Accelerating the Density-Functional Tight-Binding Method Using Graphical Processing Units

Van-Quan Vuong ⊥,[1] Caterina Cevallos ⊥,[2] Ben Hourahine,[3] Bálint Aradi,[4] Jacek Jakowski,[5, a)] Stephan Irle,[5, b)] and Cristopher Camacho[2, c)]

[1] *Bredesen Center for Interdisciplinary Research and Graduate Education, University of Tennessee, Knoxville, TN, United States*

[2] *School of Chemistry, University of Costa Rica, San José, 11501-2060, Costa Rica*

[3] *SUPA, Department of Physics, The John Anderson building, 107 Rottenrow East, Glasgow G4 0NG, United Kingdom*

[4] *Bremen Center for Computational Materials Science, Universität Bremen, Bremen, Germany*

[5] *Computational Sciences & Engineering Division, Oak Ridge National Laboratory, Oak Ridge, TN, United States*

(Dated: 30 January 2023)

Acceleration of the density-functional tight-binding (DFTB) method on single and multiple graphical processing units (GPUs) was accomplished using the MAGMA linear algebra library. Two major computational bottlenecks of DFTB ground-state calculations were addressed in our implementation: the Hamiltonian matrix diagonalization and the density matrix construction. The code was implemented and benchmarked on two different computer systems: (1) the SUMMIT IBM Power9 supercomputer at the Oak Ridge National Laboratory Leadership Computing Facility (OLCF) with 1 to 6 NVIDIA Volta V100 GPUs per computer node, and (2) an in-house Intel Xeon computer with 1 to 2 NVIDIA Tesla P100 GPUs. The performance and parallel scalability were measured for three molecular models of 1-, 2- and 3-dimensional chemical systems, represented by carbon nanotubes, covalent organic frameworks, and water clusters.

⊥ These authors contributed equally to this work

[a)] corresponding author, e-mail: jjakowski@ornl.gov

[b)] corresponding author, e-mail: irles@ornl.gov

[c)] corresponding author, e-mail: cristopher.camacho@ucr.ac.cr

I. INTRODUCTION

Computational simulation of chemical systems and materials has enriched our understanding of matter from the atomic to the microscopic scales[1–3]. For the simulation of complex chemical processes, quantum mechanics (QM)-based methods like *ab initio* wave function theory (WFT) or Density Functional Theory (DFT) are currently the most commonly used methods because of their capability to explicitly describe electronic structures, chemical bond breaking and making, including electron transfer occurring on electrodes[4,5] among other chemical reactions on the surface of heterogeneous catalysts[6,7] or in biosystems[8–10]. However, the applications of these methods are typically limited to systems containing a few hundred atoms focusing on static minimum energy reaction mechanism pathways or only short-time molecular dynamics (MD) simulations, due to the tremendous computational cost required to compute large numbers of integrals over electronic degrees of freedom[11,12]. Even though DFT calculations comprising of more than 10^6 atoms are possible, these must be considered heroic calculations and are not feasible for their routinely use in science[13,14].

To overcome the challenge in computational cost, semi-empirical WFT-based (MNDO, AM1, PMx, OMx, etc.) and approximate DFT-based methods (DFTB, xTB) have long been developed[1,15–18] and have not lost their relevance[3,19]. Among them, the density-functional tight-binding (DFTB) method is particularly attractive due to its hierarchical way of approaching the DFT level of theory at a considerably lower computational cost, greatly reducing the time-to-solution ratio by several orders of magnitude[16,20–23]. A key feature of DFTB originates from is the construction of the Slater-Koster tight-binding approximation[24] to the Kohn-Sham Hamiltonian[25,26] based on a two-center approximation which enables the use of tabulated, distance-dependent electronic integrals and repulsive potentials. Explicit computation of the highly resource-demanding integrals[27,28] can thereby be avoided and the construction of the corresponding Hamiltonian matrices is very efficient and scales quadratically with the number of atoms[29,30]. For example, on a single central processing unit (CPU) of an ordinary laptop computer, the DFTB+ code allows for geometry optimizations of molecular and bulk solid systems containing thousands of atoms within a few hours, and can perform MD simulations for systems containing hundreds of atoms on the nanosecond time-scale with a similar time-to-solution[20,31].

Unfortunately, the wall-clock time of DFTB calculations increases cubically with the size of the simulated systems, which imposes a heavy toll on large-scale applications (systems containing more than several thousand atoms). The cubic scaling of DFTB is due to BLAS3 type dense matrix-matrix operations and is primarily driven by its most time-consuming step, namely a generalized eigenvalue problem used for solving the time-independent Schrodinger equation[32–35]. Several reduced-scaling DFTB approaches have been suggested to address this problem, such as the divide-and-conquer (DC)-DFTB method[33,36,37], modified DC (mDC)-DFTB[38], the fragment molecular orbital (FMO)-DFTB method[32,39–42], a graph-based, sparse linear algebra approach to formulating the eigenvalue equation[34], and various methods for Fermi-function expansion via matrix polynomials or expansion of it's poles[3,43]. In addition, for quasi-2D and 3D bulk systems, $\mathcal{O}(N^{3/2})$ and $\mathcal{O}(N^2)$ scaling can be achieved, respectively, for large systems by means of the PEXSI method[44,45]. While these approaches allow for DFTB-based simulations of systems containing up to 100 million atoms by taking advantage of conventional homogeneous parallel computational architectures[46], the procedures cannot be easily applied to chemically reactive systems or metallic systems[32]. However, general purpose methods with no worse than quadratic scaling *can* be used in these cases[43]. Moreover, several of the techniques mentioned above sacrifice the accuracy of the energy and the gradient in an uncontrollable way[33]. The accuracy of the method can be particularly affected in MD simulations due to error accumulation, where a small error would lead to qualitatively different MD trajectories.

While DFTB calculations can in principle be accelerated in homogeneous parallel computer architectures using open multi-processing (OpenMP) or message passing interface (MPI) techniques, the resulting speedup factor is usually low for multiple-node calculations due to the limited parallel scalability of matrix diagonalization and slow speed of inter-node communication. Alternatively, the calculations can be accelerated by taking advantage of modern computer graphics processing units (GPUs) in a similar fashion as other QM-based methods[47–55]. In addition, modern supercomputer architectures are increasingly employing heterogeneous CPU architectures[56]. In the past, several attempts to accelerate the DFTB calculations on heterogeneous CPU+GPU computational architectures by porting only the Hamiltonian matrix diagonalization onto the GPUs and leaving the remainder of the DFTB calculations on the CPU have been reported[57–59]. The latest implementation by Allec et al. showed promising results, for instance, a speedup factor of ∼6x (when 4 NVIDIA P100 GPUs and 24 threads of an Intel Xeon E5-2680v3 CPU was compared to 24 threads of the same CPU) on the diagonalization of the Hamiltonian matrix for a water cluster containing 16,000 atoms was reported[59]. However, only a modest ∼1.5x speedup factor was achieved for a single-point energy calculation of the same water cluster under the same conditions. Furthermore, no parallel scalability with the number of GPUs has been reported. In this work, we report a new implementation in which the two most time-consuming steps for ground state calculations, (1) diagonalization of the Hamiltonian matrix and (2) the construction of density matrix, are ported to the GPU (being the only cubic scaling parts of the calculation). As in the ELPA2 library[55], the MAGMA GPU-accelerated library[60] is employed for the linear algebra manipulations on GPUs. The general performance and parallel scalability of the new implementation was benchmarked with various numbers of GPUs employed in parallel for a wide range of system dimensionalities (linear 1D, planar 2D, and cluster-type 3D) and system sizes. The benchmark was performed on two different CPU+GPU architectures, namely one on the Summit supercomputer at Oak Ridge National Laboratory featuring IBM Power9 CPU architectures and up to six NVIDIA Volta GPUs per node, and another, workstation-type computer using ordinary Intel Xeon CPUs with up to 2 NVIDIA Tesla GPUs. Finally, we analyzed and report in detail the benefits of porting the construction of the density matrix onto the GPU.

II. METHODOLOGY

A. Density-Functional Tight-Binding (DFTB)

The electron density of a system can be approximated using a Taylor series around a reference density[61]. Truncating the expansion at the first, second, or third orders gives rise to a hierarchical family of DFTB “flavors”: beginning from the simplest formulation, the non-self-consistent charge (non-SCC)-DFTB (or DFTB1)[29,30]; secondly, the self-consistent charge (SCC)-DFTB (or DFTB2)[62,63], and finishing with the most involved third-order DFTB3 flavor[16,64,65]. The DFTB methods have already been the subjects of many comprehensive reviews[16,22,63,66–69]; hence, they will not be discussed in detail here. In this section, as a reminder, a brief review of the SCC-DFTB (DFTB2) method[62] is given. In this method, the total energy is defined as

$$E = \sum_{i}^{occ.} n_i \langle \Psi_i | \hat{H}^0 | \Psi_i \rangle + \frac{1}{2} \sum_{AB}^{atoms} \gamma_{AB} \Delta q_A \Delta q_B + \sum_{A>B}^{atoms} E_{AB}^{\mathrm{rep}} \quad , \tag{1}$$

where $\hat{H}^0$ is the initial Hamiltonian constructed from the superposition of neutral atomic densities in a two-center approximation[29,30]; $|\Psi_i\rangle$ are molecular orbitals (MOs); Δq_A are the Mulliken charges[70], and $\gamma_{AB}\Delta q_A \Delta q_B$ represents the electrostatic interaction energy between the two Mulliken charges on atom A and atom B[62]. The molecular orbitals $|\Psi_i\rangle$ are expanded as a linear combination of atomic orbitals $|\phi_\mu\rangle$,

$$|\Psi_i\rangle = \sum_\mu c_{i\mu} |\phi_\mu\rangle . \tag{2}$$

As with other QM-based methods, the molecular orbitals $|\Psi_i\rangle$ as well as the total energy are determined by applying the variational principle. By recasting the problem of solving the coefficient $c_{i\mu}$ into a secular matrix equation, the task becomes solving the generalized eigenvalue problem

$$\sum_\nu H_{\mu\nu} c_{i\nu} = \epsilon_i \sum_\nu S_{\mu\nu} c_{i\nu}, \tag{3}$$

where H and S are the Hamiltonian and overlap matrices, respectively. The SCC-DFTB Hamiltonian is then given by

$$H_{\mu\nu} = \langle\phi_\mu| \hat{H}^0 |\phi_\nu\rangle + \frac{1}{2} S_{\mu\nu} \sum_C (\gamma_{AC} + \gamma_{BC}) \Delta q_C, \quad \text{with} \quad \mu \in A, \nu \in B. \tag{4}$$

In the framework of the two-center approximation, the initial Hamiltonian integrals $\langle\phi_\mu| \hat{H}^0 |\phi_\nu\rangle$ and the overlap integrals $\langle\phi_\mu|\phi_\nu\rangle$ are pre-tabulated for each atomic pair. The Mulliken charge on atom A is defined as

$$\Delta q_A = \sum_i^{occ} n_i \sum_{\mu\in A} \sum_\nu c_{i\mu} c_{i\nu} S_{\mu\nu} - q_A^0 = \sum_{\mu\in A} \sum_\nu P_{\mu\nu} S_{\mu\nu} - q_A^0, \tag{5}$$

where $P_{\mu\nu} = \sum_i^{occ} n_i c_{i\mu} c_{i\nu}$ are the single particle density matrix elements. Since the gross atomic charges, Δq_A, depend on the molecular orbitals $|\Psi_i\rangle$, the eigenvalue problem must be solved self-consistently. This entails solving equation 3 iteratively until convergence of the eigenvalues is reached, as depicted in Figure 1. Due to the pre-tabulated integrals in the formalism of DFTB, the most time-consuming step in traditional DFT is avoided. In DFTB calculations for systems containing more than tens of atoms, the performance-critical routines correspond to the diagonalization of the Hamiltonian matrix, taking on the order of 90−95% of the total running time, and the construction of the density matrix accounts approximately for 5 − 10%.

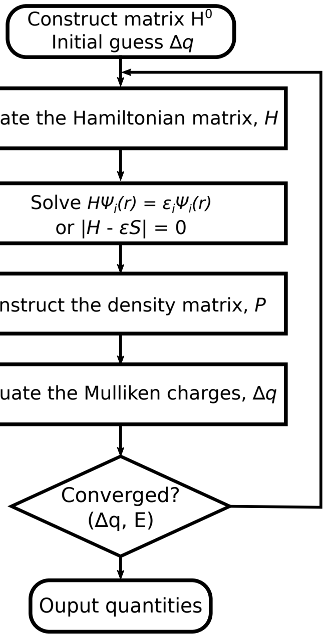

Construct matrix H^0
Initial guess Δq
Update the Hamiltonian matrix, H
Solve $H\Psi_i(r) = \varepsilon_i\Psi_i(r)$
or $|H - \varepsilon S| = 0$
Construct the density matrix, P
Evaluate the Mulliken charges, Δq
Converged?
(Δq, E)
Ouput quantities

FIG. 1. Workflow illustrates the self-consistent-charge (SCC) procedure in the DFTB method.

B. Implementation

In this work, we utilize the MAGMA GPU-based eigensolver to diagonalize the Hamiltonian matrix and replaced the BLAS routines required to construct the density matrix with the corresponding GPU-based routines. In our implementation, the number of GPUs of the "magma" solver can be specified by the user via the environment variable "$MAGMA\_NUM\_GPUS$". The code detects all available GPUs and ensure that the requested number does not exceed the number of physically available units. Otherwise, only one GPU is used by default. Furthermore, a Fortran wrapper, namely *device_ info*, based exclusively on MAGMA Device Management routines, was created to guarantee future portability with GPUs manufactured by companies other than NVIDIA. Such an interface facilitates the acquisition of information regarding the available devices by enabling seamless communication between the GPUs and the DFTB+ program. Subsequently, the MAGMA routines are called from the *eigensolver* module; an appropriate subroutine is selected from the *gpu_gvd* interface, depending on the type of chemical system, periodic or molecular.

The second most time-consuming routine in DFTB calculations is the construction of the density matrix, P, from the matrix of eigenvectors, C. The matrix construction is carried out after obtaining the eigenvectors and the occupation numbers n_i, as explained in Figure 1. For dense matrices, the computational cost typically scales as $O(N^3)$, but it can be reduced to $O(N^2)$ by exploiting the sparsity of P[20]. The building of this matrix is done by calling the BLAS level-3 routines performing the Hermitian rank-k ($HERK$) operations, $P \leftarrow \alpha CC^H + \beta P$. We replaced these CPU-based $HERK$ routines with their single-GPU-based equivalents, as provided by the MAGMA library. A patch for the 19.1 DFTB+ release[71] can be found at `https://doi.org/10.5281/zenodo.7566762`. The transfer of the eigenvectors, density matrix between the host and devices, and the memory allocation are carefully managed by handling the GPU pointers directly.

C. Computational Details

As mentioned above, the performance of our implementations was examined on two different computer architectures: (1) the SUMMIT supercomputer and (2) an in-house workstation-type Intel-based Linux computer which we call KOFUN. On the SUMMIT computer, two "IBM Power9 CPUs" at 4.00GHz in conjunction with 6 "NVIDIA Volta V100" GPUs were used. Each GPU is paired with 16 GB of High Bandwidth Memory (HBM) with a bandwidth of 900 GB/s. Bidirectional GPUs-HBMs interconnect at 50 GB/s. Communication between CPUs and GPUs-HBMs units rates at 50 GB/s. CPU to RAM possesses a bandwidth of 170 GB/s. Each "NVIDIA Volta V100" GPU has a peak performance of 7.8 teraFLOPS. The two CPUs in each node interconnect at a bandwidth of 64 GB/s. On the KOFUN computer, two "Intel Xeon CPU E5-2630 v4" at 2.20GHz paired with 2 "NVIDIA Tesla P100" GPUs were used. Each Xeon CPU is capable of 704 GFLOPS. The system possesses a max memory bandwidth of 25.6 GB/s. GPUs interconnect with a bandwidth of 16 GB/s. Each "NVIDIA Tesla P100" GPU has a peak performance of 4.7 teraFLOPS. Firstly, computational performance and parallel scalability of only the Hamiltonian matrix diagonalization ported onto the GPUs were benchmarked on SUMMIT for various carbon nanotubes (CNTs, 1D), covalent organic frameworks (COFs, 2D), and water clusters (3D). For the test of parallel scalability, the number of GPUs varied from 1 to 6. Secondly, an equivalent test was carried out on KOFUN with the constraint to using only up to 2 GPUs. Finally, the time savings of building the density matrix on the GPU was investigated on both SUMMIT and KOFUN computers. For all tests, the wall-clock time of SCC-DFTB single-point energy calculations was used to evaluate the computational cost. The wall-clock time was measured using the Unix "time" command. In these benchmarks, we only discuss the single-point energy calculation because the force calculation often takes on the order of only 1-3% of the total running time, even when the Hamiltonian matrix diagonalization is carried out on the GPUs, as shown in Table

S1 in the Supplementary Material. All DFTB single-point energy calculations were performed with the number of SCC iteration cycles set equal to 10 to ensure a consistent comparison, except in comparison with energy+gradient calculations where the default options were used. The constant number of SCC iteration cycles was achieved with the following options: "SCCTolerance = 1e-12" and "MaxSCCIterations = 10". DFTB calculations were performed using a development version of the DFTB+ program[20,72] in combination with the MAGMA (2.5.3) library[60]. The math kernel library (MKL) was used to maximize the performance of its "Intel CPU," similarly to the engineering scientific subroutine library (ESSL) used in the SUMMIT computer.

III. RESULTS AND DISCUSSION

A. Effect of the Hamiltonian matrix diagonalization on the GPUs: performance and parallel scalability

To assess the effects, we measured the wall-clock time of DFTB single-point energy calculations using various computer configurations (with and without GPUs) for a set of water clusters, varying from 3 to 11,944 water molecules. The calculations with CPU+GPUs were carried out using 21 or 42 Power9 CPU threads combined with 1, 2, 3, or 6 V100 GPUs; the calculations with CPU-only were carried out using the corresponding 21 or 42 CPU threads. We emphasize that, in this test, only the Hamiltonian matrix diagonalization utilized the GPUs in the calculations with CPU+GPUs.

Figure 2 shows how the wall-clock time increases with the system size, and Table S2 shows actual values of the wall-clock time. In the case of the CPU-only calculations, the time increases drastically for systems having more than 10,000 basis functions. It is important to note that the wall-clock time increases with the system size cubically in all cases, with and without GPUs. However, the rate of change (slope) is much lower in the cases of GPU-accelerated calculations compared to CPU-only calculations.

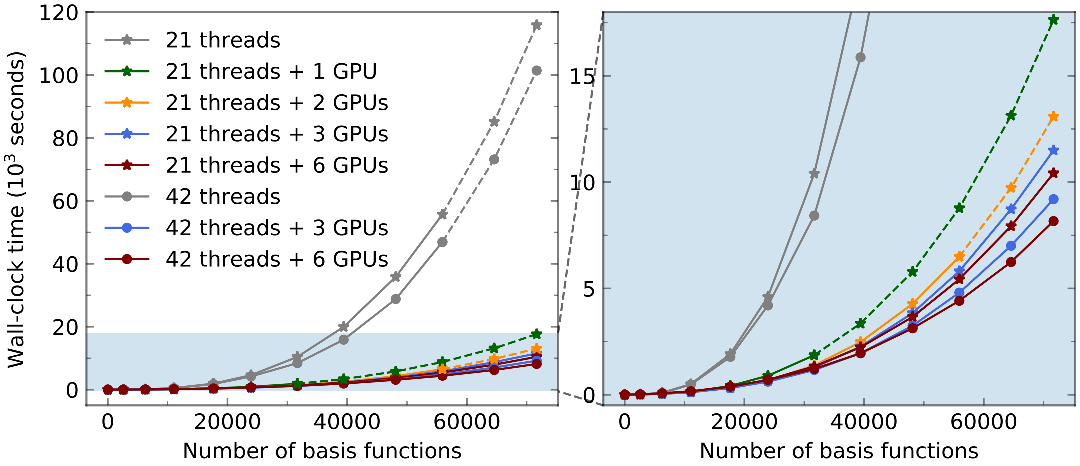


FIG. 2. Comparison of the wall-clock time of DFTB single-point energy calculation for water clusters as a function of the number of basis functions. For the heterogeneous CPU+GPU architecture, only the Hamiltonian diagonalization was performed on GPUs. All calculations were carried out on the SUMMIT supercomputer with IBM Power9 CPU + NVIDIA Volta GPUs. The dotted lines represent extrapolated data. Detail of the blue shaded region is shown on the right. A log-log version of the left panel is provided in Figure S1 in the Supplementary Material.

While doubling the number of CPU threads from 21 to 42 only sped up the calculation in a minimal way, using GPUs to diagonalize the Hamiltonian matrix significantly sped up the calculations in general. When the GPUs were employed, the wall-clock time was greatly reduced for moderate- and large-size systems with more than 10,000 basis functions. For instance, while the CPU-only calculation took 10,403 seconds for the $(H_2O)_{5280}$ cluster, the calculation took only 1,285 seconds with the aid of 3 GPUs. Nevertheless, the acceleration is less substantial for smaller systems: The wall-clock time only reduces from 97 seconds to 43 seconds when the 3 GPUs were used for the $(H_2O)_{1046}$ cluster. The calculation can be even slower with GPUs for the smallest system, the $(H_2O)_3$ cluster. The optimization of the algorithms and kernels implemented in the MAGMA library for different matrix sizes has been the subject of several studies, and we refer the reader to Refs.[73,74]

For a more comprehensive benchmark, we extended the test to include three representative types of systems: (1) carbon nanotubes (CNTs) for 1D materials, (2) covalent organic frameworks (COFs) for 2D materials, and (3) water clusters for 3D materials. The wall-clock time is listed in Table S2 for water clusters, Table S4 and Table S3 in the Supplementary Material for CNTs and COFs, respectively. Figure 3 compares the speedup factor as a function of the number of GPUs and the number of basis functions for these systems.

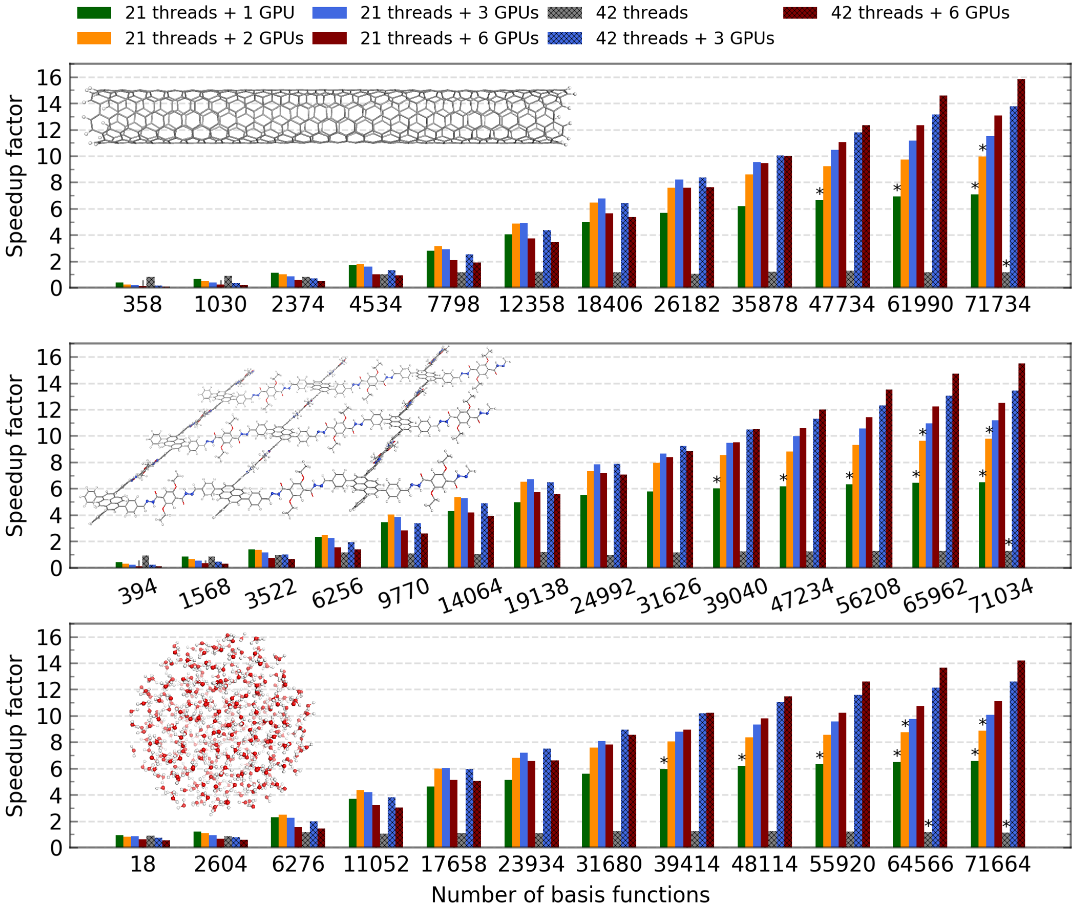


FIG. 3. Comparison of the speedup as a function of the number of basis functions gained by different computer configurations on DFTB single-point energy calculation for carbon nanotube (CNT), covalent organic frameworks (COF), and water clusters. For each system, the speedup is referenced to the running time of the corresponding calculation on the homogeneous computer with 21 CPU threads. For the heterogeneous CPU+GPU architecture, only the Hamiltonian diagonalization was performed on GPUs. The asterisk symbols label extrapolated data.

Generally, the advantages of using GPUs, with a speedup factor larger than one, were observed for systems having more than 4,000 basis functions. The parallel scalability increases with the system size and only shows a good scaling factor with the number of GPUs for large systems. For systems with less than 36,000 basis functions, using 6 GPUs slows down the calculation, as data must be transferred through two different sockets in the computer. We note that in the SUMMIT supercomputer all three GPUs in a single socket are interconnected via NVLink with a bi-directional bandwidth of 50 GB/s, while data transferred between sockets and thus between GPUs in different sockets share a single bandwidth of 64 GB/s. Although using 42 threads is marginally faster than 21 threads, the combination of 42 threads + 6 GPUs is significantly faster than 21 threads + 6 GPUs. Interestingly, for a similar size, the larger speedup factor was observed with 2D systems (COFs) compared to the 3D materials (water clusters), and the largest speedup factor was observed with 1D materials (CNTs). The trend might be attributed to the sparsity of the Hamiltonian matrix of these systems.

B. Xeon CPU + P100 GPU versus Power CPU + V100 GPU

To further evaluate the effects of different hardware configurations on the performance of our implementation, we carried out the same test for CNTs, COFs, and water clusters on our in-house KOFUN computer. The wall-clock time of DFTB single-point energy calculations for CNTs, COFs, and water clusters is listed in Table I.

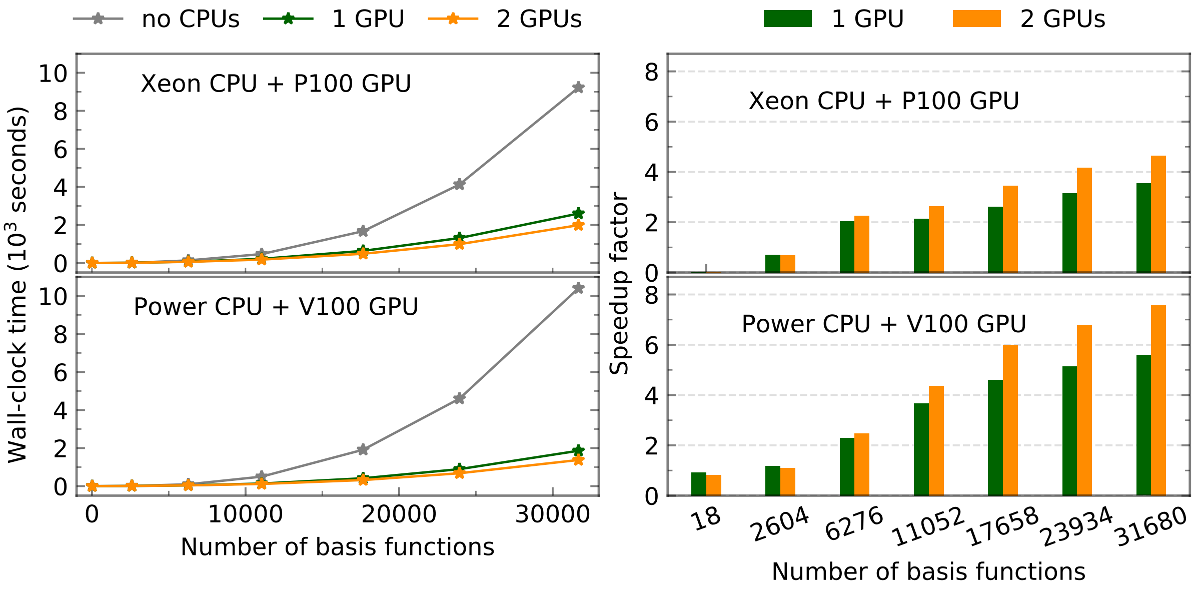


FIG. 4. Comparison of the wall-clock time and speedup factor of DFTB single-point energy calculation for water clusters as a function of the number of basis functions on Xeon CPU + P100 GPU versus Power CPU + V100 GPU. For each system, the speedup is referenced to the running time of the corresponding CPU-only calculation with: “20 Xeon CPU threads” in the case of “Xeon CPU + P100 GPU” and “21 Power CPU threads” in the case of “Power CPU + V100 GPU”. For the heterogeneous CPU+GPU architecture, only the Hamiltonian diagonalization was performed on GPUs. A log-log version of the left panel is provided in Figure S2 in the Supplementary Material.

Figure 4 compares the wall-clock time and the speedup factor of DFTB single-point energy calculations for water clusters on SUMMIT and KOFUN computers. For water clusters, the calculation is faster on KOFUN than on SUMMIT when they are carried out using CPU-only, suggesting that the Intel Xeon CPU/MKL combination is faster than the IBM Power CPU/ESSL for this specific case of study. Nevertheless, the V100

GPU is faster than the P100 GPU, making the GPU-accelerated DFTB calculations faster on SUMMIT than on KOFUN. As a result, remarkably more significant speedup factors were observed on SUMMIT. For example, for water clusters of $(H_2O)_{5280}$ computed using 2 GPUs, the speedup factor of 7.6x and 4.6x was observed on SUMMIT and KOFUN, respectively.

We noticed that in the case of CNTs, the homogeneous computation is drastically slower on KOFUN than on SUMMIT, as shown in Table I. As a result, an unreasonable high speedup factor of 32x was observed. The issue can most likely be attributed to the implementation of the MKL library. Similar behavior was observed for COFs but in a less severe manner.

TABLE I. The wall-clock time in seconds of DFTB single-point energy calculations, each with 10 self-consistent cycles, for CNTs, COF, and water clusters on SUMMIT (using 21 Power-CPU threads + V100 GPUs) and KOFUN (using 20 Xeon-CPU threads + P100 GPUs) computers. For the heterogeneous CPU+GPU architecture, only the Hamiltonian diagonalization was performed on GPUs.

Natoms	Nbasis	Power CPU + V100 GPUs			Xeon CPU + P100 GPUs		
		no GPU	1 GPU	2 GPUs	no GPU	1 GPU	2 GPUs
				CNTs			
106	358	1.0	2.6	3.9	0.4	3.5	4.1
274	1030	2.4	3.6	4.8	1.4	4.6	5.4
610	2374	9.1	8.0	9.0	7.4	10.6	11.0
1150	4534	40.2	23.2	22.4	48.8	32.7	29.8
1966	7798	176.4	62.6	56.0	309.9	101.0	86.7
3106	12358	678.7	166.8	139.0	1554.2	269.4	215.7
4618	18406	2132.6	426.6	329.9	6722.5	666.3	505.8
6562	26182	5900.6	1036.1	776.3	30901.0	1593.0	1166.9
8986	35878	14820.7	2387.0	1723.8	83040.5	3694.2	2545.4
				COFs			
142	394	1.1	2.7	3.9	0.5	6.2	6.9
560	1568	4.0	4.9	6.1	3.4	6.5	7.3
1254	3522	21.2	15.2	15.9	18.3	20.5	18.6
2224	6256	92.2	40.0	37.4	136.0	63.4	57.2
3470	9770	341.0	99.1	84.9	339.7	162.6	136.5
4992	14064	965.4	223.5	181.0	1000.6	354.7	282.0
6790	19138	2363.9	477.4	363.4	2842.3	739.1	552.3
8864	24992	5075.4	920.3	691.3	6865.0	1408.9	1039.7
11214	31626	10140.7	1750.3	1275.5	14796.6	2516.0	1868.5
				Water clusters			
9	18	0.8	0.9	1.0	0.1	2.3	2.3
1302	2604	11.3	9.5	10.2	8.8	12.4	12.8
3138	6276	96.9	42.2	39.0	135.6	66.6	60.0
5526	11052	498.9	135.7	114.3	469.1	218.3	178.2
8829	17658	1914.2	414.8	319.4	1672.7	638.2	485.5
11967	23934	4601.1	893.0	677.9	4128.3	1312.1	990.1
15840	31680	10403.2	1860.6	1374.3	9223.7	2599.8	1988.8

C. Effect of building the density matrix on GPU

For DFTB calculations using CPU-only, the Hamiltonian matrix diagonalization is the bottleneck, taking $90-95\%$ of the total wall-clock time, and the density matrix construction generally takes in the order of $5-10\%$ of the total wall-clock time. When the GPUs are used to accelerate the diagonalization by a factor of $10-15$x, the density matrix construction becomes a relatively more time-consuming step. Therefore, it is natural to question whether porting the density matrix construction to the GPU is worthwhile. This section assesses the effects of constructing the density matrix on the GPU. The differences in wall-clock time of DFTB single-point energy calculations with and without the density matrix constructed on the GPU are compared for water clusters; the wall-clock time values are listed in Table II. The corresponding wall-clock time values for COFs and CNTs are listed in Tables S5-S6 and Tables S7-S8 in the Supplementary Material, respectively.

Figure 5 shows the comparison of running times and speedup factors obtained on the SUMMIT computer. Similar to the case of porting only the Hamiltonian matrix diagonalization to the GPUs, the effects of porting the density matrix construction to the GPU increase with the system size: The larger the system, the higher the speedup factor is observed. While the trend is similar, having both the Hamiltonian matrix diagonalization and the density matrix construction on the GPUs speeds up the calculation more significantly. For instance, in the case of $(H_2O)_{5280}$ with 3 GPUs, the speedup factor increases from 8x to 13x. Figure 5 shows that including the density matrix construction on a single GPU can make the calculations as fast as having only the Hamiltonian matrix diagonalization on 2 or 3 GPUs.

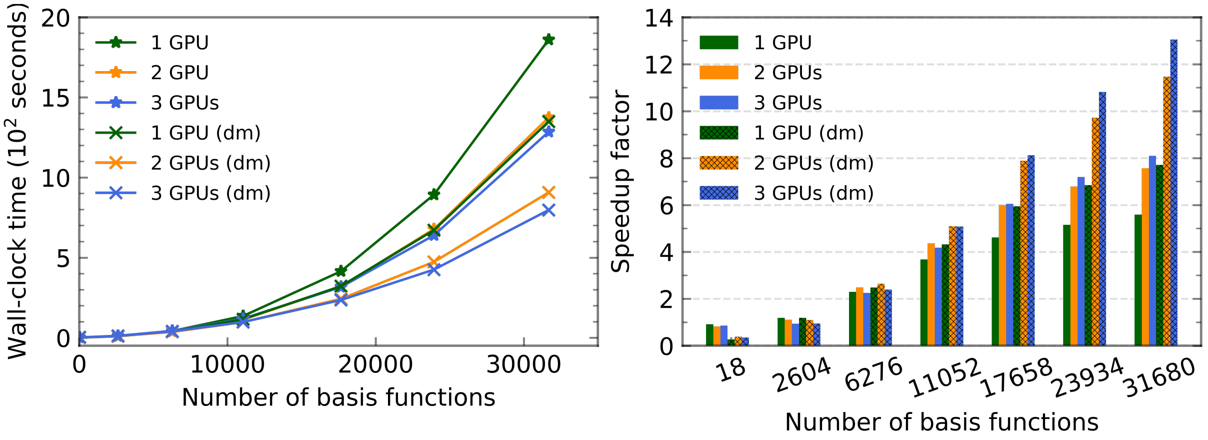


FIG. 5. Comparison of wall-clock time and the speedup as a function of the number of basis functions of DFTB single-point energy calculation for water clusters. For each system, the speedup is referenced to the running time of the corresponding calculation on the homogeneous computer with 21 CPU threads. ‘dm’ denotes the calculations, where both the Hamiltonian matrix diagonalization and the density matrix construction were carried out on GPUs. A log-log version of the left panel is provided in Figure S3 in the Supplementary Material.

Analogous to the results observed on the SUMMIT supercomputer, a similar effect was measured on KOFUN by performing the density matrix construction on a GPU, shown in Table II. Nonetheless, the overall speedup is less pronounced as the P100 GPU on KOFUN is less powerful than the V100 GPU on SUMMIT. For example, compared to the GPU-accelerated only-diagonalization implementation in the case of $(H_2O)_{5280}$ with 2 GPUs, having the density matrix construction on GPU speeds up the calculations by 1.51x on SUMMIT, and the speedup factor is only 1.26x on KOFUN.

TABLE II. Wall-clock time in second of DFTB single-point energy calculations for water clusters on SUMMIT (using 21 Power-CPU threads + V100 GPUs) and KOFUN (using 20 Xeon-CPU threads + P100 GPUs) computers without/with the density matrix constructed on a GPU.

Natoms	Nbasis	no GPU	1 GPU	2 GPUs	3 GPUs
		Power CPU + V100 GPUs			
9	18	0.8	0.9 / 2.9	1.0 / 2.1	0.9 / 2.3
1302	2604	11.3	9.5 / 9.5	10.2 / 10.4	12.1 / 12.0
3138	6276	96.9	42.2 / 39.1	39.0 / 36.7	43.1 / 40.5
5526	11052	498.9	135.7 / 115.5	114.3 / 97.9	119.3 / 98.2
8829	17658	1914.2	414.8 / 322.1	319.4 / 242.8	316.6 / 235.7
11967	23934	4601.1	893.0 / 671.9	677.9 / 473.4	639.5 / 425.3
15840	31680	10403.2	1860.6 / 1349.0	1374.3 / 907.3	1285.4 / 797.2
		Xeon CPU + P100 GPUs			
9	18	0.1	2.3 / 2.9	2.3 / 3.0	
1302	2604	8.8	12.4 / 11.7	12.8 / 13.1	
3138	6276	135.6	66.6 / 61.9	60.0 / 54.9	
5526	11052	469.1	218.3 / 196.8	178.2 / 158.0	
8829	17658	1672.7	638.2 / 549.7	485.5 / 408.7	
11967	23934	4128.3	1312.1 / 1121.5	990.1 / 804.8	
15840	31680	9223.7	2599.8 / 2219.5	1988.8 / 1583.5	

To further analyze the effects of the Hamiltonian matrix H diagonalization and the density matrix P construction on the GPUs, we decomposed the total running time of DFTB single-point energy calculation for $(H_2O)_{5280}$ into three components: (1) time for the matrix H diagonalization, (2) time for the matrix P construction, and (3) the rest. The percentage of these components with respect to the total wall-clock time was examined. Figure 6 shows how the percentage of these components varies with the number of GPUs on SUMMIT. When only the Hamiltonian matrix diagonalization is ported to the GPUs, the matrix H diagonalization percentage reduces drastically from 95.5% with CPU-only to 56.3% with CPU+3GPUs. On the other hand, the matrix P construction percentage increases from 5.4% with CPU-only to 43.1% with CPU+3GPUs. However, having the Hamiltonian matrix diagonalization as well as the density matrix construction on the GPUs shows a modest change in these percentages, varying from 94/5% to 91.1% in the case of the matrix H diagonalization and from 5.4% to 7.9% in the case of the matrix P construction, with CPU-only and CPU+3GPUs respectively. Thus, besides the Hamiltonian matrix diagonalization, porting the density matrix construction to the GPU is worth doing and necessary.

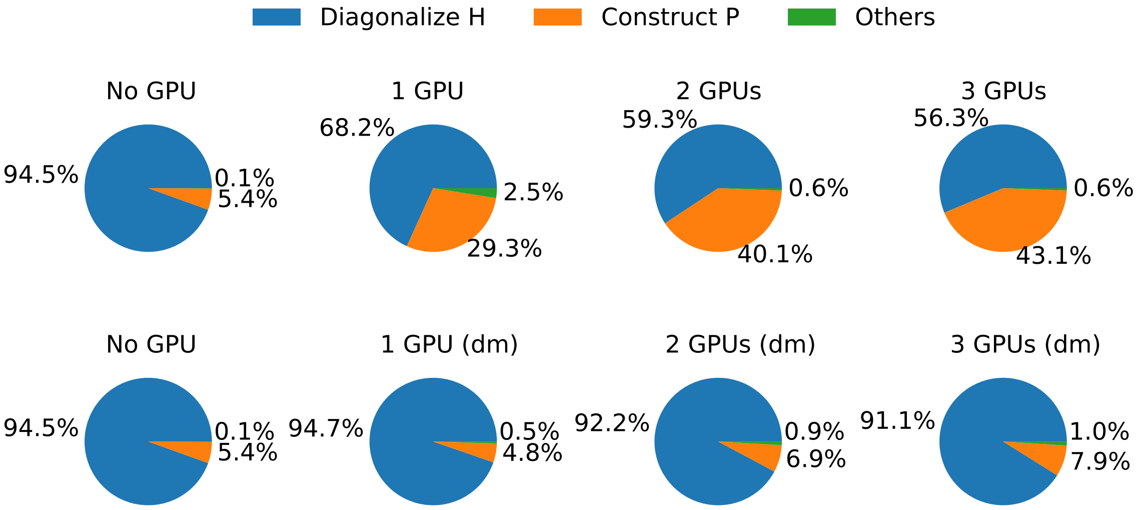


FIG. 6. Percentage of the wall-clock time of DFTB single-point energy calculation for water cluster $(H_2O)_{5280}$ decomposed to the Hamiltonian matrix H diagonalization, the density matrix P construction, and other contributions.

IV. SUMMARY AND OUTLOOK

We presented the acceleration of the density-functional tight-binding (DFTB) method on single or multiple graphical processing units (GPUs) using the MAGMA linear algebra library. In the new implementation, both the diagonalization of the Hamiltonian matrix and the construction of the density matrix were ported to the GPUs. The implementation was carefully benchmarked on two different computer systems with various numbers of GPUs for three molecular models of 1-, 2- and 3-dimensional chemical systems.

The GPUs can speed up DFTB calculations remarkably up to 15x for large-size systems having more than 70,000 basis functions (when 6 NVIDIA V100 GPUs and 42 threads of IBM Power9 CPUs was compared to 42 threads of the same CPUs on the SUMMIT supercomputer) but are less notable for the small ones. The difference in speedup for systems with different dimensionalities is rather small, as it varies only between 14x and 15x.

Likewise, the parallel scalability with number of GPUs increases with system size as well. Our benchmark results suggest that employing only 1 to 2 GPUs can be an efficient option for routine applications, as a trade-off between available computational resources and largest-possible acceleration for small- and moderate-size systems. On the other hand, using multiple GPUs on a supercomputer like SUMMIT can be worthwhile for simulations of large-size systems when the Hamiltonian matrix exceeds the size of the available memory.

Besides the Hamiltonian matrix diagonalization, constructing the density matrix on a GPU makes utilizing GPUs more effective, especially when the GPU speedup for diagonalization is significant. It can make DFTB calculations using one GPU as fast as when using two GPUs where only matrix diagonalization is performed on the GPUs.

The current implementation offers the possibility of efficiently accelerating the time–to–solution for systems in which the calculations can be parallelized as individually independent calculations, such as MD ensembles and anharmonic vibrational frequency calculations along normal modes. These type of calculations can be spanned over several nodes in a cluster, as most commonly available in computational chemistry groups.

A limit of our current implementation is the lack of control in memory allocation provided by MAGMA‘s diagonalization routines. If eigenvectors obtained from the Hamiltonian matrix diagonalization procedure could be left on the GPUs to construct the density matrix, we could further improve the performance and endow the software with superior speedup.

ACKNOWLEDGMENTS

We appreciate valuable comments and suggestions from Jack Wells. We gratefully acknowledge support for this work on OLCF Summit through the Directors Discretionary Allocation CHM152 (2018-2020). Cristopher Camacho and Caterina Cevallos acknowledge financial support from the Rectory and the Vice-rectory for research of the University of Costa Rica through grant 115-B9-461. Van-Quan Vuong acknowledges support by an Energy Science and Engineering Fellowship of the Bredesen Center for Interdisciplinary Research and Graduate Education at the University of Tennessee, Knoxville. S.I. acknowledges support from the Fluid Interface Reactions, Structures, and Transport (FIRST) Center, an Energy Frontier Research Center funded by the U.S. Department of Energy (DOE), Office of Science, Office of Basic Energy Sciences. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

SUPPLEMENTARY MATERIAL

See the Supplementary Material for the wall-clock time of DFTB calculations for carbon nanotubes (CNTs), covalent organic frameworks (COFs), and water clusters.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Van-Quan Vuong: Conceptualization (equal); Data curation (equal); Formal analysis (equal); Investigation (equal); Methodology (equal); Resources (equal); Software (equal); Supervision (equal); Validation (equal); Visualization (equal); Writing - original draft (equal); Writing - review & editing (equal). **Caterina Cevallos:** Data curation (equal); Methodology (equal); Resources (equal); Software (equal); Validation (equal); Writing - original draft (equal). **Ben Hourahine:** Methodology (equal); Software (equal); Writing - review & editing (equal). **Bálint Aradi:** Methodology (equal); Software (equal); Writing - review & editing (equal). **Jacek Jakowski:** Conceptualization (lead); Formal analysis (equal); Investigation (equal); Methodology (equal); Resources (equal); Software (equal); Writing - review & editing (supporting). **Stephan Irle:** Funding acquisition (equal); Project administration (equal); Supervision (equal); Writing - review & editing (equal). **Cristopher Camacho:** Funding acquisition (equal); Conceptualization (lead); Investigation (equal); Methodology (equal); Project administration (equal); Software (equal); Supervision (equal); Validation (equal); Writing - review & editing (equal).

DATA AVAILABILITY

The data that supports the findings of this study are available within the article and its supplementary material.

REFERENCES

[1] S. Grimme and P. R. Schreiner, Angew. Chemie Int. Ed. **57**, 4170 (2018).

[2] F. Neese, M. Atanasov, G. Bistoni, D. Maganas, and S. Ye, J. Am. Chem. Soc. **141**, 2814 (2019).

[3] S. Goedecker, Rev. Mod. Phys. **71**, 1085 (1999).

[4] C. Zhan, W. Sun, Y. Xie, D.-e. Jiang, and P. R. C. Kent, ACS Appl. Mater. Interfaces **11**, 24885 (2019).

[5] A. Van der Ven, Z. Deng, S. Banerjee, and S. P. Ong, Chem. Rev. **120**, 6977 (2020).

[6] J. K. Nørskov, F. Abild-Pedersen, F. Studt, and T. Bligaard, Proc. Natl. Acad. Sci. **108**, 937 (2011).

[7] J. Greeley, Annu. Rev. Chem. Biomol. Eng. **7**, 605 (2016).

[8] Q. Cui, J. Chem. Phys. **145**, 140901 (2016).

[9] P. Saura, M. Röpke, A. P. Gamiz-Hernandez, and V. R. I. Kaila (Humana, New York, NY, 2019) pp. 75–104.

[10] M. Manathunga, A. W. Götz, and K. M. Merz, Curr. Opin. Struct. Biol. **75**, 102417 (2022).

[11] A. V. Akimov and O. V. Prezhdo, Chem. Rev. **115**, 5797 (2015).

[12] L. E. Ratcliff, S. Mohr, G. Huhs, T. Deutsch, M. Masella, and L. Genovese, WIREs Comput. Mol. Sci. **7**, e1290 (2017).

[13] D. R. Bowler and T. Miyazaki, J. Phys. Condens. Matter **22**, 074207 (2010).

[14] J. VandeVondele, U. Borštnik, and J. Hutter, J. Chem. Theory Comput. **8**, 3565 (2012).

[15] W. Thiel, WIREs Comput. Mol. Sci. **4**, 145 (2014).

[16] M. Gaus, Q. Cui, and M. Elstner, WIREs Comput. Mol. Sci. **4**, 49 (2014).

[17] C. Bannwarth, E. Caldeweyher, S. Ehlert, A. Hansen, P. Pracht, J. Seibert, S. Spicher, and S. Grimme, WIREs Comput. Mol. Sci. **11**, e1493 (2021).

[18] Q. Cui, T. Pal, and L. Xie, J. Phys. Chem. B **125**, 689 (2021).

[19] Q. Cui and M. Elstner, Phys. Chem. Chem. Phys. **16**, 14368 (2014).

[20] B. Aradi, B. Hourahine, and T. Frauenheim, J. Phys. Chem. A **111**, 5678 (2007).

[21] A. S. Christensen, T. Kubař, Q. Cui, and M. Elstner, Chem. Rev. **116**, 5301 (2016).

[22] F. Spiegelman, N. Tarrat, J. Cuny, L. Dontot, E. Posenitskiy, C. Martí, A. Simon, and M. Rapacioli, Adv. Phys. X **5**, 1710252 (2020).

[23] V. Q. Vuong, J. M. L. Madridejos, B. Aradi, B. G. Sumpter, G. F. Metha, and S. Irle, Chem. Sci. **11**, 13113 (2020).

[24] J. C. Slater and G. F. Koster, Phys. Rev. **94**, 1498 (1954).

[25] P. Hohenberg and W. Kohn, Phys. Rev. **136**, B864 (1964).

[26] W. Kohn and L. J. Sham, Phys. Rev. **140**, A1133 (1965).

[27] J. P. Kenny, C. L. Janssen, E. F. Valeev, and T. L. Windus, J. Comput. Chem. **29**, 562 (2008).

[28] Q. Sun, J. Comput. Chem. **36**, 1664 (2015).

[29] D. Porezag, T. Frauenheim, T. Köhler, G. Seifert, and R. Kaschner, Phys. Rev. B **51**, 12947 (1995).

[30] G. Seifert, D. Porezag, and T. Frauenheim, Int. J. Quantum Chem. **58**, 185 (1996).

[31] F. J. Domínguez-Gutiérrez, F. Bedoya, P. S. Krstić, J. P. Allain, S. Irle, C. H. Skinner, R. Kaita, and B. Koel, Nucl. Fusion **57**, 086050 (2017).

[32] Y. Nishimoto, D. G. Fedorov, and S. Irle, J. Chem. Theory Comput. **10**, 4801 (2014).

[33] H. Nishizawa, Y. Nishimura, M. Kobayashi, S. Irle, and H. Nakai, J. Comput. Chem. **37**, 1983 (2016).

[34] A. M. N. Niklasson, S. M. Mniszewski, C. F. A. Negre, M. J. Cawkwell, P. J. Swart, J. Mohd-Yusof, T. C. Germann, M. E. Wall, N. Bock, E. H. Rubensson, and H. Djidjev, J. Chem. Phys. **144**, 234101 (2016).

[35] V. W.-z. Yu, F. Corsetti, A. García, W. P. Huhn, M. Jacquelin, W. Jia, B. Lange, L. Lin, J. Lu, W. Mi, A. Seifitokaldani, Á. Vázquez-Mayagoitia, C. Yang, H. Yang, and V. Blum, Comput. Phys. Commun. **222**, 267 (2018).

[36] H. Hu, Z. Lu, M. Elstner, J. Hermans, and W. Yang, J. Phys. Chem. A **111**, 5685 (2007).

[37] Y. Nishimura and H. Nakai, J. Comput. Chem. **39**, 105 (2018).

[38] T. J. Giese, H. Chen, T. Dissanayake, G. M. Giambaşu, H. Heldenbrand, M. Huang, E. R. Kuechler, T.-S. Lee, M. T. Panteva, B. K. Radak, and D. M. York, J. Chem. Theory Comput. **9**, 1417 (2013).

[39] Y. Nishimoto, D. G. Fedorov, and S. Irle, Chem. Phys. Lett. **636**, 90 (2015).

[40] Y. Nishimoto and D. G. Fedorov, J. Comput. Chem. **38**, 406 (2017).

[41] Y. Nishimoto and D. G. Fedorov, J. Chem. Phys. **148**, 064115 (2018).

[42] V. Q. Vuong, Y. Nishimoto, D. G. Fedorov, B. G. Sumpter, T. A. Niehaus, and S. Irle, J. Chem. Theory Comput. **15**, 3008 (2019).

[43] V. W.-z. Yu, C. Campos, W. Dawson, A. García, V. Havu, B. Hourahine, W. P. Huhn, M. Jacquelin, W. Jia, M. Keçeli, R. Laasner, Y. Li, L. Lin, J. Lu, J. Moussa, J. E. Roman, Á. Vázquez-Mayagoitia, C. Yang, and V. Blum, Comput. Phys. Commun. **256**, 107459 (2020).

[44] L. Lin, J. Lu, L. Ying, R. Car, and W. E, Commun. Math. Sci. **7**, 755 (2009).

[45] L. Lin, M. Chen, C. Yang, and L. He, J. Phys. Condens. Matter **25**, 295501 (2013).

[46] Y. Nishimura and H. Nakai, Chem. Lett. **50**, 1546 (2021).

[47] J. D. C. Maia, G. A. Urquiza Carvalho, C. P. Mangueira, S. R. Santana, L. A. F. Cabral, and G. B. Rocha, J. Chem. Theory Comput. **8**, 3072 (2012).

[48] X. Wu, A. Koslowski, and W. Thiel, J. Chem. Theory Comput. **8**, 2272 (2012).

[49] J. W. Mullinax, E. Maradzike, L. N. Koulias, M. Mostafanejad, E. Epifanovsky, G. Gidofalvi, and A. E. DePrince, J. Chem. Theory Comput. **15**, 6164 (2019).

[50] J. D. C. Maia, L. dos Anjos Formiga Cabral, and G. B. Rocha, J. Mol. Model. **26**, 313 (2020).

[51] M. Manathunga, Y. Miao, D. Mu, A. W. Götz, and K. M. Merz, J. Chem. Theory Comput. **16**, 4315 (2020).

[52] G. Zhou, B. Nebgen, N. Lubbers, W. Malone, A. M. N. Niklasson, and S. Tretiak, J. Chem. Theory Comput. **16**, 4951 (2020).

[53] B. S. Fales, E. R. Curtis, K. G. Johnson, D. Lahana, S. Seritan, Y. Wang, H. Weir, T. J. Martínez, and E. G. Hohenstein, J. Chem. Theory Comput. **16**, 4021 (2020).

[54] G. M. J. Barca, J. L. Galvez-Vallejo, D. L. Poole, A. P. Rendell, and M. S. Gordon, J. Chem. Theory Comput. **16**, 7232 (2020).

[55] V. W.-z. Yu, J. Moussa, P. Kůs, A. Marek, P. Messmer, M. Yoon, H. Lederer, and V. Blum, Comput. Phys. Commun. **262**, 107808 (2021).

[56] M. S. Gordon, G. Barca, S. S. Leang, D. Poole, A. P. Rendell, J. L. Galvez Vallejo, and B. Westheimer, J. Phys. Chem. A **124**, 4557 (2020).

[57] J. Jakowski, S. Irle, and K. Morokuma, in *GPU Comput. Gems Emerald Ed.* (Elsevier, 2011) Chap. 5, pp. 59–73.

[58] G.-h. Fan, K.-l. Han, and G.-z. He, Chinese J. Chem. Phys. **26**, 635 (2013).

[59] S. I. Allec, Y. Sun, J. Sun, C.-e. A. Chang, and B. M. Wong, J. Chem. Theory Comput. **15**, 2807 (2019).

[60] S. Tomov, J. Dongarra, and M. Baboulin, Parallel Comput. **36**, 232 (2010).

[61] W. M. C. Foulkes and R. Haydock, Phys. Rev. B **39**, 12520 (1989).

[62] M. Elstner, D. Porezag, G. Jungnickel, J. Elsner, M. Haugk, T. Frauenheim, S. Suhai, and G. Seifert, Phys. Rev. B **58**, 7260 (1998).

[63] G. Seifert and J.-O. Joswig, WIREs Comput. Mol. Sci. **2**, 456 (2012).

[64] Y. Yang, H. Yu, D. York, Q. Cui, and M. Elstner, J. Phys. Chem. A **111**, 10861 (2007).

[65] M. Gaus, Q. Cui, and M. Elstner, J. Chem. Theory Comput. **7**, 931 (2011).

[66] M. Elstner, Theor. Chem. Acc. **116**, 316 (2006).

[67] M. Elstner and G. Seifert, Philos. Trans. R. Soc. A Math. Phys. Eng. Sci. **372**, 20120483 (2014).

[68] J. Cuny, N. Tarrat, F. Spiegelman, A. Huguenot, and M. Rapacioli, J. Phys. Condens. Matter **30**, 303001 (2018).

[69] A. Simon, M. Rapacioli, E. Michoulier, L. Zheng, K. Korchagina, and J. Cuny, Mol. Simul. **45**, 249 (2019).

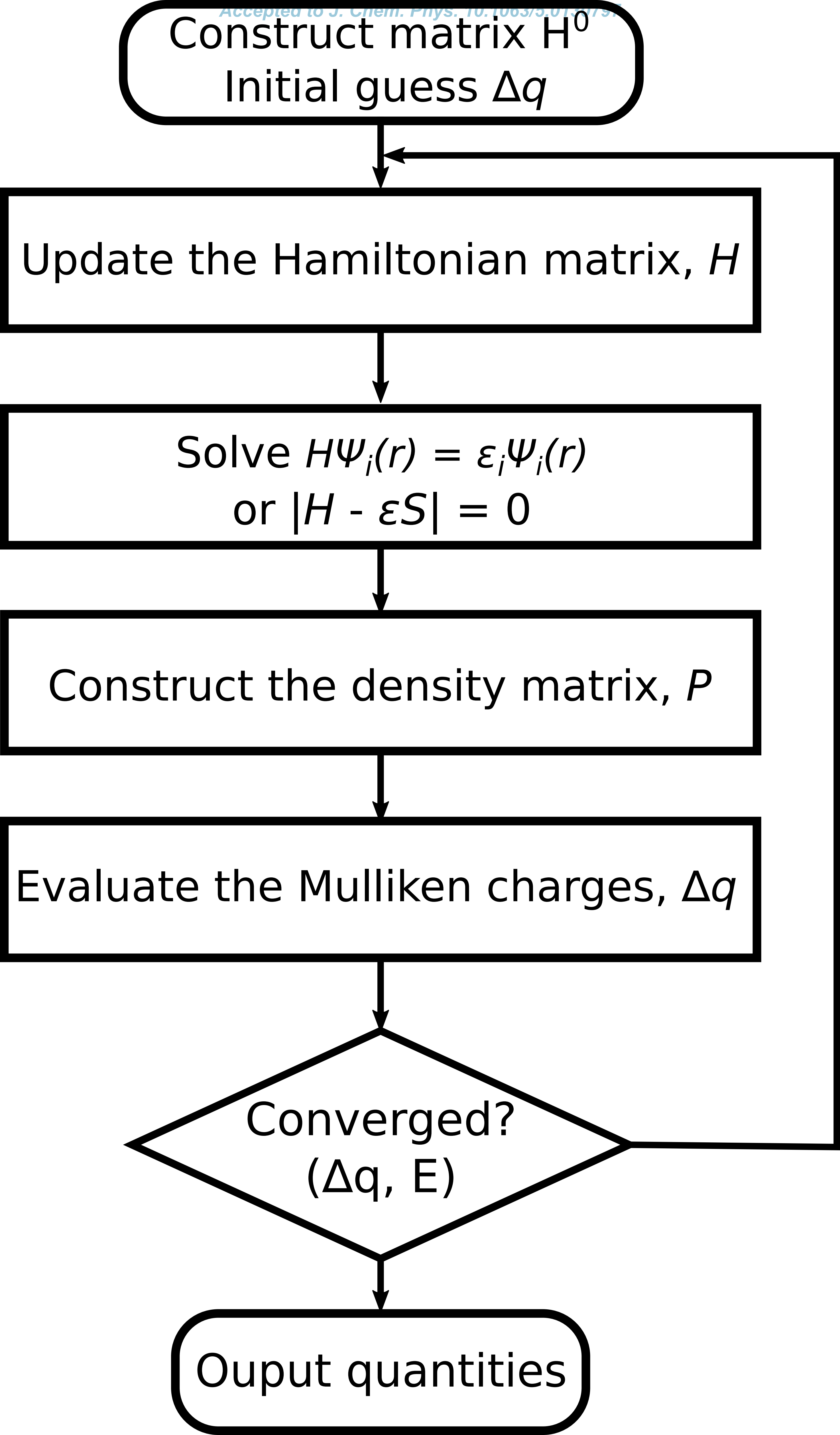
[70] R. S. Mulliken, J. Chem. Phys. **23**, 1833 (1955).

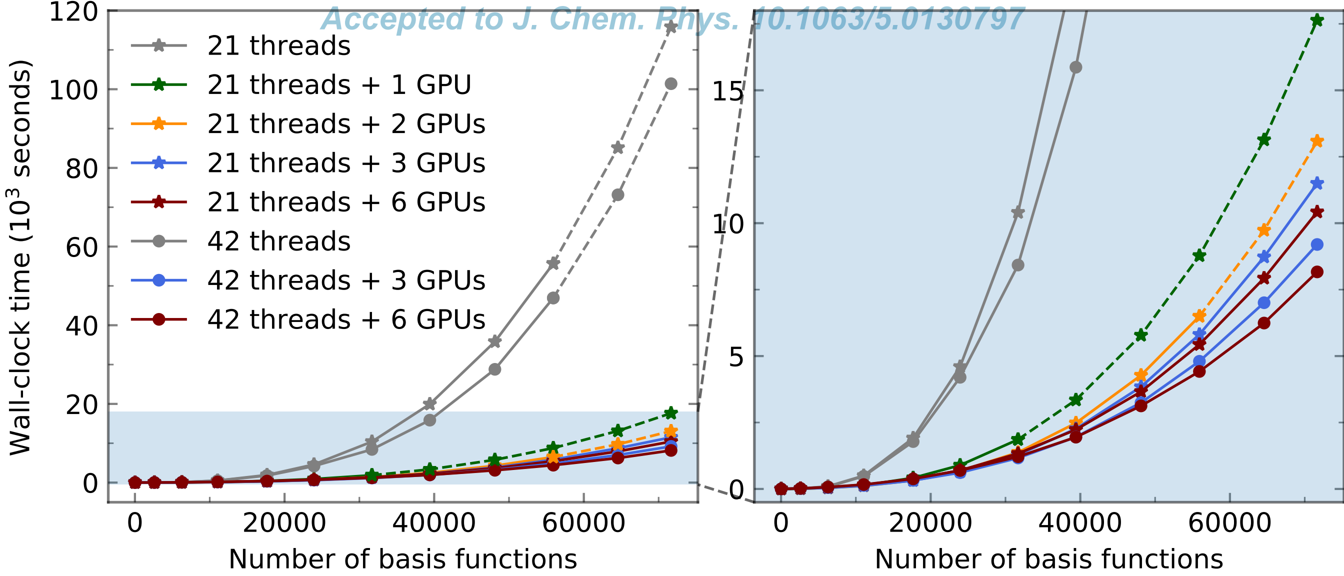
[71] B. Hourahine, B. Aradi, A. Pecchia, J. Řezáč, J. J. Kranz, C. Camacho, V. Yu, M. C. Cevallos-Brenes, yuri@FreeBSD.org, T. Niehaus, and C. Vitkun, DFTB+ release 19.1, `https://doi.org/10.5281/zenodo.3265199` (2019).

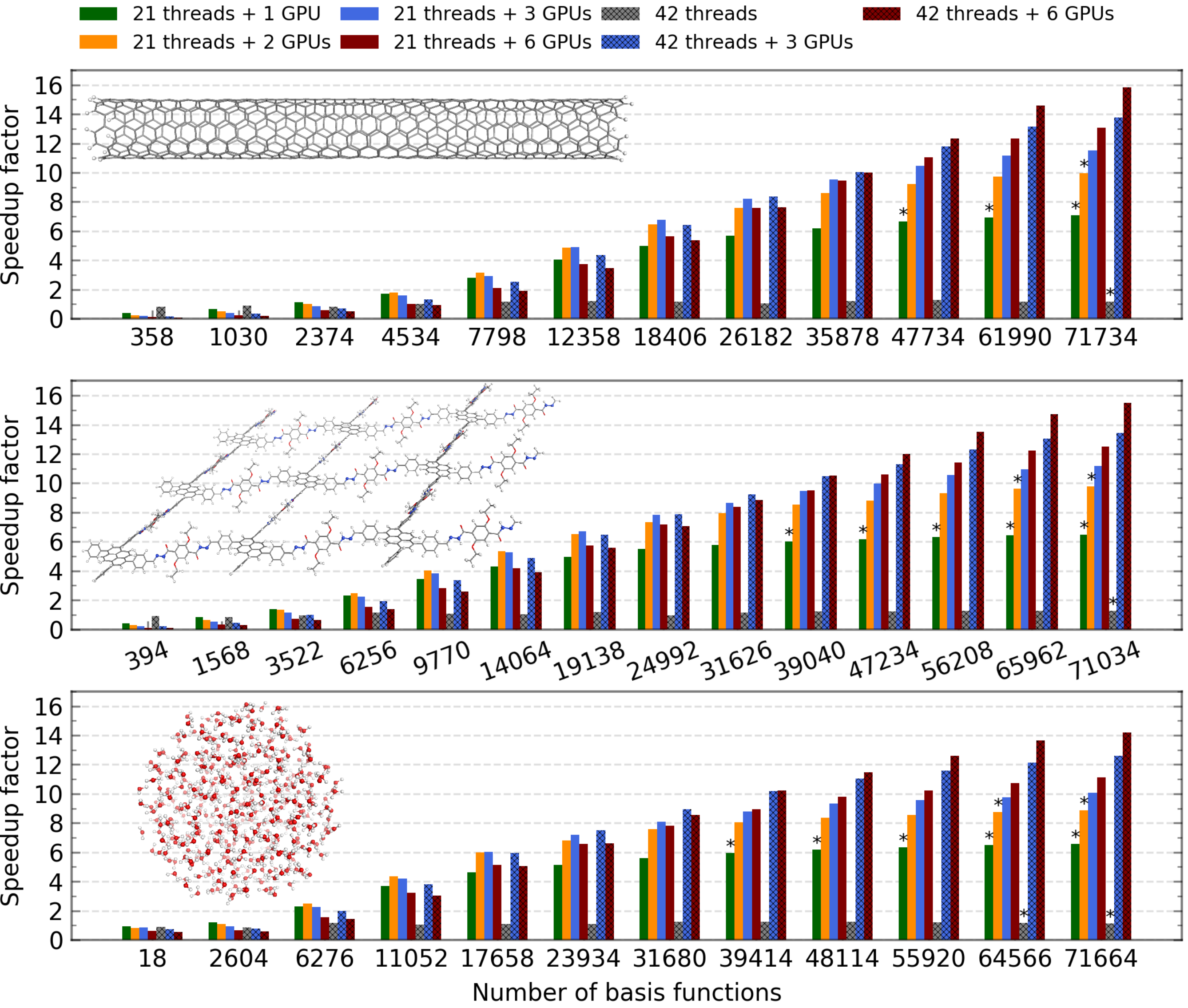
[72] B. Hourahine, B. Aradi, V. Blum, F. Bonafé, A. Buccheri, C. Camacho, C. Cevallos, M. Y. Deshaye, T. Dumitrică, A. Dominguez, S. Ehlert, M. Elstner, T. van der Heide, J. Hermann, S. Irle, J. J. Kranz, C. Köhler, T. Kowalczyk, T. Kubař, I. S. Lee, V. Lutsker, R. J. Maurer, S. K. Min, I. Mitchell, C. Negre, T. A. Niehaus, A. M. N. Niklasson, A. J. Page, A. Pecchia, G. Penazzi, M. P. Persson, J. Řezáč, C. G. Sánchez, M. Sternberg, M. Stöhr, F. Stuckenberg, A. Tkatchenko, V. W. Yu, and T. Frauenheim, J. Chem. Phys. **152**, 124101 (2020).

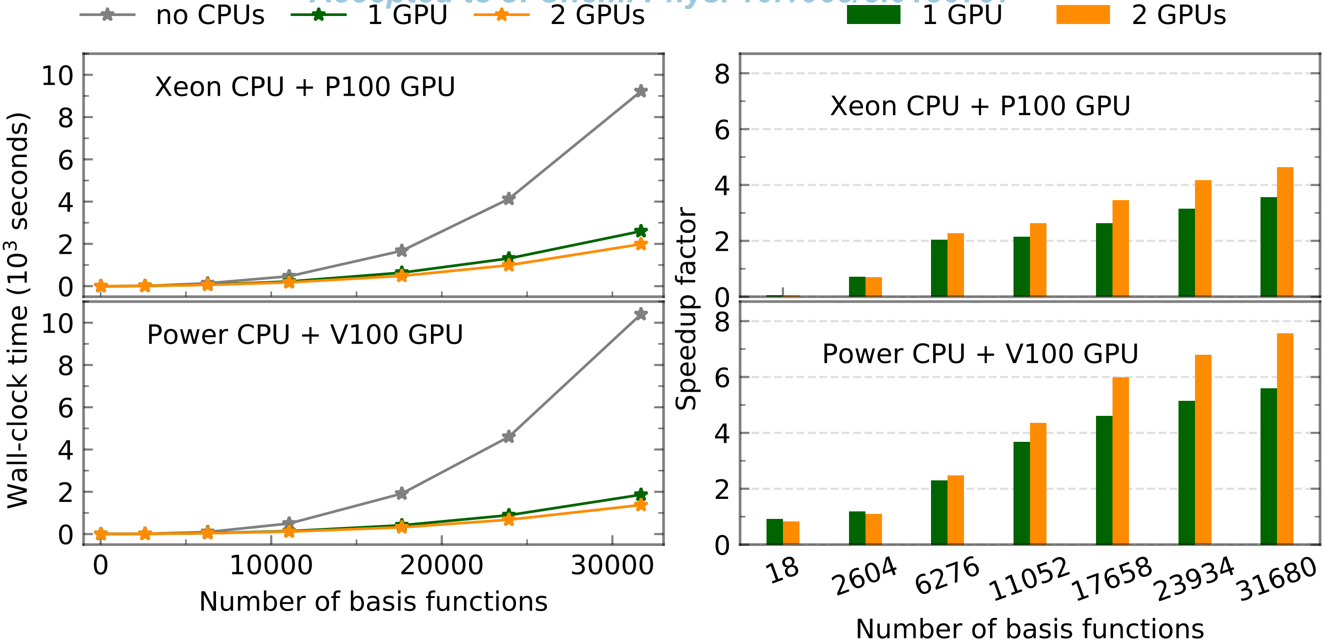
[73] A. Haidar, S. Tomov, J. Dongarra, R. Solca, and T. Schulthess, The International journal of high performance computing applications **28**, 196 (2014).

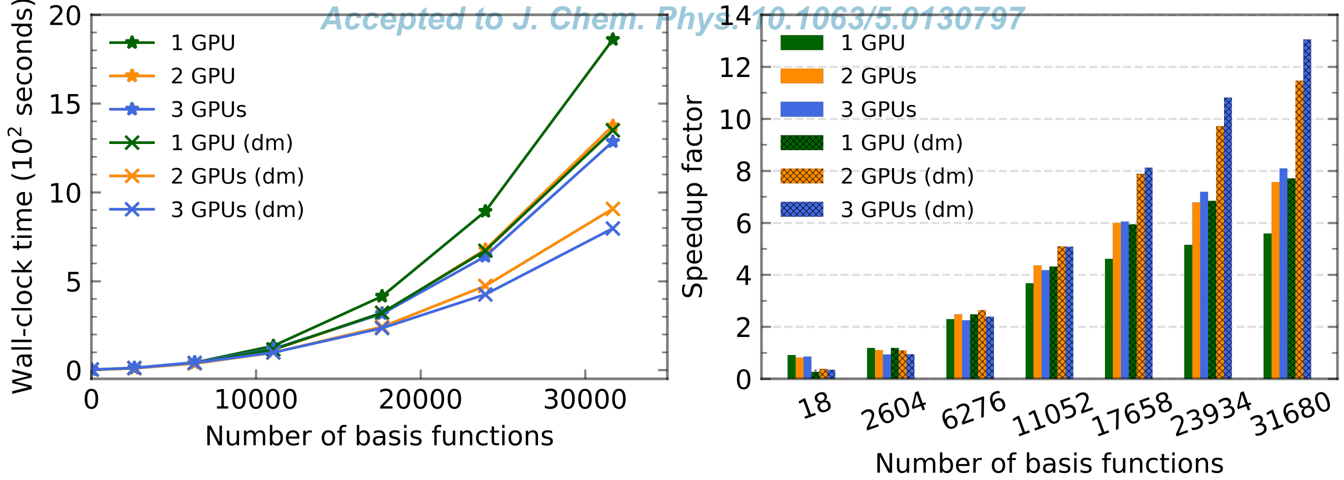
[74] T. Dong, A. Haidar, P. Luszczek, S. Tomov, A. Abdelfattah, and J. Dongarra, *Magma batched: A batched blas approach for small matrix factorizations and applications on gpus*, Tech. Rep. (Technical Report. Technical report, 2016).

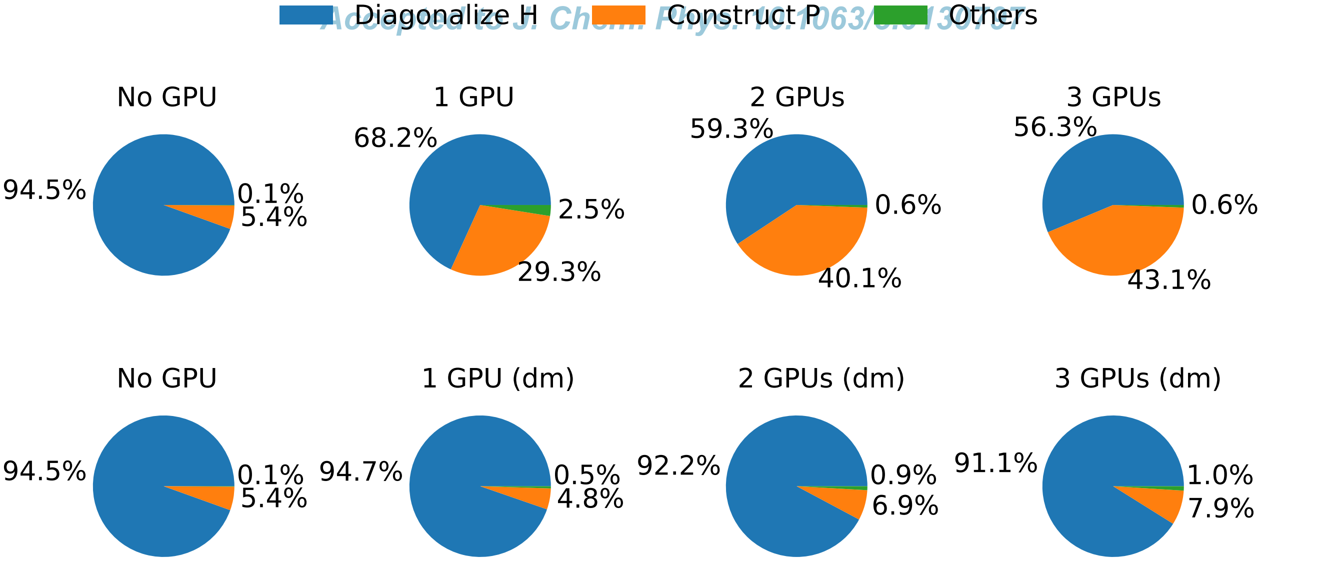

Construct matrix H^0
Initial guess Δq
Update the Hamiltonian matrix, H
Solve $H\Psi_i(r) = \varepsilon_i\Psi_i(r)$
or $|H - \varepsilon S| = 0$
Construct the density matrix, P
Evaluate the Mulliken charges, Δq
Converged?
(Δq, E)
Ouput quantities


Wall-clock time (10^3 seconds)
120
100
80
60
40
20
0
21 threads
21 threads + 1 GPU
21 threads + 2 GPUs
21 threads + 3 GPUs
21 threads + 6 GPUs
42 threads
42 threads + 3 GPUs
42 threads + 6 GPUs
0
20000
40000
60000
Number of basis functions
15
10
5
0
0
20000
40000
60000
Number of basis functions


21 threads + 1 GPU
21 threads + 3 GPUs
42 threads
42 threads + 6 GPUs
21 threads + 2 GPUs
21 threads + 6 GPUs
42 threads + 3 GPUs
Speedup factor
0
2
4
6
8
10
12
14
16
358
1030
2374
4534
7798
12358
18406
26182
35878
47734
61990
71734
*
Speedup factor
0
2
4
6
8
10
12
14
16
394
1568
3522
6256
9770
14064
19138
24992
31626
39040
47234
56208
65962
71034
Speedup factor
0
2
4
6
8
10
12
14
16
18
2604
6276
11052
17658
23934
31680
39414
48114
55920
64566
71664
Number of basis functions


no CPUs
1 GPU
2 GPUs
1 GPU
2 GPUs
Xeon CPU + P100 GPU
Power CPU + V100 GPU
Wall-clock time (10^3 seconds)
0
2
4
6
8
10
0
10000
20000
30000
Number of basis functions
Xeon CPU + P100 GPU
Power CPU + V100 GPU
Speedup factor
0
2
4
6
8
18
2604
6276
11052
17658
23934
31680
Number of basis functions


Wall-clock time (10^2 seconds)
0
5
10
15
20
0
10000
20000
30000
Number of basis functions
1 GPU
2 GPU
3 GPUs
1 GPU (dm)
2 GPUs (dm)
3 GPUs (dm)
Speedup factor
0
2
4
6
8
10
12
14
18
2604
6276
11052
17658
23934
31680
Number of basis functions
1 GPU
2 GPUs
3 GPUs
1 GPU (dm)
2 GPUs (dm)
3 GPUs (dm)



Diagonalize H
Construct P
Others
No GPU
94.5%
0.1%
5.4%
1 GPU
68.2%
2.5%
29.3%
2 GPUs
59.3%
0.6%
40.1%
3 GPUs
56.3%
0.6%
43.1%
No GPU
94.5%
0.1%
5.4%
1 GPU (dm)
94.7%
0.5%
4.8%
2 GPUs (dm)
92.2%
0.9%
6.9%
3 GPUs (dm)
91.1%
1.0%
7.9%