LASP-D3: A Linear-Scaling and Memory-Efficient GPU Implementation of DFT-D3 for Large Periodic Systems

Han-Zhi Luo[a], Cheng Shang[a*] and Zhi-Pan Liu[a,b*]

[a]State Key Laboratory of Porous Materials for Separation and Conversion, Collaborative Innovation Center of Chemistry for Energy Material, Shanghai Key Laboratory of Molecular Catalysis and Innovative Materials, Key Laboratory of Computational Physical Sci ence, Department of Chemistry, Fudan University, Shanghai 200433, China

[b]State Key Laboratory of Metal Organic Chemistry, Shanghai Institute of Organic Chemistry, Chinese Academy of Sciences, Shanghai 200032, China

*Email: cshang@fudan.edu.cn; zpliu@fudan.edu.cn

Abstract

The van-der-Waals (vdW) interaction is ubiquitous in materials and is long-range by nature. To speed up the vdW calculation in large systems of thousands of atoms, we developed LASP-D3 on Graphics Processing Unit (GPU) device, which is a highly efficient Compute Unified Device Architecture (CUDA) implementation of the DFT-D3 method, a widely utilized vdW correction form compatible with density functional theory calculations. Our algorithm achieves a linear-scaling, O(N), time complexity for large periodic systems, being much faster than all current versions, and reduces significantly the GPU memory consumption compared to existing PyTorch-based GPU implementations. The LASP-D3 is compatible with all GPU-based machine learning potentials, such as the generalized global neural network potential developed by us and thus facilitates efficient large-scale atomic simulations with vdW interaction.

Keywords: DFT-D3 algorithm, GPU acceleration, high-performance computing

1 Introduction

High-performance scientific computing is undergoing a paradigm shift toward Graphics Processing Units (GPUs)[1–7], leveraging their massive parallelism for intensive workloads. In atomic simulation, Machine Learning Potentials (MLPs) have emerged as a transformative technology[8–15]. While Density Functional Theory (DFT) has long been the standard for calculating material properties[16], its high computational cost limits system size and timescale. MLPs overcome this trade-off, reproducing DFT-level accuracy at a fraction of the cost[17–21]. Because MLPs rely on tensor algebra operations that align perfectly with GPU architecture, they enable quantum-accurate molecular dynamics (MD) simulations[22–24] and global exploration[25–27] for unprecedented system sizes.

However, MLPs by fitting DFT dataset, inherits the common DFT failure in capturing the long-range electron correlations[28–32], specifically London dispersion forces[33] which are critical factors in molecular crystals[34], biological macromolecules[35], and surface adsorption[36]. Among various correction schemes[37–40], Grimme's DFT-D methods[41–43], typically DFT-D3 method[44], is the de facto standard correction for this deficiency due to its robustness[45] and efficiency[44]. While the cost of DFT-D3 is negligible compared to traditional DFT, it becomes a significant bottleneck when paired with ultra-fast MLPs. This issue is exacerbated by a hardware mismatch: modern simulation platforms typically pair powerful GPUs with Central Processing Units (CPUs) that have limited core counts. Running the DFT-D3 correction on the CPU creates a severe bottleneck, negating the performance gains of the GPU-accelerated MLP.

Attempts to address this using GPU-accelerated libraries built on high-level frameworks (e.g., PyTorch) have yielded limited success[46,47]. While faster from CPU-based versions, these implementations still suffer from suboptimal time complexity compared to linear-scaling MLPs[7,15], leaving the correction as the rate-limiting step. Furthermore, high-level frameworks lack direct control over hardware resources, leading to excessive memory consumption. This creates a "memory capacity wall" that prevents researchers from simulating very large-scale systems such as protein-ligand complexes or amorphous solids that MLPs were designed to investigate.

In this work, we develop LASP-D3, a novel implementation of the DFT-D3 algorithm developed entirely by CUDA C++ to resolve these dual challenges. By bypassing high-level frameworks and engaging directly with the GPU architecture, LASP-D3 utilizes custom kernels and optimized memory management to achieve superior performance. Our implementation achieves optimized time complexity of O(N) for large periodic systems and linear memory scaling, effectively eliminating the overhead of the correction. When integrated with the LASP software project[48] in our group, CUDA-D3 incurs maximally a ~20% computational overhead in super-large systems (100,000 atoms), drastically outperforming both CPU and PyTorch-based alternatives. This implementation removes the key barrier in hybridizing MLP and D3 in one simulation, paving the way for accurate investigation into complex realistic systems with appreciable vdW contributions.

2 DFT-D3 algorithm

2.1 Energy calculation algorithm

The DFT-D3 algorithm[44] is an empirical correction designed to account for long-range van der Waals interactions, specifically dispersion forces, which are often inadequately described by standard density functionals (DFs). The correction involves adding a dispersion energy term, E_{disp}, to the standard Kohn-Sham DFT energy, E_{KS-D} .

$$E_{total} = E_{KS-D} - E_{disp} \tag{1}$$

The E_{disp} consists of a dominant two-body term and a less critical three-body term, $E_{disp} = E^{(2)} + E^{(3)}$. We focus on the two-body term in the current work due to its importance and its high computation efficiency. The two-body term is computed by summing the interactions between all unique atom pairs (A, B) in the system.

$$E^{(2)} = \sum_{A,B} \sum_{n=6,8} s_n \frac{C_n^{AB}}{r_{AB}^n} f_{d,n}(r_{AB}) \tag{2}$$

Here, s_n is the scaling factor that is fitted for different DFs, r_{AB} is the distance between the two atoms, C_n^{AB} is the dispersion coefficient, and $f_{d,n}(r_{AB})$ is the damping function that smoothly attenuates the dispersion energy at short distances. Various damping functions can be applied including zero-damping[44]

$$f_{d,n}(r_{AB}) = \frac{1}{1 + 6\left(r_{AB}/\left(s_{r,n} R_0^{AB}\right)\right)^{-\alpha_n}} \tag{3}$$

and Beche-Johnson damping[49].

$$f_{d,n}(r_{AB}) = \frac{r_{AB}^n}{r_{AB}^n + \left(a_1 R_0^{AB} + a_2\right)^n} \tag{4}$$

Here, $s_{r,n}, a_1, a_2$ are dependent scaling factors specific to DFs, R_0^{AB} is the pair-specific cutoff radius derived from tabulated atomic values. α_n is set to $\alpha_6 = 14$ and $\alpha_8 = 16$, respectively.

A key feature of the DFT-D3 method is that the dispersion coefficients C_6^{AB} are not static. Instead, they are dynamically calculated to reflect the local chemical environment of each atom. This is achieved by first calculating a Coordination Number (CN) for each atom,

$$CN^A = \sum_{B \neq A} \frac{1}{1 + \exp\left(-k_1\left(k_2\left(R_{A,cov} + R_{B,cov}\right)/r_{AB} - 1\right)\right)} \tag{5}$$

where $R_{A,cov}$ is the scaled covalent radius of atom A, r_{AB} is the distance between two atoms, k_1 and k_2 are fixed parameters set to 16 and 4/3, respectively.

Once the coordination numbers for a pair of atoms (A, B) are known, the specific dispersion coefficient C_6^{AB} is calculated by interpolating over a grid of pre-computed reference values.

$$C_6^{AB} = \frac{\sum_i^{n_A} \sum_j^{n_B} C_{6,ref}^{AB}\left(CN_A^i, CN_B^j\right) \exp\left(-k_3\left(\left(CN^A - CN_i^A\right)^2 + \left(CN^B - CN_j^B\right)^2\right)\right)}{\sum_i^{n_A} \sum_j^{n_b} \exp\left(-k_3\left(\left(CN^A - CN_i^A\right)^2 + \left(CN^B - CN_j^B\right)^2\right)\right)} \tag{6}$$

For each element, a small number of reference geometries (typically 1-5) are used to generate reference C_6^{AB} coefficients and their corresponding reference CNs. For any given atom pair (A, B), this creates a 2D grid of up to 25 reference values $C_{6,ref}^{AB}\left(CN_i^A, CN_j^B\right)$. The final coefficient is obtained via a weighted average across this grid, using Gaussian functions as weights based on the calculated CN^A and CN^B.

2.2 Periodic system

When applying the DFT-D3 correction in periodic systems, such as crystals, the pair-wise summation must be extended to include interactions between atoms in the reference unit cell and atoms in the surrounding periodic images. Since this would lead to an infinite number of interactions, a cutoff scheme is employed to make the calculation computationally feasible. An interaction between a central atom A and a neighboring atom B (or its periodic image) is only considered if the distance r_{AB} is within a specified cutoff radius. The DFT-D3 algorithm involves two distinct pairwise operations: the CN calculation and the energy summation. Consequently, two independent cutoffs, namely the CN cutoff and the energy cutoff, are defined. These two cutoffs can be adjusted independently. Theoretically, larger cutoffs yield more accurate results. However, the computational cost increases significantly, scaling with the volume of the cutoff sphere ($O(R_{cut}^3)$). Therefore, a balance between accuracy and efficiency is crucial. The typical values for these cutoffs are beyond 40 Bohr (~21 Å).

2.3 Force calculation algorithm

For applications like geometry optimization and MD simulation, the force acting on each atom is needed. The force $\vec{F}$ is defined as $\vec{F} = -\partial E/\partial \vec{r}$. Based on the two-body energy expression, the force on a specific atom x arises from its interactions with all other atoms.

$$\overrightarrow{F_x} = \frac{\partial E}{\partial \overrightarrow{r_x}} = \sum_{A,B}\sum_{n=6,8}\left(s_n \frac{\partial C_n^{AB}}{\partial \overrightarrow{r_x}}\frac{f_{d,n}}{r_{AB}^n} + s_n C_n^{AB}\frac{\partial f_{d,n}}{\partial \overrightarrow{r_x}}\frac{1}{r_{AB}^n} + s_n C_n^{AB} f_{d,n}\frac{\partial}{\partial \overrightarrow{r_x}}\frac{1}{r_{AB}^n}\right) \tag{7}$$

The last two entries, the derivatives of $f_{d,n}$ and $\frac{1}{r_{AB}^n}$ can be easily calculated, as they are localized and is only relative to $\overrightarrow{r_A}$ and $\overrightarrow{r_B}$. However, the first entry is more complex as C_n^{AB} is a function of CN^A and CN^B, which in turn depends on the positions of all neighboring atoms of both A and B. This leads to complex force contributions. Applying the chain rule again to the term C_n^{AB} shows that a change in the position of atom x affects its own and its neighbors' CNs, which then alters the forces that its secondary neighbors experience. Therefore, the derivative of C_n^{AB} and dispersion energy with respect to a specific atom x is written as (8) and (9), respectively.

$$\frac{\partial C_n^{AB}(CN_A, CN_B)}{\partial \overrightarrow{r_x}} = \frac{\partial C_n^{AB}}{\partial CN_A}\frac{\partial CN_A}{\partial \overrightarrow{r_x}} + \frac{\partial C_n^{AB}}{\partial CN_B}\frac{\partial CN_B}{\partial \overrightarrow{r_x}} \tag{8}$$

$$\frac{\partial E}{\partial \overrightarrow{r_x}} = \sum_{A}\sum_{n=6,8}\left(s_n C_n^{Ax}\frac{\partial f_{d,n}}{\partial \overrightarrow{r_x}}\frac{1}{r_{Ax}^n} + s_n C_n^{Ax} f_{d,n}\frac{\partial}{\partial \overrightarrow{r_x}}\frac{1}{r_{Ax}^n}\right) + \sum_{A,B}\sum_{n=6,8} s_n \frac{\partial C_n^{AB}}{\partial \overrightarrow{r_x}}\frac{f_{d,n}}{r_{AB}^n} \tag{9}$$

The calculation requires first computing the derivatives of the CNs with respect to atomic positions $\sum_A \partial CN_A/\partial \overrightarrow{r_x}$ for all atoms. With these intermediate values, the final forces can be assembled efficiently. Fortunately, the dual sum can be simplified because only $\partial C_n^{AB}/\partial CN_A$ is related with B while $\partial CN_A/\partial \overrightarrow{r_x}$ is independent from B. Therefore, the dual sum is simplified to two sequential sums.

$$\sum_{A,B}\sum_{n=6,8}\frac{\partial C_n^{AB}}{\partial CN_A}\frac{\partial CN_A}{\partial \overrightarrow{r_x}}=\sum_{n=6,8}\left(\sum_{A}\left(\sum_{B}\frac{\partial C_n^{AB}}{\partial CN_A}\right)\frac{\partial CN_A}{\partial \overrightarrow{r_x}}\right) \tag{10}$$

The same applies to the second entry of (8). Therefore, to calculate the force of atom x, we first need to iterate over all atoms in the system and obtain their $\sum_B \partial C_n^{AB} / \partial CN_A$ value and perform accumulation to obtain the $\sum_{A,B}\sum_{n=6,8} \partial C_n^{AB} / \partial \overrightarrow{r_x}$ value. After that, we can iterate over the atom's neighbors and calculate the force using (9).

3 LASP-D3 Implementation

Our LASP-D3 implementation is designed for both high efficiency and robust vdW computation with flexible Application Programming Interfaces (APIs) tailored for different atomic simulation workloads. We leverage the massive parallelism of CUDA and optimized numeric algorithms to accelerate computationally intensive tasks while minimizing GPU memory consumption. Furthermore, a custom memory reuse mechanism minimizes allocation overhead, reducing memory bandwidth requirements. We employ C++11 Resource Acquisition Is Initialization (RAII) patterns to create a maintainable, leak-free memory management system, ensuring stability during demanding, long-running simulations. Additionally, algorithmic improvements guarantee linear scaling O(N) of computation time with respect to system size, enhancing both numeric accuracy and stability.

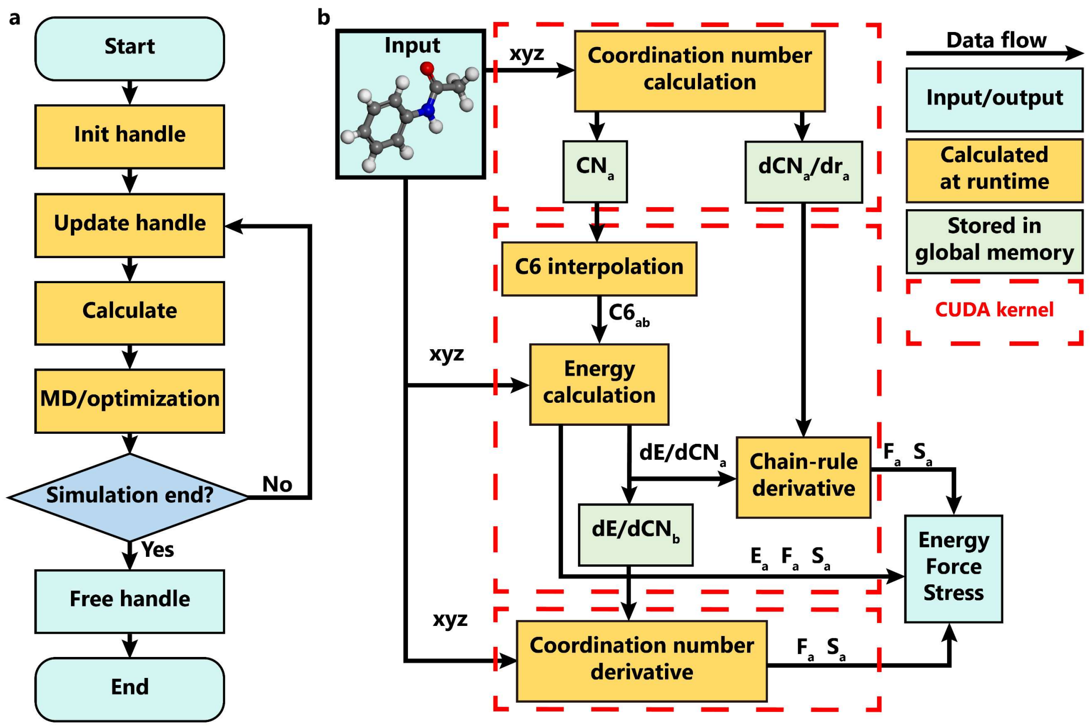

Figure 1 The workflow of (a) typical MD or optimization simulation involving LASP-D3 and (b) LASP-D3 energy, force and stress calculation. Blue, yellow and green blocks represent input/output data, runtime calculation and data stored in global memory, respectively. Red dotted blocks represent GPU kernels.

3.1 User interface

To ensure high performance and stability during iterative atomic simulations, LASP-D3 utilizes an object-oriented design encapsulated within a C-compatible API. The library relies on a central `d3_handle_t` object to manage structure configuration and the lifecycle of GPU memory. Following the C++ Resource Acquisition Is Initialization (RAII) idiom, the handle initializes resources based on system size upon creation and ensures complete deallocation upon termination, thereby preventing memory leaks. Crucially, this design mitigates the latency associated with continuous memory allocation. Pseudocode 1, shows a typical MD workflow using MLP potential and DFT-D3 correction. During the workflow, the handle persists across time steps; functions like `set_atoms` and `set_cell` simply update the existing memory blocks via efficient `cudaMemcpy` operations rather than reallocating GPU resources. This reuse mechanism significantly reduces memory bandwidth overhead and maximizes throughput for long-timescale simulations.

Pseudocode 1. Simulation process with LASP-D3

```
Input: Initial coordinates ($R_0$), Initial cell ($h_0$), Atom types
(Z)
Output: Trajectory of coordinates ($R_t$)

// Initialization: GPU resources allocated once
handle = init_d3_handle(max_atoms, Z)

For step t = 1 to N_steps do
    // 1. Update D3 Handle (Reuse GPU memory)
    set_atoms(handle, R_{t-1})
    set_cell(handle, h_{t-1})

    // 2. Compute D3 Corrections (GPU)
    E_d3, F_d3, S_d3 = compute_d3_from_handle(handle)

    // 3. Compute MLP Base Potentials
    E_mlp, F_mlp, S_mlp = MLP_inference(R_{t-1}, h_{t-1})

    // 4. Combine Contributions (Add D3 correction)
    E_tot = E_mlp + E_d3
    F_tot = F_mlp + F_d3
    S_tot = S_mlp + S_d3

    // 5. Forward Step Calculation
    R_t, h_t = forward_step(R_{t-1}, h_{t-1}, F_tot, S_tot)
End For

// Termination: Release all GPU resources
free_d3_handle(handle)
```

3.2 CUDA programming model

The CUDA programming model abstracts the GPU's massive execution architecture. Computation logic is encapsulated in kernels, which execute sequentially within a CUDA stream. Each kernel launches a grid of blocks, which execute independently in parallel. Within blocks, threads serve as the elementary execution units, communicating via fast, block-wise shared memory. This hierarchical structure allows for fine-grained control over data locality and parallel execution.

3.3 CUDA kernels

LASP-D3 accepts atomic coordinates, element types, and cell vectors as input, returning total dispersion energy, atomic forces, and the stress tensor. The calculation follows a multi-stage kernel pipeline (Figure 1b), managed via sequential launches in a single CUDA stream.

The first kernel computes CN and its derivatives based on (5). We utilize a thread-block distribution where each block handles one central atom and its threads iterate over neighbors. Block-wise reductions accumulate local CN values and derivatives, which are then written to global memory. This mapping assigns unique memory addresses to specific threads, eliminating race conditions and the need for atomic operations, thus maximizing efficiency.

The second kernel calculates direct two-body interactions (the first entry in (9)) and components of the three-body terms. Adopting a similar workload distribution, threads compute pairwise energy, force, and stress contributions. Crucially, this kernel also computes derivatives $\partial \mathrm{C}_{\mathrm{n}}^{\mathrm{AB}} / \partial \mathrm{CN}_{\mathrm{A}}$ required for complex three-body terms. While atom-specific entries are written directly to global memory, stress tensor components which are global properties are accumulated using CUDA atomic additions. To maximize resource utilization, final scalar energy accumulation is offloaded to the CPU, running asynchronously alongside the final GPU kernel.

The final kernel assembles the remaining three-body force and stress terms derived from the last entry in (9). Notably, we explicitly recalculate certain CN derivatives rather than storing them in global memory. While counterintuitive, benchmarking revealed that recalculating these values is faster than the latency incurred by reading massive pairwise matrices from global memory. Furthermore, this "recompute-over-store" strategy maintains $\mathrm{O(N)}$ memory complexity, avoiding the $\mathrm{O(N^2)}$ scaling associated with storing pairwise derivative matrices, thereby enabling the simulation of significantly larger systems.

3.4 Linear scaling via cell list algorithm

To ensure linear scaling ($O(N)$) for large periodic systems, we implemented a cell list algorithm. Standard DFT-D3 implementations often waste computational cycles iterating over distant, non-interacting atoms. To overcome this limitation, our implementation partitions the simulation box into a grid of cells, with dimensions determined by the interaction cutoff radius. Atoms are sorted into these cells, and the dispersion interaction calculation is restricted to atoms within the same or immediately neighboring cells. This spatial locality optimization reduces the algorithmic complexity from $O(N^2)$ to $O(N)$, ensuring that LASP-D3 remains efficient for massive systems and matches the linear scaling characteristics of the underlying HPNN-based MLP[15].

3.5 Algorithmic improvements

Beyond performance, we introduced two critical algorithmic enhancements to address floating-point arithmetic pitfalls, specifically optimizing for the single-precision (FP32) architecture of modern GPUs without sacrificing accuracy.

3.5.1 Avoiding arithmetic underflow

A numerical stability issue arises during the interpolation of the C_6^{AB} coefficients in (6) where weights are determined by Gaussian functions of the CN. If an atom's CN deviates significantly from the reference value, the exponential term can underflow to zero in single precision (occurring at CN differences of merely ~5). While using double precision extends this limit to CN value differences of ~13, it comes at a significant performance cost and still does not protect against extreme cases. If all weights underflow, the denominator vanishes, causing division-by-zero errors (NaNs). To resolve this, we employ the Log-Sum-Exp stabilization technique. We first compute the logarithmic arguments using (11).

$$x_{ij} = -k_3\left(\left(CN^A - CN_i^A\right)^2 + \left(CN^B - CN_j^B\right)^2\right) \tag{11}$$

We then identify the maximum argument, $x_{max} = \max\left(x_{ij}\right)$. and compute the final sum by shifting the exponents using (12).

$$C_6^{AB} = \frac{\sum_i^{n_A}\sum_j^{n_B} C_{6,ref}^{AB}\left(CN_i^A, CN_j^B\right)\exp\left(x_{ij} - x_{max}\right)}{\sum_i^{n_A}\sum_j^{n_b}\exp\left(x_{ij} - x_{max}\right)} \tag{12}$$

By rescaling the exponents such that the largest term is $\exp(0) = 1$, this method guarantees a non-zero denominator and numerical stability even when using single-precision arithmetic.

3.5.2 Reducing floating-point precision loss

The accumulation of energy and forces in large systems involves summing tens of thousands of terms spanning many orders of magnitude due to the r^{-6} scaling. Adding small contributions from distant atoms to a large running total often leads to significant precision is lost due to rounding. To preserve accuracy, we implement a Kahan compensated summation algorithm. This method explicitly tracks the rounding error lost during each addition using a compensation term and re-adds it in the subsequent step. Furthermore, we adopt a batched accumulation strategy: terms are summed into smaller independent batches using Kahan summation before being aggregated. This hierarchical approach minimizes the magnitude disparity between the accumulator and the addend, ensuring high numerical accuracy and reproducibility even for large-scale systems.

4 Benchmark and discussion

We benchmarked our LASP-D3 implementation by evaluating its accuracy, computational speed, and GPU memory consumption. For comparison, we selected two existing

DFT-D3 projects: simple-dftd3[50] and tad-dftd3[46]. The simple-dftd3 project is a CPU-based implementation written in Fortran that utilizes the OpenMP framework for multi-threading, providing significant speedups on multicore CPUs. This implementation is widely adopted in various computational chemistry software packages, including Psi4[51], PySCF[52] and Siesta[53]. The tad-dftd3 project is a highly optimized implementation for both CPUs and GPUs, built on the well-established PyTorch framework. Its use of PyTorch allows it to achieve high computational efficiency, serving as an excellent benchmark for our CUDA implementation. All implementations were tested using zero damping with arguments for the PBE functional. The cutoff radius was set to 40 Bohr for CN calculation and 60 Bohr for energy and force calculations.

4.1 Accuracy

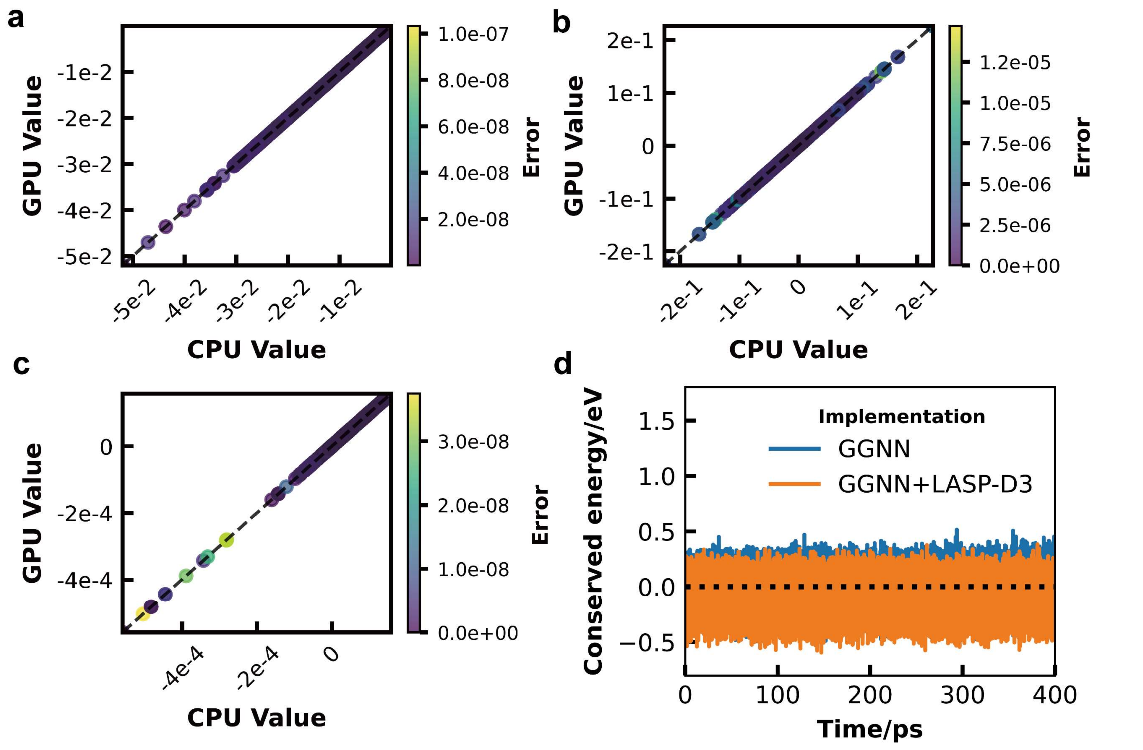

Figure 2. LASP-D3 (GPU value in figure) result of (a) energy value in Hartree, (b) force value in Hartree/Bohr and (c) stress value in Hartree/Bohr^3 in comparison with simple-dftd3 result (CPU value in figure) and (d) the conserved energy of MD simulation performed with GGNN and GGNN+LASP-D3, respectively.

To ensure the reliability of our implementation, we validated its accuracy against the double-precision simple-dftd3[50] CPU reference code using a diverse test set of 137,333 structures from the Crystal Open Database[54,55]. As shown in Figure 2a-c, the energies, forces, and stress tensors calculated with our single-precision GPU code are in excellent agreement with the double-precision CPU reference values. The maximum absolute errors observed across the entire test set are exceptionally low—on the order of 10^{-5} Hartree/Bohr for forces, 10^{-6} Hartree for energies, and 10^{-7} Hartree/Bohr^3 for stresses—which is sufficient for chemical precision. We also integrated LASP-D3 into the LASP project[48]. Using our HPNN-based MLP[15] with and without the LASP-D3 correction, we performed an MD simulation of a water droplet on a SiO_2 (110) facet at

298 K with a time step of 1 fs for a total duration of 400 ps. As shown in Figure 2d, energy conservation during the MD simulation was satisfied for both HPNN and HPNN+LASP-D3, with both exhibiting similar fluctuations in conserved energy. This demonstrates the correctness of our LASP-D3 implementation and confirms that the use of single precision does not compromise physical fidelity. These results confirm that our implementation is numerically robust and accurate for production-level simulations.

4.2 Memory consumption

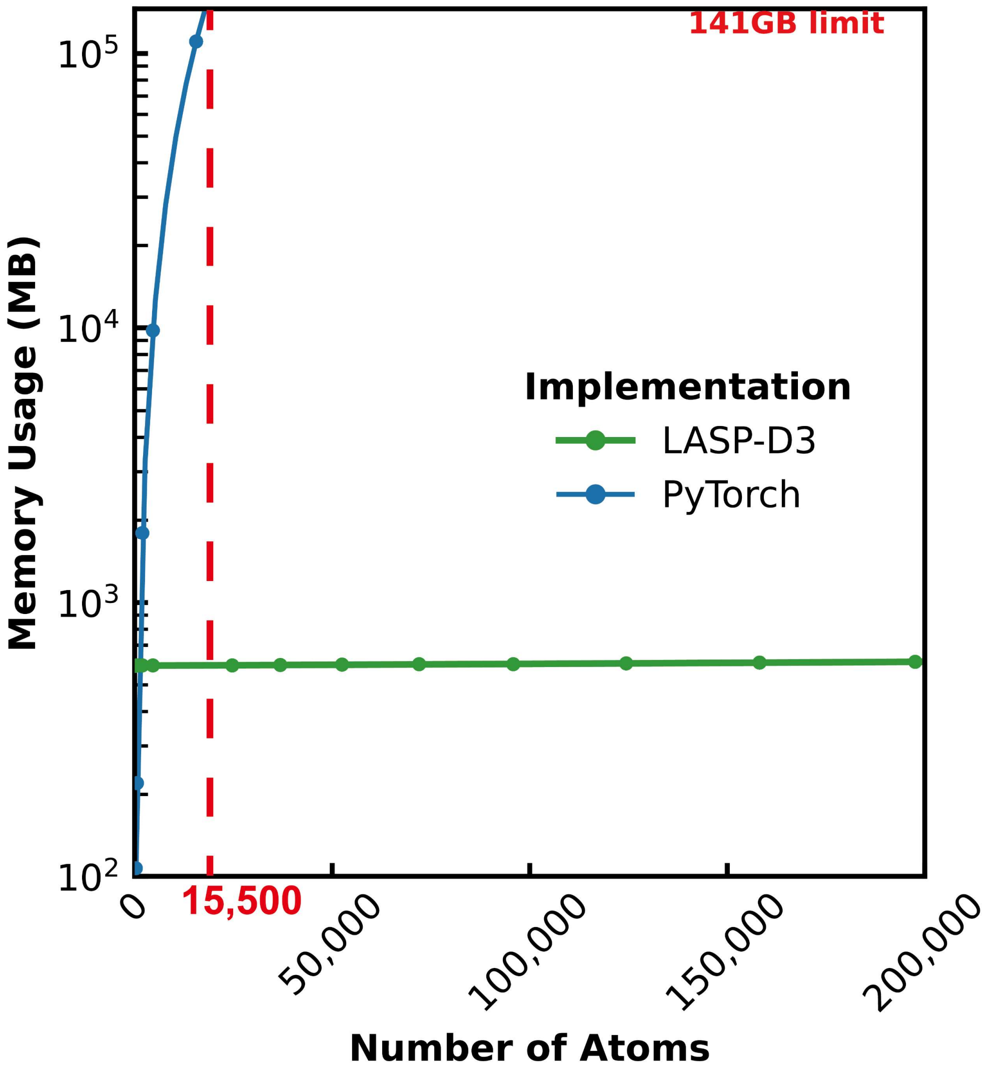

Figure 3. The memory consumption of two GPU implementations with respect to number of atoms. The green and blue dots represent benchmarked results of CUDA implementation and PyTorch implementation, respectively. The red tiled region means that the memory consumption exceeds Nvidia H200's 141GB limit. The memory consumption of both implementations is fitted using quadratic polynomial.

A critical challenge in GPU-based atomic simulation is memory consumption. GPU memory is restricted and has low expandability. For instance, a high-end Nvidia H200 GPU ships with only 141 GB of memory, which can seriously limit calculation scales. We benchmarked the GPU memory consumption of both tad-dftd3 and our LASP-D3 on an

Nvidia H200 GPU. The memory consumption for varying system sizes is plotted in Figure 3. The PyTorch-based implementation exhibits memory consumption that scales quadratically ($O(N^2)$) with the number of atoms. This is a direct consequence of its reliance on instantiating large intermediate tensors, such as the pairwise distance matrix. While straightforward from a programming perspective, this approach becomes prohibitive for larger systems, quickly exhausting available memory. This explains why tad-dftd3 failed to process systems larger than 10,000 atoms, even on a GPU with 141 GB of memory. In contrast, our LASP-D3 maintains a minimal memory footprint. Instead of explicitly storing large matrices, pairwise interactions are calculated on-the-fly within the GPU kernels. Only essential input atomic coordinates, output energies/forces, and necessary intermediate results are stored in global memory. These arrays' size scale linearly with system size. Consequently, the space complexity of our implementation is $O(N)$, which is significantly more manageable than the $O(N^2)$ complexity of tad-dftd3. Furthermore, the pre-factor for memory scaling is strictly controlled: for each additional atom, only 56 bytes of GPU memory are required. Taking the inherent ~500MB memory consumption into consideration, our implementation on a single Nvidia H200 GPU can handle massive systems potentially exceeding billions of atoms, far surpassing typical scales for atomic simulation. For standard systems with fewer than one million atoms, LASP-D3 occupies less than 1 GB of GPU memory, leaving the majority available for more memory-demanding MLPs. Thus, our CUDA implementation effectively removes the memory ceiling imposed by the PyTorch approach, enabling the treatment of significantly larger systems that would otherwise be impossible on current hardware.

4.3 Performance

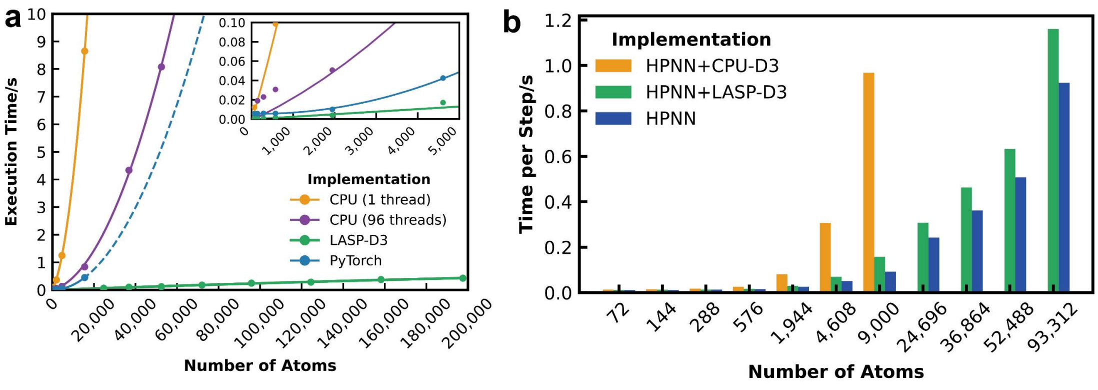

Figure 4 The execution time on varying system sizes of (a) single point DFT-D3 energy calculation by different implementations and (b) MD simulation by HPNN with different DFT-D3 implementations. The orange, purple, blue and green dots represent benchmarked results of single threaded simple-dftd3, multithreaded simple-dftd3, tad-dftd3 and LASP-D3, respectively. Corresponding lines represent quadratic polynomial fitting. The dotted blue line represents the extrapolation of PyTorch implementation execution time, which is unable to benchmark due to GPU memory limit. The blue, green and purple bar in (b) represent HPNN without D3 correction, with original CPU-based DFT-D3 implementation in LASP project and with LASP-D3, respectively. The absence of purple bars in systems with more than 20,000 atoms is caused by out of memory issue of CPU-D3 implementation.

To assess the efficiency of our CUDA-based DFT-D3 implementation, we performed a series of benchmark calculations on systems of increasing size, ranging from 72 to 197,568 atoms. The system used for benchmarking was a chloride solid electrolyte, Li_2ZrCl_6[56] in which dispersion interactions play a vital role. Comparison was made against simple-dftd3[50] and tad-dftd3[46]. The benchmarks were conducted on an AMD EPYC 9654 (96 cores) for the CPU implementation and a single Nvidia H200 (141 GB memory) for the GPU implementations. A warmup run was included for all implementations prior to the benchmark process.

The results are shown in Figure 4a. During our tests, tad-dftd3 failed to compute systems larger than 10,000 atoms due to out-of-memory (OOM) errors. The dotted blue line in Figure 4a represents an extrapolation based on its performance on sub-10,000 atom systems. The execution times for all implementations were fitted as quadratic polynomials. Among all implementations, our LASP-D3 demonstrates linear scaling of computation time with respect to the number of atoms in large systems, leading to superior performance compared to all competitors. This is a direct result of the utilization of cell list algorithms and is particularly advantageous for large-scale simulations. A detailed comparison of computation time for systems with fewer than 10,000 atoms is shown in the inset of Figure 4a. LASP-D3 outperforms both the PyTorch and CPU implementations even in small systems. Unlike tad-dftd3, where derivatives are acquired via automatic differentiation, LASP-D3 implements optimized analytical derivatives of the DFT-D3 energy. Atomic forces are directly calculated without additional overhead, maximizing efficiency and resulting in optimal performance for small systems.

We also performed an MD simulation of the Li_2ZrCl_6 electrolyte on a machine equipped with one AMD EPYC 9474F CPU (48 cores) and one Nvidia H200 GPU. The MD simulation was conducted in an NVT ensemble at 350 K using our HPNN MLP[15] for primary energy and force evaluation. LASP-D3 and the original CPU-based DFT-D3 implementation of our LASP project[48] were benchmarked respectively. In the MD simulation, LASP-D3 exhibited optimal performance, as shown in Figure 4b. Across varying system sizes, adding the LASP-D3 correction required only ~20% extra computation time. Conversely, the previous CPU-based D3 implementation, although executed in parallel with our GPU-bound MLP, became the bottleneck for large systems due to its higher time complexity. Additionally, the CPU-based D3 implementation exhausted the available system memory (750 GB) when calculating dispersion interactions for systems exceeding 20,000 atoms. In contrast, LASP-D3 maintains modest GPU memory consumption and can coexist with memory-intensive MLPs on the same device, even for systems as large as 93,312 atoms. This efficiency is the direct result of our dedicated optimization of both computing speed and memory usage.

5 Conclusion

We have developed LASP-D3, a high-performance, GPU-accelerated implementation of the DFT-D3 algorithm from scratch using CUDA C++. A specialized handle reuse mechanism significantly mitigates the overhead associated with frequent GPU memory reallocation, thereby improving efficiency during iterative tasks such as molecular dynamics (MD) simulations and geometry optimizations. Furthermore, robust algorithmic strategies were employed to ensure numerical accuracy and stability, even when utilizing single-precision arithmetic. Through rigorous optimization and the adoption of cell list algorithms, our implementation achieves linear time complexity ($O(N)$) for large periodic systems, consistently outperforming all benchmarked alternatives. In addition to speed, LASP-D3 exhibits exceptional memory efficiency. With memory consumption scaling linearly with system size, it minimizes the GPU footprint, thereby accommodating memory-intensive MLPs within the same device. This facilitates efficient calculations for systems containing tens of thousands of atoms, paving the way for real-world, large-scale simulations with atomistic precision. The implementation has been successfully integrated into the LASP software package[48] and the LASPAI web platform.

Acknowledgement

This work was supported by the National Key Research and Development Program of China (2024YFA1509600), the National Science Foundation of China (12188101, 22033003, 92472113), the Fundamental Research Funds for the Central Universities (20720250005, 20720220011), Science & Technology Commission of Shanghai Municipality (2024ZDSYS02), the robotic AI-Scientist platform of Chinese Academy of Science, and the Tencent Foundation for XPLORER PRIZE.

References

(1) Schütt, K. T.; Sauceda, H. E.; Kindermans, P.-J.; Tkatchenko, A.; Müller, K.-R. SchNet – A Deep Learning Architecture for Molecules and Materials. *J. Chem. Phys.* **2018**, *148* (24), 241722. https://doi.org/10.1063/1.5019779.

(2) Satorras, V. G.; Hoogeboom, E.; Welling, M. E(n) Equivariant Graph Neural Networks. In *Proceedings of the 38th International Conference on Machine Learning*; PMLR, 2021; pp 9323–9332.

(3) Schütt, K. T.; Unke, O. T.; Gastegger, M. Equivariant Message Passing for the Prediction of Tensorial Properties and Molecular Spectra. arXiv June 7, 2021. https://doi.org/10.48550/arXiv.2102.03150.

(4) Liao, Y.-L.; Smidt, T. Equiformer: Equivariant Graph Attention Transformer for 3D Atomistic Graphs. arXiv February 28, 2023. https://doi.org/10.48550/arXiv.2206.11990.

(5) Batzner, S.; Musaelian, A.; Sun, L.; Geiger, M.; Mailoa, J. P.; Kornbluth, M.; Molinari, N.; Smidt, T. E.; Kozinsky, B. E(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials. *Nat Commun* **2022**, *13* (1), 2453. https://doi.org/10.1038/s41467-022-29939-5.

(6) Musaelian, A.; Batzner, S.; Johansson, A.; Sun, L.; Owen, C. J.; Kornbluth, M.; Kozinsky, B. Learning Local Equivariant Representations for Large-Scale Atomistic Dynamics. *Nat Commun* **2023**, *14* (1), 579. https://doi.org/10.1038/s41467-023-36329-y.

(7) Batatia, I.; Kovacs, D. P.; Simm, G.; Ortner, C.; Csanyi, G. MACE: Higher Order Equivariant Message Passing Neural Networks for Fast and Accurate Force Fields. *Advances in Neural Information Processing Systems* **2022**, *35*, 11423–11436.

(8) Rupp, M.; Tkatchenko, A.; Müller, K.-R.; von Lilienfeld, O. A. Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning. *Phys. Rev. Lett.* **2012**, *108* (5), 058301. https://doi.org/10.1103/PhysRevLett.108.058301.

(9) Bartók, A. P.; Payne, M. C.; Kondor, R.; Csányi, G. Gaussian Approximation Potentials: The Accuracy of Quantum Mechanics, without the Electrons. *Phys. Rev. Lett.* **2010**, *104* (13), 136403. https://doi.org/10.1103/PhysRevLett.104.136403.

(10) Behler, J.; Parrinello, M. Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces. *Phys. Rev. Lett.* **2007**, *98* (14), 146401. https://doi.org/10.1103/PhysRevLett.98.146401.

(11) Bartók, A. P.; Kondor, R.; Csányi, G. On Representing Chemical Environments. *Phys. Rev. B* **2013**, *87* (18), 184115. https://doi.org/10.1103/PhysRevB.87.184115.

(12) Shi, Y.-F.; Yang, Z.-X.; Ma, S.; Kang, P.-L.; Shang, C.; Hu, P.; Liu, Z.-P. Machine Learning for Chemistry: Basics and Applications. *Engineering* **2023**, *27*, 70–83. https://doi.org/10.1016/j.eng.2023.04.013.

(13) Unke, O. T.; Chmiela, S.; Sauceda, H. E.; Gastegger, M.; Poltavsky, I.; Schütt, K. T.; Tkatchenko, A.; Müller, K.-R. Machine Learning Force Fields. *Chem. Rev.* **2021**, *121* (16), 10142–10186. https://doi.org/10.1021/acs.chemrev.0c01111.

(14) Zhang, Y.; Xia, J.; Jiang, B. Physically Motivated Recursively Embedded Atom Neural Networks: Incorporating Local Completeness and Nonlocality. *Phys. Rev. Lett.* **2021**, *127* (15), 156002. https://doi.org/10.1103/PhysRevLett.127.156002.

(15) Yang, Z.-X.; Xie, X.-T.; Wang, Z.-X.; Chen, D.-X.; Guo, Z.-X.; Du, J.-J.; Liang, Q.-M.; Liu, Q.-Y.; Shang, C.; Liu, Z.-P. High-Order Pair-Reduced Neural Network Architecture for Global Potential Energy Surface Exploration across the Periodic Table. *Sci. China Chem.* **2025**. https://doi.org/10.1007/s11426-025-3054-y.

(16) Becke, A. D. Perspective: Fifty Years of Density-Functional Theory in Chemical Physics. *J. Chem. Phys.* **2014**, *140* (18). https://doi.org/10.1063/1.4869598.

(17) Yang, Z.-X.; Xie, X.-T.; Kang, P.-L.; Wang, Z.-X.; Shang, C.; Liu, Z.-P. Many-Body Function Corrected Neural Network with Atomic Attention (MBNN-Att) for Molecular Property Prediction. *J. Chem. Theory Comput.* **2024**, *20* (15), 6717–6727. https://doi.org/10.1021/acs.jctc.4c00660.

(18) Xie, X.-T.; Guan, T.; Yang, Z.-X.; Shang, C.; Liu, Z.-P. Fine-Tuned Global Neural Network Potentials for Global Potential Energy Surface Exploration at High Accuracy. *J. Chem. Theory Comput.* **2025**, *21* (7), 3576–3586. https://doi.org/10.1021/acs.jctc.5c00051.

(19) Wang, H.; Guo, X.; Zhang, L.; Wang, H.; Xue, J. Deep Learning Inter-Atomic Potential Model for Accurate Irradiation Damage Simulations. *Applied Physics Letters* **2019**, *114* (24), 244101. https://doi.org/10.1063/1.5098061.

(20) Zhang, L.; Lin, D.-Y.; Wang, H.; Car, R.; E, W. Active Learning of Uniformly Accurate Inter-Atomic Potentials for Materials Simulation. *Phys. Rev. Materials* **2019**, *3* (2), 023804. https://doi.org/10.1103/PhysRevMaterials.3.023804.

(21) Deng, B.; Zhong, P.; Jun, K.; Riebesell, J.; Han, K.; Bartel, C. J.; Ceder, G. CHGNet as a Pretrained Universal Neural Network Potential for Charge-Informed Atomistic Modelling. *Nat Mach Intell* **2023**, *5* (9), 1031–1041. https://doi.org/10.1038/s42256-023-00716-3.

(22) Gastegger, M.; Behler, J.; Marquetand, P. Machine Learning Molecular Dynamics for the Simulation of Infrared Spectra. *Chemical Science* **2017**, *8* (10), 6924–6935. https://doi.org/10.1039/C7SC02267K.

(23) Eastman, P.; Galvelis, R.; Peláez, R. P.; Abreu, C. R. A.; Farr, S. E.; Gallicchio, E.; Gorenko, A.; Henry, M. M.; Hu, F.; Huang, J.; Krämer, A.; Michel, J.; Mitchell, J. A.; Pande, V. S.; Rodrigues, J. P.; Rodriguez-Guerra, J.; Simmonett, A. C.; Singh, S.; Swails, J.; Turner, P.; Wang, Y.; Zhang, I.; Chodera, J. D.; De Fabritiis, G.; Markland, T. E. OpenMM 8: Molecular Dynamics Simulation with Machine Learning Potentials. *J. Phys. Chem. B* **2024**, *128* (1), 109–116. https://doi.org/10.1021/acs.jpcb.3c06662.

(24) Li, Z.; Kermode, J. R.; De Vita, A. Molecular Dynamics with On-the-Fly Machine Learning of Quantum-Mechanical Forces. *Phys. Rev. Lett.* **2015**, *114* (9), 096405. https://doi.org/10.1103/PhysRevLett.114.096405.

(25) Huang, S.-D.; Shang, C.; Kang, P.-L.; Liu, Z.-P. Atomic Structure of Boron Resolved Using Machine Learning and Global Sampling. *Chem. Sci.* **2018**, *9* (46), 8644–8655. https://doi.org/10.1039/C8SC03427C.

(26) Kang, P.-L.; Shi, Y.-F.; Shang, C.; Liu, Z.-P. Artificial Intelligence Pathway Search to Resolve Catalytic Glycerol Hydrogenolysis Selectivity. *Chem. Sci.* **2022**, *13* (27), 8148–8160. https://doi.org/10.1039/D2SC02107B.

(27) Liu, Q.-Y.; Shang, C.; Liu, Z.-P. In Situ Active Site for CO Activation in Fe-Catalyzed Fischer–Tropsch Synthesis from Machine Learning. *J. Am. Chem. Soc.* **2021**, *143* (29), 11109–11120. https://doi.org/10.1021/jacs.1c04624.

(28) Pérez-Jordá, JoséM.; Becke, A. D. A Density-Functional Study of van Der Waals Forces: Rare Gas Diatomics. *Chemical Physics Letters* **1995**, *233* (1), 134–137. https://doi.org/10.1016/0009-2614(94)01402-H.

(29) Kristyán, S.; Pulay, P. Can (Semi)Local Density Functional Theory Account for the London Dispersion Forces? *Chemical Physics Letters* **1994**, *229* (3), 175–180. https://doi.org/10.1016/0009-2614(94)01027-7.

(30) Matthew D. Wodrich; Clémence Corminboeuf, and; Schleyer*, P. von R. *Systematic Errors in Computed Alkane Energies Using B3LYP and Other Popular DFT Functionals*. ACS Publications. https://doi.org/10.1021/ol061016i.

(31) Johnson, E. R.; DiLabio, G. A. Structure and Binding Energies in van Der Waals Dimers: Comparison between Density Functional Theory and Correlated Ab Initio Methods. *Chemical Physics Letters* **2006**, *419* (4), 333–339. https://doi.org/10.1016/j.cplett.2005.11.099.

(32) Grimme, S.; Hansen, A.; Brandenburg, J. G.; Bannwarth, C. Dispersion-Corrected Mean-Field Electronic Structure Methods. *Chem. Rev.* **2016**, *116* (9), 5105–5154. https://doi.org/10.1021/acs.chemrev.5b00533.

(33) Eisenschitz, R.; London, F. Über das Verhältnis der van der Waalsschen Kräfte zu den homöopolaren Bindungskräften. *Z. Physik* **1930**, *60* (7), 491–527. https://doi.org/10.1007/BF01341258.

(34) Civalleri, B.; Zicovich-Wilson, C. M.; Valenzano, L.; Ugliengo, P. B3LYP Augmented with an Empirical Dispersion Term (B3LYP-D*) as Applied to Molecular Crystals. *CrystEngComm* **2008**, *10* (4), 405–410. https://doi.org/10.1039/B715018K.

(35) Grimme, S.; Antony, J.; Schwabe, T.; Mück-Lichtenfeld, C. Density Functional Theory with Dispersion Corrections for Supramolecular Structures, Aggregates, and Complexes of (Bio)Organic Molecules. *Org. Biomol. Chem.* **2007**, *5* (5), 741–758. https://doi.org/10.1039/B615319B.

(36) Mahlberg, D.; Sakong, S.; Forster-Tonigold, K.; Groß, A. Improved DFT Adsorption Energies with Semiempirical Dispersion Corrections. *Journal of Chemical Theory and Computation* **2019**. https://doi.org/10.1021/acs.jctc.9b00035.

(37) Andersson, Y.; Langreth, D. C.; Lundqvist, B. I. Van Der Waals Interactions in Density-Functional Theory. *Phys. Rev. Lett.* **1996**, *76* (1), 102–105. https://doi.org/10.1103/PhysRevLett.76.102.

(38) Sato, T.; Tsuneda, T.; Hirao *, K. Van Der Waals Interactions Studied by Density Functional Theory. *Molecular Physics* **2005**, *103* (6–8), 1151–1164. https://doi.org/10.1080/00268970412331333474.

(39) von Lilienfeld, O. A.; Tavernelli, I.; Rothlisberger, U.; Sebastiani, D. Optimization of Effective Atom Centered Potentials for London Dispersion Forces in Density Functional Theory. *Phys. Rev. Lett.* **2004**, *93* (15), 153004. https://doi.org/10.1103/PhysRevLett.93.153004.

(40) Sun, Y. Y.; Kim, Y.-H.; Lee, K.; Zhang, S. B. Accurate and Efficient Calculation of van Der Waals Interactions within Density Functional Theory by Local Atomic Potential Approach. *J. Chem. Phys.* **2008**, *129* (15). https://doi.org/10.1063/1.2992078.

(41) Elstner, M.; Hobza, P.; Frauenheim, T.; Suhai, S.; Kaxiras, E. Hydrogen Bonding and Stacking Interactions of Nucleic Acid Base Pairs: A Density-Functional-Theory Based Treatment. *J. Chem. Phys.* **2001**, *114* (12), 5149–5155. https://doi.org/10.1063/1.1329889.

(42) Grimme, S. Accurate description of van der Waals complexes by density functional theory including empirical corrections. *Journal of Computational Chemistry* **2004**, *25* (12), 1463–1473. https://doi.org/10.1002/jcc.20078.

(43) Jurečka, P.; Černý, J.; Hobza, P.; Salahub, D. R. Density functional theory augmented with an empirical dispersion term. Interaction energies and geometries of 80 noncovalent complexes compared with ab initio quantum mechanics calculations. *Journal of Computational Chemistry* **2007**, *28* (2), 555–569. https://doi.org/10.1002/jcc.20570.

(44) Grimme, S.; Antony, J.; Ehrlich, S.; Krieg, H. A Consistent and Accurate Ab Initio Parametrization of Density Functional Dispersion Correction (DFT-D) for the 94 Elements H-Pu. *J. Chem. Phys.* **2010**, *132* (15), 154104. https://doi.org/10.1063/1.3382344.

(45) Goerigk, L. Chapter 6 - A Comprehensive Overview of the DFT-D3 London-Dispersion Correction. In *Non-Covalent Interactions in Quantum Chemistry and Physics*; Otero de la Roza, A., DiLabio, G. A., Eds.; Elsevier, 2017; pp 195–219. https://doi.org/10.1016/B978-0-12-809835-6.00007-4.

(46) Friede, M.; Hölzer, C.; Ehlert, S.; Grimme, S. *Dxtb* —An Efficient and Fully Differentiable Framework for Extended Tight-Binding. *J. Chem. Phys.* **2024**, *161* (6), 062501. https://doi.org/10.1063/5.0216715.

(47) Takamoto, S.; Shinagawa, C.; Motoki, D.; Nakago, K.; Li, W.; Kurata, I.; Watanabe, T.; Yayama, Y.; Iriguchi, H.; Asano, Y.; Onodera, T.; Ishii, T.; Kudo, T.; Ono, H.; Sawada, R.; Ishitani, R.; Ong, M.; Yamaguchi, T.; Kataoka, T.; Hayashi, A.; Charoenphakdee, N.; Ibuka, T. Towards Universal Neural Network Potential for Material Discovery Applicable to Arbitrary Combination of 45 Elements. *Nat Commun* **2022**, *13* (1), 2991. https://doi.org/10.1038/s41467-022-30687-9.

(48) Xie, X.-T.; Yang, Z.-X.; Chen, D.; Shi, Y.-F.; Kang, P.-L.; Ma, S.; Li, Y.-F.; Shang, C.; Liu, Z.-P. LASP to the Future of Atomic Simulation: Intelligence and Automation. *Precision Chemistry* **2024**, *2* (12), 612–627. https://doi.org/10.1021/prechem.4c00060.

(49) Grimme, S.; Ehrlich, S.; Goerigk, L. Effect of the Damping Function in Dispersion Corrected Density Functional Theory. *Journal of Computational Chemistry* **2011**, *32* (7), 1456–1465. https://doi.org/10.1002/jcc.21759.

(50) Ehlert, S. Simple DFT-D3: Library First Implementation of the D3dispersion Correction. *JOSS* **2024**, *9* (103), 7169. https://doi.org/10.21105/joss.07169.

(51) Smith, D. G. A.; Burns, L. A.; Simmonett, A. C.; Parrish, R. M.; Schieber, M. C.; Galvelis, R.; Kraus, P.; Kruse, H.; Di Remigio, R.; Alenaizan, A.; James, A. M.; Lehtola, S.; Misiewicz, J. P.; Scheurer, M.; Shaw, R. A.; Schriber, J. B.; Xie, Y.; Glick, Z. L.; Sirianni, D. A.; O'Brien, J. S.; Waldrop, J. M.; Kumar, A.; Hohenstein, E. G.; Pritchard, B. P.; Brooks,

B. R.; Schaefer, H. F., III; Sokolov, A. Yu.; Patkowski, K.; DePrince, A. E., III; Bozkaya, U.; King, R. A.; Evangelista, F. A.; Turney, J. M.; Crawford, T. D.; Sherrill, C. D. PSI4 1.4: Open-Source Software for High-Throughput Quantum Chemistry. *J. Chem. Phys.* **2020**, *152* (18), 184108. https://doi.org/10.1063/5.0006002.

(52) Sun, Q.; Zhang, X.; Banerjee, S.; Bao, P.; Barbry, M.; Blunt, N. S.; Bogdanov, N. A.; Booth, G. H.; Chen, J.; Cui, Z.-H.; Eriksen, J. J.; Gao, Y.; Guo, S.; Hermann, J.; Hermes, M. R.; Koh, K.; Koval, P.; Lehtola, S.; Li, Z.; Liu, J.; Mardirossian, N.; McClain, J. D.; Motta, M.; Mussard, B.; Pham, H. Q.; Pulkin, A.; Purwanto, W.; Robinson, P. J.; Ronca, E.; Sayfutyarova, E. R.; Scheurer, M.; Schurkus, H. F.; Smith, J. E. T.; Sun, C.; Sun, S.-N.; Upadhyay, S.; Wagner, L. K.; Wang, X.; White, A.; Whitfield, J. D.; Williamson, M. J.; Wouters, S.; Yang, J.; Yu, J. M.; Zhu, T.; Berkelbach, T. C.; Sharma, S.; Sokolov, A. Yu.; Chan, G. K.-L. Recent Developments in the P Y SCF Program Package. *The Journal of Chemical Physics* **2020**, *153* (2), 024109. https://doi.org/10.1063/5.0006074.

(53) García, A.; Papior, N.; Akhtar, A.; Artacho, E.; Blum, V.; Bosoni, E.; Brandimarte, P.; Brandbyge, M.; Cerdá, J. I.; Corsetti, F.; Cuadrado, R.; Dikan, V.; Ferrer, J.; Gale, J.; García-Fernández, P.; García-Suárez, V. M.; García, S.; Huhs, G.; Illera, S.; Korytár, R.; Koval, P.; Lebedeva, I.; Lin, L.; López-Tarifa, P.; Mayo, S. G.; Mohr, S.; Ordejón, P.; Postnikov, A.; Pouillon, Y.; Pruneda, M.; Robles, R.; Sánchez-Portal, D.; Soler, J. M.; Ullah, R.; Yu, V. W.; Junquera, J. S IESTA : Recent Developments and Applications. *The Journal of Chemical Physics* **2020**, *152* (20), 204108. https://doi.org/10.1063/5.0005077.

(54) Vaitkus, A.; Merkys, A.; Sander, T.; Quirós, M.; Thiessen, P. A.; Bolton, E. E.; Gražulis, S. A Workflow for Deriving Chemical Entities from Crystallographic Data and Its Application to the Crystallography Open Database. *Journal of Cheminformatics* **2023**, *15* (1), 123. https://doi.org/10.1186/s13321-023-00780-2.

(55) Merkys, A.; Vaitkus, A.; Grybauskas, A.; Konovalovas, A.; Quirós, M.; Gražulis, S. Graph Isomorphism-Based Algorithm for Cross-Checking Chemical and Crystallographic Descriptions. *Journal of Cheminformatics* **2023**, *15* (1), 25. https://doi.org/10.1186/s13321-023-00692-1.

(56) Wang, K.; Ren, Q.; Gu, Z.; Duan, C.; Wang, J.; Zhu, F.; Fu, Y.; Hao, J.; Zhu, J.; He, L.; Wang, C.-W.; Lu, Y.; Ma, J.; Ma, C. A Cost-Effective and Humidity-Tolerant Chloride Solid Electrolyte for Lithium Batteries. *Nat Commun* **2021**, *12* (1), 4410. https://doi.org/10.1038/s41467-021-24697-2.