

Linear-Scaling and Memory-Efficient Implementation of van-der-Waals Interaction (DFT-D3) for Large Systems

Han-Zhi Luo, Cheng Shang,* and Zhi-Pan Liu*



Cite This: <https://doi.org/10.1021/acs.jctc.6c00223>



Read Online

ACCESS |



Metrics & More



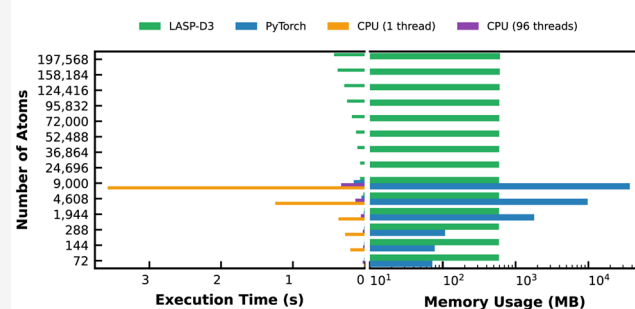
Article Recommendations



Supporting Information

ABSTRACT: The van der Waals (vdW) interaction is ubiquitous in materials and is long-range by nature. To facilitate vdW-included atomic simulations in large systems with tens of thousands of atoms, we developed LASP-D3, a Compute Unified Device Architecture (CUDA) implementation of the DFT-D3 method on graphics processing unit (GPU) devices, which realizes fast vdW corrections compatible with state-of-the-art machine-learning potential calculations. Our implementation achieves a linear-scaling time complexity, $O(N)$, for large periodic systems, being up to 2 orders of magnitude faster than all current versions for systems above 100,000 atoms, and significantly reduces GPU memory consumption compared to existing PyTorch-based GPU implementations. By combining LASP-D3 with the generalized global neural network potential developed by us, we show that the leading solid electrolyte LiTaCl₆ can achieve high conductivity, where the vdW interaction plays a key role in governing Li-ion diffusion and the simulated conductivity reproduces experimental measurements.

LASP-D3: GPU Based Linear Scaling DFT-D3 Implementation



1. INTRODUCTION

While quantum mechanics (QM), particularly density functional theory (DFT) calculations, has demonstrated its value and power in predicting material properties,¹ its high computational cost limits its application to large system simulations over long time scales. As a transformative technology,^{2–9} machine learning potentials (MLPs) overcome the size-accuracy trade-off, reproducing QM-level accuracy at a fraction of the cost^{10–14} for large systems with thousands of atoms. Recent MLPs rely on tensor algebra operations that align perfectly with graphics processing unit (GPU) architecture^{15–21} and thus undergo a paradigm shift toward GPU implementation, which would enable accurate molecular dynamics (MD) simulations^{22–24} and global exploration^{25–27} for unprecedented system sizes.

However, MLPs, by fitting DFT data sets, inherit 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 Attempts have been made to fit dispersion interaction using machine learning models,^{37,38} but they often come at the cost of increased architectural complexity and high computational overhead due to the long-range nature of the interaction. A more portable and robust alternative is directly applying corrections specifically designed for DFT functionals.^{39,40} Among various correction schemes,^{41–44} the DFT-D3 method,⁴⁵ the most popular Grimme's DFT-D method,^{46–48} is the de facto standard correction for this deficiency due to its robustness⁴⁹ and

efficiency.⁴⁵ While the computational cost of DFT-D3 is negligible compared to that of traditional DFT, it becomes a significant bottleneck when paired with fast MLP simulations. This issue is exacerbated by hardware mismatch: modern simulation platforms typically pair powerful GPUs with central processing units (CPUs) that have limited core counts. The addition of D3 corrections via available software implementations, regardless of CPU or GPU implementations, creates a severe bottleneck, negating the performance gains of the GPU-accelerated MLPs.

Recent attempts utilized GPU-accelerated libraries built on high-level frameworks (e.g., PyTorch⁵⁰) to speed up the D3 calculation, but the performance has fallen well short of expectations.^{51,52} While these implementations are indeed faster than earlier CPU-based versions, their time and memory complexity^{9,21} are still not compatible with modern linear-scaling MLP calculations, leaving the vdW correction alone as the rate-limiting step in atomic simulation. This could be due to the lack of direct and fine control over hardware resources in high-level GPU-accelerated frameworks, often reflected by

Received: February 5, 2026

Revised: April 1, 2026

Accepted: April 6, 2026

excessive memory consumption. The presence of the “memory capacity wall” prevents the simulation of super large-scale systems (e.g., >100,000 atoms) such as protein–ligand complexes or amorphous solids, which MLPs were purposely designed to address.

In this work, to match the highly efficient MLP simulations in our LASP software (www.lasphub.com),⁵³ we develop LASP-D3, a novel implementation of DFT-D3 entirely by Compute Unified Device Architecture (CUDA) C++. By engaging directly with GPU architecture without using high-level frameworks, 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 generalized global neural network (GGNN)⁹ potential simulations in LASP, we show that LASP-D3 maximally incurs a ~20% computational overhead in superlarge systems (100,000 atoms), drastically outperforming both CPU and PyTorch-based alternatives. Our LASP-D3 implementation is openly available and has a general interface that can be readily adapted to other MLPs. By removing the key barrier in hybridizing MLP and DFT-D3, LASP-D3 paves the way for accurate investigation of complex systems with appreciable vdW contributions, as demonstrated here in Li-ion diffusion of the LiTaCl₆ system.

2. DFT-D3 ALGORITHM

2.1. Energy Formula of D3

First, we overview the DFT-D3 algorithm,⁴⁵ a semiempirical correction designed to account for long-range vdW interactions, specifically dispersion forces, which are often inadequately described by standard density functionals. The correction involves adding a dispersion energy term, E_{disp} , to the standard Kohn–Sham DFT energy, $E_{\text{KS-DFT}}$.

$$E_{\text{total}} = E_{\text{KS-DFT}} - E_{\text{disp}} \quad (1)$$

The E_{disp} term consists of a two-body term and a three-body term, $E_{\text{disp}} = E^{(2)} + E^{(3)}$; the two-body term is computed by summing the interactions between all unique atom pairs (A and B) in the system, while the three-body term accumulates all distinct atomic triplets (A, B, and C) in the system.

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

$$E^{(3)} = \sum_{\text{ABC}} f_{d,(3)}(\bar{r}_{\text{ABC}}) E_{\text{ABC}} \quad (3)$$

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 $f_{d,n}(r_{\text{AB}})$ is the damping function that smoothly attenuates the dispersion energy at short distances. Since the three-body term significantly increases computational complexity but contributes little to the total energy (maximally ~5 meV/atom⁴⁵), this term is switched off by default in previous implementations^{45,54} and we will discuss our implementation of the three-body term in Supporting Information (SI). In the following, we will focus on the two-body term, considering the tiny contribution of the three-body term and our aim to match D3 calculation speed with that of GGNN utilized in the LASP code.

Various damping functions can be applied in eq 2, including zero damping⁴⁵

$$f_{d,n}(r_{\text{AB}}) = \frac{1}{1 + 6(r_{\text{AB}}/(s_n R_0^{\text{AB}}))^{\alpha_n}} \quad (4)$$

and Beche-Johnson damping⁵⁵

$$f_{d,n}(r_{\text{AB}}) = \frac{r_{\text{AB}}^n}{r_{\text{AB}}^n + (a_1 R_0^{\text{AB}} + a_2)^n} \quad (5)$$

Here, s_n , a_1 , and a_2 are dependent scaling factors specific to DFs, and 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 the coordination number (CN) for each atom.

$$\text{CN}^A = \sum_{B \neq A} \frac{1}{1 + \exp(-k_1(k_2(R_{A,\text{cov}} + R_{B,\text{cov}})/r_{\text{AB}} - 1))} \quad (6)$$

Here, $R_{A,\text{cov}}$ is the scaled covalent radius of atom A, r_{AB} is the distance between two atoms, and 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 and B) are known, the specific dispersion coefficient C_6^{AB} is calculated by interpolating over a grid of precomputed reference values.

$$C_6^{\text{AB}} = \frac{\sum_i^{n_A} \sum_j^{n_B} C_{6,\text{ref}}^{\text{AB}}(\text{CN}_i^A, \text{CN}_j^B) L_{ij}}{\sum_i^{n_A} \sum_j^{n_B} L_{ij}} \quad (7)$$

$$L_{ij} = \exp(-k_3((\text{CN}^A - \text{CN}_i^A)^2 + (\text{CN}^B - \text{CN}_j^B)^2)) \quad (8)$$

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 and B), this creates a 2D grid of up to 25 reference values $C_{6,\text{ref}}^{\text{AB}}(\text{CN}_i^A, \text{CN}_j^B)$. The final coefficient is obtained via a weighted average across this grid, using Gaussian functions as weights based on calculated CN^A and CN^B .

2.2. Cutoff Radius of vdW Interaction

When applying the DFT-D3 correction for large systems, including periodic solids, the pairwise summation must be extended to a long distance due to the relatively slow decay of the vdW interaction. For solids, this requires including the pairwise interactions between atoms and their surrounding periodic images. To avoid an infinite number of pairwise interactions, a long-distance cutoff scheme must be employed to make the calculation computationally feasible. It means that interaction between two atoms is considered only if the distance r_{AB} is within a specified cutoff radius. The DFT-D3 algorithm involves two distinct pairwise operations: the CN calculation and energy summation. Consequently, two independent cutoffs, namely, the CN cutoff and the energy cutoff, are defined. These two cutoffs can be adjusted independently. Although larger cutoffs yield more accurate results, the computational cost increases significantly with the scaling $O(R_{\text{cut}}^3)$ proportional to the volume of the cutoff sphere. The typical values for these cutoffs are therefore set as beyond 40 Bohr (~21 Å) for a balance between accuracy and efficiency.

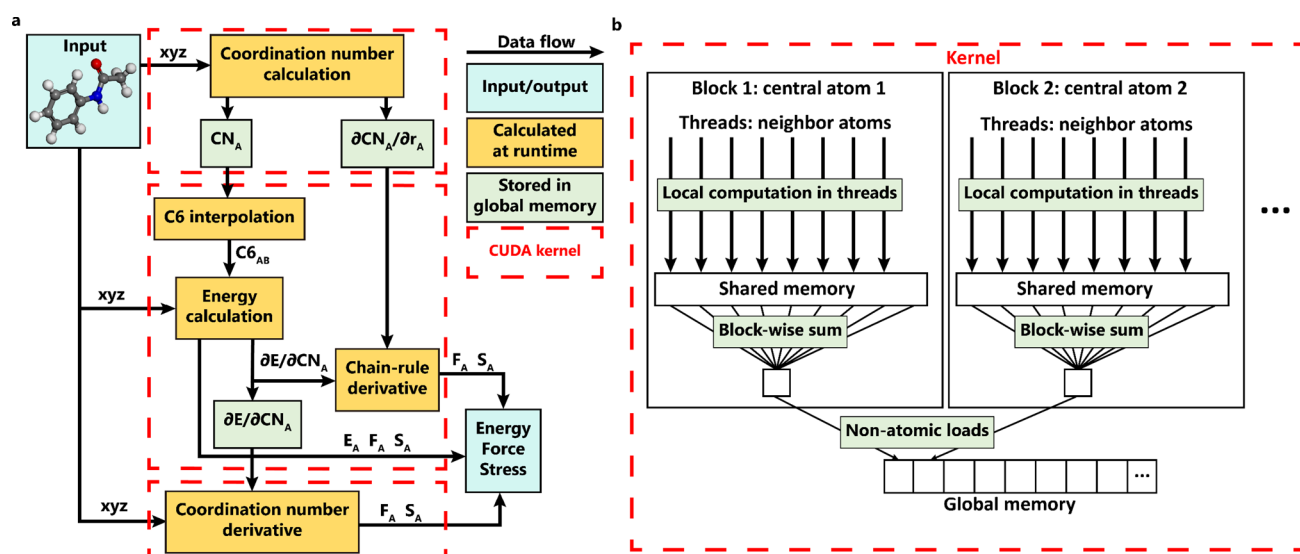


Figure 1. (a) Workflow of LASP-D3 energy, force, and stress calculation. Blue, yellow, and green blocks represent input/output data, runtime calculation, and data stored in global memory, respectively. Red dotted blocks represent GPU kernels. (b) Workload distribution across blocks and threads of LASP-D3 implementation. The number of arrows is not representative of actual number of threads.

2.3. Force Calculation

For applications such as geometry optimization and MD simulation, the analytic force acting on each atom needs to be calculated using energy derivatives, $F = \partial E / \partial r$. In the two-body term, the force can be calculated using eq 9, where the subscript A denotes atom A whose force is being calculated, and the subscripts B and C denote the neighbors of A.

$$F_A = \sum_B \sum_{n=6,8} \left(s_n C_n^{AB} \frac{\partial f_{d,n}}{\partial r_A} \frac{1}{r_{AB}^n} + s_n C_n^{AB} f_{d,n} \frac{\partial}{\partial r_A} \frac{1}{r_{AB}^n} \right) + \sum_{B,C} \sum_{n=6,8} s_n \frac{\partial C_n^{BC}}{\partial r_A} \frac{f_{d,n}}{r_{BC}^n} \quad (9)$$

The first two entries, the derivatives of $f_{d,n}$ and $1/r_{AB}^n$ can be easily calculated, as they are localized and are only relative to r_A and r_B . However, the third entry is more complex, as C_n^{AB} is a function of CN^A and CN^B , which in turn depend on the positions of all neighboring atoms of both A and B. This leads to complex force contributions. Applying the chain rule to the term, C_n^{AB} shows that a change in the position of atom A affects its own and its neighbors' CNs, which then alters the forces that its secondary neighbors experience. Therefore, the derivative of C_n^{AB} with respect to a specific atom A is written as eq 10.

$$\frac{\partial C_n^{BC}(CN_B, CN_C)}{\partial r_A} = \frac{\partial C_n^{BC}}{\partial CN_B} \frac{\partial CN_B}{\partial r_A} + \frac{\partial C_n^{BC}}{\partial CN_C} \frac{\partial CN_C}{\partial r_A} \quad (10)$$

Applying eq 10 to the last entry of eq 9 and simplifying the dual accumulation yields eq 11, the first two entries and $\partial E / \partial CN_A$ can be calculated along with the energy calculation; the last entry can be added after acquiring the $\partial E / \partial CN_A$ values using a separate loop.

$$F_A = \sum_B \sum_{n=6,8} \left(s_n C_n^{AB} \frac{\partial f_{d,n}}{\partial r_A} \frac{1}{r_{AB}^n} + s_n C_n^{AB} f_{d,n} \frac{\partial}{\partial r_A} \frac{1}{r_{AB}^n} \right) + \sum_B \sum_{n=6,8} \left(\left(\frac{\partial E}{\partial CN_A} + \frac{\partial E}{\partial CN_B} \right) \frac{\partial CN_{AB}}{\partial r_A} \right) \quad (11)$$

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 employ C++ Resource Acquisition Is Initialisation (RAII) patterns to create a maintainable, leak-free memory management system, ensuring stability during demanding, long-running simulations. Furthermore, a custom memory reuse mechanism minimizes allocation overhead, reducing the memory bandwidth requirements. We leverage the massive parallelism of CUDA and optimized numeric algorithms to accelerate computationally intensive tasks while minimizing GPU memory consumption, achieving linear scaling ($O(N)$) of computation time and memory consumption with respect to system size, enhancing both numeric accuracy and stability.

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 handle object to manage the structure configuration and the lifecycle of GPU memory. Following the C++ 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 S1 in the Supporting Information (SI) shows a typical MD workflow using MLP potential and LASP-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 (provided by CUDA API) rather than reallocating GPU resources. This reuse mechanism significantly reduces memory bandwidth overhead and maximizes throughput for long-time scale simulations.

Based on the C-compatible API, we also provide Fortran and Python wrappers to facilitate seamless integration with existing models and modern simulation engines. To demonstrate practical utility, we provide an example integrating the MACE potential²¹ with LASP-D3 corrections within the Atomic Simulation Environment (ASE) Python framework.⁵⁶ This implementation, available publicly on GitHub, enables the execution of typical atomic simulation tasks, such as MD.

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 are executed independently in parallel. Within blocks, threads serve as the elementary execution units, communicating via block-wise shared memory, which have restricted size (typically in kilobytes) but with high speed. Global memory, although slow compared with shared memory, has a much larger size and serves as a persisting memory across kernels, enabling kernels to store temporary results. It is also the generally recognized GPU memory. This hierarchical structure allows for fine-grained control over the data locality and parallel execution.

3.3. CUDA Kernels

LASP-D3 accepts atomic coordinates, element types, and lattice vectors as input, returning the total dispersion energy, atomic forces, and stress tensor. The calculation follows a multistage kernel pipeline (Figure 1a), managed via sequential launches in a single CUDA stream. The first kernel computes CN_A and its derivatives $\partial CN/\partial r_A$ based on eq 6. Each block calculates one floating-point variable for CN_A , and three floating-point variables for $\partial CN/\partial r_A$, which are stored in global memory. The second kernel extracts CN_A and $\partial CN/\partial r_A$ stored by the former kernel, performs the C6 interpolation using eq 7, and then calculates the dispersion energy using eq 2. It also calculates part of the force contributions derived from the first two entries in eq 9. The results are also stored in global memory. This results in an additional four floating points stored in global memory for each atom. The last kernel follows a workflow similar to that of the first kernel, but instead of accumulating all $\partial CN_{AB}/\partial r_A$, it assembles the last entry in eq 11 using the $\partial E/\partial CN_A$ value extracted from global memory. Notably, we explicitly recalculate CN derivatives rather than storing them in global memory. 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 $O(N)$ memory complexity, avoiding $O(N^2)$ scaling associated with storing pairwise derivative matrices, thereby enabling the simulation of significantly larger systems.

As shown in Figure 1b, we launch one block of threads for each central atom A and threads inside the block cooperate to iterate over all of its neighbors. At the end of each kernel, block-wise reductions accumulate local variables, and the first thread in each block stores the result to global memory. This construction guarantees that each block of global memory is only accessed by 1 thread, minimizing the need for time-consuming atomic operations while preventing racing conditions.

Among the three kernels, five floating points are stored in global memory for each atom as intermediate results (one for

CN_A , one for $\partial E/\partial CN_A$, and three for $\partial CN_A/\partial r_A$). Taking input variables (two short integers for atom types, three floating points for r_A , and two long integers to identify the atom's location when using cell list optimization) and output variables (four floating points, one for energy, and three for atomic force), LASP-D3 requires 56 bytes of memory for each atom.

Pseudocode 1. Cell list algorithm to compute atom-wise D3 contribution

```

1 // 1. Grid Partitioning
2 Init grids
3
4 // 2. Binning, complexity: O(N)
5 For atom_i in atoms do
6   Identify grid of atom_i
7   Bind atom_i to grids
8 End For
9
10 // 3. CalculateD3, complexity:
11 O(N)
12 For atom_i in atoms do
13   For atom_j in neighbor_grids do
14     CalculateD3(atom_i, atom_j)
15   End For
16 End For

```

3.4. Linear Scaling via Cell List Algorithm

In large periodic systems, many atom pair distances are beyond the cutoff radius, thus being ignored. Standard DFT-D3 implementations often iterate over all atom pairs, leading to $O(N^2)$ time complexity and wasting computational cycles iterating over distant, noninteracting atoms. To overcome this limitation and ensure linear scaling ($O(N)$) for large periodic systems, our implementation partitions the simulation box into a grid of cells with dimensions determined by the cutoff radius. As shown in Pseudocode 1, atoms are binned into these cells, and the dispersion interaction calculation is restricted to atoms within the same or immediately neighboring cells. This spatial locality optimization ensures that LASP-D3 remains efficient for massive systems and matches the linear scaling characteristics of the underlying GGNN-based MLPs.⁹

Pseudocode 2. Hierarchical Kahan accumulation algorithm

```

1 For i = 1 to N do
2   // 1. Perform Kahan summation
3   y=xi-compensation
4   t=base_sum+y
5   compensation=(t-base_sum)-y
6   base_sum=t
7   // 2. Accumulate to high levels
8   num_addends+=1
9   if num_addends%STRIDE==0 then
10    levelled_sum+=base_sum
11    base_sum=0
12  End If
13 End For
14
15 // Return the accumulated value
16 Return (base sum + levelled sum)

```

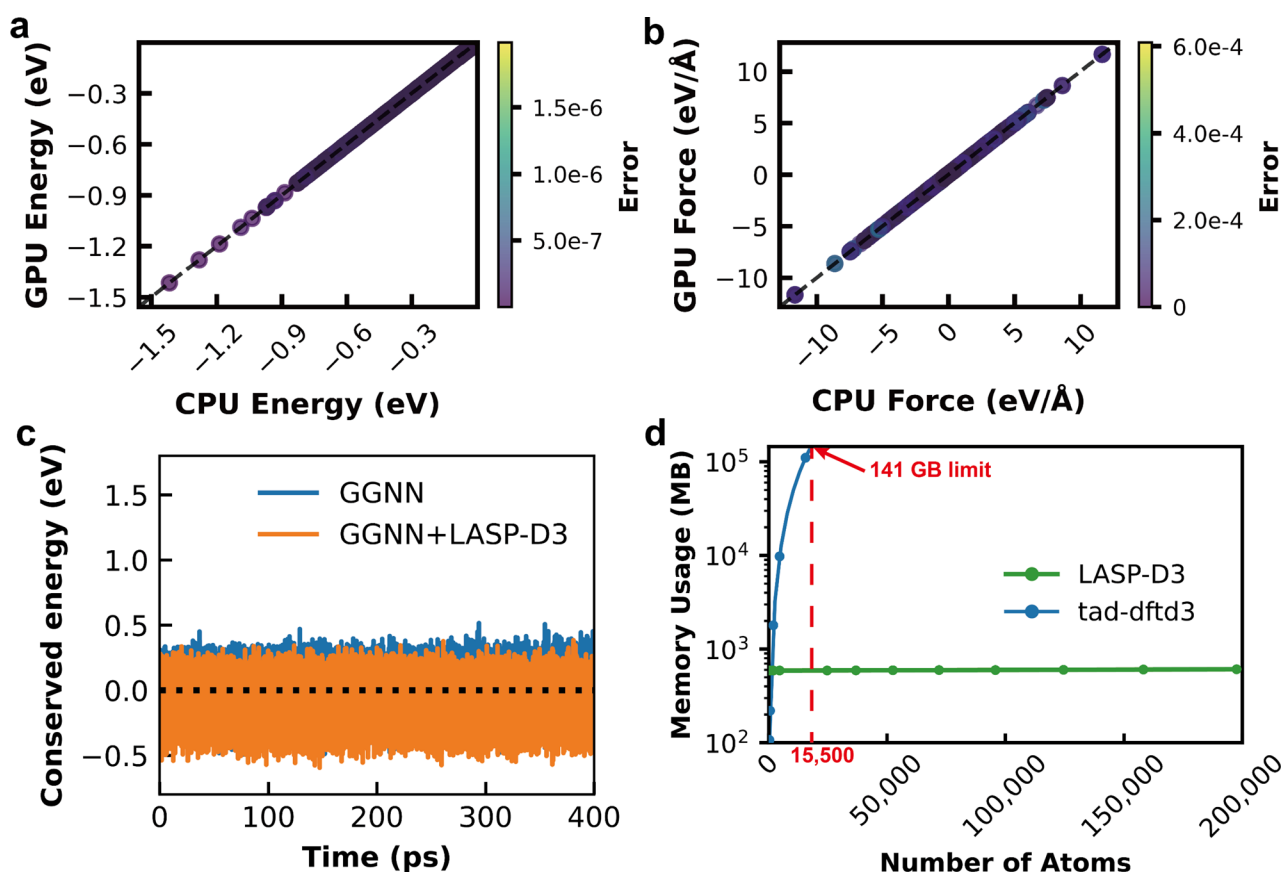


Figure 2. LASP-D3 (GPU values in figure) results of (a) atomic energy value in eV and (b) force value in eV/Å in comparison with simple-dftd3 result (CPU values in figure). (c) is the conserved energy of MD simulation performed with GGNN and GGNN+LASP-D3, respectively, and (d) is the memory consumption of LASP-D3 in comparison with PyTorch-based tad-dftd3.

3.5. Numerical Stability Improvements

Beyond performance, we introduced two critical algorithmic enhancements to address floating-point arithmetic pitfalls, specifically optimizing 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 eq 7 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, then the denominator vanishes, causing division-by-zero errors. To resolve this, we employ the log-sum-exp stabilization technique. We first compute the logarithmic arguments using eq 12.

$$x_{ij} = -k_3((CN^A - CN_i^A)^2 + (CN^B - CN_j^B)^2) \quad (12)$$

We then identify the maximum argument, $x_{\max} = \max(x_{ij})$, and compute the final sum by shifting the exponents using eq 13.

$$C_6^{AB} = \frac{\sum_i^{n_A} \sum_j^{n_B} C_{6,\text{ref}}^{AB} (CN_i^A, CN_j^B) \exp(x_{ij} - x_{\max})}{\sum_i^{n_A} \sum_j^{n_B} \exp(x_{ij} - x_{\max})} \quad (13)$$

By rescaling the exponents such that the largest term is $\exp(0) = 1$, this method guarantees a nonzero 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 loss due to rounding. To preserve accuracy, we implement a Kahan compensated summation algorithm.⁵⁷ This algorithm is described in the first part of Pseudocode 2. We use a temporary variable (variable t in the pseudocode) to store the result of the direct accumulation. Then, we calculate and track the lost parts of this addend (variable *compensation* in the pseudocode), which will be added in the following accumulation. Furthermore, we adopt a batched accumulation strategy: terms are summed into smaller independent batches using Kahan summation before being aggregated. This mixed approach minimizes the magnitude disparity between the accumulator and the addends, ensuring high numerical accuracy and reproducibility, even for large-scale systems.

4. PERFORMANCE BENCHMARKS

We benchmarked our LASP-D3 implementation by evaluating its accuracy, memory consumption, and computational speed. For comparison, we selected two existing DFT-D3 projects: simple-dftd3⁵⁴ and tad-dftd3.⁵¹ The simple-dftd3 project is a CPU-based implementation written in Fortran and utilizes the OpenMP framework⁵⁸ for multithreading, providing significant

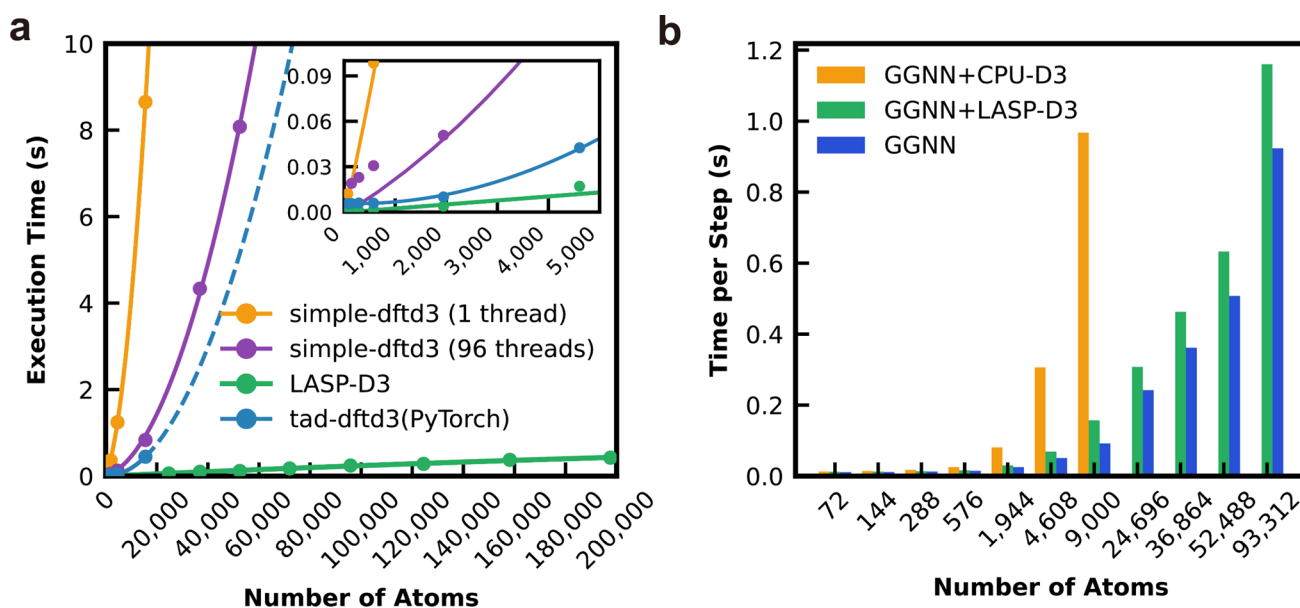


Figure 3. (a) Execution time on varying system sizes of the single-point DFT-D3 energy calculation by different implementations. The orange, purple, blue and green dots and lines represent benchmarked results of single threaded simple-dftd3, multithreaded simple-dftd3, tad-dftd3, and LASP-D3, respectively. The dotted blue line represents the extrapolation of tad-dftd3 execution time, which is unable to benchmark due to GPU memory limit. The inset shows detailed execution time of sub-5000 systems. (b) MD simulation speed by GGNN with different DFT-D3 implementations. The blue, green, and purple bars represent GGNN without D3 correction, with original CPU-based 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 the out-of-memory issue of CPU-D3 implementation.

speedups on multicore CPUs. This implementation is widely adopted in various computational chemistry software packages, including Psi4,⁵⁹ PySCF⁶⁰ and Siesta.⁶¹ The tad-dftd3 project is a highly optimized implementation for both CPUs and GPUs, built on the well-established PyTorch framework.⁵⁰ Its use of PyTorch allows it to achieve a high computational efficiency, serving as an excellent baseline 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 calculations and 60 Bohr for energy and force calculations.

4.1. Accuracy

To ensure the reliability of our implementation, we validated its accuracy against the double-precision simple-dftd3⁵⁴ CPU reference code using a diverse test set of 137,333 structures from the Crystal Open Database.^{62,63} As shown in Figure 2a and b, the energies and forces calculated with our single-precision GPU code are in excellent agreement with the double-precision CPU reference values. We also validated the accuracy of the three-body term, but due to its high computational cost, we randomly sampled 30,770 structures and performed three-body validation on the reduced test set. The test result after adding the three-body term is shown in Figure S1. The maximum errors observed across the entire test set are exceptionally low—on the order of 10^{-4} eV/Å (10^{-5} Hartree/Bohr) for forces and 10^{-6} eV (10^{-8} Hartree) per atom for energies, which is sufficient for chemical precision. To verify that these discrepancies are attributed to the precision limits of single-precision floating-point arithmetic, we benchmarked LASP-D3 on the same structures using double precision. The result is shown in Figure S2; the maximum errors diminish by several orders of magnitude to 10^{-12} eV/Å (10^{-14} Hartree/Bohr) for forces and 10^{-14} eV (10^{-15} Hartree) per atom for energy. This reduction confirms the algorithmic correctness of our implementation, demonstrat-

ing that the deviations in the standard build are purely numerical artifacts of 32-bit accumulation.

We also integrated LASP-D3 into the LASP project.⁵³ The GGNN potential⁹ employed in this work was trained using the GGA-PBE functional. Consequently, it inherits the well-known limitations of this functional in capturing long-range dispersion interactions. A brief description of the model parameters and the training data set is provided in the Supporting Information. Using our GGNN potential with and without the LASP-D3 correction, we performed an MD simulation of a water droplet on a SiO₂ (110) facet consisting of a total of 8,541 atoms at 298 K with a time step of 1 fs for a total duration of 400 ps. As shown in Figure 2c, energy conservation during the MD simulation was satisfied for both GGNN and GGNN+LASP-D3, with both exhibiting similar fluctuations in the conserved energy (0.0045 meV/atom). This demonstrates the correctness of our LASP-D3 implementation and confirms that the use of single precision does not compromise physical fidelity.

4.2. Memory Consumption

A critical challenge in GPU-based atomic simulations 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-dftd3⁵¹ and LASP-D3 on an NVIDIA H200 GPU. The memory consumption for varying system sizes is plotted in Figure 2d. PyTorch-based tad-dftd3 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 the available memory. This explains why tad-dftd3 failed to process systems larger than 15,500 atoms, even on a GPU with 141 GB of memory. In contrast,

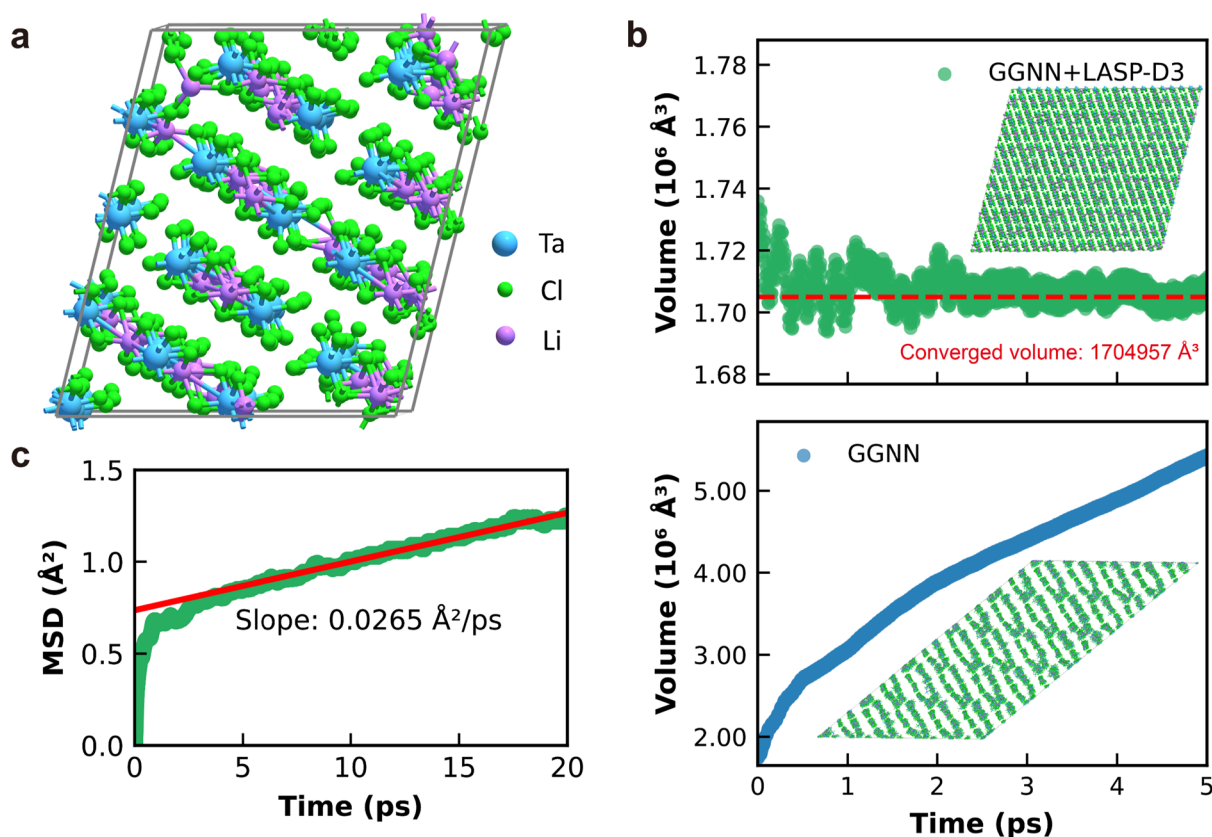


Figure 4. (a) Structure of the LiTaCl_6 electrolyte and (b) the cell volume with respect to time during the NPT MD simulation by GGNN+LASP-D3 (top) and GGNN potentials (bottom), respectively. The insets show the structure at the end of MD trajectories. (c) MSD of Li ions with respect to time during NVT MD simulation. Purple, green, and blue spheres represent Li, Cl, and Ta atoms, respectively.

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/output data and necessary intermediate results are stored in global memory. These arrays' size scales 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 prefactor for memory scaling is strictly controlled: for each additional atom, only 56 bytes of GPU memory are required. Taking the inherent ~ 500 MB 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, LASP-D3 effectively removes the memory ceiling imposed by the PyTorch approach, enabling the treatment of significantly larger systems that would otherwise be impossible on the current hardware.

4.3. Speed Scaling

To assess the efficiency of our CUDA-based D3 implementation, we performed a series of benchmark calculations on systems of increasing size, ranging from 72 to 197,568 atoms. Comparison was made against *simple-dftd3*⁵⁴ and *tad-dftd3*.⁵¹ The benchmarks were conducted on an AMD EPYC 9654 (96 cores) for CPU implementation (*simple-dftd3*) and an NVIDIA H200 for the GPU implementations (*tad-dftd3* and LASP-D3). A warm-up run was included for all implementations prior to the benchmark process.

The results are shown in Figure 3a. In our tests, *tad-dftd3* failed to compute systems larger than 15,000 atoms due to out-of-memory (OOM) errors. The dotted blue line in Figure 3a represents an extrapolation based on its performance on sub-15,000-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 system size, 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 5000 atoms is shown in the inset of Figure 3a, showing that LASP-D3 achieves optimal performance even in small systems. Utilizing the high computation throughput of the GPU, LASP-D3 achieves much higher performance compared to a multithreaded *simple-dftd3* implementation. Unlike LASP-D3, *tad-dftd3* relies on the PyTorch framework, leading to higher overhead, which can be seen from the higher intercept with the y-axis. LASP-D3, on the other hand, requires minimal preparation, including memory allocation and initialization, which is further replaced by a simple memory copy call when the MD simulation is performed, resulting in a minimal launching overhead.

We also performed an MD simulation of Li_2ZrCl_6 electrolyte⁶⁴ on a machine equipped with one AMD EPYC 9474F CPU (48 cores) and one NVIDIA H200 GPU. The MD simulation was conducted in an NPT ensemble at 350 K using our GGNN potential⁹ for primary energy and force evaluation. LASP-D3 and the original CPU-based DFT-D3 implementation of our

LASP project⁵³ were benchmarked, respectively. As shown in Figure 3b, LASP-D3 exhibited optimal performance in the MD simulation workload. Across varying system sizes, adding the LASP-D3 correction required only ~20% extra computation time. Conversely, the previous CPU-based DFT-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 GPU, 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. EXAMPLE: THE CONDUCTIVITY OF LiTaCl_6 SOLID ELECTROLYTE

To validate the performance of LASP-D3 under realistic simulation workloads, we investigated the ionic conductivity of LiTaCl_6 ,⁶⁵ a promising solid electrolyte emerged in recent years. As illustrated in Figure 4a, for the atomic structure, the LiTaCl_6 lattice features a 1D-channel-like framework composed of TaCl_6 octahedra with lithium ions dispersed within the interstitial voids. To capture long-range diffusion pathways and minimize finite-size effects, we constructed a large supercell containing 64,000 atoms for MD simulation, 8000 of which are Li ions. We first performed MD simulations in the NPT ensemble at 300 K and 1 atm for 10 ps to equilibrate the cell volume using the GGNN potential. Independent simulations with and without LASP-D3 dispersion correction are conducted for comparison.

Figure 4b depicts the cell volume evolutions for the D3-corrected (top panel) and uncorrected (bottom panel) simulations along the MD trajectory, respectively. With the LASP-D3 correction, the system maintained structural integrity, and the cell volume rapidly converged and oscillated within a narrow range (~0.5%) of the average value ($1,704,957 \text{ \AA}^3$). This can be confirmed by the structure in the inset at the top panel of Figure 4b. In sharp contrast, the simulation without D3 correction exhibited a continuous expansion in cell volume, indicative of structural disintegration. This failure, visually confirmed by the disordered final structure shown in the corresponding inset, highlights the critical role of dispersion forces in maintaining the interchannel cohesion of the TaCl_6 framework. After the equilibration of the D3-corrected simulation, we further performed a production run in the isothermal (NVT) ensemble at 300 K for 20 ps. The mean squared displacement (MSD) of the Li ions (Figure 4c) exhibits a linear regime with a slope of $0.0265 \text{ \AA}^2/\text{ps}$. By applying the Nernst–Einstein equation, we yield a calculated ionic conductivity of 12.84 mS/cm , in good agreement with the experimental value of 10.95 mS/cm ,⁶⁵ confirming the very high conductivity of LiTaCl_6 as the solid electrolyte. Our results demonstrate the ability to model large-scale systems and simultaneously reproduce accurately both structural and physicochemical properties that rely critically on both the correctness of the physical model (e.g., vdW interaction) and the high computational efficiency, which can now be conveniently achieved via the seamless joint of GGNN potential and LASP-D3.

6. CONCLUSIONS

This work developed a key gradient for modern atomistic simulations, namely, LASP-D3, a high-performance, GPU-accelerated implementation of the DFT-D3 algorithm using CUDA C++ to be utilized together with state-of-the-art MLP PES calculations. We design a specialized object-oriented design and memory reuse mechanism to maximally mitigate the overhead associated with frequent GPU memory reallocation, thereby improving the efficiency during iterative tasks such as MD simulations and geometry optimizations. Furthermore, robust algorithmic strategies are employed to ensure numerical accuracy and stability, particularly in single-precision arithmetic. We show that LASP-D3 achieves linear time complexity ($O(N)$) for large systems through rigorous optimization and the adoption of cell list algorithms, consistently outperforming all benchmarked alternatives. In addition, LASP-D3 also exhibits exceptional memory efficiency, with the memory consumption scaling linearly with system size, facilitating memory-intensive MLPs within the same GPU device. Taking Li-ion diffusion in LiTaCl_6 solid electrolyte as an example, we demonstrate that LASP-D3 implementation facilitates long-time atomistic simulations for systems containing tens of thousands of atoms with high precision. The implementation has been successfully integrated into the LASP software package⁵³ and the LASPAI material design platform (www.laspai.com),⁶⁶ and is now publicly available.

■ ASSOCIATED CONTENT

Data Availability Statement

The source code of LASP-D3 is publicly available at <https://github.com/LipidL/LASP-D3>. Instructions for its use are provided on the same webpage.

Supporting Information

The Supporting Information is available free of charge at <https://pubs.acs.org/doi/10.1021/acs.jctc.6c00223>.

Pseudocode for using LASP-D3 in a handle-oriented way during MD simulation; three-body term of DFT-D3; accuracy test for three-body term; and accuracy test using double precision (PDF)

■ AUTHOR INFORMATION

Corresponding Authors

Cheng Shang – State Key Laboratory of Porous Materials for Separation and Conversion, Collaborative Innovation Center of Chemistry for Energy Materials, Shanghai Key Laboratory of Molecular Catalysis and Innovative Materials, Key Laboratory of Computational Physical Science, Department of Chemistry, Fudan University, Shanghai 200433, China; orcid.org/0000-0001-7486-1514; Email: cshang@fudan.edu.cn

Zhi-Pan Liu – State Key Laboratory of Porous Materials for Separation and Conversion, Collaborative Innovation Center of Chemistry for Energy Materials, Shanghai Key Laboratory of Molecular Catalysis and Innovative Materials, Key Laboratory of Computational Physical Science, Department of Chemistry, Fudan University, Shanghai 200433, China; State Key Laboratory of Metal Organic Chemistry, Shanghai Institute of Organic Chemistry, Chinese Academy of Sciences, Shanghai 200032, China; orcid.org/0000-0002-2906-5217; Email: zpliu@fudan.edu.cn

Author

Han-Zhi Luo – State Key Laboratory of Porous Materials for Separation and Conversion, Collaborative Innovation Center of Chemistry for Energy Materials, Shanghai Key Laboratory of Molecular Catalysis and Innovative Materials, Key Laboratory of Computational Physical Science, Department of Chemistry, Fudan University, Shanghai 200433, China

Complete contact information is available at:
<https://pubs.acs.org/10.1021/acs.jctc.6c00223>

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Program of China (2024YFA1509600 and 2025YFA1510900), the National Science Foundation of China (12188101, U25B6005, 22033003, and 92472113), the Fundamental Research Funds for the Central Universities (20720250005 and 20720220011), Science and Technology Commission of Shanghai Municipality (2024ZDSYS02), and the Discipline Breakthrough Pilot Project of the Ministry of Education of China (JYB2025XDXM307).

REFERENCES

- (1) Becke, A. D. Perspective: Fifty Years of Density-Functional Theory in Chemical Physics. *J. Chem. Phys.* **2014**, *140* (18), No. 18A301.
- (2) 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), No. 058301.
- (3) 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), No. 136403.
- (4) Behler, J.; Parrinello, M. Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces. *Phys. Rev. Lett.* **2007**, *98* (14), No. 146401.
- (5) Bartók, A. P.; Kondor, R.; Csányi, G. On Representing Chemical Environments. *Phys. Rev. B* **2013**, *87* (18), No. 184115.
- (6) 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.
- (7) 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.
- (8) Zhang, Y.; Xia, J.; Jiang, B. Physically Motivated Recursively Embedded Atom Neural Networks: Incorporating Local Completeness and Nonlocality. *Phys. Rev. Lett.* **2021**, *127* (15), No. 156002.
- (9) 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**.
- (10) 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.
- (11) 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.
- (12) Wang, H.; Guo, X.; Zhang, L.; Wang, H.; Xue, J. Deep Learning Inter-Atomic Potential Model for Accurate Irradiation Damage Simulations. *Appl. Phys. Lett.* **2019**, *114* (24), No. 244101.
- (13) 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), No. 023804.
- (14) 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.
- (15) 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), No. 241722.
- (16) 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.
- (17) Schütt, K. T.; Unke, O. T.; Gastegger, M. Equivariant Message Passing for the Prediction of Tensorial Properties and Molecular Spectra. arXiv 2021. .
- (18) Liao, Y.-L.; Smidt, T. Equiformer: Equivariant Graph Attention Transformer for 3D Atomistic Graphs. arXiv 2023. .
- (19) 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.
- (20) 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.
- (21) 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. In *Advances in Neural Information Processing Systems*; 2022; Vol. 35, pp 11423–11436.
- (22) Gastegger, M.; Behler, J.; Marquetand, P. Machine Learning Molecular Dynamics for the Simulation of Infrared Spectra. *Chemical Science* **2017**, *8* (10), 6924–6935.
- (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.
- (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), No. 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.
- (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.
- (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.
- (28) Pérez-Jordá, J. M.; Becke, A. D. A Density-Functional Study of van Der Waals Forces: Rare Gas Diatomics. *Chem. Phys. Lett.* **1995**, *233* (1), 134–137.
- (29) Kristyán, S.; Pulay, P. Can (Semi)Local Density Functional Theory Account for the London Dispersion Forces? *Chem. Phys. Lett.* **1994**, *229* (3), 175–180.
- (30) Schleyer, P. V. R.; Corminboeuf, C.; Schleyer, P. v. R. Systematic Errors in Computed Alkane Energies Using B3LYP and Other Popular DFT Functionals. *Org. Lett.* **2006**, *8*, 3631.
- (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. *Chem. Phys. Lett.* **2006**, *419* (4), 333–339.
- (32) Grimme, S.; Hansen, A.; Brandenburg, J. G.; Bannwarth, C. Dispersion-Corrected Mean-Field Electronic Structure Methods. *Chem. Rev.* **2016**, *116* (9), 5105–5154.

- (33) Eisenschitz, R.; London, F. Über das Verhältnis der van der Waalschen Kräfte zu den homöopolaren Bindungskräften. *Z. Physik* **1930**, *60* (7), 491–527.
- (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.
- (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.
- (36) Mahlberg, D.; Sakong, S.; Forster-Tonigold, K.; Groß, A. Improved DFT Adsorption Energies with Semiempirical Dispersion Corrections. *J. Chem. Theory Comput.* **2019**, *15*, 3250.
- (37) Moerman, E.; Kabylda, A.; Khabibrakhmanov, A.; Tkatchenko, A. MBD-ML: Many-Body Dispersion from Machine Learning for Molecules and Materials. arXiv 2026.
- (38) Tu, N. T. P.; Rezajooei, N.; Johnson, E. R.; Rowley, C. N. A Neural Network Potential with Rigorous Treatment of Long-Range Dispersion. *Digital Discovery* **2023**, *2* (3), 718–727.
- (39) Anstine, D.; Zubatyuk, R.; Isayev, O. AIMNet2: A Neural Network Potential to Meet Your Neutral, Charged, Organic, and Elemental-Organic Needs. *ChemRxiv* **2024**.
- (40) Unke, O. T.; Meuwly, M. PhysNet: A Neural Network for Predicting Energies, Forces, Dipole Moments, and Partial Charges. *J. Chem. Theory Comput.* **2019**, *15* (6), 3678–3693.
- (41) Andersson, Y.; Langreth, D. C.; Lundqvist, B. I. Van Der Waals Interactions in Density-Functional Theory. *Phys. Rev. Lett.* **1996**, *76* (1), 102–105.
- (42) Sato, T.; Tsuneda, T.; Hirao, K. Van Der Waals Interactions Studied by Density Functional Theory. *Mol. Phys.* **2005**, *103* (6–8), 1151–1164.
- (43) 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), No. 153004.
- (44) 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), No. 154102.
- (45) 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), No. 154104.
- (46) 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.
- (47) Grimme, S. Accurate description of van der Waals complexes by density functional theory including empirical corrections. *J. Comput. Chem.* **2004**, *25* (12), 1463–1473.
- (48) Jurečka, P.; Cerný, 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. *J. Comput. Chem.* **2007**, *28* (2), 555–569.
- (49) 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.
- (50) Ansel, J.; Yang, E.; He, H.; Gimelshein, N.; Jain, A.; Voznesensky, M.; Bao, B.; Bell, P.; Berard, D.; Burovski, E.; Chauhan, G.; Chourdia, A.; Constable, W.; Desmaison, A.; DeVito, Z.; Ellison, E.; Feng, W.; Gong, J.; Gschwind, M.; Hirsh, B.; Huang, S.; Kalambarkar, K.; Kirsch, L.; Lazos, M.; Lezcano, M.; Liang, Y.; Liang, J.; Lu, Y.; Luk, C. K.; Maher, B.; Pan, Y.; Puhrsch, C.; Reso, M.; Saroufim, M.; Siraichi, M. Y.; Suk, H.; Zhang, S.; Suo, M.; Tillet, P.; Zhao, X.; Wang, E.; Zhou, K.; Zou, R.; Wang, X.; Mathews, A.; Wen, W.; Chanan, G.; Wu, P.; Chintala, S. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, Vol. 2; ACM: La Jolla, CA, USA, 2024; pp 929–947.
- (51) 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), No. 062501.
- (52) 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.
- (53) 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.
- (54) Ehlert, S. Simple DFT-D3: Library First Implementation of the D3dispersion Correction. *JOSS* **2024**, *9* (103), 7169.
- (55) Grimme, S.; Ehrlich, S.; Goerigk, L. Effect of the Damping Function in Dispersion Corrected Density Functional Theory. *J. Comput. Chem.* **2011**, *32* (7), 1456–1465.
- (56) Hjorth Larsen, A.; Jørgen Mortensen, J.; Blomqvist, J.; Castelli, I. E.; Christensen, R.; Dulak, M.; Friis, J.; Groves, M. N.; Hammer, B.; Hargus, C.; Hermes, E. D.; Jennings, P. C.; Bjerre Jensen, P.; Kermoder, J.; Kitchin, J. R.; Leonhard Kolsbjerg, E.; Kubal, J.; Kaasbjerg, K.; Lysgaard, S.; Bergmann Maronsson, J.; Maxson, T.; Olsen, T.; Pastewka, L.; Peterson, A.; Rostgaard, C.; Schiøtz, J.; Schütt, O.; Strange, M.; Thygesen, K. S.; Vegge, T.; Vilhelmsen, L.; Walter, M.; Zeng, Z.; Jacobsen, K. W. The Atomic Simulation Environment—a Python Library for Working with Atoms. *J. Phys.: Condens. Matter* **2017**, *29* (27), No. 273002.
- (57) Kahan, W. Pracniques: Further Remarks on Reducing Truncation Errors. *Commun. ACM* **1965**, *8* (1), 40.
- (58) Dagum, L.; Menon, R. OpenMP: An Industry Standard API for Shared-Memory Programming. *IEEE Computational Science and Engineering* **1998**, *5* (1), 46–55.
- (59) 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. Y.; 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), No. 184108.
- (60) 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 γ SCF Program Package. *J. Chem. Phys.* **2020**, *153* (2), No. 024109.
- (61) 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. *J. Chem. Phys.* **2020**, *152* (20), No. 204108.

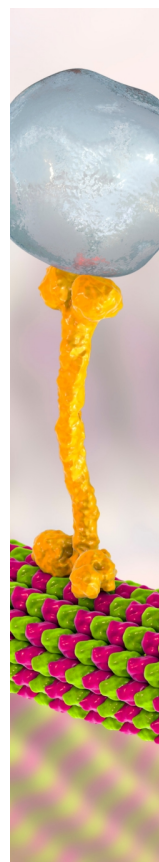
(62) Vaitkus, A.; Merkys, A.; Sander, T.; Quirós, M.; Thiessen, P. A.; Bolton, E. E.; Gražulis, S. A Workflow for Deriving Chemical Entities from Crystallographic Data and Its Application to the Crystallography Open Database. *Journal of Cheminformatics* **2023**, *15* (1), 123.

(63) 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.

(64) 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.

(65) Ishiguro, Y.; Ueno, K.; Nishimura, S.; Iida, G.; Igarashib, Y. TaCl₅-Glassified Ultrafast Lithium Ion-Conductive Halide Electrolytes for High-Performance All-Solid-State Lithium Batteries. *chem. Lett.* **2023**, *52* (4), 237–241.

(66) Luo, H.-Z.; Liang, Q.-M.; Guo, Z.-X.; Xie, X.-T.; Tang, J.-P.; Guan, T.; Li, Y.-F.; Ma, S.-C.; Xu, Y.-C.; Wang, Z.-X.; Shang, C.; Liu, Z.-P. LASPAI: AI-Powered Platform for the Future Atomic Simulation. *Acta Physico-Chimica Sinica* **2026**, *42*, No. 100235.



CAS BIOFINDER DISCOVERY PLATFORM™

BRIDGE BIOLOGY AND CHEMISTRY FOR FASTER ANSWERS

Analyze target relationships,
compound effects, and disease
pathways

Explore the platform

