Efficient GPU Parallelization of Electronic Transport and Nonequilibrium Dynamics from Electron-Phonon Interactions in the PERTURBO Code

Shiyu Peng,[1,*] Donnie Pinkston,[2,*] Jia Yao,[1] Sergei Kliavinek,[1] Ivan Maliyov,[1] and Marco Bernardi[1,†]

[1] *Department of Applied Physics and Materials Science, and Department of Physics, California Institute of Technology, Pasadena, California 91125, USA*

[2] *Schmidt Academy for Software Engineering, California Institute of Technology, Pasadena, California 91125, USA*

The Boltzmann transport equation (BTE) with electron-phonon (e-ph) interactions computed from first principles is widely used to study electronic transport and nonequilibrium dynamics in materials. Calculating the e-ph collision integral is the most important step in the BTE, but it remains computationally costly, even with current MPI+OpenMP parallelization. This challenge makes it difficult to study materials with large unit cells and to achieve high resolution in momentum space. Here, we show acceleration of BTE calculations of electronic transport and ultrafast dynamics using graphical processing units (GPUs). We implement a novel data structure and algorithm, optimized for GPU hardware and developed using OpenACC, to process scattering channels and efficiently compute the collision integral. This approach significantly reduces the overhead for data referencing, movement, and synchronization. Relative to the efficient CPU implementation in the open-source package PERTURBO (v2.2.0), used as a baseline, this approach achieves a speed-up of 40 times for both transport and nonequilibrium dynamics on GPU hardware, and achieves nearly linear scaling up to 100 GPUs. The novel data structure can be generalized to other electron interactions and scattering processes. We released this GPU implementation in the latest public version (v3.0.0) of PERTURBO. The new MPI+OpenMP+GPU parallelization enables sweeping studies of e-ph physics and electron dynamics in conventional and quantum materials, and prepares PERTURBO for exascale supercomputing platforms.

* These authors contributed equally to this work.

† bmarco@caltech.edu

I. INTRODUCTION

Electron-phonon (e-ph) interactions govern a wide range of properties in materials, including electronic transport, nonequilibrium dynamics of excited electrons, superconductivity, optical behavior, and polarons[1,2]. First-principles calculations based on density functional theory (DFT) and related techniques can calculate e-ph interactions with quantitative accuracy[3,4]. In recent years, the combination of semiclassical Boltzmann transport equation (BTE) with first-principles e-ph interactions has enabled accurate predictions of electronic transport in metals[5,6], inorganic and organic semiconductors[7–16], complex oxides[17–19], and quantum materials[20–23].

For studies of ultrafast nonequilibrium dynamics, solving the BTE in the time domain − a scheme called the real-time BTE (rt-BTE) − provides a favorable balance between accuracy and computational cost[24,25]. At present, this method can address the ultrafast dynamics of excited electrons[26–28], the coupled nonequilibrium dynamics of electrons and phonons[29–32], and with appropriate extensions to the formalism, the ultrafast dynamics of excitons[33–35]. To accelerate the solution, the rt-BTE method can also take advantage of data-driven techniques[36], such as dynamic mode decomposition[37,38] and interaction compression[39], as well as advanced adaptive and multi-rate time-stepping schemes[32]. The rt-BTE scheme offers a practical approach focused on incoherent dynamics on femtosecond to picosecond timescales that complements methods targeting coherent dynamics, such as time-dependent DFT[40–43] and nonequilibrium Green's functions[44–47].

Despite the overall efficiency, the rt-BTE method still requires parallelization and extensive software optimization, particularly for simulations of materials with large unit cells, and/or using dense momentum grids and targeting long simulation times beyond the picosecond timescale[32]. In the PERTURBO code, after identifying the relevant e-ph scattering processes, the collision integral − the key quantity in BTE calculations − is computed by looping over these scattering channels[48]. Even after selecting a relevant energy window and retaining only energy-conserving scattering channels, in a typical calculation, the total number of active scattering channels can still be as large as 10^8 or higher, which poses a major computational challenge for CPU hardware. Therefore, for both transport and time-domain dynamics, it is particularly important to design a data structure that addresses the high-dimensionality and sparsity of e-ph interactions and scattering processes[32].

Using graphic processing units (GPUs)[49], which are now prevalent and widely available, could be game changing for accelerating BTE calculations of transport and nonequilibrium dynamics, and potentially a broader range of e-ph physics. Unlike CPUs, which typically feature a limited number of high-performance cores and are optimized to handle complex workloads, GPUs can process a large volume of lightweight tasks. Originally designed for processing images, in the last decade GPUs have greatly expanded their scope in scientific computing, becoming an increasingly popular option for high-performance tasks[50]. For example, on the Perlmutter cluster at the National Energy Research Scientific Computing Center (NERSC)[51], at present about 40% of the compute nodes are GPU nodes, each equipped with 28,000 CUDA cores. In contrast, each CPU node contains only 128 cores. Although each GPU core has lower computing power than a CPU core, GPUs can perform a large number of simple tasks with a high degree of parallelism.

However, two key considerations need to be addressed when designing algorithms for GPU execution. First, it is important to minimize data movement between the host and GPUs because it causes substantial overhead. Second, atomic operations must be optimized to avoid communication and synchronization between GPU cores, which causes significant performance loss[52]. This is particularly evident in a parallel CPU implementation of the BTE method, where different scattering channels contributing to the collision integral are typically handled by different processes or threads[48]. Therefore, we seek to design a data structure for e-ph scattering in the BTE that is optimal for use on GPUs.

Several programming frameworks are available for GPU algorithms, such as the widely used CUDA and OpenACC. Due to its low-level architecture, CUDA offers greater control and optimization, but at the cost of increased complexity for code implementation and maintenance. In contrast, OpenACC offers a directive-based approach that enhances code readability and maintainability, with only a slight performance loss. In addition, OpenACC is designed for portability across platforms and thus is not limited to GPUs from any specific vendor. These strengths led us to use OpenACC in this work.

Here, we design an efficient GPU data structure and algorithm for the BTE, and implement them with OpenACC in Perturbo, to accelerate calculations of transport and ultrafast dynamics using GPUs. Our new data structure optimizes data allocation and movement, as well as communication and synchronization between GPU cores. We show benchmarks for performance, memory consumption, and strong scaling in several materials.

Our analysis shows a substantial performance improvement relative to the (already efficient) reference CPU implementation: we achieve a speed-up by ∼40 times for BTE calculations of transport and nonequilibrium dynamics on GPUs, realizing nearly linear scaling up to 100 GPUs. This new implementation was released in PERTURBO v3.0 in early 2025.

II. RESULTS

A. BTE for transport and ultrafast dynamics

The semiclassical BTE models the change in electronic occupations in response to external fields and collision processes. In a solid, these processes are naturally described in momentum space[48]:

$$\frac{\partial f_{n\boldsymbol{k}}(t)}{\partial t} = -\Big[\hbar^{-1}\nabla_{\boldsymbol{k}} f_{n\boldsymbol{k}}(t)\cdot\boldsymbol{F}\Big] + \mathcal{I}[f_{n\boldsymbol{k}}(t)], \tag{1}$$

where $f_{n\boldsymbol{k}}(t)$ is the occupation factor at time t of an electronic state with crystal momentum $\boldsymbol{k}$ and band index n[2,3]. Note that we assume slowly varying fields and homogeneous material and electronic occupations, which removes spatial derivatives. Under an external field $\boldsymbol{F}$ ($\boldsymbol{F} = -e\boldsymbol{E}$ for electrons in the presence of an electric field $\boldsymbol{E}$), the change in electron occupations is determined by the drift term (first term on the right-hand side) and the collision integral $\mathcal{I}$. The BTE is solved in the time domain for ultrafast dynamics and at steady state for transport (see Methods for details). Computing the collision integral is the main bottleneck in the algorithm for both ultrafast dynamics, where it is computed at each time step, and for transport calculations, where it is computed in each iteration step.

B. Collision integral, *e*-ph coupling matrix, and scattering channels

Using Fermi's golden rule, the collision integral $\mathcal{I}$ for nonequilibrium dynamics reads[2,48]

$$\begin{aligned}\mathcal{I}^{e-\mathrm{ph}}[f_{n\boldsymbol{k}}(t)] = -\frac{2\pi}{\hbar}\frac{1}{\mathcal{N}_{\boldsymbol{q}}}\sum_{m\boldsymbol{q}\nu}\Big|g_{mn\nu}(\boldsymbol{k},\boldsymbol{q})\Big|^2\times \\ \Big[\delta\Big(\varepsilon_{n\boldsymbol{k}}-\varepsilon_{m\boldsymbol{k}+\boldsymbol{q}}+\hbar\omega_{\nu\boldsymbol{q}}\Big)\times F_{\mathrm{abs}} + \delta\Big(\varepsilon_{n\boldsymbol{k}}-\varepsilon_{m\boldsymbol{k}+\boldsymbol{q}}-\hbar\omega_{\nu\boldsymbol{q}}\Big)\times F_{\mathrm{em}}\Big].\end{aligned} \tag{2}$$

The above equation describes the e-ph scattering process for an electron from the initial state $|n\boldsymbol{k}\rangle$ with energy $\varepsilon_{n\boldsymbol{k}}$ to the final state $|m\boldsymbol{k}+\boldsymbol{q}\rangle$ with energy $\varepsilon_{m\boldsymbol{k}+\boldsymbol{q}}$, by absorbing

(emitting) a phonon with wave vector $\boldsymbol{q}$ ($-\boldsymbol{q}$), mode index ν, and frequency $\omega_{\nu\boldsymbol{q}}$.

Each absorption or emission process can be labeled using the notation $(\boldsymbol{k}, \boldsymbol{q}, n, m, \nu)$, here referred to as a scattering channel. The factors F_{em} and F_{abs} depend on the carrier occupations $f_{n\boldsymbol{k}}(t)$ and the phonon occupations $N_{\nu\boldsymbol{q}}$. In addition, $g_{mn\nu}(\boldsymbol{k}, \boldsymbol{q})$ are elements of the e-ph coupling matrix $\boldsymbol{g}(\boldsymbol{k}, \boldsymbol{q})$ computed from first principles, the δ functions enforce energy conservation, and crystal momentum conservation is satisfied by using commensurate electron and phonon grids (with $\mathcal{N}_{\mathbf{q}}$ grid points) and selecting appropriate $(\boldsymbol{k}, \boldsymbol{q})$ pairs for each scattering process. For convenience, in the following we denote the collision integral for state $|n, \boldsymbol{k}\rangle$ as $\mathcal{I}(n, \boldsymbol{k}) = \mathcal{I}^{e-\text{ph}}[f_{n\boldsymbol{k}}(t)]$. Although the expression for $\mathcal{I}(n, \boldsymbol{k})$ is different in transport calculations (see Methods), the same data structure and algorithm apply to both transport and ultrafast dynamics, and are illustrated here for the ultrafast dynamics case.

Computing the collision integral $\mathcal{I}$ for all electronic states involves summing over all active scattering channels. The electronic structure and lattice dynamics are first obtained in the entire Brillouin zone. Then, we restrict the electronic states of interest to a given energy window, significantly reducing the total number of $(\boldsymbol{k}, \boldsymbol{q})$ pairs[48]. For each $(\boldsymbol{k}, \boldsymbol{q})$ pair, the nominal number of scattering channels prior to imposing any conservation constraints is $N_b^2 \times N_\nu$, where N_b is the number of included bands and N_ν is the number of phonon modes. By imposing an approximate energy conservation (with a Gaussian δ-function) and discarding channels with $|g_{mn\nu}(\boldsymbol{k}, \boldsymbol{q})|$ below a prescribed cutoff, the set of active scattering channels is further reduced to a small fraction of the total. This process makes $g_{nm\nu}(\boldsymbol{k}, \boldsymbol{q})$ highly sparse in the parameter space $(\boldsymbol{k}, \boldsymbol{q}, n, m, \nu)$ (see Supplementary Fig. 1), saving extensive memory and computational cost[39]. This scattering channel selection algorithm is implemented in PERTURBO v2.2.0 and earlier CPU-based versions[48], and achieves efficient performance and memory usage on CPUs.

However, the calculation of the collision integral $\mathcal{I}$ remains the most computationally demanding part of the BTE workflow and would greatly benefit from GPU acceleration. The CPU implementation is highly optimized using hybrid MPI and OpenMP parallelization but is not readily adapted to GPU parallelization for two main reasons. First, when computing the collision integral, the CPU implementation uses a large number of atomic operations to avoid competition (so-called "race conditions") between threads parallelized over scattering channels. Second, the use of numerous arrays of variable lengths to store information about the scattering channels $(\boldsymbol{k}, \boldsymbol{q}, n, m, \nu)$ requires referencing millions of individual heap alloca-

tions, introducing substantial overhead that limits GPU performance.

We propose a GPU-optimized data structure and algorithm that address these limitations and efficiently calculate the collision integral on GPUs. In the following, we describe this data structure and algorithm, and show benchmarks of performance, memory usage, and scaling behavior, using the CPU implementation of PERTURBO as a reference.

C. Data structure and implementation

1. *Reference CPU algorithm*

As discussed above, the electronic bands, phonon modes, and momenta collectively label an active scattering channel between states $|n\boldsymbol{k}\rangle$ and $|m\boldsymbol{k}+\boldsymbol{q}\rangle$. The e-ph coupling matrix $\boldsymbol{g}$ is effectively sparse and irregular in this parameter space $(\boldsymbol{k}, \boldsymbol{q}, m, n, \nu)$. Consequently, using a single 5-dimensional array to store $g_{nm\nu}(\boldsymbol{k}, \boldsymbol{q})$ leads to highly inefficient code as many of its entries are zero or very small (see Supplementary Fig. 1). In practice, we group together all active scattering channels involving the same $(\boldsymbol{k}, \boldsymbol{q})$ pairs, and create an abstract type containing all relevant information for each pair. This design leverages benefits of object-oriented programming, such as flexibility and maintainability, while at the same time decoupling the $(\boldsymbol{k}, \boldsymbol{q})$ pairs, which enables efficient distributed programming using MPI. Specifically, we define the object `scatter_base` to store all the relevant information for all the active scattering channels of each $(\boldsymbol{k}, \boldsymbol{q})$ pair, thus filtering out all redundant e-ph and scattering channel data. A schematic of this data structure is shown in Fig. 1.

We use this implementation and data structure, available in PERTURBO v2.2.0 and earlier versions[48], as a reference or baseline for benchmarking code performance. This "Baseline-CPU" algorithm features hybrid MPI plus OpenMP CPU parallelization, where the $\boldsymbol{k}$ points are evenly distributed over different MPI processes. For ultrafast dynamics simulations, this code has two nested loops for each $\boldsymbol{k}$ point: an outer loop over $(\boldsymbol{k}, \boldsymbol{q})$ pairs accelerated with OpenMP, and an inner loop over active scattering channels for the current $(\boldsymbol{k}, \boldsymbol{q})$ pair, executed sequentially by each OpenMP thread. For transport, there is an additional inner loop over Cartesian components of the external field. When the same collision integral $\mathcal{I}(n, \boldsymbol{k})$ is updated by multiple threads simultaneously, as in the red arrows in Fig. 1, the race condition between threads is avoided using OpenMP atomic operations.

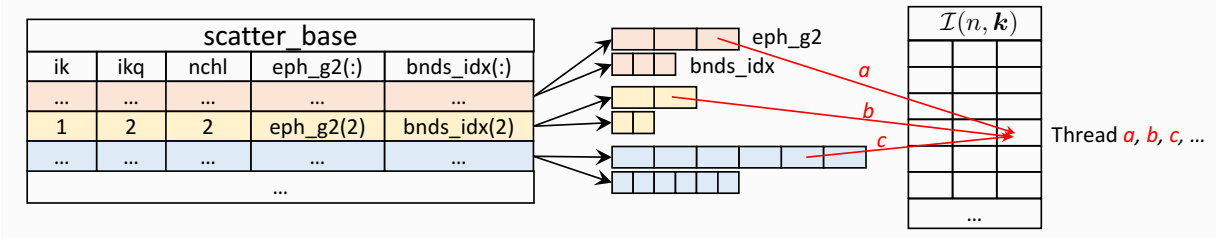

Figure 1. **Schematic of the `scatter_base` data structure.** The information for each $(\boldsymbol{k}, \boldsymbol{q})$ pair is stored as a separate entry in `scatter_base`, as shown using different colors. The variables `ik`, `ikq`, `nchl` are indices of the $\boldsymbol{k}$ and $\boldsymbol{k}+\boldsymbol{q}$ points and the number of active scattering channels, respectively. The variables `eph_g2` and `bnds_idx` are arrays holding, respectively, the squared norm of the e-ph matrix elements and the joint indices of bands and phonon modes for each scattering channel. The relation of these variables to the collision integral $\mathcal{I}(n, \boldsymbol{k})$, whose components can be updated by multiple processes and threads simultaneously, is shown using red arrows.

2. Optimized GPU algorithm

A direct implementation of the Baseline-CPU algorithm on GPUs would be inherently inefficient as GPUs are optimized for executing many lightweight threads concurrently, and therefore perform poorly when frequent atomic updates are required. Moreover, OpenACC generates one data transfer per memory allocation, so allocating a large number of variable-length arrays incurs significant overhead. The combination of numerous heap allocations and frequent atomic operations across threads would severely limit the efficiency of a GPU version of the above algorithm.

To achieve GPU acceleration, we redesign the data structure and code implementation for the key step of the BTE algorithm, the calculation of the collision integral. In the optimized data structure, shown in Fig. 2, we allocate `scatter` and `scatter_channels`, which store the same information as `scatter_base` but using fixed-size buffers instead of variable-length arrays. This avoids dynamic GPU allocations and improves performance, while preserving the flexibility of object-oriented programming. In addition, to eliminate atomic updates on the GPU, we develop an algorithm that inverts the accumulation scheme: rather than assigning multiple threads to update the contribution of each scattering channel to one collision integral $\mathcal{I}(n, \boldsymbol{k})$, it distributes threads over $\mathcal{I}(n, \boldsymbol{k})$ itself. Each thread then identifies the scattering channels that contribute to its assigned $\mathcal{I}(n, \boldsymbol{k})$. The calculation of

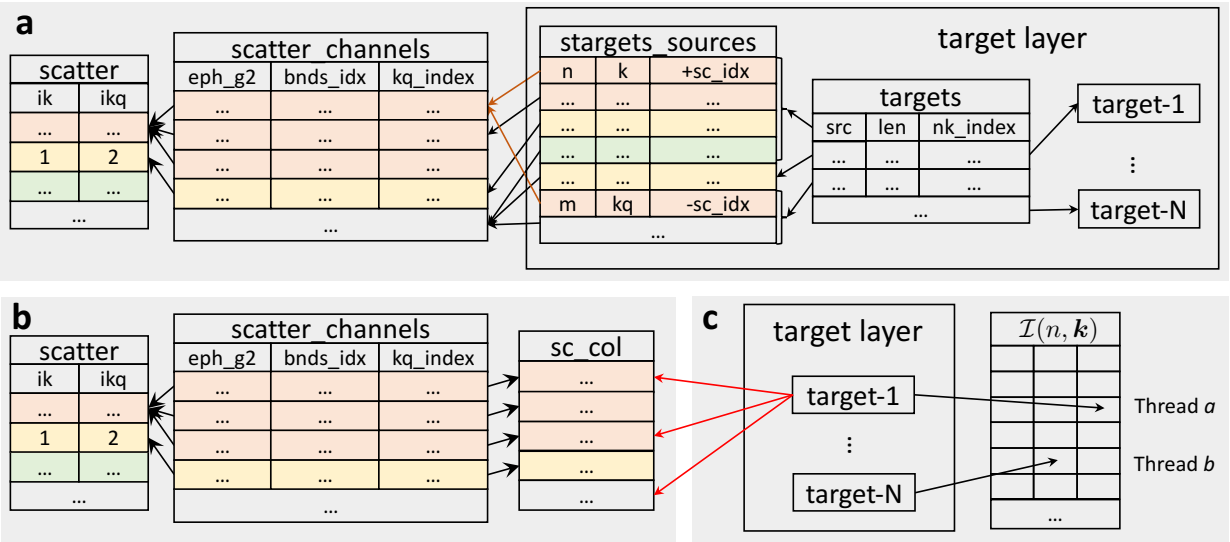

Figure 2. Data structure optimized for GPUs. **a** Setup of the scattering channels and target layer. Relevant quantities for the $(\boldsymbol{k},\boldsymbol{q})$ pairs are stored into multiple arrays: `scatter`, which stores the indexes of the $\boldsymbol{k}$ and $\boldsymbol{k}+\boldsymbol{q}$ points, and `scatter_channels`, which stores information for all scattering channels, such as the square of the e-ph matrix elements (`eph_g2`), the joint indices of bands and phonon modes (`bnds_idx`), and the index of the $(\boldsymbol{k},\boldsymbol{q})$ pair (`kq_index`). Scattering channels shown with the same color are associated with the same $(\boldsymbol{k},\boldsymbol{q})$ pair. In addition, `stargets_sources` indexes the elements of the collision integral $\mathcal{I}(n,\boldsymbol{k})$ and the position of the scattering channels (`sc_idx`) in `scatter_channels`. The positive (negative) sign of `sc_idx` reflects how that entry contributes to the collision integral. The rows of `stargets_sources` are arranged in order, with rows sharing the same $(n,\boldsymbol{k})$ grouped together, as shown with curly braces. Each such group is called a target, and for each group, the position in `stargets_sources` (`src`), the length (`len`), and the combined $(n,\boldsymbol{k})$ index (`nk_index`) are stored in `targets`. Together, `stargets_sources` and `targets` constitute the target layer. **b** Calculation of the contribution to the collision integral from each scattering channel, defined in Eq. 2, which is computed and stored in `sc_col`. **c** Update of collision integrals $\mathcal{I}(n,\boldsymbol{k})$. Each element of $\mathcal{I}$ is updated by one target and one thread of execution. Using the target layer described in (**a**), each target is able to find the contribution of all the associated scattering channels in `sc_col`, as shown with red arrows.

the collision integral with the optimized GPU algorithm and data structure consists of three steps:

1. *Create the target layer:*

   In the first step, the scattering channels are traversed to determine to which elements of the collision integral $\mathcal{I}(n, \boldsymbol{k})$ they contribute. Channels contributing to the same element of $\mathcal{I}$ are grouped together and form a `target`. This grouping is handled in the target layer (Fig. 2a), where each scattering channel of `scatter_channels` contributes to two different collision integrals, $\mathcal{I}(n, \boldsymbol{k})$ and $\mathcal{I}(m, \boldsymbol{k}+\boldsymbol{q})$. These contributions are equal in magnitude but opposite in sign: In Fig. 2a, the positive sign of `sc_idx` denotes the contribution to $\mathcal{I}(n, \boldsymbol{k})$ and the negative sign to $\mathcal{I}(m, \boldsymbol{k}+\boldsymbol{q})$). The rows of `stargets_sources` are sorted based on their contribution to the collision integral and grouped into a `target`. This step is not computationally intensive and is performed only once on CPUs. See Supplementary Note 1 for demo code related to this step.

2. *Compute the contribution from each scattering channel:*

   The second step evaluates the contribution of each scattering channel to the collision integral. As shown in Fig. 2b, the contribution to $\mathcal{I}(n, \boldsymbol{k})$ from each channel is stored in the variable `sc_col`, which has the same number of rows as `scatter_channels`. This computationally demanding step is accelerated on GPUs using heterogeneous programming with OpenACC. Example code is provided in Supplementary Note 2.

3. *Collect the contributions from all targets:*

   This step updates the collision integrals using contributions from all targets computed in Step 2. The routine loops over `targets` to update the elements of $\mathcal{I}$ by summing over contributions from all scattering channels in each target (Fig. 2c). This way, only one element of $\mathcal{I}$ will be updated for each thread, as shown with black arrows in Fig. 2c. Using `target`-1 as an example, the first row of `targets` records the position of this target in `stargets_sources`. By using the value of `sc_idx` for all relevant elements in `stargets_sources`, the code finds all the scattering channels in `sc_col` (see red arrows in Fig. 2c). Then those values are summed together to update the corresponding element of $\mathcal{I}$ in the current thread. This step is performed on GPUs for acceleration. Demo code for this step is provided in Supplementary Note 3.

The new data structure resolves the inefficiency of the Baseline-CPU method by minimizing host–device data transfers, reducing the number of atomic operations across threads, as well as eliminating memory padding through data alignment.

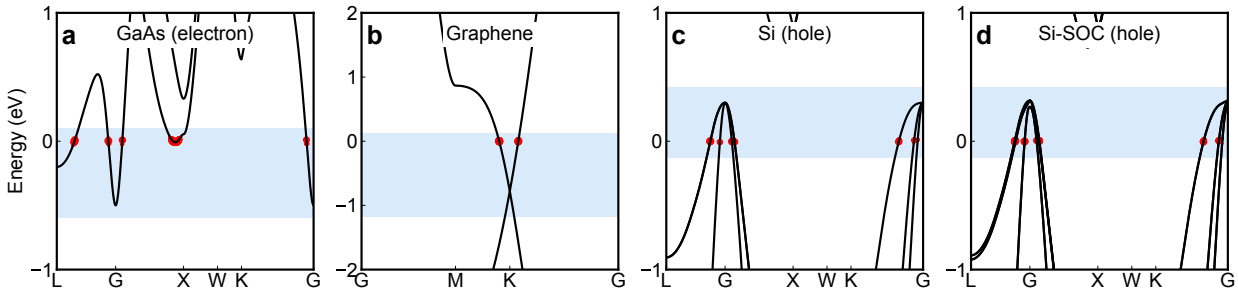

Figure 3. **Simulation setup for four systems.** **a** Electrons in GaAs, **b** electrons in graphene, **c** hole carriers in silicon, and **d** holes in silicon with SOC. Band structures are shown together with the selected energy windows (shaded regions) and the initial populations for the nonequilibrium dynamics simulations (red dots). Energies are shifted so that the Fermi energy is at 0 eV.

As our discussion has focused on ultrafast dynamics, we briefly mention the main differences in the implementation for transport calculations. For ultrafast dynamics, the contribution to the collision integrals $\mathcal{I}(n, \boldsymbol{k})$ and $\mathcal{I}(m, \boldsymbol{k}+\boldsymbol{q})$ from a single scattering channel is equal in magnitude and opposite in sign. This allows us to define only a 1D array for `sc_col`(:) and use the sign of `sc_idx` for bookkeeping. For transport, this is not possible, and thus `sc_col` needs an additional dimension. In practice, `sc_col` is defined as a 3D array to account for the external field, and the code has an additional loop over the directions of the field. These small differences do not affect the performance.

D. Performance, memory usage, and scaling analysis of GPU code

1. Simulation setup

We benchmark the GPU implementation on four selected systems: electron carriers in gallium arsenide (GaAs), graphene, hole carriers in silicon (Si) modeled without spin-orbit coupling (SOC), and hole carriers in silicon with SOC (Si-SOC). These cases cover a range of scenarios, including metals and semiconductors, calculations with and without SOC, electron and hole carriers, and 2D and bulk materials. The band structures, energy windows for nonequilibrium dynamics, and the initial population for nonequilibrium simulation for all four systems are shown in Fig. 3.

As shown in Table 1, in the ultrafast dynamics simulation of electrons in GaAs, after imposing an energy window of 0.7 eV above the conduction band edge, the number of $\boldsymbol{k}$ and $\boldsymbol{q}$ points in GaAs are reduced from 135^3 to 56713 (2% of the original value) and 637412 (26% of the original value), respectively. Imposing energy conservation and a cutoff on $|\mathbf{g}(\boldsymbol{k}, \boldsymbol{q})|$ further reduces the number of scattering channels, from a nominal value of 10^{11} to an actual value of 10^8. These values justify our approach of keeping only the active scattering channels (10^8 in the case of GaAs) and looping over these channels in the code.

TABLE 1. Summary of simulation parameters for the four systems studied, including the number of bands and phonon modes (n,ν), the number of $(\boldsymbol{k}, \boldsymbol{q})$ points, and the number of nominal and active scattering channels.

Systems	(n,ν)	$(\#\boldsymbol{k}, \#\boldsymbol{q})$		# channels	
		Nominal	Energy window	Nominal	Active
GaAs	(1,6)	(135^3, 135^3)	(56713, 637412)	10^{11}	10^8
graphene	(1,6)	(1300^2, 1300^2)	(43206, 131605)	10^{11}	10^8
Si	(3,6)	(105^3, 105^3)	(44275, 232728)	10^{12}	10^8
Si-SOC	(6,6)	(95^3, 95^3)	(29317, 151560)	10^{12}	10^7

2. Performance comparisons

We first compare the wall-time of the optimized-GPU code with the Baseline-CPU algorithm to directly show the performance improvement. Figure 4a-d compare calculations carried out with the baseline CPU algorithm, used as a reference, and the optimized GPU algorithm, for both transport and ultrafast dynamics, for the four systems studied. For transport calculations, the wall time used in the plots is the average elapsed wall time of each iteration in the iterative BTE solution, and for ultrafast dynamics, the wall time is for one time step of the rt-BTE simulation.

The speed up of the GPU implementation relative to the baseline CPU code is noteworthy. Our optimized GPU code achieves a speedup by a factor of 44 for transport and 35 for ultrafast dynamics. We find similar speed-ups for all systems in our test set, showing

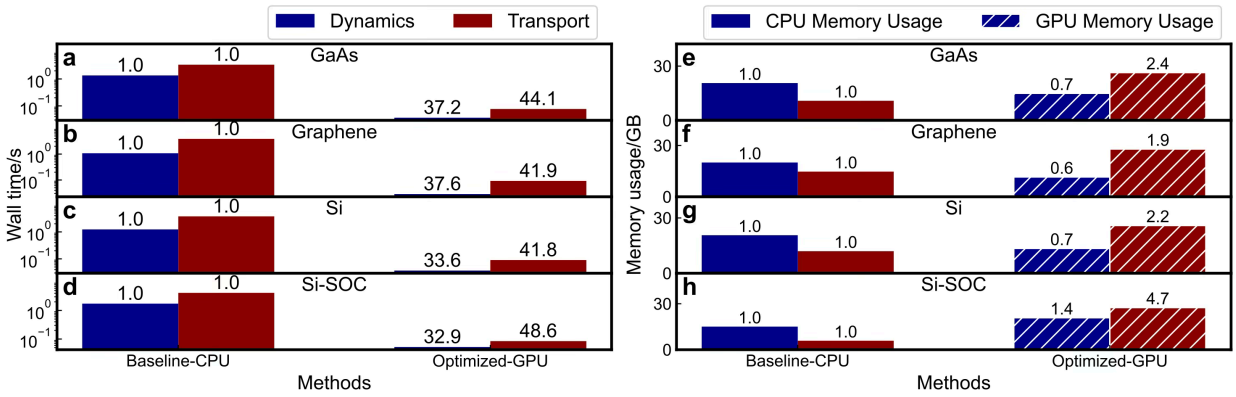

Figure 4. **Performance of the optimized GPU implementation. a-d**, performance, and **e-h**, memory usage, for the four systems studied here, respectively. The left panels, **a**–**d**, show the wall time (in seconds, on a logarithmic scale) for ultrafast dynamics (blue) and transport (red) calculations. The speedup values, obtained as the ratio of Baseline-CPU to optimized-GPU code wall times, are given above each bar in the optimized-GPU results. The right panels, **e-h**, give the memory usage (in GB) on CPU (solid colors) and GPU (striped bars) for the same systems. Memory usage values annotated in the plot are referenced to the baseline CPU results.

that the speed-up is an intrinsic feature of the GPU algorithm independent of the system studied. This order-of-magnitude speed up is the result of careful design and optimization of array structures, data transfer, and thread management in our GPU algorithm.

To demonstrate the importance of our novel data structure optimized for GPUs, we run the optimized GPU code with and without OpenACC directives in the same HPC settings. This test compares the performance of the same GPU-optimized code executed on GPU versus CPU hardware; this is a fair and established approach to compare CPU and GPU code. As shown in the Supplementary Fig. 2, the optimized-GPU code runs 25−50 times faster on GPUs than on CPU hardware, for both transport and ultrafast dynamics. This result demonstrates the considerable acceleration achieved on GPUs.

Next, we compare memory consumption in the baseline CPU and optimized GPU algorithms. In Fig. 4e-h, we show data on memory usage. For all calculations without SOC, we find that the memory allocation in the optimized GPU code is approximately 70% for ultrafast dynamics, and 200% for transport, relative to the memory used in the baseline CPU code. This difference arises from the need to store several intermediate variables in the GPU implementation of transport, in particular the direction of the applied electric field,

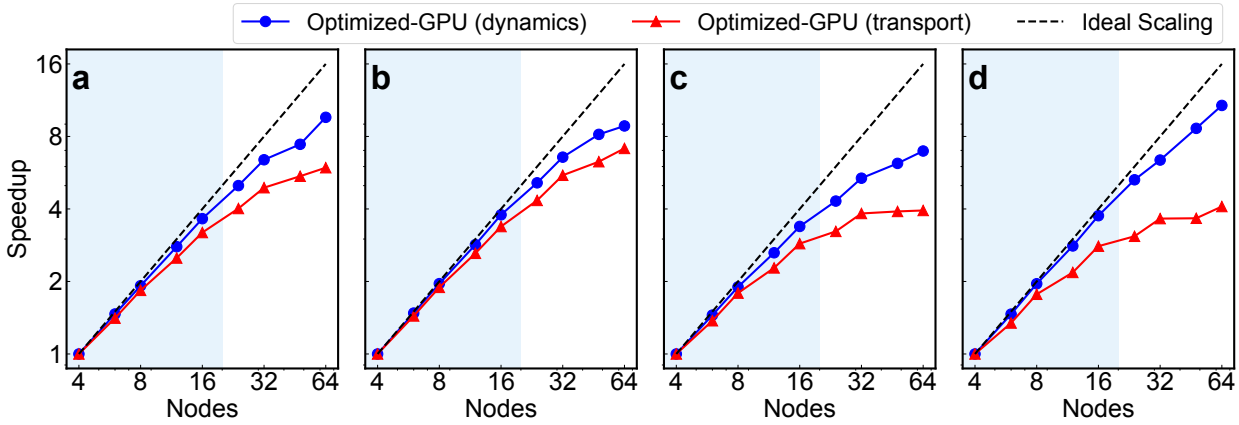

Figure 5. **Strong-scaling performance.** Speedup versus number of GPU nodes for **a** GaAs, **b** graphene, **c** Si, and **d** Si with SOC. Results for the optimized-GPU code are shown using solid lines with symbols for ultrafast dynamics (purple) and transport (red). The dashed line shows the ideal linear scaling. Common scenarios for most users ($\leq$ 20 GPU nodes) are indicated with shaded regions.

which in the Baseline-CPU code is replaced by an iterative loop without loss of performance. Finally, the calculation with SOC in Fig. 4h uses more memory than the cases without SOC. The reason is that the number of scattering channels in the arrays scales with the number of bands, resulting in a higher memory usage when SOC is included. Despite the higher memory usage, the wall-time speed-up is unchanged in the presence of SOC.

3. Strong scaling analysis

The strong scaling measures how the speedup scales with the number of computing nodes for a fixed simulation size. Therefore, strong-scaling benchmarks address a key question for computationally intensive simulations: What performance improvement can one obtain by increasing the computational resources? To avoid confusion with the speed-up of GPU versus CPU code discussed above, we define the strong-scaling speed-up as:

$$\text{Strong-scaling speedup}_N = \frac{T_{4-\text{nodes}}}{T_{\text{N}-\text{nodes}}} \tag{3}$$

where $T_{4-\text{nodes}}$ and $T_{\text{N}-\text{nodes}}$ are the wall times for simulations using 4 and N nodes, respectively.

We carry out strong-scaling benchmarks for the optimized GPU code, for both transport and ultrafast dynamics, using between 4 and 64 GPU nodes. Setup details of these simulations, which employ very dense grids in momentum space, are provided in Table 2. Strong scaling results for the four systems studied here are shown in Fig. 5. Based on the definition of strong-scaling speedup given above, the ideal speedup is $\frac{N}{4}$, where N is the number of nodes. This ideal value is shown in Fig. 5 with a dashed line and used as a reference.

Our results in Fig. 5 show that in common scenarios for most users, consisting of calculations with up to 20 GPU nodes, our GPU code exhibits nearly ideal scaling performance, which extends up to 24 nodes or more nodes in most cases. The only case that deviates from this trend is transport calculations in Si with or without SOC (red lines in Fig. 5c-d). Due to the high memory demand for these calculations, the scaling remains nearly ideal up to 8 GPU nodes, but deviates increasingly beyond that. For all systems, the acceleration efficiency drops to around 40–60% of the ideal scaling at 64 nodes. This reduction is common in GPU codes and is mainly due to insufficient workload per GPU when such a large number of nodes is employed. This result implies that allocating excessive GPU resources to a calculation that can be completed on a much smaller number of nodes is unnecessary and inefficient.

TABLE 2. Momentum grid size and computational resources for performance and strong scaling analysis on four systems. Note that the momentum grid is two-dimensional in graphene and three-dimensional for the other materials.

		GaAs	graphene	Si	Si-SOC
Performance	Transport	120^3	1200^2	95^3	75^3
(1 node, size fixed)	Dynamics	135^3	1300^2	105^3	90^3
Strong scaling	Transport	155^3	1700^2	125^3	95^3
(#nodes vary, size fixed)	Dynamics	195^3	2300^2	150^3	120^3

III. DISCUSSION

In this work, we have developed an efficient GPU algorithm and data structure to accelerate BTE calculations of electronic transport and ultrafast dynamics governed by e-ph interactions. The code is developed in OpenACC and is included in version 3.0 of the open-source code PERTURBO. Due to the directive-based approach used in OpenACC, the implementation is easy to maintain. With GPU acceleration, this new version of PERTURBO provides a highly efficient MPI+OpenMP+GPU parallelization that can fully take advantage of modern Exascale HPC computing environments with multi-core CPUs and GPU accelerators.

Through extensive benchmarks, we report speed-ups by a factor of $\sim$40 for transport and ultrafast dynamics relative to the reference, state-of-the-art CPU implementation in the previous version of PERTURBO. This result is achieved by reformulating the data structure and algorithm to store e-ph interactions and carry out calculations of collision integrals. Our approach optimizes data allocation and movement between host and GPUs and synchronization between numerous GPU cores. We analyze the performance, memory consumption, and strong scaling for three materials. The strong scaling analysis shows nearly ideal scaling up to 16 GPU nodes (64 GPUs), with only a slight decrease in performance up to 24 GPU nodes (96 GPUs) for most cases.

While the optimized GPU data structure is discussed in the context of e-ph interactions, the same data structure can also be used for other scattering mechanisms, such as electron-defect[53], exciton-phonon[33], magnon-phonon[54], and phonon-phonon[29,32] interactions. Note that we did not consider GPU acceleration for the interpolation of e-ph matrices, which has been studied in previous work[55,56], mainly because recently developed compression algorithms[39] make the e-ph interpolation routines already highly efficient. In a future release, we will extend the GPU acceleration feature to other modules of PERTURBO.

IV. METHODS

A. Ultrafast dynamics and carrier transport simulations

1. *Ultrafast dynamics simulations*

The ultrafast carrier dynamics, described by the time evolution of carrier occupations $f_{n\boldsymbol{k}}(t)$, can be simulated by solving the rt-BTE defined in Eq. 1 starting from an initial nonequilibrium distribution. The absorption and emission occupation factors in Eq. 2 are defined as[48]:

$$\begin{aligned} F_{\text{abs}} &= f_{n\boldsymbol{k}}(1 - f_{m\boldsymbol{k}+\boldsymbol{q}})N_{\nu\boldsymbol{q}} - f_{m\boldsymbol{k}+\boldsymbol{q}}(1 - f_{n\boldsymbol{k}})(N_{\nu\boldsymbol{q}} + 1) \\ F_{\text{em}} &= f_{n\boldsymbol{k}}(1 - f_{m\boldsymbol{k}+\boldsymbol{q}})(N_{\nu\boldsymbol{q}} + 1) - f_{m\boldsymbol{k}+\boldsymbol{q}}(1 - f_{n\boldsymbol{k}})N_{\nu\boldsymbol{q}}, \end{aligned} \tag{4}$$

where $f_{n\boldsymbol{k}}$ and $N_{\nu\boldsymbol{q}}$ denote electron and phonon occupations. Starting with an initial value of $f_{n\boldsymbol{k}}(t)$, the rt-BTE can be solved by explicit time-stepping, in our case using the fourth-order Runge-Kutta method. The collision integrals for each band and momentum are evaluated at each time step, with a typical simulation comprising 1,000−10,000 time steps. More advanced methods such as adaptive and multirate time integration have also recently been proposed for more efficient time stepping[32]. The choice of the initial electron occupation depends on the specific problem being studied, with common options including Lorentzian, Fermi-Dirac, and Gaussian distributions in energy. We use Gaussian energy distributions to model the initial electronic occupations for ultrafast dynamics simulations of all materials studied in this work.

2. *Electronic transport calculations*

The BTE reaches a steady state when the time derivative $\partial f_{n\boldsymbol{k}}(t)/\partial t$ in Eq. 1 vanishes. At steady state, the drift term from the external field $\boldsymbol{E}$ is balanced by the e-ph collisions. For weak electric fields, the electron occupations $f_{n\boldsymbol{k}}$ can be expanded to first-order in the field as[48]

$$\begin{aligned} f_{n\boldsymbol{k}} &= f^0_{n\boldsymbol{k}} + f^1_{n\boldsymbol{k}} + \mathcal{O}(E^2) \\ &= f^0_{n\boldsymbol{k}} + e\boldsymbol{E} \cdot \boldsymbol{F}_{nk} \frac{\partial f^0_{n\boldsymbol{k}}}{\partial \epsilon_{n\boldsymbol{k}}} + \mathcal{O}(E^2), \end{aligned} \tag{5}$$

where $f^0_{n\boldsymbol{k}}$ is the equilibrium Fermi-Dirac distribution and $\boldsymbol{F}_{nk}$ characterizes the first-order deviation from the equilibrium distribution.

Substituting Eq. 5 into Eq. 1, we obtain an iterative approach to compute the deviation from equilibrium[48]:

$$\boldsymbol{F}^{i+1}_{n\boldsymbol{k}} = \boldsymbol{F}^{0}_{n\boldsymbol{k}} + \frac{\tau_{n\boldsymbol{k}}}{\mathcal{N}_{\boldsymbol{q}}} \sum_{m,\nu\boldsymbol{q}} \boldsymbol{F}^{i}_{m\boldsymbol{k}+\boldsymbol{q}} W^{\nu\boldsymbol{q}}_{n\boldsymbol{k},m\boldsymbol{k}+\boldsymbol{q}}, \tag{6}$$

from which the conductivity and other transport coefficients can be obtained. Above, the e-ph scattering rate is defined as

$$\begin{aligned} W^{\nu\boldsymbol{q}}_{n\boldsymbol{k},m\boldsymbol{k}+\boldsymbol{q}} = \frac{2\pi}{\hbar} \Big| g_{mn\nu}(\boldsymbol{k},\boldsymbol{q}) \Big|^2 \\ \times \Big[\delta\Big(\epsilon_{n\boldsymbol{k}} - \hbar\omega_{\nu\boldsymbol{q}} - \epsilon_{m\boldsymbol{k}+\boldsymbol{q}}\Big)(1 + N_{\nu\boldsymbol{q}} - f_{m\boldsymbol{k}+\boldsymbol{q}}) \\ + \delta\Big(\epsilon_{n\boldsymbol{k}} + \hbar\omega_{\nu\boldsymbol{q}} - \epsilon_{m\boldsymbol{k}+\boldsymbol{q}}\Big)(N_{\nu\boldsymbol{q}} + f_{m\boldsymbol{k}+\boldsymbol{q}}) \Big] \end{aligned} \tag{7}$$

and $\boldsymbol{F}^{0}_{n\boldsymbol{k}} = \tau_{n\boldsymbol{k}} v_{n\boldsymbol{k}} = \left(\frac{1}{N_{\boldsymbol{q}}} \sum_{m,\nu\boldsymbol{q}} W^{\nu\boldsymbol{q}}_{n\boldsymbol{k},m\boldsymbol{k}+\boldsymbol{q}} \right)^{-1} v_{n\boldsymbol{k}}$ is the deviation from equilibrium in the relaxation time approximation, which is used as the initial guess in the iterative method.

3. Collision integral for transport calculations

The collision integral for transport calculations can be obtained from Eq. 6 as[48]

$$\begin{aligned} \mathcal{I}(n,\boldsymbol{k},\alpha) = \frac{2\pi}{\hbar\mathcal{N}_{\boldsymbol{q}}} \tau_{n\boldsymbol{k}} \sum_{m\boldsymbol{q}\nu} |g_{mn\nu}(\boldsymbol{k},\boldsymbol{q})|^2 \times \boldsymbol{F}^{i}_{m\boldsymbol{k}+\boldsymbol{q},\alpha} \times \\ \Big[\delta\Big(\epsilon_{n\boldsymbol{k}} - \hbar\omega_{\nu\boldsymbol{q}} - \epsilon_{m\boldsymbol{k}+\boldsymbol{q}}\Big)(1 + N_{\nu\boldsymbol{q}} - f_{m\boldsymbol{k}+\boldsymbol{q}}) \\ + \delta\Big(\epsilon_{n\boldsymbol{k}} + \hbar\omega_{\nu\boldsymbol{q}} - \epsilon_{m\boldsymbol{k}+\boldsymbol{q}}\Big)(N_{\nu\boldsymbol{q}} + f_{m\boldsymbol{k}+\boldsymbol{q}}) \Big], \end{aligned} \tag{8}$$

where α denotes the Cartesian directions of the applied electric field $\boldsymbol{E}$. For transport calculations using the iterative solution of the BTE, collision integrals are typically evaluated for 10−100 iterations to reach convergence, depending on the specific system and conditions.

B. Benchmark setup and computational environment

We design benchmarks that use different momentum-grid sizes and computational resources (see Table 2). Performance tests are conducted on a single compute node and the strong-scaling analysis is performed on 4 to 64 nodes. The grid size for each material is chosen to fit on one node for performance tests and on four nodes for strong scaling tests. As the memory requirements for transport and ultrafast dynamics are different, we use different grid sizes for these two types of calculations.

For a fair comparison, all benchmark tests − including both CPU and GPU calculations − are performed on the heterogeneous GPU nodes of the Perlmutter cluster at NERSC. Each heterogeneous GPU node consists of one CPU (AMD EPYC 7763) with 64 cores and four GPUs (Nvidia A100 with 40GB)[51]. For transport and ultrafast dynamics simulations using the GPU-optimized code, only the collision integral, which is the most computationally expensive part in the solution of the BTE, is performed on GPUs, while the remaining part of the algorithm is executed on CPUs. Each CPU (GPU) calculation is repeated 10 (50) times independently, and we report the average wall-time to ensure statistical significance of the results. Finally, the accuracy of the GPU implementation was validated through integration tests, not discussed in this work, using the Python-based package PERTURBOPY[57].

C. First-principles calculations

The first-principles calculations are performed following the established workflow discussed in Ref.[48]. In the first step, the electronic ground states and band structures of the four systems are obtained from DFT calculations using the QUANTUM ESPRESSO package[58]. We use the local density approximation (LDA)[59] and norm-conserving pseudopotentials[60]. The lattice dynamics and e-ph perturbation potentials are computed using density functional perturbation theory (DFPT) on $8 \times 8 \times 8$, $18 \times 18 \times 1$, and $8 \times 8 \times 8$ $\boldsymbol{q}$-point grids for GaAs, graphene, and silicon, respectively.

Using these inputs, together with Wannier functions generated with the WANNIER90 code[61], we calculate the e-ph coupling matrices and interpolate them onto denser grids using the open-source PERTURBO code. The momentum grids are chosen separately for different simulations and are given in the main text. The delta functions enforcing energy conservation during scattering processes are approximated using Gaussian functions with a 5 meV broadening. Relatively large energy windows − respectively, 0.7 eV for GaAs, 1.3 eV for graphene, and 0.55 eV for silicon − are used to generate grids that fill the computational capacity of one node in benchmark tests and 4 nodes in strong-scaling tests, as discussed above. For ultrafast dynamics simulations, the initial distributions of excited carriers are modeled as narrow Gaussian functions with broadening parameters of 20 meV for GaAs and graphene, and 10 meV for silicon. The fourth-order Runge-Kutta method is used to time-step the rt-BTE.

V. DATA AVAILABILITY

Additional data supporting the findings of this study are available in the Supplementary Information. The source data and plotting scripts used to generate the figures are provided via Zenodo at https://doi.org/10.5281/zenodo.17522786.

VI. CODE AVAILABILITY

The newly developed GPU implementation has been publicly released in PERTURBO v3.0, which is available for download at https://perturbo-code.github.io.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Grant No. OAC-2209262. The ultrafast carrier dynamics calculations are based on work performed within the Liquid Sunlight Alliance, which is supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, and Fuels from Sunlight Hub under Award DE-SC0021266. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 using NERSC award NERSC DDR-ERCAP0026831.

VII. COMPETING INTERESTS

The authors declare no competing interests.

[1] Ziman, J. M. *Electrons and Phonons: the Theory of Transport Phenomena in Solids* (Oxford university press, 2001). https://academic.oup.com/book/32666.

[2] Mahan, G. D. *Condensed Matter in a Nutshell* (Princeton University Press, 2011). https://press.princeton.edu/books/hardcover/9780691140162/condensed-matter-in-a-nutshell.

[3] Bernardi, M. First-principles dynamics of electrons and phonons. *Eur. Phys. J. B* **89**, 1–15 (2016). https://link.springer.com/article/10.1140/epjb/e2016-70399-4.

[4] Giustino, F. Electron-phonon interactions from first principles. *Rev. Mod. Phys.* **89**, 015003 (2017). https://journals.aps.org/rmp/abstract/10.1103/RevModPhys.89.015003.

[5] Park, C.-H. *et al.* Electron–phonon interactions and the intrinsic electrical resistivity of graphene. *Nano Lett.* **14**, 1113–1119 (2014). https://pubs.acs.org/doi/full/10.1021/nl402696q.

[6] Mustafa, J. I., Bernardi, M., Neaton, J. B. & Louie, S. G. Ab initio electronic relaxation times and transport in noble metals. *Phys. Rev. B* **94**, 155105 (2016). https://journals.aps.org/prb/abstract/10.1103/PhysRevB.94.155105.

[7] Li, W. Electrical transport limited by electron-phonon coupling from Boltzmann transport equation: An ab initio study of Si, Al, and MoS_2. *Phys. Rev. B* **92**, 075405 (2015). https://journals.aps.org/prb/abstract/10.1103/PhysRevB.92.075405.

[8] Zhou, J.-J. & Bernardi, M. Ab initio electron mobility and polar phonon scattering in GaAs. *Phys. Rev. B* **94**, 201201(R) (2016). https://journals.aps.org/prb/abstract/10.1103/PhysRevB.94.201201.

[9] Liu, T.-H., Zhou, J., Liao, B., Singh, D. J. & Chen, G. First-principles mode-by-mode analysis for electron-phonon scattering channels and mean free path spectra in GaAs. *Phys. Rev. B* **95**, 075206 (2017). https://link.aps.org/doi/10.1103/PhysRevB.95.075206.

[10] Poncé, S., Margine, E. R. & Giustino, F. Towards predictive many-body calculations of phonon-limited carrier mobilities in semiconductors. *Phys. Rev. B* **97**, 121201 (2018). https://link.aps.org/doi/10.1103/PhysRevB.97.121201.

[11] Lee, N.-E., Zhou, J.-J., Agapito, L. A. & Bernardi, M. Charge transport in organic molecular semiconductors from first principles: The bandlike hole mobility in a naphthalene crys-

tal. *Phys. Rev. B* **97**, 115203 (2018). https://journals.aps.org/prb/abstract/10.1103/PhysRevB.97.115203.

[12] Cheng, L., Zhang, C. & Liu, Y. Why two-dimensional semiconductors generally have low electron mobility. *Phys. Rev. Lett.* **125**, 177701 (2020). https://link.aps.org/doi/10.1103/PhysRevLett.125.177701.

[13] Poncé, S., Li, W., Reichardt, S. & Giustino, F. First-principles calculations of charge carrier mobility and conductivity in bulk semiconductors and two-dimensional materials. *Rep. Prog. Phys.* **83**, 036501 (2020). https://iopscience.iop.org/article/10.1088/1361-6633/ab6a43/meta.

[14] Desai, D. C., Zviazhynski, B., Zhou, J.-J. & Bernardi, M. Magnetotransport in semiconductors and two-dimensional materials from first principles. *Phys. Rev. B* **103**, L161103 (2021). https://journals.aps.org/prb/abstract/10.1103/PhysRevB.103.L161103.

[15] Chang, B. K., Zhou, J.-J., Lee, N.-E. & Bernardi, M. Intermediate polaronic charge transport in organic crystals from a many-body first-principles approach. *Npj Comput. Mater.* **8**, 63 (2022). https://www.nature.com/articles/s41524-022-00742-6.

[16] Chang, B. K. & Bernardi, M. Bandlike charge transport and electron-phonon coupling in organic molecular crystals. *J. Phys.: Condens. Matter* **37**, 095704 (2025). https://iopscience.iop.org/article/10.1088/1361-648X/ad9da6/meta.

[17] Zhou, J.-J., Hellman, O. & Bernardi, M. Electron-phonon scattering in the presence of soft modes and electron mobility in $SrTiO_3$ perovskite from first principles. *Phys. Rev. Lett.* **121**, 226603 (2018). https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.121.226603.

[18] Zhou, J.-J. & Bernardi, M. Predicting charge transport in the presence of polarons: the beyond-quasiparticle regime in $SrTiO_3$. *Phys. Rev. Res.* **1**, 033138 (2019). https://journals.aps.org/prresearch/abstract/10.1103/PhysRevResearch.1.033138.

[19] Luo, Y., Park, J. & Bernardi, M. First-principles diagrammatic Monte Carlo for electron-phonon interactions and polaron. *Nat. Phys.* **21**, 1275–1282 (2025). https://www.nature.com/articles/s41567-025-02954-1.

[20] Abramovitch, D. J., Zhou, J.-J., Mravlje, J., Georges, A. & Bernardi, M. Combining electron-phonon and dynamical mean-field theory calculations of correlated materials: Transport in the correlated metal Sr_2RuO_4. *Phys. Rev. Mater.* **7**, 093801 (2023). https://journals.aps.

org/prmaterials/abstract/10.1103/PhysRevMaterials.7.093801.

[21] Abramovitch, D. J., Mravlje, J., Zhou, J.-J., Georges, A. & Bernardi, M. Respective roles of electron-phonon and electron-electron interactions in the transport and quasiparticle properties of $SrVO_3$. *Phys. Rev. Lett.* **133**, 186501 (2024). https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.133.186501.

[22] Gao, S., Zhou, J.-J., Luo, Y. & Bernardi, M. First-principles electron-phonon interactions and electronic transport in large-angle twisted bilayer graphene. *Phys. Rev. Mater.* **8**, L051001 (2024). https://journals.aps.org/prmaterials/abstract/10.1103/PhysRevMaterials.8.L051001.

[23] Desai, D. C., Park, J., Zhou, J.-J. & Bernardi, M. Dominant two-dimensional electron-phonon interactions in the bulk Dirac semimetal Na_3Bi. *Nano Lett.* **23**, 3947–3953 (2023). https://pubs.acs.org/doi/full/10.1021/acs.nanolett.3c00713.

[24] Bernardi, M. Computing electron dynamics in momentum space. *Nat. Comput. Sci* **3**, 480–481 (2023). https://doi.org/10.1038/s43588-023-00473-8.

[25] Caruso, F. & Novko, D. Ultrafast dynamics of electrons and phonons: from the two-temperature model to the time-dependent boltzmann equation. *Adv. Phys. X* **7**, 2095925 (2022). https://www.tandfonline.com/doi/full/10.1080/23746149.2022.2095925.

[26] Bernardi, M., Vigil-Fowler, D., Lischner, J., Neaton, J. B. & Louie, S. G. Ab initio study of hot carriers in the first picosecond after sunlight absorption in silicon. *Phys. Rev. Lett.* **112**, 257402 (2014). https://link.aps.org/doi/10.1103/PhysRevLett.112.257402.

[27] Jhalani, V. A., Zhou, J.-J. & Bernardi, M. Ultrafast hot carrier dynamics in GaN and its impact on the efficiency droop. *Nano Lett.* **17**, 5012–5019 (2017). https://pubs.acs.org/doi/10.1021/acs.nanolett.7b02212.

[28] Sjakste, J., Tanimura, K., Barbarino, G., Perfetti, L. & Vast, N. Hot electron relaxation dynamics in semiconductors: assessing the strength of the electron–phonon coupling from the theoretical and experimental viewpoints. *J. Phys. Condens. Matter* **30**, 353001 (2018). https://iopscience.iop.org/article/10.1088/1361-648X/aad487/meta.

[29] Tong, X. & Bernardi, M. Toward precise simulations of the coupled ultrafast dynamics of electrons and atomic vibrations in materials. *Phys. Rev. Res.* **3**, 023072 (2021). https://journals.aps.org/prresearch/abstract/10.1103/PhysRevResearch.3.023072.

[30] Caruso, F. Nonequilibrium lattice dynamics in monolayer MoS_2. *J. Phys. Chem. Lett.* **12**, 1734–1740 (2021). https://pubs.acs.org/doi/full/10.1021/acs.jpclett.0c03616.

[31] Emeis, C. *et al.* Coherent phonons and quasiparticle renormalization in semimetals from first principles. *Phys. Rev. X* **15**, 021039 (2025). https://journals.aps.org/prx/abstract/10.1103/PhysRevX.15.021039.

[32] Yao, J., Maliyov, I., Gardner, D. J., Woodward, C. S. & Bernardi, M. Advancing simulations of coupled electron and phonon nonequilibrium dynamics using adaptive and multirate time integration. *Npj Comput. Mater.* **11**, 256 (2025). https://www.nature.com/articles/s41524-025-01738-8.

[33] Chen, H.-Y., Sangalli, D. & Bernardi, M. Exciton-phonon interaction and relaxation times from first principles. *Phys. Rev. Lett.* **125**, 107401 (2020). https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.125.107401.

[34] Chen, H.-Y., Sangalli, D. & Bernardi, M. First-principles ultrafast exciton dynamics and time-domain spectroscopies: Dark-exciton mediated valley depolarization in monolayer WSe_2. *Phys. Rev. Res.* **4**, 043203 (2022). https://journals.aps.org/prresearch/abstract/10.1103/PhysRevResearch.4.043203.

[35] Chan, Y.-h. *et al.* Exciton thermalization dynamics in monolayer MoS_2: A first-principles boltzmann equation study. *Phys. Rev. B* **111**, 184305 (2025). https://link.aps.org/doi/10.1103/PhysRevB.111.184305.

[36] Brunton, S. L. & Kutz, J. N. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* (Cambridge University Press, 2022). https://www.cambridge.org/core/books/datadriven-science-and-engineering/77D52B171B60A496EAFE4DB662ADC36E.

[37] Maliyov, I., Yin, J., Yao, J., Yang, C. & Bernardi, M. Dynamic mode decomposition of nonequilibrium electron-phonon dynamics: accelerating the first-principles real-time boltzmann equation. *npj Comput. Mater.* **10**, 123 (2024). https://www.nature.com/articles/s41524-024-01308-4.

[38] Reeves, C. C. *et al.* Dynamic mode decomposition for extrapolating nonequilibrium Green's-function dynamics. *Phys. Rev. B* **107**, 075107 (2023). https://journals.aps.org/prb/abstract/10.1103/PhysRevB.107.075107.

[39] Luo, Y., Desai, D., Chang, B. K., Park, J. & Bernardi, M. Data-driven compression of electron-phonon interactions. *Phys. Rev. X* **14**, 021023 (2024). https://journals.aps.org/prx/abstract/10.1103/PhysRevX.14.021023.

[40] Tancogne-Dejean, N. *et al.* Octopus, a computational framework for exploring light-driven phenomena and quantum dynamics in extended and finite systems. *J. Chem. Phys.* **152** (2020). https://pubs.aip.org/aip/jcp/article/152/12/124119/954926.

[41] Falke, S. M. *et al.* Coherent ultrafast charge transfer in an organic photovoltaic blend. *Science* **344**, 1001–1005 (2014). https://www.science.org/doi/10.1126/science.1249771.

[42] Ward, L. *et al.* Accelerating multiscale electronic stopping power predictions with time-dependent density functional theory and machine learning. *Npj Comput. Mater.* **10**, 214 (2024). https://www.nature.com/articles/s41524-024-01374-8.

[43] Zheng, Z. *et al.* Ab initio real-time quantum dynamics of charge carriers in momentum space. *Nat. Comput. Sci.* **3**, 532–541 (2023). https://www.nature.com/articles/s43588-023-00456-9.

[44] Sangalli, D. & Marini, A. Ultra-fast carriers relaxation in bulk silicon following photo-excitation with a short and polarized laser pulse. *Europhys. Lett.* **110**, 47004 (2015). https://iopscience.iop.org/article/10.1209/0295-5075/110/47004/meta.

[45] Sangalli, D. Excitons and carriers in transient absorption and time-resolved arpes spectroscopy: An ab initio approach. *Phys. Rev. Mater.* **5**, 083803 (2021). https://link.aps.org/doi/10.1103/PhysRevMaterials.5.083803.

[46] Perfetto, E., Pavlyukh, Y. & Stefanucci, G. Real-time GW: Toward an ab initio description of the ultrafast carrier and exciton dynamics in two-dimensional materials. *Phys. Rev. Lett.* **128**, 016801 (2022). https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.128.016801.

[47] Hou, B. *et al.* Data-driven low-rank approximation for the electron-hole kernel and acceleration of time-dependent gw calculations. *Npj Comput. Mater.* **11**, 204 (2025). https://www.nature.com/articles/s41524-025-01680-9.

[48] Zhou, J.-J. *et al.* Perturbo: A software package for ab initio electron–phonon interactions, charge transport and ultrafast dynamics. *Comput. Phys. Commun.* **264**, 107970 (2021). https://www.sciencedirect.com/science/article/pii/S0010465521000837.

[49] Taher, M. Accelerating scientific applications using GPU's. In *2009 4th International Design and Test Workshop (IDT)*, 1–6 (IEEE, 2009). https://ieeexplore.ieee.org/abstract/document/5404114.

[50] Abramov, N. S. & Abramov, S. M. November 2022 Top500 list overview. *Supercomputing Frontiers and Innovations* **10**, 4–17 (2023). https://superfri.org/index.php/superfri/article/view/499.

[51] NERSC. Perlmutter architecture (2025). https://docs.nersc.gov/systems/perlmutter/architecture/.

[52] Storti, D. & Yurtoglu, M. *CUDA for engineers: an introduction to high-performance parallel computing* (Addison-Wesley Professional, 2015). https://dl.acm.org/doi/10.5555/2911064.

[53] Lu, I.-T., Zhou, J.-J. & Bernardi, M. Efficient ab initio calculations of electron-defect scattering and defect-limited carrier mobility. *Phys. Rev. Mater.* **3**, 033804 (2019). https://journals.aps.org/prmaterials/abstract/10.1103/PhysRevMaterials.3.033804.

[54] Le, K. B. *et al.* Magnon-phonon interactions from first principles. *Phys. Rev. B* **112**, L180403 (2025). https://link.aps.org/doi/10.1103/489b-n5pd.

[55] Cepellotti, A., Coulter, J., Johansson, A., Fedorova, N. S. & Kozinsky, B. Phoebe: a high-performance framework for solving phonon and electron Boltzmann transport equations. *J. Phys. Mater.* **5**, 035003 (2022). https://iopscience.iop.org/article/10.1088/2515-7639/ac86f6/meta.

[56] Liu, Z., Zhang, B., Fan, Z. & Li, W. A high-performance GPU implementation of the electron-phonon Wannier interpolation and the related transport properties. *arXiv 2306.16493* (2023).

[57] Perturbopy: a suite of Python scripts for Perturbo testing and postprocessing. Official documentation (2025). https://perturbopy.readthedocs.io/en/latest/index.html.

[58] Giannozzi, P. *et al.* Advanced capabilities for materials modelling with QUANTUM ESPRESSO. *J. Phys. Condens. Matter* **29**, 465901 (2017). https://doi.org/10.1088/1361-648X/aa8f79.

[59] Perdew, J. P. & Wang, Y. Accurate and simple analytic representation of the electron-gas correlation energy. *Phys. Rev. B* **45**, 13244 (1992). https://doi.org/10.1103/PhysRevB.45.13244.

[60] Troullier, N. & Martins, J. L. Efficient pseudopotentials for plane-wave calculations. *Phys. Rev. B* **43**, 1993 (1991). https://doi.org/10.1103/PhysRevB.43.1993.

[61] Mostofi, A. A. *et al.* An updated version of WANNIER90: A tool for obtaining maximally-localised Wannier functions. *Comput. Phys. Commun.* **185**, 2309 (2014). https://doi.org/10.1016/j.cpc.2014.05.003.