RESEARCH ARTICLE | MAY 12 2022

A neural network-assisted open boundary molecular dynamics simulation method

J. E. Floyd; J. R. Lukes

J. Chem. Phys. 156, 184114 (2022)
https://doi.org/10.1063/5.0083198

A neural network-assisted open boundary molecular dynamics simulation method

Cite as: J. Chem. Phys. 156, 184114 (2022); doi: 10.1063/5.0083198
Submitted: 22 December 2021 • Accepted: 25 April 2022 •
Published Online: 12 May 2022

J. E. Floyd and J. R. Lukes[a)]

AFFILIATIONS

Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, Pennsylvania 19104, USA

[a)]**Author to whom correspondence should be addressed:** jrlukes@seas.upenn.edu

ABSTRACT

A neural network-assisted molecular dynamics method is developed to reduce the computational cost of open boundary simulations. Particle influxes and neural network-derived forces are applied at the boundaries of an open domain consisting of explicitly modeled Lennard-Jones atoms in order to represent the effects of the unmodeled surrounding fluid. Canonical ensemble simulations with periodic boundaries are used to train the neural network and to sample boundary fluxes. The method, as implemented in the LAMMPS, yields temperature, kinetic energy, potential energy, and pressure values within 2.5% of those calculated using periodic molecular dynamics and runs two orders of magnitude faster than a comparable grand canonical molecular dynamics system.

I. INTRODUCTION

Bulk fluids have traditionally been modeled in molecular dynamics (MD) simulations by employing periodic boundary conditions in which atoms leaving one side of the computational domain enter the opposite side.[1] While appropriate for homogeneous systems at equilibrium, periodic boundary conditions do not allow treatment of nonequilibrium systems in which the number of atoms in the domain increases or decreases with time, long range interactions are present, or bridging between regions modeled at differing length scales occurs. These include systems in which mass flows across the boundary are driven by phase change or molecular adsorption,[2] ionic liquid systems, which are dominated by electrostatic interactions,[3] and systems that permit particle fluxes between regions modeled atomistically and those modeled at mesoscopic and continuum scales.[4]

One possible approach to treat these systems is the use of open boundaries, which re-imagines the simulation domain as a core domain and a buffer region where the exterior world can "communicate" with the system.[2] The buffer region allows atoms to flow into (and out of) the computational domain from (to) its surroundings and permits accumulation (depletion) of atoms in the domain. Schemes based on adaptive resolution molecular dynamics (AdResS)[5–8] are commonly used to implement open boundaries. In these schemes, multiple different resolutions are combined by adapting the degrees of freedom on the fly. Atoms within the center region are simulated in full detail, while coarse-grained models are used to reduce the complexity of molecules in regions distant from the center. Recent developments of AdResS have improved the modeling of particle and energy exchange with the bulk/reservoir region,[9] connected systems to nonequilibrium conditions,[10,11] and significantly reduced the cost of modeling the reservoir by replacing coarse grained particles with point-like, non-interacting particles.[12,13] These improvements, along with other improvements related to insertions[14] and the use of solvent scaling,[15] have increased the overall computational performance of the method. However, further work is needed to incorporate nonlinear coupling between the system and its boundaries (e.g., Ref. 9) into nonequilibrium open boundary implementations of AdResS.[11] Grand canonical molecular dynamics (GCMD)[16,17] is another method that permits transfer of atoms into and out of a computational domain. GCMD is a hybrid modeling approach that uses MD to advance atomic trajectories and Monte Carlo (MC) to implement atomic insertions and deletions. Grand canonical MD has been used to model the behavior of a variety of systems in which mass is accumulated or depleted, including gas adsorption-induced stress in flexible nanoporous materials[18] and diffusion-induced reactions.[19]

One drawback of GCMD is that particle insertions and deletions can be costly. In dense phases, the probability of finding a cavity for an energetically favorable insertion and the probability of performing an energetically favorable deletion can become extremely small.[20] For this reason, many Monte Carlo steps are often required to attain suitable configurations, leading to significant computational expense. The perturbation caused by insertions and/or deletions can also require additional simulation time for the system to relax.[20] Additionally, the Monte Carlo code developed for hybrid simulations with molecular dynamics can be extremely difficult to parallelize and may not be able to take advantage of multicore performance.[21]

Machine learning with neural networks allows new possibilities for computationally efficient open system MD simulations. It has been used to develop interatomic potentials[22–24] to coarse grain simulations[25–27] and to enhance long timescale molecular dynamics.[28] In addition, it has been used to classify trajectory data[29] and to use such data to analyze the water droplet contact angle and the hydrogen bond network.[30] The increasing prevalence of machine learning-enabled molecular dynamics has led to the development of packages designed to generate easily implementable machine learning code (JAX MD).[31] Additionally, neural networks have been used to recreate the force imposed on each atom in a simulation domain by atoms outside the domain.[32] This force, termed the boundary force because it acts on atoms just inside (within the cutoff radius of) the domain boundary, must be included in open boundary MD simulations in order to account for the pressure exerted by the surroundings on the atoms in the domain. Boundary forces have previously been accounted for in MD simulations using various methods including stochastic boundary conditions,[33] an empirical formula fit to MD data that depends on temperature, density, and the atom's distance from the boundary,[34] and AdResS.[6]

Stochastic boundary conditions and empirical formulas derived assuming static lattice positions provide open boundary force descriptions that do not incorporate the full configurational complexity of atoms outside the domain. The reduced complexity inherent in such methods may lead to reduced physical realism in simulations employing these boundary force descriptions. The AdResS method circumvents this by explicitly and efficiently modeling a coarse-grained exterior region that directly produces forces on the central region of interest but requires analytically tractable boundary descriptions. Neural networks have no such requirement and, thus, offer the potential to treat a broader range of situations than AdResS, for example, local nonlinear fluctuations of temperature, density, or pressure, ionic surfaces, or porous media.

For this reason, the use of neural networks to apply boundary forces[32] is a promising approach for open boundary MD simulations. To take advantage of the flexibility of neural networks and the computational efficiency of AdResS, it is envisioned that the two could ultimately be combined in a hybrid code that switches the neural network "on" under nonlinear or other complex boundary configurations and switches AdResS "on" under equilibrium or analytically tractable conditions. However, care must be taken to incorporate all important physical processes in the neural network simulations. In Ref. 32, reflecting boundaries are used to prevent particles from leaving the domain, meaning that the boundaries are not open and also that an artificial impulse force is imparted on the reflected particles. Additionally, the neural network in Ref. 32 is trained solely on the distance of particles from one boundary, which does not appropriately account for situations where multiple boundaries are within the vicinity of an atom. Finally, it does not incorporate the effects of particle configuration outside the domain on the forces applied to atoms inside the domain, which as discussed above may be needed to provide more physically meaningful simulations.

In this paper, we implement open boundaries and incorporate the pressure exerted by the external fluid by applying neural network-predicted forces to atoms just inside the boundary of the simulation domain. The neural networks are trained on both distance from the boundary and the net force exerted on each atom by other atoms within the domain. Inclusion of the net force is important because this feeds configurational information from particles in the system back to the neural network used to compute boundary forces. As such, our method provides a complementary nonlinear coupling approach to that described in Ref. 9. The training data are obtained from equilibrium bulk fluid simulations with periodic boundary conditions. Section II provides an overview of the method used in this paper. Section III details the data sampling and flux calculation process, and Sec. IV discusses the construction of the neural network. Section V details how the modeling framework is integrated in the MD software package LAMMPS.[35] Section VI reports the accuracy and computational efficiency of the framework as compared to existing methods, and Sec. VII presents concluding remarks and directions for future work.

II. OVERVIEW OF THE METHOD

The goal of this work is to develop a computationally efficient method to model an open boundary MD system that is surrounded by a bulk fluid. This is done by replacing the explicitly modeled image atoms in a periodic simulation [Fig. 1(a)] with a featureless bulk whose effect on the atoms in the simulation domain is represented by boundary forces and fluxes [Fig. 1(b)]. The boundary fluxes in Fig. 1(b) represent atoms that flow into and out of the simulation domain through the open boundaries. Unlike periodic boundaries, which require the same atom that leaves the domain to re-enter on the other side [Fig. 1(a)], distinct particles flow into and out of the domain by being inserted near the boundary and by being deleted after they cross it. The boundary force in Fig. 1(b) represents the combined force from all image atoms on an atom within the central simulation box.

While forces can be analytically calculated for situations in which external particle positions are fixed,[34,36] doing so does not capture the variations in particle position over time nor the time-dependent forces resulting from these variations. A neural network can easily handle these variations and has been adopted in the present work.

III. DATA SAMPLING

The first step in constructing the neural network is to create a representative bulk system to form a dataset to learn from. To do this, we use a Lennard-Jones system with periodic boundary conditions to emulate bulk fluids at dimensionless temperatures and densities ranging from 1 to 1.1 and 0.05 to 0.8, respectively.

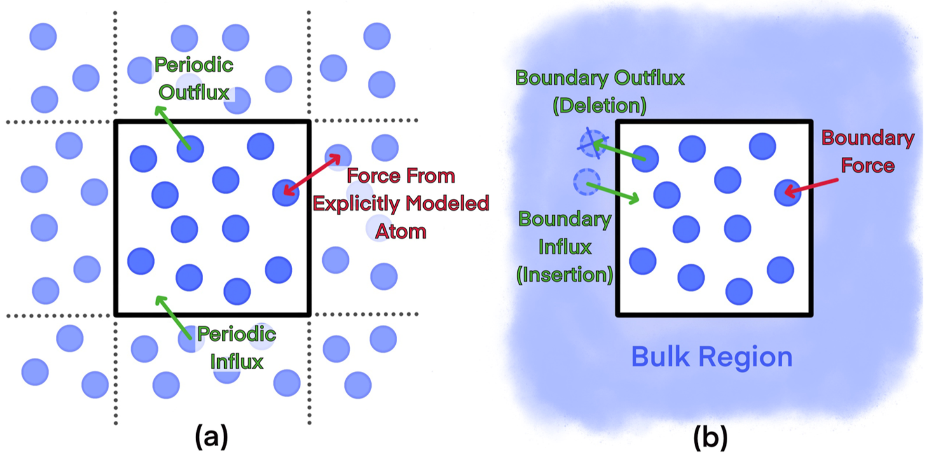
Periodic Outflux
Force From Explicitly Modeled Atom
Periodic Influx
(a)
Boundary Outflux (Deletion)
Boundary Force
Boundary Influx (Insertion)
Bulk Region
(b)

FIG. 1. Molecular simulation domain (central box) surrounded by bulk fluid modeled as (a) periodic image atoms and (b) the featureless continuum used in this work. Atomic forces and fluxes are handled differently in (a) and (b).

It is important to sample the system in many different configurations so that the training dataset provides a reasonably realistic representation of the NVT ensemble. This was done by recording various atomic quantities, including atom position, velocity, and force. Among these quantities, only the distance from the atom to a boundary and the force exerted on the atom from other atoms inside the domain (F_{in}) had any clear effect on the mean squared error of neural network predictions. Accordingly, these two quantities were employed to construct the neural networks used to produce all data reported in this paper.

In addition to the force from other atoms inside the domain, each atom also experiences a force from the bulk fluid outside the domain. In simulations with periodic boundary conditions, this latter force is imparted by the periodic images of atoms within the cutoff radius of the boundary. Boundaries have no physical meaning in periodic simulations, so the division between "inside" and "outside" need not be limited to the specific numerical coordinates used for bookkeeping purposes to determine whether an atomic position must be wrapped back into the simulation domain. Accordingly, imaginary boundaries can be defined anywhere in the domain. To sample the "inside" and "boundary" forces, we choose a single random atom (atom *i*) and then define three artificial orthogonal "boundary" planes at randomly chosen positions where at least one boundary is within the cutoff distance of the atom. These planes divide space into octants that are useful for classifying the forces and are chosen for convenience to be normal to the x, y, and z coordinate directions. Forces from atoms that are in the same octant as atom i are summed to compute the "inside" force, and forces that are in any of the other seven octants are summed to compute the "boundary" force. Figure 2 illustrates this concept in two dimensions with the yellow shaded region indicating the inside quadrant (octant).

NVT ensemble simulations were performed on cubic domains with 512 000 atoms and side lengths varying from 86 to 468 Lennard-Jones units to achieve the desired densities. Atom positions were initialized in simple cubic lattice structures corresponding to the density of interest. The Nose–Hoover thermostat with a damping parameter of 1.0 was applied to control the system temperature. Each simulation was run for 100 000 time steps until the system equilibrated, with a time step of 0.005 Lennard-Jones time units. Once equilibrated, the simulations were run with sampling occurring every 5 time steps. In order to ensure proper sampling, 200 different random seeds were used to initialize the atomic velocities for each density and temperature combination. The size of the system was selected to mitigate size effects. This was done by increasing the volume at each density until the standard deviation of time traces of system energy, temperature, average system velocity, and pressure displayed little change with system size.

In addition to sampling forces and distances from the boundary, our method samples other data related to boundary fluxes.

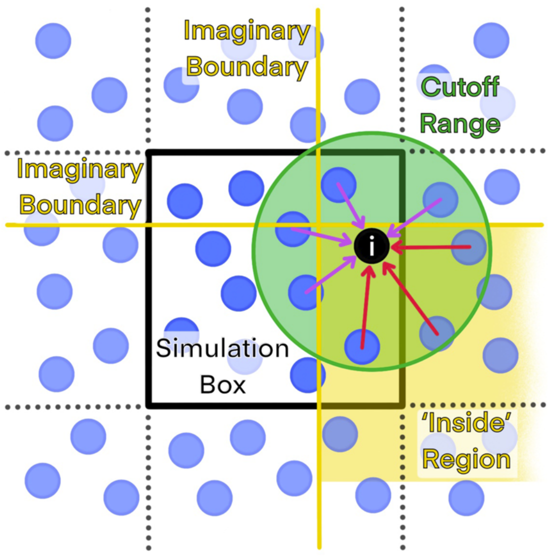

FIG. 2. Visualization of the sampling method used to determine boundary forces. Imaginary "boundaries" are placed at random near atom i to define an "inside" region that is distinct from the main simulation box. Forces on atom i from other atoms within the cutoff radius are summed separately to compute F_{in} (red arrows) and the "outside" force F_{NN} (purple arrows).

Three sets of values are collected: the velocities of atoms that cross the boundary, the distances that these atoms travel beyond the boundary in one time step, and the frequency with which atoms cross the boundary. During the simulations described above, every instance of an atom crossing an arbitrarily selected plane in the simulation domain is tracked. While any choice of plane would yield the same results in our homogeneous, equilibrium system, we chose the plane x = 1 for convenience. The velocity and penetration depth of the atom are recorded at the first time step after crossing the boundary. This information is compiled for each simulation set to form velocity and penetration depth distributions. Gaussian estimates of these distributions are created, and an average number of atoms crossing per time step is calculated. While these distributions can also be predicted using the Maxwell–Boltzmann velocity distribution and were found to agree with such predictions, it is convenient to sample them directly since sampling is already being performed to build the training dataset for the neural network. The computational cost to collect the flux information while running simulations for the neural network data collection is insignificant.

IV. NEURAL NETWORK CREATION

The data extracted from the simulations are incorporated into the machine learning software Pytorch[37] in order to construct the neural networks. The objective is to build a set of three fully connected neural networks for each thermodynamic state point of the fluid that predict the force exerted by periodic image atoms outside the simulation domain (F_{NN}) on atoms inside the domain. The decision to create one network set per state point was made to improve accuracy and performance. Specializing the network allowed for a smaller overall network and focused the network on predicting in a more narrowly defined space.

Depending on where they are located in the computational domain, atoms may interact with zero, one, two, or three boundaries. No neural networks are constructed for atoms far from the three boundaries because the boundary forces are identically zero in this case. For atoms within a cutoff distance of at least one boundary, three different kinds of networks are constructed (Fig. 3). These include a "side" network for atoms that interact with only one boundary, an "edge" network for atoms that interact with two, and a "corner" network for atoms that interact with three. The reason for constructing three separate networks is to reduce the complexity and computational cost of the network by only including variables necessary for the force predictions. In addition, these aim to reduce any artifacts imposed by the edges and corners of the domain. For example, for atoms in the side region, only one boundary contributes to the boundary force, and thus, the only parameter with predictive value is the distance from that boundary (Δx). Δy and Δz, being larger than the cutoff, are unnecessary for force predictions, and, thus, there is no need to include them in the network.

Each network is a fully connected network built with four layers, each with 30 nodes (Fig. 4). Two sets of MD data were established to create the network: the training set, which is actively used to build the network, and the test set, which is used to evaluate the network after it has been built. The size of the network was determined by testing different sizes and observing the error of both test and training sets to ensure accuracy and to make sure no over fitting occurs. Each node in the network represents a matrix of weights that govern how the node converts an input matrix into an output matrix. Each row of the matrix corresponds to data sampled for a single atom at a single time step for a single initialization seed. The number of rows in the matrix represents the total number of different samples (different atoms, time steps, and seeds) fed into the network. This is done in conjunction with a ReLU activation function that describes the output as the following:

$$output = \max(0, (N_{i,j} * input)). \tag{1}$$

Here, $N_{i,j}$ is the matrix of weights corresponding to the *j*th node of the *i*th layer, whose weights are initialized using standard PyTorch implementation. The input data, composed of the components of F_{in}, the distance of the atom from the first boundary (Δx), and, if applicable, the distance of the atom from the second and third boundaries (Δy and Δz), are forward propagated through the network, and the outputs (predicted boundary forces F_{NN}) are compared to the known boundary forces that have been directly computed from MD training data. The mean squared error, computed from the difference between the predicted and known boundary force values, is calculated, and the error is backpropagated through the network using autograd implementation in PyTorch.[37] As this backpropagation occurs, node values are updated by an incremental amount to train the network. This process is repeated until the error levels out, and a minimum is reached.

These combined networks give the total force (F), in three Cartesian coordinates, on an atom in neural network-assisted MD,

$$F = F_{in} + F_{NN}, \tag{2}$$

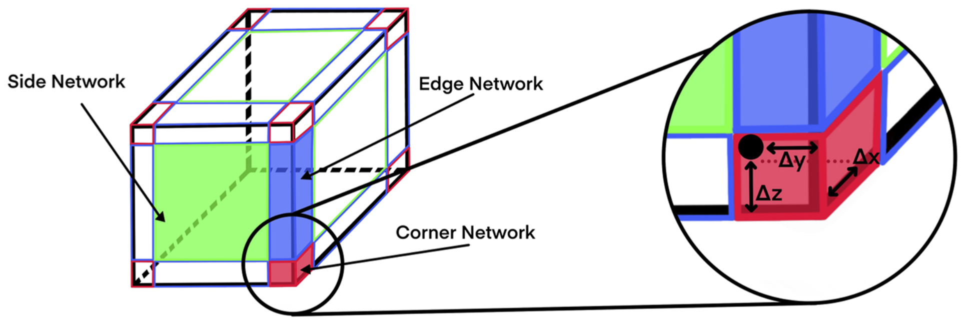

FIG. 3. Regions represented by side, edge, and corner networks. In total, there are 26 regions where a network is applied. The center region is not affected by atoms outside the simulation box, so F_{NN} is always zero and no network is needed. The zoomed-in view shows how Δx, Δy, and Δz are mapped from an atom to the corresponding boundary.

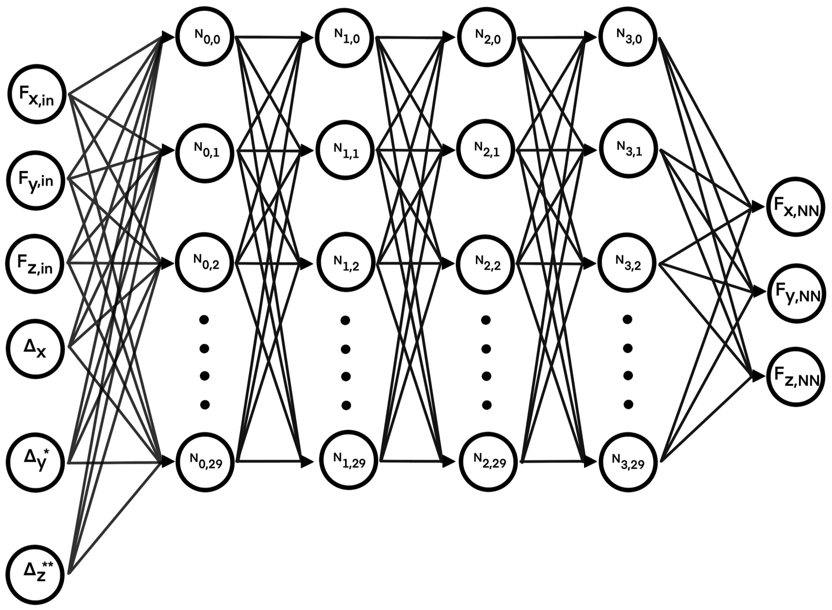

FIG. 4. General structure of the neural networks incorporated into open boundary molecular dynamics simulations. Each network is built for a specific temperature and density. $^*\Delta$y is included in the edge and corner networks only. $^{**}\Delta$z is included in the corner network only.

$$F_{NN} = F_{side}(F_{x,in}, F_{y,in}, F_{z,in}, \Delta x) \quad \text{if the atom is in a side region,} \tag{3}$$

$$F_{NN} = F_{edge}(F_{x,in}, F_{y,in}, F_{z,in}, \Delta x, \Delta y) \quad \text{if the atom is in an edge region,} \tag{4}$$

$$F_{NN} = F_{corner}(F_{x,in}, F_{y,in}, F_{z,in}, \Delta x, \Delta y, \Delta z) \quad \text{if the atom is in a corner region.} \tag{5}$$

Like F_{NN}, potential energy also has contributions from particles outside the boundary. While not needed to advance particle positions, potential energy is often of interest in MD calculations. To implement potential energy calculations in neural network-assisted MD, a set of networks was constructed with the same structure as those in the F_{NN} calculations but with the output being potential energy instead of three force components. These neural networks are isolated so that they are only called when the user requests the system's potential energy.

V. IMPLEMENTATION IN LAMMPS

A. Boundary force

LAMMPS[35] was modified to incorporate the neural net boundary force calculation and the open boundary condition. F_{in} is computed using the standard pairwise force calculation procedure in LAMMPS. To compute F_{NN}, a new code has been implemented in which each individual processor checks each of its atoms to determine whether that atom falls into one of the 26 regions where a neural network applies. This is performed via a series of nested logic statements. If an atom is not in one of these regions, the atom's force value is unmodified as the resultant force added from the boundary would be zero.

If the atom is located in a region where a neural network force should be applied, a transformation of the data is made via rotation, reflection, and translation as needed to align with the neural network coordinate system. The original network is referenced from the point (0,0,0) with vectors pointing in the positive directions of each coordinate. Then, the atom information is passed to the code for the neural networks, which are custom coded into LAMMPS as a series of addition, multiplication, and if statements involving the elements of all nodal matrices in the network. This code replicates the structure and nodal values of the Python neural network, which was separately trained as discussed in Sec. IV. This structure can be run on individual processors without affecting the ability of the code to be multithreaded. Once the neural network code is run, the force components are transformed to ensure that they are applied in the correct direction. These forces are applied to each atom and this total force is used in the next step of the velocity Verlet method.

B. Open boundary

The open boundaries are implemented by removing the periodic boundary conditions, which allows atoms to exit the simulation box permanently instead of forcing them to wrap around and re-enter it and by incorporating a new particle insertion process at the boundary that mimics the cross-plane atomic fluxes that naturally occur in fluids. This approach removes the artificial constraint that the number of atoms in the simulation box remains perfectly constant at each time step while also maintaining a stable time-averaged number density. Particles that leave the domain of the simulation box are simply removed in the following time step. The particle insertion was coded by modifying the "fix deposit" code currently in LAMMPS. In this code, particles are inserted at random locations in a region with velocities sampled from a Gaussian distribution. The modified code uses positions, velocities, and insertion rates sampled from the training data.

The atom insertion rate is chosen based on the output of the flux sampling procedure described in Sec. III, instead of the standard "fix deposit" procedure in which m atoms are inserted every n time steps (m and n are set by the user). In the present approach, the sampled number of atoms crossing a boundary in a given time step is inverted to obtain the number of time steps between insertions. As this number is, in general, not an integer, a counter is used at each of the six boundaries of the simulation domain to track when an atom should be inserted. This counter increments by one each time step until it surpasses the determined number of time steps per insertion. Then, an atom insertion is attempted. If successful, the counter is decreased by the number of time steps per insertion. If unsuccessful, a new insertion is attempted until success is achieved, up to a user-defined number of trials. If the trial limit is reached, the system is advanced by one time step in an attempt to obtain a more favorable configuration, and the insertion is attempted again. The user must be careful to set a trial limit large enough to permit successful insertions at the appropriate time step, but small enough to prevent the system from getting "stuck" due to an unfavorable atomic configuration. In this work, the trial limit was set to 2000. In higher density fluids, there will be situations in which multiple atoms are inserted per time step. In this case, multiple insertions take place until the counter is less than the value for time steps per atom insertion. An insert is deemed successful if the inserted atom is farther than a specified set distance from neighboring atoms. Inspired by the radial distribution function, we set this distance to 1.0 Lennard-Jones units.

For each face, the y and z coordinates of the inserted atom are selected from a uniform distribution across the entire face because bulk systems should display no bias toward any specific position parallel to the face. The x coordinate, which represents the distance of the particle from the boundary, is selected from a Gaussian distribution fit to the training data above. This distance is typically extremely small such that particles always insert close to the boundary. This Gaussian is centered at zero because it is based on the positions of atoms immediately after they cross a plane in either direction. To ensure that the atom is placed inside the simulation domain, the absolute value of the selected x value is chosen. For velocity, the y and z components are both chosen from a Gaussian distribution centered around zero that was fit to the corresponding components from the training data. The same Gaussian is used for both components because the distribution is the same in y and z directions. The x component of velocity is selected from a Gaussian fit to the relevant training data. As with the x-position, the absolute value of the sampled x-velocity is taken because inserted atoms can only cross the boundary if they travel in the positive x-direction. It should be noted that the present method, like other open boundary molecular dynamics methods and in contrast to many Monte Carlo methods, is not set up to enforce detailed balance during particle insertion and deletion processes.

The insertions and deletions occur in a different part of the code from the force calculation. Because the neural network inputs are updated every time step, the network can appropriately change its prediction if an atom is removed from one of the boundary regions.

VI. RESULTS AND DISCUSSION

In order to show that the neural network-assisted open boundary method is able to reproduce the behavior of bulk fluid systems, we computed various system metrics with our method and compared them to those computed using conventional molecular dynamics simulations with periodic boundary conditions. The metrics being compared include radial distribution function, atom count, kinetic and potential energy, pressure, and excess chemical potential. Excess chemical potential was calculated using the Widom insertion method found within LAMMPS.[35] Using the configuration in Fig. 1(b) for our method and in Fig. 1(a) for periodic boundaries, we performed simulations with a time step of 0.005 dimensionless Lennard-Jones time units on domains initialized with the same box size, number of atoms, atomic lattice positions, and initial velocity distributions. Atomic velocities were initialized and maintained throughout the simulations using a Nose–Hoover thermostat with a damping parameter of 1.0 for the periodic simulations and a Langevin thermostat for the open boundary simulations. The Langevin thermostat damping parameter varied depending on the state point but typically ranged from 0.01 to 0.1. The Langevin thermostat was used for the open boundary system instead of the Nose–Hoover thermostat from the periodic as it handled the change in N with significantly greater success. At each of the eight temperature/density state points studied in this work, 25 different seeds were chosen and the results were averaged.

Each simulation was equilibrated for 50 000 time steps, and then, the metrics above were computed for 100 000 time steps and averaged. The average quantities for the open boundary and periodic simulations are given in the Appendix, and the error in the open boundary simulations, computed as $100 \times (M_{PBC} - M_{Open})/M_{PBC}$, where M represents the metric of interest, is given in Table I.

Overall, the neural network-assisted open boundary method performs very well, showing good agreement with periodic simulations across multiple performance metrics. Temperature, number of atoms in the simulation box, kinetic energy, and pressure from the open boundary simulations are very close to those from the periodic simulations with differences well below 1% for all state points. The open boundary potential energy values also agree well, displaying errors within 2.6%. They are, however, consistently higher than the periodic values, leading to negative error percentages in Table I. It is important to note that the potential energy computed in the open boundary simulations was directly output from a neural network specifically trained and tested on potential energy. As discussed in Sec. IV, this additional network is necessary to account for the direct contributions to potential energy from atoms outside the simulation box. The other physical properties in Table I do not require additional networks because their values do not directly depend on the configurations of atoms outside the box. They only require the correct system pressure, which the force network that produces F_{NN} provides already. For this reason, we believe that the small negative errors in potential energy arise from the potential network itself, and not from the force network, which does not cause systematic errors in any other properties. The errors in potential energy are more pronounced for the higher density (liquid) simulations than the lower density (gas) simulations. This likely occurs because a larger fraction of the total liquid atoms is located in the boundary region in the present simulations. Thus,

TABLE I. Percent error of system metrics for the open boundary simulation.

System parameters		Error of system metrics (in %)					
Set temperature	Set density	Temperature	Atom count	Kinetic energy	Potential energy	Pressure	Excess chemical potential
1	0.005	−0.0023	−0.0989	0.0144	−0.6473	−0.1015	1.0462
1	0.01	−0.0590	−0.3903	0.0262	−0.5254	−0.3917	0.3046
1.05	0.01	0.0353	0.0697	0.0028	−0.9514	0.0281	−0.5450
1.1	0.01	−0.0023	−0.0608	0.0024	−0.8267	−0.0964	−1.1594
1	.75	0.0337	0.0523	0.0087	−2.4913	−0.8171	−4.1856
1	0.8	0.0248	−0.0514	0.0044	−2.4785	0.1243	−2.7487
1.05	0.8	0.0059	−0.0920	0.0053	−2.4375	0.0410	−78.5185
1.1	0.8	0.0256	0.0204	−0.0028	−2.5291	0.9431	6.2322

more liquid atoms than gas atoms are affected by the potential energy network.

Excess chemical potential also shows good agreement, except for the data point at $T^* = 1.05$, $\rho^* = 0.8$. Our explanation of the spike in percent error at this point is as follows: We believe that the training error of the neural network sets a lower bound on the magnitude of the difference between properties computed using open and periodic simulations (including the excess chemical potential difference $\mu_{excess, PBC} - \mu_{excess, Open}$). When this difference is comparable to the periodic reference value $\mu_{excess, PBC}$, which is almost zero at this state point (Table II), the error becomes large.

Additionally, the open boundary simulation should produce the same fluid structure as that produced by the periodic simulation. Using OVITO,[38] the radial distribution function was plotted from a snapshot taken at a single time step for periodic and open boundary systems. Figure 5 shows that these sampled radial distribution functions are nearly identical for the two systems.

Local densities were also computed by slicing the simulation box into subvolumes and computing the number of atoms per subvolume. Figure 6 shows the local density vs position, averaged across each of the three coordinate directions. It can be seen that the open boundary simulations produce local densities that agree very well with those from the periodic simulations along the length of the box, with almost imperceptible increases in density very near the boundaries of the box.

The probability distribution of the number of particles in identical system volumes was also compared for open and periodic boundary systems, following a similar procedure to that in Ref. 39. The periodic data were obtained by placing a box with the same volume as that used in the open system into a larger (1,000 000 atom) periodic system. The particle count in the box was tabulated at each time step and a Gaussian was fit to the data. Since the mean atom count had previously been compared, it was subtracted from the distributions to elucidate the spread in N. Figure 7 shows that distributions for each of the test cases compare favorably, with minor differences in distribution width that are in all cases less than 0.05% of the average particle number. These differences do not result in any significant error in observable metrics as seen in Table I.

The computational efficiency of the two systems was also compared. As discussed above, grand canonical molecular dynamics can be used with open boundaries but can be computationally taxing. When open boundaries exist, atoms are constantly leaving the system and this puts strain on the Monte Carlo method to continually insert particles.

Table III compares the simulation time of our method to that of the grand canonical molecular dynamics approach (GCMD)

TABLE II. Excess chemical potential data.

System parameters		Excess chemical potential			
Set temperature	Set density	Periodic boundaries	Open boundaries	Difference	Percent error (in %)
1	0.005	0.393	0.389	0.004 11	1.046
1	0.01	0.549	0.547	0.001 67	0.304
1.05	0.01	0.371	0.373	−0.002 02	−0.545
1.1	0.01	0.394	0.398	−0.004 57	−1.159
1	0.75	−1.418	−1.359	−0.059 3	−4.185
1	0.8	−0.470	−0.457	−0.012 9	−2.748
1.05	0.8	−0.0681	−0.0146	−0.053 5	−78.518
1.1	0.8	0.376	0.353	0.023 4	6.232

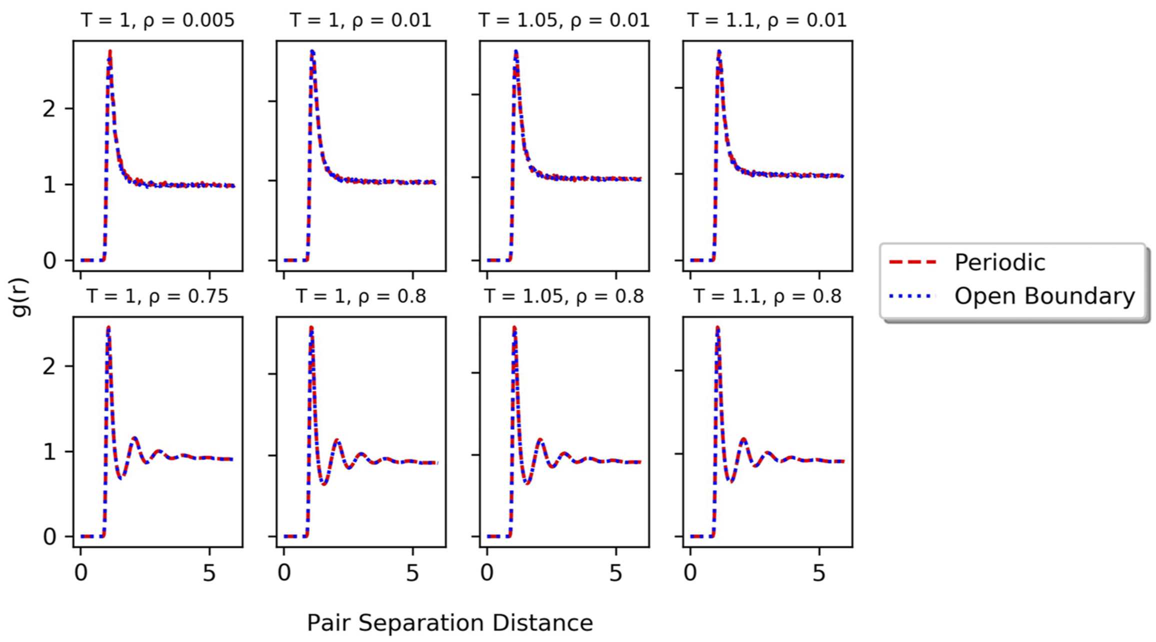

FIG. 5. Radial distribution functions for various state points in periodic and neural network-assisted open boundary simulations. Figures were produced using OVITO.[38] For all state points, the two boundary conditions produce distributions with no noticeable differences.

available in LAMMPS. Open boundaries and identical system sizes (50 653 atoms) were used in both cases for eight different state points. Ten different random number seeds for each system type were used to initialize the velocity distribution at each state point and the same initial configurations were used for each seed. Each system was run for 10 000 time steps in periodic boundary conditions to allow the system to come to equilibrium. This state was saved and used as the starting point for the open boundary simulation time

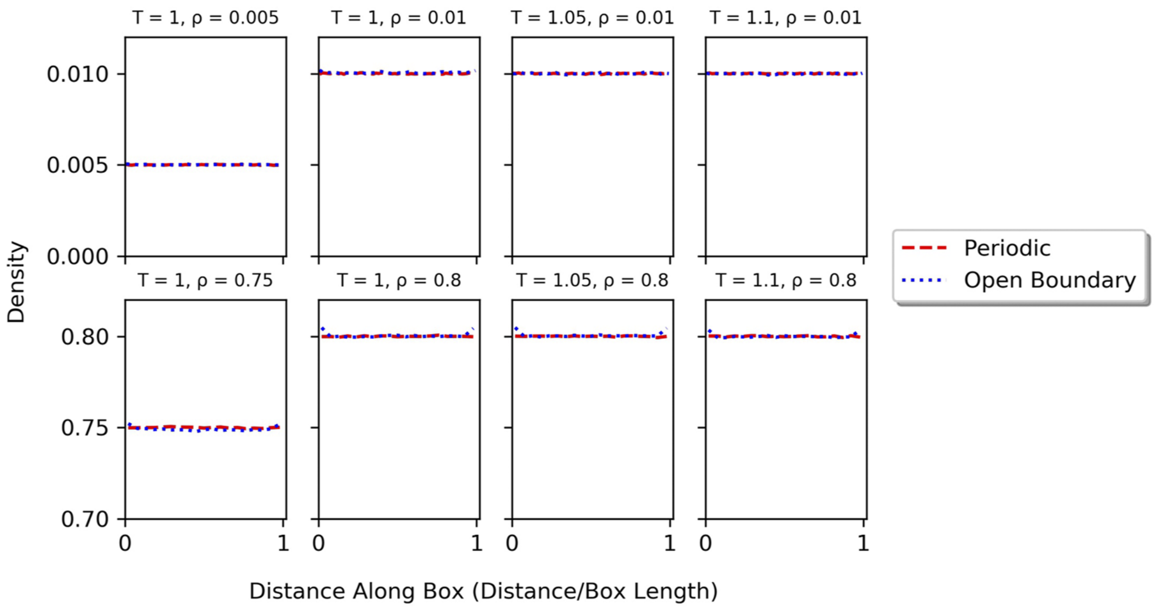

FIG. 6. Local density vs position in the simulation box. Each graph averages the density in each of the three coordinate directions.

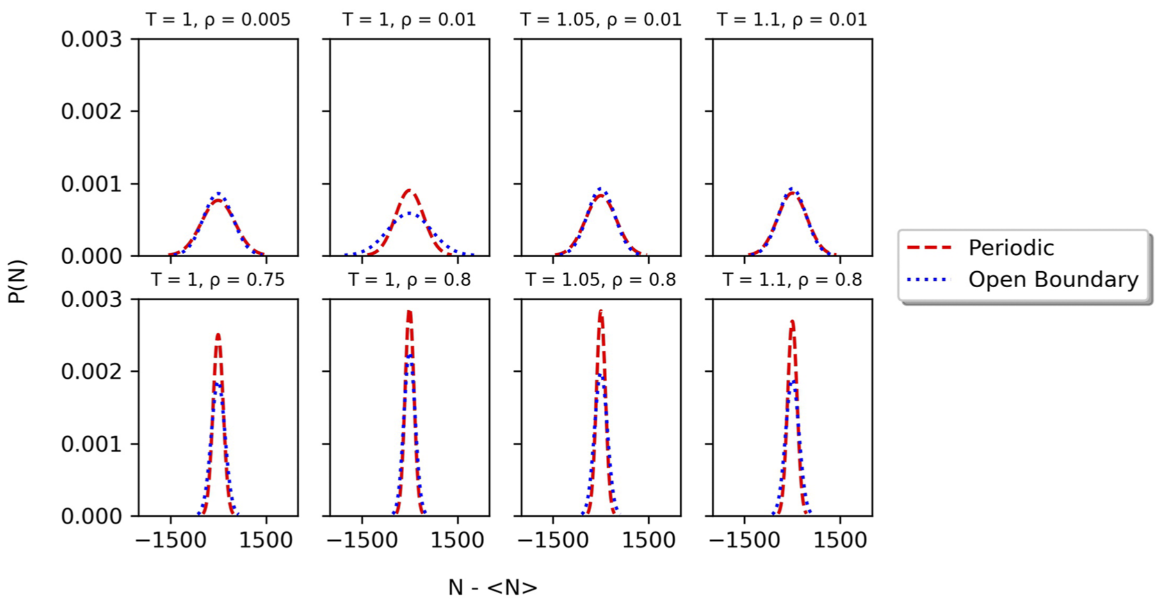

FIG. 7. The probability of the number of particles for equal volume boxes in both periodic and open boundary conditions. Each graph shows Gaussian estimates of the distribution with the mean subtracted.

trials. The systems were then run for 10 000 time steps using the two open boundary methods. The runtime to complete 10 000 time steps was averaged over the ten seeds at each state point and reported in Table III for both methods.

The key difference between the methods is the location and method of particle insertion. GCMD inserts particles throughout the entire box, using an insertion probability drawn from a chemical potential set to maintain the atom count at its initial value. The set value of chemical potential was found by trial and error. In contrast, our method inserts particles near the boundary at a rate sampled from periodic simulations as previously described.

The data show that our method performs over two orders of magnitude faster than GCMD. There are several potential reasons for this. First, the code that evaluates whether an insertion is accepted or rejected is much more efficient in our method than in GCMD. This is because our modified "fix deposit" code only requires an evaluation of the inserted particle's position relative to its nearest neighbor, while GCMD requires calculation of the energy of the entire system's atomic configuration to evaluate the probability of insertion. The use of the cutoff in the present Lennard-Jones GCMD simulations reduces this calculation to only neighbors within a cutoff distance; however, this is still a more intensive calculation than the evaluation of distance to the nearest neighbor. In addition, our method's use of prior information about where an atom insert gives rise to more frequent success, which dramatically cuts down on the time it takes to insert the needed number of atoms. Finally, the Monte Carlo method must check for deletions as well, therefore, half of all attempts are not leading to the desired insertion.

TABLE III. Open boundary simulation times for neural network-assisted and grand canonical molecular dynamics methods.

System parameters		System metrics		
Set temperature	Set density	NN time (min)	GCMD time (min)	Time ratio
1	0.005	0.233	26.525	113.678
1	0.01	0.266	34.098	127.868
1.05	0.01	0.253	32.233	127.236
1.1	0.01	0.26	58.920	226.615
1	.75	1.903	279.408	146.799
1	0.8	2.296	254.238	110.698
1.05	0.8	2.315	351.68	151.913
1.1	0.8	2.308	336.508	145.779

VII. CONCLUSIONS AND OUTLOOK

This work introduces a neural network-assisted method to perform open boundary simulations. The method samples position, velocity, and force data from periodic boundary condition NVT simulations in order to (1) train neural networks that predict the "missing" forces exerted by bulk fluid atoms outside the simulation domain and (2) calculate the position and velocity distributions of particles that are inserted at the boundaries to maintain the proper mass influx. LAMMPS code was modified to implement the method using neural networks generated by PyTorch from LAMMPS training data. The method yields physical observables that are almost identical to those generated from standard periodic boundary condition molecular simulations, and its efficiency in handling particle insertions leads to a two order of magnitude speedup. The method embeds a molecular dynamics simulation domain within an unmodeled bulk fluid reservoir, enabling explicit atomistic simulation of transport processes in local regions of interest. While applied here to Lennard-Jones systems in equilibrium with their surroundings, the method could be extended to model fluids with long-range interactions to handle mass flows from the domain to its surroundings and to treat complex situations involving nonlinearities.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation MRSEC program (Grant No. DMR-1720530) and the Ashton Fellowship program. This work used the Extreme Science and Engineering Discovery Environment (XSEDE),[40] which was supported by the National Science Foundation under Grant No. ACI-1548562. Specifically, it used the Bridges system,[41] which was supported by the NSF under Award No. ACI-1445606, at the Pittsburgh Supercomputing Center (PSC).

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

APPENDIX: ADDITIONAL DATA

Tables IV–VIII contain raw numerical values for different properties used in Table I. The entries in each table are averaged from 25 different seeds. All units are in dimensionless Lennard-Jones units, unless otherwise described.

TABLE IV. Temperature data.

System parameters		Temperature			
Set temperature	Set density	Periodic boundaries	Open boundaries	Difference	Error (in %)
1	0.005	1.000 02	1.000 04	−0.000 0228	−0.0023
1	0.01	1.000 0007	1.000 5	−0.000 590	−0.0590
1.05	0.01	1.049 9	1.049 6	0.000 370	0.0353
1.1	0.01	1.099 9	1.100 003	−0.000 0256	−0.0023
1	0.75	1.000 04	0.999 7	0.000 337	0.0337
1	0.8	0.999	0.999 7	0.000 248	0.0248
1.05	0.8	1.050 02	1.049 95	0.000 0629	0.0059
1.1	0.8	1.099 9	1.099 7	0.000 281	0.0256

TABLE V. Atom count.

System parameters		Atom count			
Set temperature	Set density	Periodic boundaries	Open boundaries	Difference	Percent error (in %)
1	0.005	512 000	512 506.304	−506.304	−0.0989
1	0.01	512 000	513 998.337	−1998.337	−0.3903
1.05	0.01	512 000	511 642.871	357.128	0.0697
1.1	0.01	512 000	512 311.560	−311.560	−0.0608
1	0.75	512 000	511 732.112	267.887	0.0523
1	0.8	512 000	512 263.148	−263.148	−0.0514
1.05	0.8	512 000	512 471.176	−471.176	−0.0920
1.1	0.8	512 000	511 895.154	104.845	0.0204

TABLE VI. Kinetic energy data.

System parameters		Kinetic energy			
Set temperature	Set density	Periodic boundaries	Open boundaries	Difference	Percent error (in %)
1	0.005	1.500	1.499	0.000 216	0.0144
1	0.01	1.499	1.499	0.000 393	0.0262
1.05	0.01	1.574	1.574	0.000 0448	0.0028
1.1	0.01	1.649	1.649	0.000 0397	0.0024
1	0.75	1.5000	1.499	0.000 130	0.0087
1	0.8	1.499	1.499	0.000 0657	0.0044
1.05	0.8	1.575	1.574	0.000 0847	0.0053
1.1	0.8	1.649	1.650	−0.000 0477	−0.0028

TABLE VII. Potential energy data.

System parameters		Potential energy			
Set temperature	Set density	Periodic boundaries	Open boundaries	Difference	Percent error (in %)
1	0.005	−0.0430	−0.0427	−0.000 278	−0.647
1	0.01	−0.0861	−0.0856	−0.000 452	−0.525
1.05	0.01	−0.0834	−0.0826	−0.000 794	−0.951
1.1	0.01	−0.0811	−0.0805	−0.000 671	−0.826
1	0.75	−4.825	−4.705	−0.120	−2.491
1	0.8	−5.112	−4.986	−0.126	−2.478
1.05	0.8	−5.068	−4.945	−0.123	−2.437
1.1	0.8	−5.025	−4.898	−0.127	−2.529

TABLE VIII. Pressure data.

System parameters		Pressure			
Set temperature	Set density	Periodic boundaries	Open boundaries	Difference	Percent error (in %)
1	0.005	0.004 89	0.004 90	−0.000 004 97	−0.101
1	0.01	0.009 58	0.009 62	−0.000 037 5	−0.391
1.05	0.01	0.010 1	0.010 1	0.000 002 84	0.028
1.1	0.01	0.010 6	0.010 6	−0.000 0102	−0.096
1	0.75	0.991	1.000	−0.008 10	−0.817
1	0.8	1.692	1.690	0.002 10	0.124
1.05	0.8	1.934	1.933	0.000 793	0.041
1.1	0.8	2.171	2.151	0.020 4	0.943

REFERENCES

[1] D. Frenkel and B. Smit, *Understanding Molecular Simulation: From Algorithms to Applications* (Elsevier, 2001), Vol. 1.

[2] R. Delgado-Buscalioni, J. Sablić, and M. Praprotnik, "Open boundary molecular dynamics," Eur. Phys. J.: Spec. Top. **224**, 2331–2349 (2015).

[3] G. Ciccotti and L. Delle Site, "The physics of open systems for the simulation of complex molecular environments in soft matter," Soft matter **15**, 2114–2124 (2019).

[4] R. Delgado-Buscalioni, K. Kremer, and M. Praprotnik, "Coupling atomistic and continuum hydrodynamics through a mesoscopic model: Application to liquid water," J. Chem. Phys. **131**, 244107 (2009).

[5] M. Praprotnik, L. Delle Site, and K. Kremer, "Adaptive resolution molecular-dynamics simulation: Changing the degrees of freedom on the fly," J. Chem. Phys. **123**, 224106 (2005).

[6] L. Delle Site and M. Praprotnik, "Molecular systems with open boundaries: Theory and simulation," Phys. Rep. **693**, 1–56 (2017).

[7] M. Heidari, K. Kremer, R. Golestanian, R. Potestio, and R. Cortes-Huerto, "Open-boundary Hamiltonian adaptive resolution. From grand canonical to non-equilibrium molecular dynamics simulations," J. Chem. Phys. **152**, 194104 (2020).

[8] D. Mukherji, N. F. A. van der Vegt, K. Kremer, and L. Delle Site, "Kirkwood–Buff analysis of liquid mixtures in an open boundary simulation," J. Chem. Theory Comput. **8**, 375–379 (2012).

[9]L. Delle Site and R. Klein, "Liouville-type equations for the n-particle distribution functions of an open system," J. Math. Phys. **61**, 083102 (2020).
[10]R. Ebrahimi Viand, F. Höfling, R. Klein, and L. Delle Site, "Theory and simulation of open systems out of equilibrium," J. Chem. Phys. **153**, 101102 (2020).
[11]R. Klein, R. Ebrahimi Viand, F. Höfling, and L. Delle Site, "Nonequilibrium induced by reservoirs: Physico-mathematical models and numerical tests," Adv. Theory Simul. **4**, 2100071 (2021).
[12]A. Gholami, F. Höfling, R. Klein, and L. Delle Site, "Thermodynamic relations at the coupling boundary in adaptive resolution simulations for open systems," Adv. Theory Simul. **4**, 2000303 (2021).
[13]L. Delle Site, C. Krekeler, J. Whittaker, A. Agarwal, R. Klein, and F. Höfling, "Molecular dynamics of open systems: Construction of a mean-field particle reservoir," Adv. Theory Simul. **2**, 1900014 (2019).
[14]S. Thaler, M. Praprotnik, and J. Zavadlav, "Back-mapping augmented adaptive resolution simulation," J. Chem. Phys. **153**, 164118 (2020).
[15]A. Kubincová, S. Riniker, and P. H. Hünenberger, "Solvent-scaling as an alternative to coarse-graining in adaptive-resolution simulations: The adaptive solvent-scaling (AdSoS) scheme," J. Chem. Phys. **155**, 094107 (2021).
[16]S. Boinepalli and P. Attard, "Grand canonical molecular dynamics," J. Chem. Phys. **119**, 12769–12775 (2003).
[17]H. Eslami and F. Müller-Plathe, "Molecular dynamics simulation in the grand canonical ensemble," J. Comput. Chem. **28**, 1763–1773 (2007).
[18]G. Fraux and F.-X. Coudert, "Recent advances in the computational chemistry of soft porous crystals," Chem. Commun. **53**, 7211–7221 (2017).
[19]M. J. Del Razo, H. Qian, and F. Noé, "Grand canonical diffusion-influenced reactions: A stochastic theory with applications to multiscale reaction-diffusion simulations," J. Chem. Phys. **149**, 044102 (2018).
[20]H. Eslami, F. Mojahedi, and J. Moghadasi, "Molecular dynamics simulation with weak coupling to heat and material baths," J. Chem. Phys. **133**, 084105 (2010).
[21]R. F. L. Evans, W. J. Fan, P. Chureemart, T. A. Ostler, M. O. A. Ellis, and R. W. Chantrell, "Atomistic spin model simulations of magnetic nanomaterials," J. Phys.: Condens. Matter **26**, 103202 (2014).
[22]H. Chan, B. Narayanan, M. J. Cherukara, F. G. Sen, K. Sasikumar, S. K. Gray, M. K. Y. Chan, and S. K. R. S. Sankaranarayanan, "Machine learning classical interatomic potentials for molecular dynamics from first-principles training data," J. Phys. Chem. C **123**, 6941–6957 (2019).
[23]J. Behler, "Perspective: Machine learning potentials for atomistic simulations," J. Chem. Phys. **145**, 170901 (2016).
[24]A. Khorshidi and A. A. Peterson, "Amp: A modular approach to machine learning in atomistic simulations," Comput. Phys. Commun. **207**, 310–324 (2016).
[25]H. Chan, M. J. Cherukara, B. Narayanan, T. D. Loeffler, C. Benmore, S. K. Gray, and S. K. R. S. Sankaranarayanan, "Machine learning coarse grained models for water," Nat. Commun. **10**, 379 (2019).
[26]W. Wang and R. Gómez-Bombarelli, "Coarse-graining auto-encoders for molecular dynamics," npj Comput. Mater. **5**, 125 (2019).
[27]M. Ceriotti, "Unsupervised machine learning in atomistic simulations, between predictions and understanding," J. Chem. Phys. **150**, 150901 (2019).
[28]F. Noé, "Machine learning for molecular dynamics on long timescales," *Machine Learning Meets Quantum Physics* (Springer, 2020), pp. 331–372.
[29]B. K. Carpenter, G. S. Ezra, S. C. Farantos, Z. C. Kramer, and S. Wiggins, "Empirical classification of trajectory data: An opportunity for the use of machine learning in molecular dynamics," J. Phys. Chem. B **122**, 3230–3241 (2017).
[30]S. K. Singh, K. K. Bejagam, Y. An, and S. A. Deshmukh, "Machine-learning based stacked ensemble model for accurate analysis of molecular dynamics simulations," J. Phys. Chem. A **123**, 5190–5198 (2019).
[31]S. S. Schoenholz and E. D Cubuk, "JAX MD: A framework for differentiable physics," J. Stat. Mech. 124016 (2020), Vol. 33.
[32]P. Neumann and N. Wittmer, "Open boundary modeling in molecular dynamics with machine learning," *International Conference on Computational Science* (Springer, 2020), pp. 334–347.
[33]M. Berkowitz and J. A. McCammon, "Molecular dynamics with stochastic boundary conditions," Chem. Phys. Lett. **90**, 215–217 (1982).
[34]W. Zhou, H. Luan, Y. He, J. Sun, and W. Tao, "A study on boundary force model used in multiscale simulations with non-periodic boundary condition," Microfluid. Nanofluid. **16**, 1 (2014).
[35]S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," J. Comput. Phys. **117**, 1–19 (1995).
[36]H. C. Hamaker, "The London—van der waals attraction between spherical particles," Physica **4**, 1058–1072 (1937).
[37]A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., 2019), Vol. 32, pp. 8024–8035.
[38]A. Stukowski, "Visualization and analysis of atomistic simulation data with OVITO—The open visualization tool," Modell. Simul. Mater. Sci. Eng. **18**, 015012 (2010).
[39]J. Whittaker and L. Delle Site, "Investigation of the hydration shell of a membrane in an open system molecular dynamics simulation," Phys. Rev. Res. **1**, 033099 (2019).
[40]J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, *et al.*, "XSEDE: Accelerating scientific discovery," Comput. Sci. Eng. **16**, 62–74 (2014).
[41]N. A. Nystrom, M. J. Levine, R. Z. Roskies, and J. R. Scott, "Bridges: A uniquely flexible HPC resource for new communities and data analytics," in *Proceedings of the 2015 Annual Conference on Extreme Science and Engineering Discovery Environment* (ACM, 2015), pp. 1–8.