Computational Materials Science xxx (xxxx) xxxx

Contents lists available at ScienceDirect

Computational Materials Science

journal homepage: www.elsevier.com/locate/commatsci

HOOMD-blue: A Python package for high-performance molecular dynamics and hard particle Monte Carlo simulations

Joshua A. Anderson[a], Jens Glaser[a], Sharon C. Glotzer[a,b,c,*]

[a] Department of Chemical Engineering, University of Michigan, Ann Arbor, MI 48109, USA
[b] Department of Materials Science and Engineering, University of Michigan, Ann Arbor, MI 48109, USA
[c] Biointerfaces Institute, University of Michigan, Ann Arbor, MI 48109, USA

ARTICLE INFO

Keywords:
Python
Molecular dynamics
Monte Carlo
Molecular simulation
GPU
CUDA

ABSTRACT

HOOMD-blue is a particle simulation engine designed for nano- and colloidal-scale molecular dynamics and hard particle Monte Carlo simulations. It has been actively developed since March 2007 and available open source since August 2008. HOOMD-blue is a Python package with a high performance C++/CUDA backend that we built from the ground up for GPU acceleration. The Python interface allows users to combine HOOMD-blue with other packages in the Python ecosystem to create simulation and analysis workflows. We employ software engineering practices to develop, test, maintain, and expand the code.

1. Introduction

Molecular, nano-, and colloidal-scale simulations are powerful tools to probe the structure and dynamics of materials. Simulations can offer insight to the fundamental physics of a phenomenon and be used to efficiently scan parameter space to find promising structures, properties or behavior. Molecular dynamics (MD) is commonly used in the biomolecular community with long established codes such as AMBER [1], GROMACS [2], and NAMD [3]. LAMMPS [4] is a general purpose MD engine with many capabilities designed for materials science applications, and is also capable of biomolecular simulations. Monte Carlo (MC) simulations of molecular systems are well suited for determining phase equilibria and made possible by codes like Cassandra [5], GOMC [6], and MCCCS Towhee [7]. Recent codes like FEN ZI [8,9], HOOMD-blue [10], and OpenMM [11] were developed around new functionalities or use cases not possible with the established codes.

Over the past decade, Python has become a popular scripting language for scientific computing in general [12] and the molecular simulation community in particular. Many Python based tools are now available for system initialization, trajectory analysis, and workflow management such as mBuild [13], MDAnalysis [14], MDTraj [15], pysimm [16], and signac [17]. Of the simulation engines mentioned, only HOOMD-blue and OpenMM provide first-class Python application programming interfaces (APIs).

This paper describes HOOMD-blue v2.6. HOOMD-blue is a particle simulation engine designed for nano- and colloidal-scale simulations. As a general-purpose tool HOOMD-blue is capable of standard MD and biomolecular simulations, but we focus our efforts on providing unique functionality that is not already available in other codes.

HOOMD-blue has been under development for more than 10 years (Section 2) as an open source code. It provides MD and hard particle MC capabilities (Section 3). We implement HOOMD-blue as a Python package (Section 4) that seamlessly interoperates with the scientific Python ecosystem. The high level Python interface abstracts a high performance backend (Section 5) that executes simulations on one or many GPUs or CPUs. As an open source project (Section 6), users can extend the code with new models and methods to enable their research. We employ software engineering (Section 7) practices to design, implement, and test the code.

2. History

HOOMD began development in March 2007 at Iowa State University. The initial implementation consisted of algorithms and data structures demonstrating high performance MD on the GPU for coarse-grained polymer simulations [10], implemented in C++ and CUDA. The second open source release in 2008 added a Python interface. Research groups discovered HOOMD, contributed new functionalities

* Corresponding author at: Department of Chemical Engineering, University of Michigan, Ann Arbor, Michigan, 48109, USA.
E-mail address: sglotzer@umich.edu (S.C. Glotzer).

https://doi.org/10.1016/j.commatsci.2019.109363
Received 12 April 2019; Received in revised form 17 October 2019; Accepted 18 October 2019

to the code, and published research papers using it [18–22]. In August 2009, HOOMD development moved to the University of Michigan and the software became HOOMD-blue (HOOMD, blue edition). Development on the project has continued with more than 10,000 commits by 68 contributors since inception. Two major milestones in HOOMD-blue development were the releases of v1.0 in 2014, which added MPI domain decomposition [23], and v2.0 in 2016 which added Monte Carlo [24,25] and discrete element method MD [26] for hard shapes. In addition to these major developments, the code has grown organically with new capabilities and performance improvements through a process of lazy refactoring [27]. As of September 2019, we are aware of 298 peer-reviewed research articles that made use of HOOMD-blue.

3. Capabilities

3.1. Molecular dynamics

Listing 1 Molecular dynamics simulation

```
from hoomd import *
from hoomd import md
# place particles
context.initialize ()
unitcell = lattice.sc (a = 2.0, type_name='A')
init.create_lattice (unitcell, n = 10)
# define Lennard-Jones interactions
nl  = md.nlist.cell ()
lj  = md.pair.lj (r_cut = 2.5, nlist = nl)
lj.pair_coeff.set ('A', 'A', epsilon = 1.0,sigma = 1.0)
# NVT integration
all  = group.all ();
md.integrate.mode_standard (dt = 0.005)
nvt  = md.integrate.nvt (group = all, kT = 1.2, tau = 1.0)
nvt.randomize_velocities (seed = 1)
# run the simulation
run (10e3)
```

HOOMD-blue has MD integrators for many different thermodynamic ensembles including NVE, NVT, NPH, NPT, Langevin dynamics, Brownian dynamics, and dissipative particle dynamics, and also supports FIRE [28] energy minimization. The NVT, NPH, and NPT integrators are based on the Martyna-Tobias-Klein method [29]. All of these support the integration of rotational degrees of freedom directly, and are able to couple a single thermostat to a system consisting of particles with and without rotational degrees of freedom. HOOMD-blue also implements the multi-particle collision dynamics solvent model [30]. Users can apply different integrators to distinct subsets of the system.

The general-purpose MD engine in HOOMD-blue can apply many different types of forces to particles. Users can employ any number of these to achieve the desired model. Anisotropic potentials treat particles with extended shape and produce both forces and torques on particles. HOOMD-blue implements DEM potentials for faceted shapes [26], Gay-berne ellipsoids, and dipole potentials. Composite particle constraints connect many constituent particles so that they move as a rigid body [21,31]. HOOMD-blue supports the active matter community with an active force module that can apply constant magnitude forces or torques to particles. HOOMD-blue also supports commonly used pair, bond, angle, dihedral, improper, special pair potentials, and PPPM electrostatics [22]. We provide a wide variety of pair potentials utilized in different fields including Buckingham, DLVO, DPD, Lennard-Jones, Gaussian, Mie, WCA, Yukawa, and others. Users can also develop and test custom potentials with tabulated pair, bond, angle and dihedral potentials. HOOMD-blue includes EAM [20,32], Tersoff, and square density many-body potentials as well as periodic, electric field, constant force, and wall external potentials. Users can fix the bond length between pairs of particles with distance constraints [33,34], or constrain particles to the surface of a sphere or an ellipsoid.

3.2. Monte Carlo

Listing 2 Hard particle Monte Carlo simulation

```
from hoomd import *
from hoomd import hpmc
# place particles
context.initialize ()
unitcell = lattice.sc (a = 2.0, type_name='A')
init.create_lattice (unitcell, n = 10)
# hard particle Monte Carlo
mc  = hpmc.integrate.convex_polyhedron(d = 0.1, a = 0.1, seed = 2)
cube_verts = [[-0.5,-0.5,-0.5], [0.5,-0.5,-0.5],
[-0.5,-0.5, 0.5], [0.5,-0.5, 0.5],
[-0.5, 0.5,-0.5], [0.5, 0.5,-0.5],
[-0.5, 0.5, 0.5], [0.5, 0.5, 0.5]]
mc.shape_param.set ('A', vertices = cube_verts)
# run the simulation
run (10e3)
```

In addition to MD simulations, HOOMD-blue can perform hard particle MC simulations [24] with the HPMC component. We have implemented a wide variety of shape classes, including spheres, disks, unions of spheres, convex spheropolygons, simple polygons, ellipsoids, convex spheropolyhedra, unions of convex spheropolyhedra, faceted spheres, and general triangle meshes. All particles in a simulation must be of the same shape class, but may be of different particle types, where each particle type has separate shape parameters. User-defined wall constraints confine particles to particular regions of space. HPMC implements trial moves that enable NVT, NPT, grand, and Gibbs ensembles. An implicit depletant algorithm [25] enables efficient simulations of colloids with depletants. HPMC can sample the simulation pressure in NVT ensembles [35,36,24] and the free volume available to the system. We have also implemented the Frenkel-Ladd free energy method [37].

In hard particle Monte Carlo, the energy of the system is infinite when two particles overlap or zero when there are no overlaps. There are research applications that apply attractive patchy interactions in addition to the hard particle core. Each project customizes the form of this enthalpic interaction to the specific research question. HOOMD provides flexibility to the user while maintaining high performance using just-in-time compilation. The user provides a C++ code snippet in their script, which HPMC compiles at runtime with Clang [38] and executes it with LLVM [39] when needed to determine the energy of each interaction. We provide hooks for cutoff pair potentials, cutoff pair potentials evaluated at the points of a sphere union shape, and external potentials applied to each particle.

4. Python package

HOOMD-blue is a Python package. Through the imperative Python API, a user can configure which capabilities are enabled, set parameters, and control the progression of the simulation run. Listing 1 performs a simple MD simulation. It uses `init.create_lattice` to initialize a simple cubic lattice of particles, `md.pair.lj` to define the Lennard-Jones particle pair interaction potential, and `md.integrate.nvt` to employ NVT ensemble integration. Listing 2 performs a simple MC simulation of hard cubes. It uses `hpmc.integrate.convex_polyhedron` to specify HPMC integration of the convex polyhedron shape class and sets the parameters for type *A* to the vertices of a cube. In both examples `run` executes the simulation for the given number of steps.

Users can use the Python API in a script, via job queue submission, inside Jupyter notebooks, or by any other method one can use a Python package. For example, HOOMD-blue can easily be used for workflow steps in the *signac* framework [17]. Options passed to the device context control whether the simulation executes on the CPU or the GPU. Without modifications to the script, a user may execute a script on

many CPUs or GPUs by launching it as an MPI parallel job or on GPU nodes with NVLINK (such as OLCF Summit) with the multiple GPU device option `gpu=0,1,2`.

We provide complete documentation [40] for HOOMD-blue, including installation instructions, tutorials in Jupyter notebooks, overview documentation of general concepts, and a complete listing of every API call along with the options and parameters they accept. Users can browse static copies of the tutorials online or download the notebooks and execute them with Jupyter. Jupyter notebooks provide a mixture of formatted text description, code, and output. The HOOMD-blue example notebooks examine the simulation results and show relevant output, including plots of system quantities vs time and trajectory visualizations. When running the notebooks, users can modify parameters or introduce new commands, then re-execute the notebook and see how the output changes. HOOMD-blue's user and developer community is available to answer questions on the hoomd-users mailing list, which has 521 members as of September 2019.

HOOMD-blue runs on Linux and macOS. We provide binary packages on the *conda-forge* [41] Anaconda channel and also in Docker and Singularity [42] images. The Anaconda packages provide an installation mechanism suitable for testing jobs on a laptop or workstation. However, there are limitations that prevent Anaconda packages from taking full advantage of resources on HPC clusters. We provide Singularity images with performance optimized builds of HOOMD for a number of national HPC resources, including PSC Bridges, SDSC Comet, and TACC Stampede2. Users can also build HOOMD-blue from source using CMake, numpy, and Python. NVIDIA CUDA is optional, but required to enable GPU acceleration. An MPI library is optional, but required to enable execution in parallel on multiple nodes.

5. Performance

We optimize all capabilities in HOOMD-blue extensively so that it runs nano- and colloidal-scale simulations as fast as possible. With very few exceptions, all of the time-consuming operations in HOOMD-blue can execute on the GPU, including all of the data structures, force evaluations, integrators, and MPI communication. We have also parallelized the majority of HOOMD-blue capabilities with MPI for execution on multiple GPUs or multiple CPU cores. The exceptions are cases where particular methods are typically used for simulations where GPUs and/or MPI simulations are not necessary or code paths that are rarely called. The components that lack GPU support in v2.6 are external fields and user-defined pair potentials in HPMC, and the temperature rescale and zero momentum updaters in MD. The components that lack MPI support in v2.6 are active forces, the IMD (interactive MD) communication protocol, the Berendsen integrator, and ellipsoid constraints.

Simulation performance is highly dependent on the type of simulation, particle sizes, force fields, system density, and other parameters. Typical research-relevant MD simulations execute an order of magnitude faster on a single GPU than on all cores of a single CPU socket for system sizes of four thousand particles per GPU or more [23]. Over the years, we have continually re-tuned and rewritten the MD CUDA code for each new generation of GPU hardware. The current version of the code (v2.6) utilizes many optimization techniques standard in the CUDA developer community, including multiple threads per particle with warp-level reductions, atomic operations to build cell lists, autotuning kernel parameters, and others which readers can find in the HOOMD-blue source code. Recently, we added support for improved intra-node scaling to many GPUs using NVIDIA's NVLINK technology [43] (this issue).

HOOMD-blue includes unique performance optimizations. In colloidal systems with large and small particle sizes, cell-list based neighbor list algorithms do not operate efficiently. The cell list must either be sized to the largest particle, or many hundreds of cells must be traversed to find tens of neighbors. Bounding volume hierarchy (BVH) data structures, commonly used in computer graphics and video games, dynamically adapt to the density fluctuations with minimal memory usage and high performance. HOOMD-blue provides a BVH method to build neighbor lists with `nlist.tree`, contributed by Michael Howard [44], which is faster than the cell list for size ratios of 2:1 or greater. Howard is currently working to optimize this code path further with the goal to make it faster than the standard cell list in all cases [45] (this issue). Users choose which neighbor list implementation to use in their simulations.

Hard particle Monte Carlo (HPMC) simulations also execute an order of magnitude faster on a single GPU than on all cores of a single CPU socket, but only when system sizes are larger than tens of thousands of particles per GPU. We are actively working on further optimizing this GPU code path for hard particle simulations smaller than this. We have also spent considerable effort optimizing HPMC's CPU code path. HPMC uses BVH data structures and CPU vector intrinsics, and executes many trial moves in parallel to attain the best possible performance on the CPU. See Ref. [24] for complete details about the HPMC component of HOOMD-blue.

To put these general performance statements into context, we include benchmarks for two representative cases: the Lennard-Jones liquid with $N = 64,000$ particles at a number density of $0.382\sigma^{-3}$ in the NVT ensemble with $k_B T/\varepsilon = 1.2$, $\delta t = 0.005\sqrt{m\sigma^2/\varepsilon}$ and a cutoff distance $r_{cut} = 3.0\sigma$ [10]; and the hard hexagon hexatic phase with $N = 1,048,576$ particles at a packing fraction of $\phi_P = 0.7$ in the NVT ensemble [46].

On a single 24-core Intel Xeon Platinum 8160 CPU (TACC Stampede2), HOOMD-blue v2.6.0 performs the MD Lennard-Jones liquid benchmark at $16.1 \cdot 10^6$ particle time steps per second and the HPMC hexagon benchmark at $20.8 \cdot 10^6$ trial moves per second. On a single V100 GPU (PSC Bridges), performance increases to $275 \cdot 10^6$ million particle time steps per second for the Lennard-Jones liquid benchmark and $132 \cdot 10^6$ trial moves per second for the hexagon benchmark.

We recommend that users test their models with different numbers of CPU cores and GPUs so that they can make informed choices on the performance and efficiency tradeoffs specific to their systems. We urge authors who wish to publish comparative benchmarks to build the latest version of HOOMD-blue and perform direct comparisons on the most current hardware available. The performance numbers we include here are representative only of a single point in time. Manufactures regularly produce new processors and developers frequently write new code performance optimizations. As a case in point, compare the above *full double precision* V100 performance of HOOMD-blue v2.6.0 to our 2008 publication, where HOOMD v0.6.0 performed the Lennard-Jones liquid benchmark *in single precision* on a single G80 GPU at $12.9 \cdot 10^6$ particle time steps per second [10] and our 2015 publication where HOOMD v1.0.0 performed the Lennard-Jones benchmark (with $N = 32,000$) *in single precision* on a single K20 GPU (OLCF Titan) at $64 \cdot 10^6$ particle time steps per second [23].

6. Open source

HOOMD-blue is available open source under the permissive 3-clause BSD license. To implement changes, users can fork the HOOMD-blue code and add new functionalities directly, or create a plugin in a separate code repository and link it to HOOMD-blue at build time. Many users have published papers and software frameworks using HOOMD-blue with extensions they have developed. Six recent examples of this include: raaSAFT, a framework enabling coarse-grained simulations based on the SAFT-γ Mie force field [47]; reverse non-equilibrium molecular dynamics simulation applied to transitions between lamellar orientations in shear flow [48]; protracted colored noise dynamics applied to linear polymer systems [49]; epoxpy, a package that dynamically adds bonds during DPD simulations to model expoxy curing [50];

accurate hydrodynamic interactions to study how surface heterogeneity affects percolation and gelation of colloids [51]; and Hoobas, a highly objected-oriented builder for molecular dynamics to generate complex initial conditions for polymer and DNA-coated nanoparticle systems [52].

As of September 2019, GitHub (https://github.com/glotzerlab/hoomd-blue) facilitates the open source development of HOOMD-blue (previously, we have hosted development on Assembla, Redmine, and Bitbucket). The code repository houses the entire history of HOOMD-blue's development and allows many developers to simultaneously work on changes to the code. Issue lists allow users to submit bug reports and track their progress and developers to track planned feature development. We encourage contributions to HOOMD-blue from the community. Pull requests allow the community to review and discuss proposed changes to the code. We merge pull requests into the main line of development after they are reviewed and approved and pass all tests.

7. Software engineering

HOOMD-blue v2.6 consists of 173,662 lines of code. While 68 individuals have contributed to the project over its lifetime, at any one time a group of 2–3 developers maintains the code and improves core functionalities. HOOMD-blue developers are researchers and do not develop full-time. We automate processes and carefully consider library dependencies, tools, and design patterns in order to provide the highest quality code that will remain stable over a long time period with a minimum of maintenance effort. For example, HOOMD-blue's simulation engine is written with a modular, isolated, object oriented design. Each capability of the code is implemented in a separate class that communicates with the core particle data structures and only with other classes as necessary.

We implement simulation algorithms in C++ and CUDA, with MPI for domain decomposition. CUDA and MPI are optional, but are needed to provide GPU and parallel runs, respectively. The user-facing side HOOMD-blue is written in Python. We use the pybind11 [53] library to interface Python and C++ classes, and the CMake system to configure builds. We also utilize a number of header-only libraries which we embed with submodules so that users and developers do not need to separately download them. HOOMD-blue's documentation contains the full list of all libraries utilized and their corresponding license notices.

7.1. Testing

We employ white box unit testing for every module in HOOMD-blue. Each test is designed with knowledge of how the module is implemented and sufficient cases are tested so as to exercise all of the possible code paths that the module can take. The automated unit tests can easily be executed for any build of HOOMD-blue to validate that all capabilities are operating correctly.

For example, a unit test for the Lennard-Jones pair force class creates a system with six particles in it. Some of these interact across periodic boundary conditions while others interact directly. It then sets the potential parameters ε, σ, and r_{cut} to different values and verifies that the correct force, energy, and virial are computed each time. We compute reference values independently (e.g. with a calculator) for individual unit tests. A test passes only when all computed values are within a tolerance of the reference.

We also perform black box system integration tests to ensure that modules work together correctly. These validation tests assume no implementation specific knowledge. The tests enter simulation parameters in a job script and analyze output files the same way a user would run HOOMD-blue.

Automated validation tests must meet the following requirements. The test simulation must be long enough to sample an average value with reasonable error bars, but short enough that they complete in a reasonable amount of time (ideally, individual tests should complete in less than 10 min). When available, we validate equations of state against published reference values. For example, we use the pressure and density of the hard disk fluid to validate HPMC [54] and the NIST standard simulation reference [55] to validate Lennard-Jones simulations in both HPMC and MD. In other cases, such references are not available in the literature and we instead cross-validate multiple code paths in HOOMD-blue. For example, one validation test compares MD and MC simulations of WCA dimers, both performed by HOOMD-blue.

7.2. Continuous integration

We employ continuous integration practices to ensure code quality. Scripts trigger on every commit to the HOOMD-blue source code repository and on pull requests. These scripts compile that commit with a selection of compilers, CUDA versions, Python versions, CMake versions, LLVM versions, and other build options. Then the script runs the unit tests. Selected builds also run the longer validation tests. All of the build output is captured and any failing builds or failing test are flagged. The continuous integration testing system provides developers with feedback on the test outcomes.

As of September 2019, we build Docker containers to provide the build and test environment and execute tests on Microsoft Azure Pipelines [56]. We execute CPU tests on Microsoft-Hosted cloud agents and GPU tests on self-hosted agents we run locally. We also utilize the readthedocs [57] service to automatically build and host HOOMD-blue's documentation when new commits are pushed to the repository.

8. Conclusions

HOOMD-blue is a particle simulation engine designed for nano- and colloidal-scale simulations. It was built from the ground up for GPU acceleration, and has been actively developed since March 2007 with more than 10,000 commits and 68 contributors. As a general-purpose code, it is used by researchers the fields of colloidal self-assembly, active matter, coarse grained polymers, and others.

We develop the open source HOOMD-blue as a Python package with a high performance C++/CUDA backend. HOOMD-blue's Python API allows users to define and execute simulations and combine them with other tools in the scientific Python ecosystem. We optimize all core functionalities to obtain the fastest possible performance on whatever architecture (CPU or GPU) is best suited to the problem. This requires us to continually refactor and rewrite the core kernels to ensure HOOMD-blue performs well on the latest hardware. As an open source package, users can modify and extend the code to meet their specific needs.

We employ software engineering practices including unit tests, system validation tests and continuous integration to ensure that the code operates correctly. Collaborative development and code review practices to ensure code meets standards set by the developers and to give the community an opportunity to provide input on the development process. HOOMD-blue is open source and we welcome contributions of new capabilities of interest to a wide audience of users.

We are impressed by the amazing science the research community is able to accomplish using HOOMD-blue. We hope that more of this great work continues, and that more developers contribute to make HOOMD-blue a better simulation code for the community.

Data availability

HOOMD-blue source code is available on github:https://github.com/glotzerlab/hoomd-blue.

CRediT authorship contribution statement

Joshua A. Anderson: Conceptualization, Investigation, Validation,

Writing - original draft, Writing - review & editing. **Jens Glaser:** Conceptualization, Investigation, Validation, Writing - original draft, Writing - review & editing. **Sharon C. Glotzer:** Conceptualization, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Initial HOOMD development (v0.6-v0.8) was supervised by Alex Travesset and funded by the National Science Foundation through Grant DMR-0426597 and by DOE through the Ames lab under Contract No. DE-AC02-07CH11358. HOOMD-blue development has been supported by the DOD/ASD (R&E) under Award No. N00244-09-1-0062 (2009-2014, early design and implementation, v0.9 – v1.x) and the National Science Foundation, Division of Materials Research Award # DMR 1409620 (2014–2018, especially DEM and HPMC capabilities in v2.x). Software was validated and benchmarked on the Extreme Science and Engineering Discovery Environment (XSEDE) [58], which is supported by National Science Foundation Grant No. ACI-1053575 (XSEDE award DMR 140129); on resources of the Oak Ridge Leadership Computing Facility which is a DOE Office of Science User Facility supported under Contract No. DE- AC05-00OR22725; and through computational resources and services provided by Advanced Research Computing at the University of Michigan, Ann Arbor. Hardware provided by NVIDIA Corp. is gratefully acknowledged. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the DOD/ASD(R&E).

We would like to thank all HOOMD-blue contributors: Carl Simon Adorf, Khalid Ahmed, James Antonaglia, Steve Barr, Joseph Berleant, Isaac Bruss, Chengyu Dai, Kevin Daly, Avisek Das, Bradley Dice, Paul Dodd, Chrisy Du, Åsmund Ervik, Jenny Fothergill, Grey Garrett, Eric Harper, Mike Henry, Michael Howard, Alexander Hudson, M. Eric Irrgang, Eric Jankowski, Kwanghwi Je, Bjørnar Jensen, Christoph Junghans, Aaron Keys, Christoph Klein, Axel Kohlmeyer, Kevin Kohlstedt, David LeBard, Andrew Mark, Ryan Marson, Tim Moore, Shannon Moran, Igor Morozov, Pavani Medapuram Lakshmi Narasimha, Richmond Newman, Trung Dac Nguyen, Sam Nola, Antonio Osorio, Carolyn Phillips, James Proctor, Cong Qiao, Vyas Ramasubramani, Malcolm Ramsay, Sumedh R. Risbud, Luis Y. Rivera-Rivera, Ludwig Schneider, Benjamin Schultz, Peter Schwendeman, Wenbo Shen, Kevin Silmore, Rastko Sknepnek, Brandon Denis Smith, Ross Smith, Matthew Spellings, Ben Swerdlow, Erin Teich, Stephen Thomas, Alex Travesset, Alyssa Travitz, Greg van Anders, Bryan VanSaders, Lin Yang, Pengji Zhou, and William Zygmunt

References

[1] D. Case, I. Ben-Shalom, S. Brozell, D. Cerutti, T. Cheatham III, V. Cruzeiro, T. Darden, R. Duke, D. Ghoreishi, M. Gilson, H. Gohlke, A. Goetz, D. Greene, R. Harris, N. Homeyer, S. Izadi, A. Kovalenko, T. Kurtzman, T. Lee, S. LeGrand, P. Li, C. Lin, J. Liu, T. Luchko, R. Luo, D. Mermelstein, K. Merz, Y. Miao, G. Monard, C. Nguyen, H. Nguyen, I. Omelyan, A. Onufriev, F. Pan, R. Qi, D. Roe, A. Roitberg, C. Sagui, S. Schott-Verdugo, J. Shen, C. Simmerling, J. Smith, R. Salomon-Ferrer, J. Swails, R. Walker, J. Wang, H. Wei, R. Wolf, X. Wu, L. Xiao, D. York, P. Kollman, AMBER (2018) 2018.

[2] M.J. Abraham, T. Murtola, R. Schulz, S. Páll, J.C. Smith, B. Hess, E. Lindahl, GROMACS: high performance molecular simulations through multi-level parallelism from laptops to supercomputers, SoftwareX 1–2 (2015) 19–25, https://doi.org/10.1016/j.softx.2015.06.001.

[3] J.C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R.D. Skeel, L. Kalé, K. Schulten, Scalable molecular dynamics with NAMD, J. Comput. Chem. 26 (2005) 1781–1802, https://doi.org/10.1002/jcc.20289.

[4] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J. Comp. Phys. 117 (1995) 1–19, https://doi.org/10.1006/jcph.1995.1039.

[5] J.K. Shah, E. Marin-Rimoldi, R.G. Mullen, B.P. Keene, S. Khan, A.S. Paluch, N. Rai, L.L. Romanielo, T.W. Rosch, B. Yoo, E.J. Maginn, Cassandra: an open source Monte Carlo package for molecular simulation, J. Comput. Chem. 38 (2017) 1727–1739, https://doi.org/10.1002/jcc.24807.

[6] Y. Nejahi, M. Soroush Barhaghi, J. Mick, B. Jackman, K. Rushaidat, Y. Li, L. Schwiebert, J. Potoff, GOMC: GPU optimized Monte Carlo for the simulation of phase equilibria and physical properties of complex fluids, SoftwareX 9 (2019) 20–27 DOI: 10.1016/j.softx.2018.11.00.

[7] M.G. Martin, MCCCS Towhee: a tool for Monte Carlo molecular simulation, Mol. Simul. 39 (2013) 1212–1222, https://doi.org/10.1080/08927022.2013.828208.

[8] N. Ganesan, B.A. Bauer, T.R. Lucas, S. Patel, M. Taufer, Structural, dynamic, and electrostatic properties of fully hydrated DMPC bilayers from molecular dynamics simulations accelerated with graphical processing units (GPUs), J. Comput. Chem. 32 (2011) 2958–2973, https://doi.org/10.1002/jcc.21871.

[9] M. Taufer, N. Ganesan, S. Patel, GPU enabled macromolecular simulation: challenges and opportunities, IEEE Comput. Sci. Eng. (CiSE) 15 (2013) 56–64, https://doi.org/10.1109/MCSE.2012.42.

[10] J.A. Anderson, C.D. Lorenz, A. Travesset, General purpose molecular dynamics simulations fully implemented on graphics processing units, J. Comput. Phys. 227 (2008) 5342–5359, https://doi.org/10.1016/j.jcp.2008.01.047.

[11] P. Eastman, V.S. Pande, Efficient nonbonded interactions for molecular dynamics on a graphics processing unit, J. Comput. Chem. 31 (2010) 1268–1272, https://doi.org/10.1002/jcc.21413.

[12] K.J. Millman, M. Aivazis, Python for scientists and engineers, Comput. Sci. Eng. 13 (2011) 9–12, https://doi.org/10.1109/MCSE.2011.36.

[13] C. Klein, J. Sallai, T.J. Jones, C.R. Iacovella, C. McCabe, P.T. Cummings, A Hierarchical, Component Based Approach to Screening Properties of Soft Matter, Foundations of Molecular Modeling and Simulation, 2016, pp. 79–92, , https://doi.org/10.1007/978-981-10-1128-3_5.

[14] N. Michaud-Agrawal, E.J. Denning, T.B. Woolf, O. Beckstein, MDAnalysis: a toolkit for the analysis of molecular dynamics simulations, J. Comput. Chem. 32 (2011) 2319–2327, https://doi.org/10.1002/jcc.21787.

[15] R.T. McGibbon, K.A. Beauchamp, M.P. Harrigan, C. Klein, J.M. Swails, C.X. Hernández, C.R. Schwantes, L.-P. Wang, T.J. Lane, V.S. Pande, MDTraj: a modern open library for the analysis of molecular dynamics trajectories, Biophys. J. 109 (2015) 1528–1532, https://doi.org/10.1016/j.bpj.2015.08.015.

[16] M.E. Fortunato, C.M. Colina, pysimm: a python package for simulation of molecular systems, SoftwareX 6 (2017) 7–12, https://doi.org/10.1016/j.softx.2016.12.002.

[17] C.S. Adorf, P.M. Dodd, V. Ramasubramani, S.C. Glotzer, Simple data and workflow management with the SIGNAC framework, Comput. Mater. Sci. 146 (2018) 220–229, https://doi.org/10.1016/j.commatsci.2018.01.035.

[18] C.L. Phillips, J.A. Anderson, S.C. Glotzer, Pseudo-random number generation for Brownian dynamics and dissipative particle dynamics simulations on GPU devices, J. Comput. Phys. 230 (2011) 7191–7201, https://doi.org/10.1016/j.jcp.2011.05.021.

[19] B.G. Levine, D.N. Lebard, R. Devane, W. Shinoda, A. Kohlmeyer, M.L. Klein, Micellization studied by GPU-accelerated coarse-grained molecular dynamics, J. Chem. Theory Comput. 7 (2011) 4135–4145, https://doi.org/10.1021/ct2005193.

[20] I. Morozov, A. Kazennov, R. Bystryi, G. Norman, V. Pisarev, V. Stegailov, Molecular dynamics simulations of the relaxation processes in the condensed matter on GPUs, Comput. Phys. Commun. 182 (2011) 1974–1978, https://doi.org/10.1016/j.cpc.2010.12.026.

[21] T.D. Nguyen, C.L. Phillips, J.A. Anderson, S.C. Glotzer, Rigid body constraints realized in massively-parallel molecular dynamics on graphics processing units, Comput. Phys. Commun. 182 (2011) 2307–2313, https://doi.org/10.1016/j.cpc.2011.06.005.

[22] D.N. LeBard, B.G. Levine, P. Mertmann, S.A. Barr, A. Jusufi, S. Sanders, M.L. Klein, A.Z. Panagiotopoulos, Self-assembly of coarse-grained ionic surfactants accelerated by graphics processing units, Soft Matter 8 (2012) 2385–2397, https://doi.org/10.1039/C1SM06787G.

[23] J. Glaser, T.D. Nguyen, J.A. Anderson, P. Lui, F. Spiga, J.A. Millan, D.C. Morse, S.C. Glotzer, Strong scaling of general-purpose molecular dynamics simulations on GPUs, Comput. Phys. Commun. 192 (2015) 97–107, https://doi.org/10.1016/j.cpc.2015.02.028.

[24] J.A. Anderson, M. Eric Irrgang, S.C. Glotzer, Scalable metropolis Monte Carlo for simulation of hard shapes, Comput. Phys. Commun. 204 (2016) 21–30, https://doi.org/10.1016/j.cpc.2016.02.024.

[25] J. Glaser, A.S. Karas, S.C. Glotzer, A parallel algorithm for implicit depletant simulations, J. Chem. Phys. 143 (2015) 184110, https://doi.org/10.1063/1.4935175.

[26] M. Spellings, R.L. Marson, J.A. Anderson, S.C. Glotzer, GPU accelerated Discrete Element Method (DEM) molecular dynamics for conservative, faceted particle simulations, J. Comput. Phys. 334 (2017) 460–467, https://doi.org/10.1016/j.jcp.2017.01.014.

[27] C.S. Adorf, V. Ramasubramani, J.A. Anderson, S.C. Glotzer, How to professionally develop reusable scientific software? and when not to, Comput. Sci. Eng. (2018), https://doi.org/10.1109/MCSE.2018.2882355 1–1.

[28] E. Bitzek, P. Koskinen, F. Gähler, M. Moseler, P. Gumbsch, Structural relaxation made simple, Phys. Rev. Lett. 97 (2006), https://doi.org/10.1103/PhysRevLett.97.170201.

[29] G.J. Martyna, D.J. Tobias, M.L. Klein, Constant pressure molecular dynamics algorithms, J. Chem. Phys. 101 (1994) 4177–4189, https://doi.org/10.1063/1.467468.

[30] M.P. Howard, A.Z. Panagiotopoulos, A. Nikoubashman, Efficient mesoscale hydrodynamics: multiparticle collision dynamics with massively parallel GPU

acceleration, Comput. Phys. Commun. 230 (2018) 10–20, https://doi.org/10.1016/j.cpc.2018.04.009.
[31] J. Glaser, X. Zha, J.A. Anderson, S.C. Glotzer, A. Travesset, Pressure in Rigid Body Molecular Dynamics, In Preparation (this issue), 2019.
[32] L. Yang, F. Zhang, C.-Z. Wang, K.-M. Ho, A. Travesset, Implementation of metal-friendly EAM/FS-type semi-empirical potentials in HOOMD-blue: a GPU-accelerated molecular dynamics software, J. Comput. Phys. 359 (2018) 352–360, https://doi.org/10.1016/j.jcp.2018.01.015.
[33] M. Yoneya, H.J.C. Berendsen, K. Hirasawa, A non-iterative matrix method for constraint molecular dynamics simulations, Mol. Simul. 13 (1994) 395–405, https://doi.org/10.1080/08927029408022001.
[34] M. Yoneya, A generalized non-iterative matrix method for constraint molecular dynamics simulations, J. Comput. Phys. 172 (2001) 188–197, https://doi.org/10.1006/jcph.2001.6819.
[35] R. Eppenga, D. Frenkel, Monte Carlo study of the isotropic and nematic phases of infinitely thin hard platelets, Mol. Phys. 52 (1984) 1303–1334, https://doi.org/10.1080/00268978400101951.
[36] P.E. Brumby, A.J. Haslam, E. de Miguel, G. Jackson, Subtleties in the calculation of the pressure and pressure tensor of anisotropic particles from volume-perturbation methods and the apparent asymmetry of the compressive and expansive contributions, Mol. Phys. 109 (2011) 169–189, https://doi.org/10.1080/00268976.2010.530301.
[37] D. Frenkel, A.J.C. Ladd, New Monte Carlo method to compute the free energy of arbitrary solids. Application to the fcc and hcp phases of hard spheres, J. Chem. Phys. 81 (1984) 3188–3193, https://doi.org/10.1063/1.448024.
[38] Clang: a C language family frontend for LLVM, 2019.https://clang.llvm.org/.
[39] C. Lattner, V. Adve, LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation, in: Proceedings of the International Symposium on Code Generation and Optimization: Feedback-directed and Runtime Optimization, CGO '04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 75.
[40] HOOMD-blue documentation, 2019. https://hoomd-blue.readthedocs.io.
[41] Conda Forge, 2019. https://conda-forge.org/.
[42] G.M. Kurtzer, V. Sochat, M.W. Bauer, Singularity: scientific containers for mobility of compute, PLOS ONE 12 (2017) e0177459, https://doi.org/10.1371/journal.pone.0177459.
[43] J. Glaser, P. Schwendeman, J.A. Anderson, S.C. Glotzer, Unified memory in HOOMD-blue for node-level strong scaling, In Preparation (this issue), 2019.
[44] M.P. Howard, J.A. Anderson, A. Nikoubashman, S.C. Glotzer, A.Z. Panagiotopoulos, Efficient neighbor list calculation for molecular simulation of colloidal systems using graphics processing units, Comput. Phys. Commun. 203 (2016) 45–52, https://doi.org/10.1016/j.cpc.2016.02.003.
[45] M.P. Howard, A. Statt, F. Madutsa, T.M. Truskett, A.Z. Panagiotopoulos, Quantized bounding volume hierarchies for neighbor search in molecular simulations on graphics processing units, (this issue) (2019). http://arxiv.org/abs/1901.08088.
[46] J.A. Anderson, J. Antonaglia, J.A. Millan, M. Engel, S.C. Glotzer, Shape and symmetry determine two-dimensional melting transitions of hard regular polygons, Phys. Rev. X 7 (2017) 021001, , https://doi.org/10.1103/PhysRevX.7.021001 URL: http://arxiv.org/abs/1606.00687.
[47] Å. Ervik, G.J. Serratos, E.A. Müller, raaSAFT: a framework enabling coarse-grained molecular dynamics simulations based on the SAFT-γ Mie force field, Comput. Phys. Commun. 212 (2017) 161–179, https://doi.org/10.1016/j.cpc.2016.07.035.
[48] L. Schneider, M. Heck, M. Wilhelm, M. Müller, Transitions between lamellar orientations in shear flow, Macromolecules 51 (2018) 4642–4659, https://doi.org/10.1021/acs.macromol.8b00825.
[49] A.J. Peters, B.D. Nation, D. Nicoloso, P.J. Ludovice, C.L. Henderson, Protracted colored noise dynamics applied to linear polymer systems, Macromol. Theory Simul. 27 (2018) 1700062, https://doi.org/10.1002/mats.201700062.
[50] S. Thomas, M. Alberts, M.M. Henry, C.E. Estridge, E. Jankowski, Routine million-particle simulations of epoxy curing with dissipative particle dynamics, J. Theoretical Computat. Chem. 17 (2018) 1840005, https://doi.org/10.1142/S0219633618400059.
[51] G. Wang, J.W. Swan, Surface heterogeneity affects percolation and gelation of colloids: dynamic simulations with random patchy spheres, Soft Matter 15 (2019) 5094–5108, https://doi.org/10.1039/C9SM00607A.
[52] M. Girard, A. Ehlen, A. Shakya, T. Bereau, M.O. de la Cruz, Hoobas: a highly object-oriented builder for molecular dynamics, Comput. Mater. Sci. 167 (2019) 25–33, https://doi.org/10.1016/j.commatsci.2019.05.003.
[53] W. Jakob, J. Rhinelander, D. Moldovan, pybind11 – Seamless operability between C++11 and Python, 2017.
[54] E.P. Bernard, W. Krauth, Two-step melting in two dimensions: first-order liquid-hexatic transition, Phys. Rev. Lett. 107 (2011) 155704, https://doi.org/10.1103/PhysRevLett. 107.155704.
[55] V.K. Shen, D.W. Siderius, W.P. Krekelberg, H.W. Hatch, NIST Standard Reference Simulation Website, NIST Standard Reference Database Number 173, National Institute of Standards and Technology, Gaithersburg MD, 2017. https://doi.org/10.18434/T4M88Q.
[56] Microsoft Azure Pipelines, 2019. https://azure.microsoft.com/en-us/services/devops/pipelines/.
[57] readthedocs, 2019. https://readthedocs.org/.
[58] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G.D. Peterson, R. Roskies, J.R. Scott, N. Wilkens-Diehr, XSEDE: accelerating scientific discovery, Comput. Sci. Eng. 16 (2014) 62–74, https://doi.org/10.1109/MCSE.2014.80.