, and J. Y. Kim, "Pseudo-halide anion engineering for α-FAPbI3 perovskite solar cells," *Nature*, vol. 592, no. 7854, pp. 381–385, 2021. [Online]. Available: https://doi.org/10.1038/s41586-021-03406-5

[3] J. Y. Kim, J.-W. Lee, H. S. Jung, H. Shin, and N.-G. Park, "High-Efficiency Perovskite Solar Cells," *Chemical Reviews*, vol. 120, no. 15, pp. 7867–7918, Jul. 2020. [Online]. Available: https://doi.org/10.1021/acs.chemrev.0c00107

[4] W. Kohn, "Nobel Lecture: Electronic structure of matter—wave functions and density functionals," *Rev. Mod. Phys.*, vol. 71, pp. 1253–1266, Oct 1999. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.71.1253

[5] W. Kohn and J. M. Luttinger, "Ground-State Energy of a Many-Fermion System," *Phys. Rev.*, vol. 118, pp. 41–45, 1960. [Online]. Available: http://dx.doi.org/10.1103/PhysRev.118.41

[6] M. D. Ben, C. Yang, Z. Li, F. H. d. Jornada, S. G. Louie, and J. Deslippe, "Accelerating Large-Scale Excited-State GW Calculations on Leadership HPC Systems," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020, pp. 1–11.

[7] E. E. Salpeter and H. A. Bethe, "A Relativistic Equation for Bound-State Problems," *Phys. Rev.*, vol. 84, pp. 1232–1242, 1951. [Online]. Available: http://dx.doi.org/10.1103/PhysRev.84.1232

[8] S. M. Dancoff, "Non-Adiabatic Meson Theory of Nuclear Forces," *Phys. Rev.*, vol. 78, pp. 382–385, 1950. [Online]. Available: http://dx.doi.org/10.1103/PhysRev.78.382

[9] C. Bischof and C. Van Loan, "The WY Representation for Products of Householder Matrices," *SIAM Journal on Scientific and Statistical Computing*, vol. 8, no. 1, pp. s2–s13, 1987.

[10] J. J. Cuppen, "A divide and conquer method for the symmetric tridiagonal eigenproblem," *Numerische Mathematik*, vol. 36, pp. 177–195, 1980.

[11] G. Kresse and J. Furthmüller, "Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set," *Phys. Rev. B*, vol. 54, pp. 11 169–11 186, Oct 1996. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.54.11169

[12] G. Kresse and J. Furthmüller, "Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set," *Computational Materials Science*, vol. 6, no. 1, pp. 15–50, 1996. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0927025696000080

[13] J. Deslippe, G. Samsonidze, D. A. Strubbe, M. Jain, M. L. Cohen, and S. G. Louie, "BerkeleyGW: A massively parallel computer package for the calculation of the quasiparticle and optical properties of materials and nanostructures," *Computer Physics Communications*, vol. 183, no. 6, pp. 1269–1289, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465511003912

[14] M. Shao, F. H. da Jornada, C. Yang, J. Deslippe, and S. G. Louie, "Structure preserving parallel algorithms for solving the Bethe–Salpeter eigenvalue problem," *Linear Algebra and its Applications*, vol. 488, pp. 148–167, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0024379515005637

[15] A. Marek, V. Blum, R. Johanni, V. Havu, B. Lang, T. Auckenthaler, A. Heinecke, H.-J. Bungartz, and H. Lederer, "The ELPA library: scalable parallel eigenvalue solutions for electronic structure theory and computational science," *Journal of Physics: Condensed Matter*, vol. 26, no. 21, p. 213201, may 2014. [Online]. Available: https://dx.doi.org/10.1088/0953-8984/26/21/213201

[16] C. Penke, A. Marek, C. Vorwerk, C. Draxl, and P. Benner, "High performance solution of skew-symmetric eigenvalue problems with applications in solving the Bethe-Salpeter eigenvalue problem," *Parallel Computing*, vol. 96, p. 102639, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167819120300326

[17] J. Choi, J. J. Dongarra, R. Pozo, and D. W. Walker, "ScaLAPACK: a scalable linear algebra library for distributed memory concurrent computers," in *[Proceedings 1992] The Fourth Symposium on the Frontiers of Massively Parallel Computation*. IEEE Computer Society, 1992, pp. 120–127. [Online]. Available: https://www.computer.org/csdl/proceedings-article/fmpc/1992/00234898/12OmNCctfdR

[18] L. S. 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. C. Whaley, *ScaLAPACK Users' Guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1997.

[19] L. S. Blackford, A. Petitet, R. Pozo, K. Remington, R. C. Whaley, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry *et al.*, "An updated set of basic linear algebra subprograms (BLAS)," *ACM Transactions on Mathematical Software*, vol. 28, no. 2, pp. 135–151, 2002.

[20] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney *et al.*, *LAPACK users' guide*. SIAM, 1999.

[21] F. Tisseur and J. Dongarra, "A Parallel Divide and Conquer Algorithm for the Symmetric Eigenvalue Problem on Distributed Memory Architectures," *SIAM Journal on Scientific Computing*, vol. 20, no. 6, pp. 2223–2236, 1999.

[22] V. W. zhe Yu, J. Moussa, P. Kůs, A. Marek, P. Messmer, M. Yoon, H. Lederer, and V. Blum, "GPU-acceleration of the ELPA2 distributed eigensolver for dense symmetric and hermitian eigenproblems," *Computer Physics Communications*, vol. 262, p. 107808, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465520304021

[23] G. Wlazlowski, M. M. Forbes, S. R. Sarkar, A. Marek, and M. Szpindler, "Characterizing the Cascade of Energy in Fermionic Quantum Turbulence: Pushing the Limits of High-Performance Computing," 2023.

[24] M. Gates, J. Kurzak, A. Charara, A. YarKhan, and J. Dongarra, "SLATE: design of a modern distributed and accelerated linear algebra library," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2019, pp. 1–18. [Online]. Available: https://dl.acm.org/doi/10.1145/3295500.3356223

[25] K. Akbudak, P. Bagwell, S. Cayrols, M. Gates, D. Sukkari, A. YarKhan, and J. Dongarra, "SLATE performance improvements: QR and eigenvalues, SWAN no. 17," Innovative Computing Laboratory, University of Tennessee, Tech. Rep. ICL-UT-XX-XX, 4 2021, revision 04-2021. [Online]. Available: https://www.icl.utk.edu/publications/swan-017

[26] H. Imachi and T. Hoshi, "Hybrid Numerical Solvers for Massively Parallel Eigenvalue Computation and Their Benchmark with Electronic Structure Calculations," *Journal of Information Processing*, vol. 24, no. 1, pp. 164–172, 2016, arXiv:1504.06443 [cond-mat, physics:physics]. [Online]. Available: http://arxiv.org/abs/1504.06443

[27] T. Imamura, T. Terao, T. Ina, K. Ozaki, and Y. Uchino, "Performance benchmark of the latest EigenExa on Fugaku," Jan. 2022. [Online]. Available: https://sighpc.ipsj.or.jp/HPCAsia2022/poster/108_poster.pdf

[28] A. Haidar, R. Solcà, M. Gates, S. Tomov, T. C. Schulthess, and J. J. Dongarra, "Leading Edge Hybrid Multi-GPU Algorithms for General-

ized Eigenproblems in Electronic Structure Calculations," in *Information Security Conference*, 2013.

[29] M. Bokdam, T. Sander, A. Stroppa, S. Picozzi, D. D. Sarma, C. Franchini, and G. Kresse, "Role of Polar Phonons in the Photo Excited State of Metal Halide Perovskites," *Sci Rep*, vol. 6, p. 28618, 2016. [Online]. Available: http://dx.doi.org/10.1038/srep28618

[30] K. Ghimire, D. Zhao, Y. Yan, and N. J. Podraza, "Optical response of mixed methylammonium lead iodide and formamidinium tin iodide perovskite thin films," *AIP Advances*, vol. 7, no. 7, p. 075108, Jul. 2017. [Online]. Available: https://doi.org/10.1063/1.4994211

[31] "NVIDIA Selene: Leadership-Class Supercomputing Infrastructure," https://www.nvidia.com/en-us/on-demand/session/supercomputing2020-sc2019/, accessed: 2023-12-15.

[32] "Top500 List - November 2023," https://www.top500.org/lists/top500/list/2023/11/, accessed: 2023-12-15.

[33] "Grafana: The open observability platform," https://grafana.com/, accessed: 2023-12-15.

[34] S. Blackford and J. Dongarra, *LAPACK Working Note 41 Installation Guide for LAPACK*, 1999. [Online]. Available: https://netlib.org/lapack/lawnspdf/lawn41.pdf

[35] A. Haidar, H. Ltaief, and J. Dongarra, "Toward a High Performance Tile Divide and Conquer Algorithm for the Dense Symmetric Eigenvalue Problem," *SIAM Journal on Scientific Computing*, vol. 34, no. 6, pp. C249–C274, 2012. [Online]. Available: http://epubs.siam.org/doi/10.1137/110823699

[36] J. J. Dongarra, D. C. Sorensen, and S. J. Hammarling, "Block reduction of matrices to condensed forms for eigenvalue computations," in *Parallel Algorithms for Numerical Linear Algebra*, ser. Advances in Parallel Computing, H. A. van der Vorst and P. van Dooren, Eds. North-Holland, 1990, vol. 1, pp. 215–227. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444886217500153

[37] J. H. Wilkinson and J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, ser. Numerical Mathematics and Scientific Computation. Oxford University Press, 1988.

[38] "Green500 List - November 2022," https://www.top500.org/lists/green500/list/2023/11/, accessed: 2023-12-15.

[39] F. Khan, B. D. Rezgui, M. T. Khan, and F. Al-Sulaiman, "Perovskite-based tandem solar cells: Device architecture, stability, and economic perspectives," *Renewable and Sustainable Energy Reviews*, vol. 165, p. 112553, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S136403212200452X

[40] F. Ji, G. Boschloo, F. Wang, and F. Gao, "Challenges and Progress in Lead-Free Halide Double Perovskite Solar Cells," *Solar RRL*, vol. 7, no. 6, p. 2201112, 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/solr.202201112

[41] S. Joseph, J. Mohan, S. Lakshmy, S. Thomas, B. Chakraborty, S. Thomas, and N. Kalarikkal, "A review of the synthesis, properties, and applications of 2D transition metal dichalcogenides and their heterostructures," *Materials Chemistry and Physics*, vol. 297, p. 127332, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0254058423000408

[42] V. F. Yusuf, N. I. Malek, and S. K. Kailasa, "Review on Metal–Organic Framework Classification, Synthetic Approaches, and Influencing Factors: Applications in Energy, Drug Delivery, and Wastewater Treatment," *ACS Omega*, vol. 7, no. 49, pp. 44 507–44 531, 2022.