Supplementary Information: Open-Source Molecular Modeling Software in Chemical Engineering, with Focus on the Molecular Simulation Design Framework (MoSDeF)

Peter T. Cummings[1], Clare McCabe[1,2], Christopher R. Iacovella[1], Akos Ledeczi[3], Eric Jankowski[4], Arthi Jayaraman[5], Jeremy C. Palmer[6], Edward J. Maginn[7], Sharon C. Glotzer[8], Joshua A. Anderson[8], J. Ilja Siepmann[9], Jeffrey Potoff[10], Ray A. Matsumoto[1], Justin B. Gilmer[11], Ryan S. DeFever[7], Ramanish Singh[9] and Brad Crawford[10]

[1]*Department of Chemical and Biomolecular Engineering and Multiscale Modeling and Simulation Center, Vanderbilt University, Nashville, TN*

[2]*Department of Chemsitry, Vanderbilt University, Nashville, TN*

[3]*Department of Electrical Engineering and Computer Science and Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN*

[4]*Micron School of Materials Science and Engineering, Boise State University, Boise, ID*

[5]*Departments of Chemical and Biomolecular Engineering and of Materials Science and Engineering, University of Delaware, Newark, DE*

[6]*Department of Chemical and Biomolecular Engineering, University of Houston, Houston, TX*

[7]*Department of Chemical and Biomolecular Engineering, University of Notre Dame, Notre Dame, IN*

[8]*Departments of Chemical Engineering, of Materials Science, and of Physics, University of Michigan, Ann Arbor, MI*

[9]*Department of Chemistry and of Chemical Engineering and Materials Science, University of Minnesota, MN*

[10]*Department of Chemical Engineering and Materials Science, Wayne State University, Detroit, MI*

[11]*Interdiscplinary Graduate Program in Materials Science and Multiscale Modeling and Simulation Center, Vanderbilt University, Nashville, TN*

January 20, 2021

The latest version of this supplementary information can be found on the mosdef_slitpore GitHub repository. The url is: https://github.com/mosdef-hub/mosdef_slitpore/blob/master/supplementary_info/si_latest.pdf.

1 Introduction

One goal of the Molecular Simulation Design Framework (MoSDeF)[1,2] is to integrate with numerous open-source molecular simulation engines to provide more seamless interconversion between the different input formats required by each. The GROMACS[3–5] molecular dynamics (MD) software has been tightly integrated with MoSDeF since the early development of the project, leveraging ParmEd[6,7] to write the necessary input files to disk. MoSDeF has since enabled screening studies of diverse soft matter systems[8,9] that would have been otherwise difficult. LAMMPS[10], another open-source MD simulation engine, has also been supported since early development with custom readers and writers implemented in `mBuild`. Recently efforts have yielded tight integration with `Cassandra`[11], an open-source Monte Carlo (MC) code developed by the Maginn Group of Notre Dame. `MoSDeF-Cassandra`[12] is a python wrapper for `Cassandra` that utilizes `MoSDeF` for system setup and force field application and calls `Cassandra` to run MC simulations. Ongoing development efforts include the integration of GPU-Optimized Monte Carlo (`GOMC`)[13] and the `CP2K` quantum chemistry package[14].

Integrating numerous simulation engines with `MoSDeF` enables comparisons of simulation results from different codes in a reproducible fashion. As an example, we replicate portions of the calculations of water in carbon slitpores performed by Keith Gubbins and colleagues[15] with five modern simulation software packages. Using in-house code, Striolo *et al.* reported grand canonical Monte Carlo (GCMC) simulations of SPC/E water in carbon slitpores of various widths and reported water adsorption isotherms and the water structure within the pores. Here, we calculate the water adsorption isotherms with GCMC in `GOMC` and `Cassandra`. Once we determined the equilibrium number of water molecules in the pores at prescribed external pressures, the structure of water inside the pores was calculated using the following: MC simulations in the canonical (NVT) ensemble with `GOMC` and `Cassandra`, force-field-based NVT MD simulations with `GROMACS` and `LAMMPS`, and first-principles NVT MD simulations with `CP2K`.

2 Methods

Systems were generated with `mBuild`[16] and `foyer`[17,18]. The details of each system are provided in Table 1 and snapshots of each system are provided in Fig. 1. The slit pores were defined by layers of graphene with interlayer spacings of 0.335 nm with a carbon–carbon distance of 0.142 nm. All carbon atoms were fixed during the simulations. Periodic boundary conditions were applied in all three spatial dimensions. Vacuum space was added in the direction normal to the pore. Simulation details specific to each simulation package are detailed in the subsequent sections. Thorough installation and simulation instructions are provided at the end of this document. All the files required to run the simulations reported can be downloaded from the `mosdef_slitpore` GitHub repository[19].

Table 1: Systems evaluated in this study. Dimensions are provided in nm.

System	Pore width	Box dimensions	Cutoff	$N_{Graphene}$	N_{water} NVT
large-1.0	1.0	$2.947 \times 2.978 \times 6.0$[a]	0.9	6	-
large-1.6	1.6	$2.947 \times 2.978 \times 6.0$[a]	0.9	6	-
large-2.0	2.0	$2.947 \times 2.978 \times 6.0$[a]	0.9	6	485
small-1.0	1.0	$0.982 \times 1.063 \times 2.0$	0.49	2	24
small-1.0	1.0	$0.982 \times 1.063 \times 2.0$	0.49	2	1

[a] GOMC dimensions differed in z-direction. See section 2.2 for details.

2.1 Force-field-based simulation details

Water was described by the SPC/E[20] model. The carbon–carbon Lennard–Jones (LJ) parameters were taken from Table 1 of Striolo *et al.*, and Lorentz-Berthelot[21,22] combining rules were used to define the carbon–water LJ interactions. LJ and Coulomb interactions were truncated at 0.9 nm (unless otherwise noted in Table 1), with long-range Coulomb interactions computed via an Ewald summation or grid-based counterpart.[1] No analytical tail corrections were applied to the Lennard-Jones potential. There are technical details that differ between the original simulations of Striolo *et al.* and the work repeated here. Thus, our goal is less to reproduce the original work of Striolo *et al.*, and instead demonstrate the power of MoSDeF to show consistency between different simulation engines (GROMACS and LAMMPS for MD, Cassandra and GOMC for MC) and different simulation approaches (MD vs. MC).

2.2 GOMC

All simulations were performed using GOMC version 2.60. The long-range electrostatic interactions were calculated using Ewald summations, utilizing an energy tolerance of 10^{-5}. The simulation methods, sampling techniques, and the number of steps differed for each simulation type. Quintuplicate simulations were conducted for production runs used in the data analysis. The saturation pressure and vapor pressure simulations employed short-range Coulombic cutoffs extended to at least 12.0 nm in the vapor phase box to optimize the simulation time.

The Molecular exchange Monte Carlo type 2 (MEMC-2) move was used during the adsorption and desorption simulations.[23–25] The MEMC-2 move utilized 3 to 6 molecules of fake water in the simulation, with all the other water molecules being defined by the standard SPC/E force field.[23–25] The impure water (i.e., surrogate water) is defined as having the partial charges reduced by 2, and the epsilon for the Lennard-Jones parameter reduced by 4, while keeping

[1] Striolo *et al.* appears to have used no long-range electrostatics solver, likely in conjunction with a group-based cutoff scheme. We have chosen to proceed with long-range electrostatics since this is now the de-facto standard and many modern codes do not support group-based cutoff schemes.

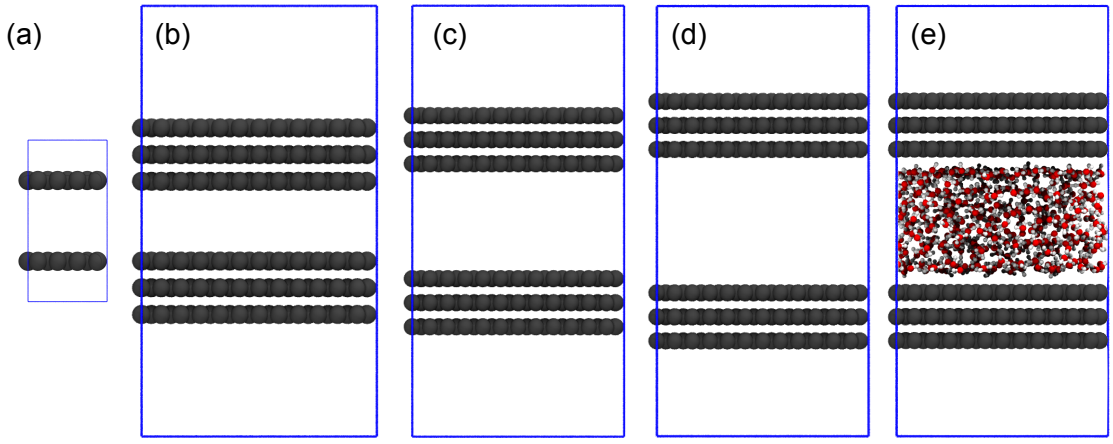

Figure 1: Snapshots of systems simulated in this work (the numbers refer to the pore width in nm): (a) small-1.0, (b) large-1.0, (c) large-1.6, (d) large-2.0, and (e) large-2.0 after pore filling. Carbon, oxygen, and hydrogen are shown as gray, red, and white spheres. Periodic boundaries are indicated by the blue lines.

the other parameters the same, all of which are compared to standard SPC/E forcefield.[23–25] The MEMC-2 move with impure water greatly enhances the system's sampling. All the other simulations only consisted of the standard water molecule.

Gibbs ensemble Monte Carlo NVT (GEMC-NVT) was used to determine the water saturation pressure, which were conducted for 60 million steps and used the volume adjustment (1%), multi-particle (2%), displacement (17%), rotation (20%), regrowth (20%), intra-swap (20%), inter-swap (20%) moves. The water vapor pressure as a function of chemical potential was determined by running GCMC-NVT simulations for 100 million steps and utilizing displacement (15%), rotation (15%), regrowth (10%), and inter-swap (60%) moves. The adsorption and desorption simulations were also performed via GCMC simulations, but the employed moves ratios and the number of steps depended on the preferred end phase of the system. However, if it was at a difficult transition point, the liquid phase moves were utilized, including the MEMC-2 move. The adsorption and desorption simulations that included the MEMC-2 move were run for at least 50 million steps, and other adsorption and desorption simulations were conducted for 150 million steps. The MEMC-2 move was not employed for systems that stayed in or quickly transitioned to the vapor phase, which leveraged the displacement (15%), rotation (15%), regrowth (20%), and inter-swap (50%) moves. The systems that maintained, ended, or were difficult to transition to the liquid or vapor phase utilized the (15%), rotation (15%), regrowth (10%), inter-MEMC-2 (20%), inter-swap (40%) moves. The water structure at the graphene surface of the small-1.0 nm pore utilized the displacement (24%), rotation (24%), regrowth (51%), and multi-particle (1%) moves. The large-2.0 nm pore's water structure at the graphene surface was determined using the displacement (24%), rotation (24%), regrowth (50%), and multi-particle (2%) moves (see Table 1 for system sizes). The small-1.0 nm and large-2.0 nm slit pores were simulated for 400 million and 40 million steps, respectively.

The systems used for GOMC during the adsorption and desorption simulation were adjusted to remove the vacuum space on the exterior of the pore. This was necessary as GOMC does not allow molecule insertions to be restricted to a sub-domain of the simulation box. In these GOMC simulations, utilizing the vacuum space using the GCMC ensemble would insert water in the vacuum space, an undesirable outcome. The final dimensions for the large-1.0 nm and large-1.6 nm systems for the adsorption and desorption calculations were $2.947 \times 2.978 \times 2.675$ and $2.947 \times 2.978 \times 3.275$, respectively. Note that the vacuum space *was* present for all GOMC simulations in the NVT ensemble.

2.3 Cassandra

Triplicate simulations of 300 million MC steps each were performed all state points. For GCMC simulations, the "restricted insertions" feature of Cassandra was used to confine molecule insertions to the region between the pore walls. GCMC simulations used insertion (25%), deletion (25%), translation (25%) and rotation (25%) moves. NVT simulations used translation (50%) and rotation (50%) moves. The translation and rotation move sizes were

selected to achieve 50% acceptance. Long range electrostatics were treated with an Ewald summation with a energy tolerance of 10^{-5}. Simulations were performed with MoSDeF Cassandra version 0.2.2 and Cassandra version 1.2.5.

2.4 GROMACS

The equations of motion were integrated with the leap-frog algorithm for 50 ns (200 ns for the small-1.0 nm/1 water system) using a time step of 1 fs. Water bonds and angles were constrained via LINCS[26] and the carbons atoms were held in place with `freezegrps`. Long range electrostatics were handled through the Fast Smooth Particle-Mesh Ewald method[27], with the `ewald-rtol` parameter set to 10^{-5}. Note that long-range electrostatics were turned off for the small-1.0 nm slit pore containing 1 water molecule. The number of k-space vectors in the large-2.0 system was set by a Fourier spacing of 0.12 nm, and was directly set to 16, 27, and 18 in cartesian coordinates for the small-1.0 nm systems. The water molecules were thermostatted to 298 K with the Bussi thermostat[28] and a time constant of 1 ps. Simulations were performed with the 2020 version of GROMACS. Other than the small-1.0 nm/1 water molecule system, analysis is performed on the last 45 ns of each simulation trajectory.

2.5 LAMMPS

The equations of motion were solved with the equations of Shinoda *et al.* for 20 ns using a 1 fs time step.[29] The carbon atoms were excluded from the MD integration, rendering them fixed. Water bonds and angles were constrained with SHAKE.[30]. Long range electrostatics were handled through the particle-particle-particle-mesh solver[31] with the relative force tolerance set to 10^{-5}. Temperature was controlled with the Nosé–Hoover chain thermostat at 298 K with the time constant set to 1 ps and chain length set to 3. Simulations were performed with the June 5, 2019 version of LAMMPS. Analysis is performed on the last 18 ns of the simulation trajectories for the small-1.0 nm system and the last 15 ns for the large-2.0 nm system.

2.6 CP2K

NVT ensemble first-principles molecular dynamics (FPMD) simulations were performed in the `CP2K` software version 7.0. The Gaussian plane wave method[14] was used to solve the Kohn–Sham formulation of density functional theory (KS-DFT). The Becke-Lee-Yang-Parr (BLYP) exchange-correlation functional[32,33] with the third generation Grimme dispersion correction[34] was used along with the MOLOPT–DZVP–SR–GTH basis set and Goedecker–Teter–Hutter (GTH) pseudopotentials[35,36]. Short test simulations indicated that a plane wave cutoff of 650 Ry yields a satisfactory balance of accuracy and computational efficiency.

For the FPMD trajectories, a time step of 0.5 fs was used and the system was maintained at 298 K using a Nosé–Hoover chain thermostat[37–39] with a time constant of 1 ps and chain length of 5. Each system was equilibrated for 30 ps, and the production trajectory was sampled for 120 ps.

3 Results

3.1 Water Adsorption

Adsorption and desorption isotherms were computed for 1.0 nm and 1.6 nm slit pores with GCMC (large-1.0 nm and large-1.6 nm slit pore systems run with both GOMC and Cassandra, small-1.0 nm system run with Cassandra to determine number of water molecules for Section 3.2 below). Adsorption simulations were initialized with an empty pore. Desorption simulations were initialized with a pre-filled pore, where the number of water molecules in the pore at the start of the simulation corresponded to the pore loading at high pressure as determined from the GCMC adsorption simulations.

3.1.1 Relating μ to $\mathbf{P/P}^{sat}$

Prior to completing the adsorption and desorption calculations a series of gas-phase simulations of SPC/E water were performed to establish a mapping between chemical potential and pressure. The ideal gas law was used to select the simulation box sizes such that there were a minimum $\sim$ 30–60 water molecules present. A comparison of GOMC and Cassandra results is shown in Fig. 2. Note that the two software packages employ a different definition of the chemical potential. In the case of SPC/E water the conversion from GOMC to Cassandra is $\mu' = \mu^{\mathrm{GOMC}} + 3k_BT\ln(\Lambda)$, where μ' is the Cassandra chemical potential, k_B is the Boltzmann constant, T is the temperature, and Λ is the thermal de Broglie wavelength in units of angstroms.

The saturation pressure (P^{sat}) of water using the SPC/E water model was computed at 298 K with Gibbs ensemble Monte Carlo (GEMC-NVT) in GOMC. Water's saturation pressure was employed to create a baseline in many of the analyses in this study (i.e., pressure divided by the saturation pressure or P/P^{sat}). This type of baseline, (P/P^{sat}), allows the simulation data to estimate the real-world pressure by merely plugging in the experimental saturation pressure. Grand Canonical Monte Carlo (GCMC-NVT) simulations are transferable to experimental conditions via the chemical potential (μ') and simulation pressure (P), which can be easily be determined in these simulations. The relationship between μ' and $\log(P)$ is approximately linear up to $P/P^{\mathrm{sat}} = 10$, a linear fit between μ' and $\log(P)$ was used to relate a given chemical potential, μ' to P/P^{sat}.

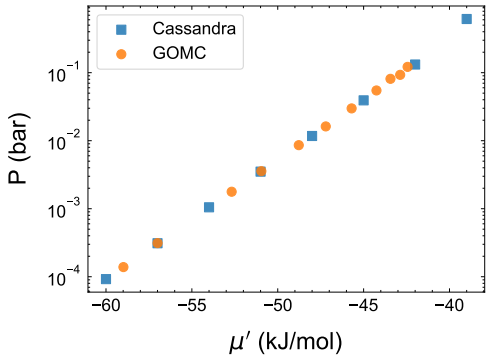

Figure 2: Computed pressure as a function of chemical potential (μ') for SPC/E water with Cassandra and GOMC.

3.1.2 Comparison with Striolo *et al.*

An equation of state method was used to relate the activity ($\zeta = \Lambda^{-3} \exp(\mu/k_B T)$) to P/P^{sat} in Ref. 15. P/P^{sat} was given by:

$$\frac{P}{P_0} = \frac{2\zeta\zeta_0^2 + (\rho_0 - \zeta_0)\zeta^2}{(\rho_0 + \zeta_0)\zeta_0^2} \tag{1}$$

The method[40] requires the activity at saturation (ζ_0) and vapor density at saturation (ρ_0). Neither the values themselves nor method used to compute them were explicitly provided in Ref. 15, and the resulting P/P_0 is extremely sensitive to these values. A future publication[41] from the same authors indicate that their procedure was most likely as follows: perform GCMC of water vapor in a box with a 3.0 nm side length and gradually increase the chemical potential until the system condenses to the liquid phase, then take ρ_0 and ζ_0 as the values just prior to condensation. In order to compare our results to Ref. 15, we repeated this same procedure to estimate the values used for ρ_0 and ζ_0. Our calculations yielded $\zeta_0 = 2.15 \times 10^{-6}$ Å^{-3} and $\rho_0 = 2.58 \times 10^{-6}$ Å^{-3}. The figures from Ref. 15 were digitized and equation 1 was used to estimate the activities at which the original simulations were performed. We caution the reader that our comparison of the adsorption/desorption isotherms are imperfect and that a rigorous quantitative comparison between our results and Ref. 15 is not our objective. As already noted we chose to proceed with long range electrostatics solvers that were absent in the original work, and our simulations contain other small differences (e.g., carbon–carbon distances, 2D vs. 3D periodic boundary conditions, etc). Nevertheless, as shown below, it appears that we are able to achieve semi-quantitative agreement between the results of Ref. 15 and the simulations performed here.

3.1.3 Adsorption and desorption isotherms

Adsorption and desorption simulations were performed at a range of μ' that corresponded to $10^{-3} < P/P^{\text{sat}} < 10^{1}$. The results are reported in Fig. 3, where the amount of water in the pore is reported as the number of water molecules per unit of pore surface area. The results from Cassandra and GOMC show good agreement with regards to the loading of the pore at different pressures. There is satisfactory agreement for the adsorption and desorption pressure. We suspect the slight discrepancies between Cassandra and GOMC may have two possible causes: (1) the use of the MEMC-2 move and surrogate water molecules in GOMC (see section 2.2) and (2) the use of restricted insertion volume in Cassandra (see section 2.3).

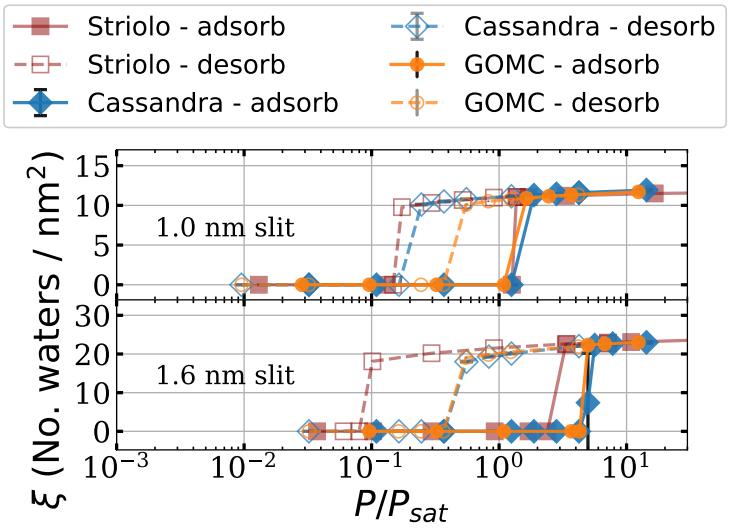

Figure 3: Adsorption and desorption isotherms for Cassandra, GOMC and the Striolo *et al.* data.[15] The P_{sat} value was obtained from GEMC-NVT water simulations. The Striolo *et al.* data was re-scaled as described in the text to match the P/P^{sat} definition in this work.[15]

3.2 Structure of Water

3.2.1 Calculation of Number Density and Order Parameter S

The number of water molecules inside of the pores were selected based upon the water adsorption results calculated from GCMC. All simulations used to compute the water structure in the pores were performed in the NVT ensemble. Fig. 4a and Fig. 4b report the number density profiles in the large-2.0 nm slit pore with 485 water molecules ($P/P^{\text{sat}} = 1.6$). The results from Cassandra, GOMC, GROMACS, and LAMMPS show nearly perfect agreement in terms of both oxygen and hydrogen number density inside of the slit pore, showing two fluid layers near each graphene wall. Additionally, all number density profiles are in good agreement with Fig. 9b from Ref. 15.

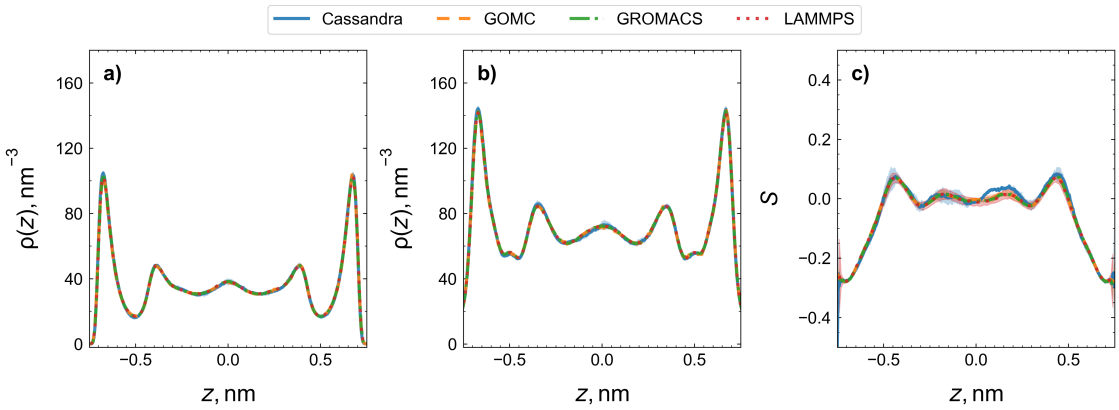

Figure 4: a) Oxygen-atom and b) Hydrogen-atom number density profiles, and c) S order parameter across the large-2.0 nm slit pore with 485 water molecules ($P/P^{\text{sat}} = 1.6$). z denotes the direction normal to pore walls, where $z = 0$ is set to the center of the pore. Uncertainties in the number density are on less than or equal to the line width.

The orientational S order parameter used in Ref. 15 is defined as:

$$S = \frac{3\langle \cos^2\theta \rangle - 1}{2} \tag{2}$$

where θ is the angle between the vector normal to the graphene walls and the vector drawn from the midpoint of the hydrogen atoms through center of the the oxygen atom (i.e. along the dipole). The S order parameter was calculated for the large-2.0 nm slit pore and is presented in Fig. 4c. Near the pore wall, the dipoles of the water molecules have a tendency to orient in a parallel fashion as indicated by the negative S order parameter. The water molecules near the middle of the pore are more bulk-like as shown by the random orientations. Overall, the results between the simulation engines are in good agreement with one other, and with Fig. 10b in Ref. 15.

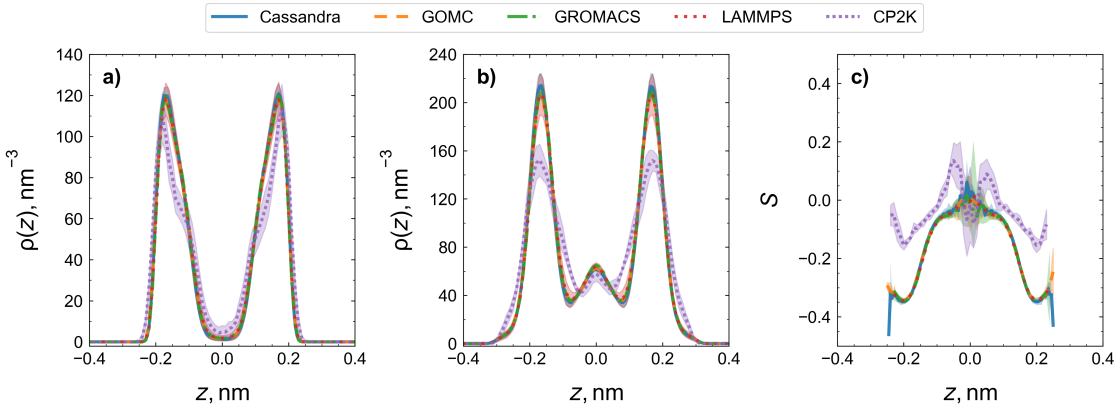

Figure 5: a) Oxygen-atom and b) hydrogen-atom number density profiles, and c) S order parameter across the small-1.0 nm slit pore with 24 water molecules ($P/P^{\text{sat}} = 1.6$). z denotes the direction normal to pore walls, where $z = 0$ is set to the center of the pore.

The number density profiles for the small-1.0 nm slit pore are displayed in Fig. 5a and Fig. 5b. This system contains 24 water molecules and has been constructed to be smaller than the large-2.0 nm slit pore system to allow for a direct comparison with first-principles simulations. The number density profiles computed from all four simulation codes using the SPC/E-water/LJ-graphene model (Cassandra, GOMC, GROMACS, LAMMPS) are in agreement with each other as well as with the result in Fig. 11 of Ref. 15. The oxygen atoms are arranged in a distinct layer near each pore wall. The hydrogen atoms also form a distinct layer neat each pore wall along with a small layer in the middle of the pore. The results from the FPMD performed in CP2K show some differences. The hydrogen number density peaks computed from CP2K are lower and broader when compared with the force field-based simulations. The oxygen number density profiles also display subtle differences that become more pronounced at low loading (see below).

The S order parameter for the small-1.0 nm slit pore is displayed in Fig. 5c. The error bars in the middle of the pore are larger as a result of the small system size and small probability of finding a water molecule in that region (see Fig. 5a). Similar to the large-2.0 nm pore, the water molecules near the walls are oriented parallel to the walls. Towards the middle of the pore, the water molecules are once again more randomly ordered. Simulations performed with the SPC/E-water/LJ-graphene model yield near perfect agreement for S across four simulation engines (Cassandra, GOMC, GROMACS, LAMMPS). The results reported here also show good agreement with Fig. 11b of Ref. 15.

To further investigate the differences in the density and orientational profiles observed between the SPC/E-water/LJ-graphene model and the KS-DFT description, both types of simulations were conducted for the small-1.0 nm slit pore system with just one water molecule. The symmetrized number density profile, the symmetrized S order parameter profile, and the normalized distribution of the angle θ for the one-molecule system are shown in Fig. 6. θ is again defined as the angle between the dipole moment vector of the water molecule and the surface normal vector (Note that for the angle distribution profiles, the surface normal vector is chosen based on nearest proximity to a given water molecule). For the FPMD simulation, a peak in the hydrogen number density is observed at $|z| = 0.24$ nm, i.e., a distance of 0.26 nm from the graphene surface. The θ distribution exhibits a peak at $\theta = 0$ degrees with a relatively extended tail. These two observations suggest a favorable electrostatic interaction of the water dipole with the π-electrons of the graphene sheet. This favorable electrostatic interaction also leads to a strong localization of the oxygen atom about 0.32 nm away from the graphene sheet with negligible density in the central region of the pore. In the force-field-based simulations, however, no peak is observed in the hydrogen number density near the graphene surface. Such a behavior of the hydrogen number density is expected as the water–graphene surface interactions only comprise of Lennard-Jones interactions between the water oxygen and the graphene carbon atoms, and partial charges are not used to represent the charge distribution of the graphene sheet. As a result, the water–graphene interactions are inherently isotropic and a uniform distribution is observed for θ. Therefore, the strength of interactions between water molecules and graphene surface is different between force-field-based and first-principles simulations which causes the differences in the density and S profiles for the two types of simulations. As an aside, it should be noted that

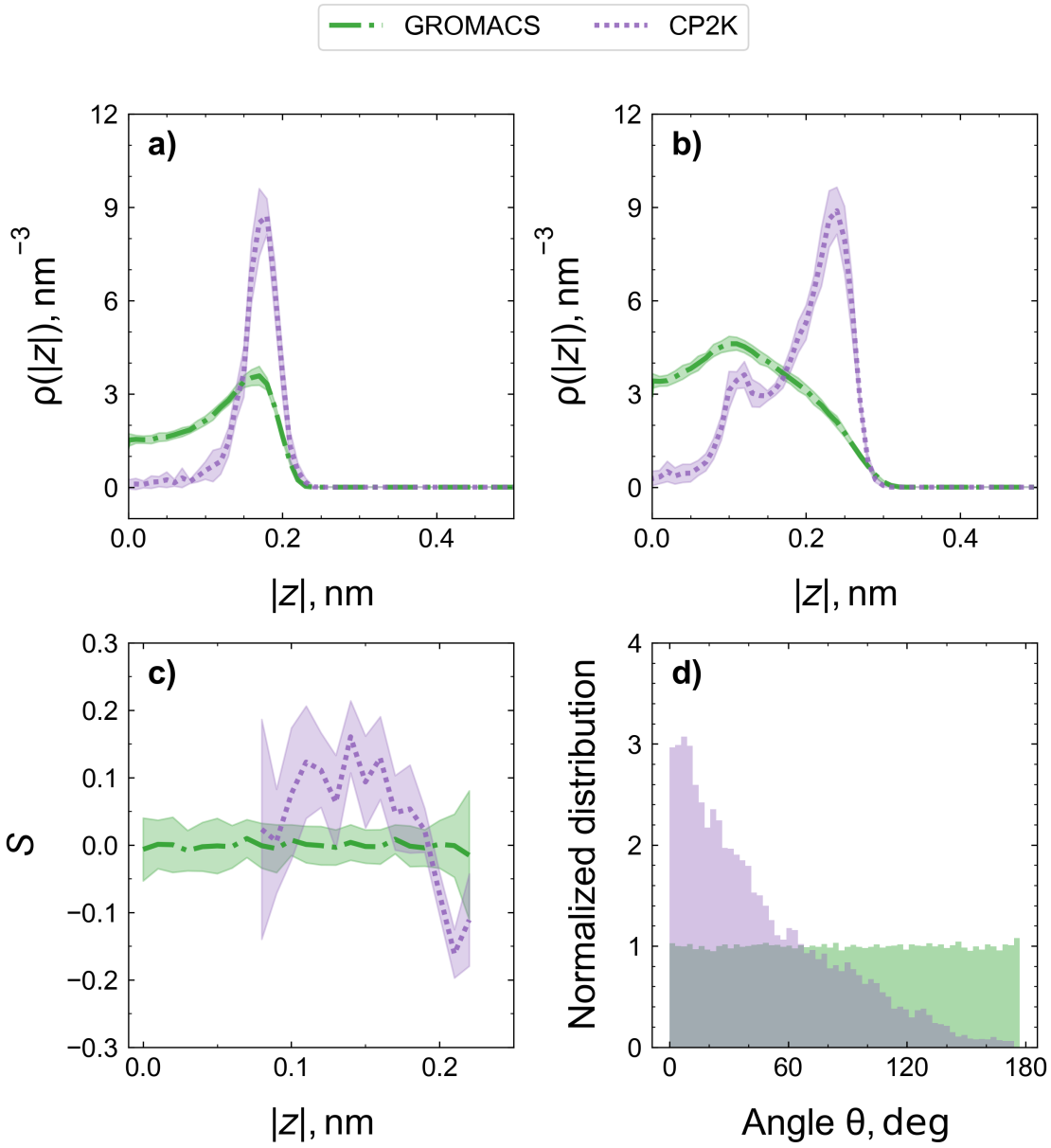

Figure 6: a) Oxygen-atom and b) hydrogen-atom number density profiles, c) S order parameter across the absolute value of a small-1.0 nm slit pore with a single water molecule, d) θ distribution across the pore. z denotes the direction normal to pore walls, where $z = 0$ is set to the center of the pore and θ is the angle between the dipole moment vector of the water molecule and the graphene surface normal vector.

$S = 0$ can reflect either a uniform orientation or a strong preference for the magic angle of 54.74° that is roughly half of the H-O-H angle of water. However, the θ distribution does not indicate a propensity for 'hydrogen-bonded' configurations in which one of the hydrogen atoms of a water molecule is perpendicular with the graphene surface.

4 Conclusions

Portions of the water in carbon slitpores calculations conducted by Striolo *et al.*[15] have been replicated using five modern open-source simulation engines in which input files were generated through the use of MoSDeF. GCMC was performed with Cassandra and GOMC to calculate the adsorption and desorption isotherms for water in carbon slitpores. Our results show reasonably good agreement with the previous calculations performed by Striolo *et al.* Simulations were then performed in the canonical ensemble with Cassandra, GOMC, GROMACS, LAMMPS, and CP2K to characterize the structure of water inside the carbon slitpores. A comparison of number density profiles and S order parameters shows good agreement between the force-field-based MC and MD simulations. The FPMD simulations show differences in the oxygen and hydrogen atom distributions and the water orientations in the pore. These differences were further investigated through the study of a single water molecule inside the carbon slitpores. In contrast to the force-field-based simulations, a peak in the hydrogen density near the wall and a strong orientational preference are observed in the FPMD simulations suggesting that KS-DFT with the BLYP functional predicts stronger water–graphene interactions than the SPC/E-water/LJ-graphene model. Overall this exercise demonstrates how MoSDeF enables the comparison of simulation results across multiple simulation codes and methods in a reproducible and user-friendly manner.

5 Installation and Simulation Instructions

In an effort to make our work reproducible, all steps to run and analyze the graphene slit pore simulations are included. To install all packages manually, detailed instructions are shown below to install of the required software packages. For the packages that are installed from source or through PIP, we recommend installing in a single directory on your machine. For this example we will use the directory, `software`, as the location for the packages installed from source.

5.1 Installation of the `conda` Package Manager

The `conda` package manager can be installed by running the following commands in your shell session:

The `$` *denotes a line in your terminal emulator and is not part of the command.*

For MacOS:

```
$ cd ${HOME}
$ curl -O https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-x86_64.sh
$ /bin/bash Miniconda3-latest-MacOSX-x86_64.sh
```

For GNU/Linux:

```
$ cd ${HOME}
$ curl -O https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
$ /bin/bash Miniconda3-latest-Linux-x86_64.sh
```

A series of prompts will install the package based on your preferences. The default location, your home directory, can be expected to work. Please refer to the documentation (https://conda.io/projects/conda/en/latest/user-guide/index.html) for any additional help or if your installation is on a computing cluster. You may also have to close your current terminal window and open a new one to execute `conda` commands.

5.2 Creating the `conda` Environment

Once `conda` has been installed, you can proceed to install the software packages and libraries required to run and analyze the simulations described herein. Start by creating a new `conda` environment with `Python 3.7` as the base `Python` interpreter. A `conda` environment is a directory that contains all of the packages and libraries based on your installation instructions. Multiple environments can be created on a single machine that contain different packages and/or different versions of `Python`. The environments are independent of each other, meaning modifications of one specific environment will not affect packages in another environment. Environments can easily be swapped by *deactivating* out of an environment and *activating* another environment. To simplify the process, an environment.yml file has been included in the `mosdef_slitpore` repository. This file contains all the information required for conda to install the required packages in the `slitpore37` environment. Note that some packages cannot be installed via conda, and instructions to install such packages are outlined below. All simulation and analysis steps should be run within the `slitpore37` environment. To create the `slitpore37` environment:

```
# change into `software` directory
$ cd ~/software
# clone mosdef_slitpore
$ git clone https://github.com/mosdef-hub/mosdef_slitpore.git
$ cd mosdef_slitpore
# create the `slitpore37` environment
$ conda env create -f environment.yml
# activate the `slitpore37` environment
$ conda activate slitpore37
```

Follow the series of command prompts to install all of the packages and their dependencies. A list of all packages installed in your current `conda` environment can be viewed by running:

```
$ conda list
```

5.2.1 Installation of `mosdef_slitpore`

All of the Python code necessary to initialize, run, and analyze the MC and MD simulations are contained within the `mosdef_slitpore` repository on GitHub. Once the `slitpore37` conda environment has been installed and activated,the following steps should be taken to install this repository.

```
# install mosdef_slitpore
$ pip install -e .
```

5.3 Installation of `Pore-Builder`

The design of `mBuild` encourages the development of recipes, which are external packages that utilize `mBuild` to initialize specific chemical systems. Please see the `mBuild` GitHub landing page for more information on recipes: (https://github.com/mosdef-hub/mbuild). Here, we use the `Pore-Builder`[42] recipe to initialize the carbon slitpores for each simulation. To install this package, run the the following commands. Please note that if you followed the instructions to this point, the package requirements for `Pore-Builder` should already be installed in the `slitpore37` environment.

```
# change into `software` directory
$ cd ~/software
# clone Pore-Builder
$ git clone https://github.com/rmatsum836/Pore-Builder.git
$ cd Pore-Builder
$ pip install -e .
```

Recipes have been designed to be imported through `mBuild`. To test that `Pore-Builder` has been correctly installed, run the following commands.

Below, the >>> indicates these commands are to be executed within the python interpreter.

```
# run Python and import Pore-Builder recipe
$ python
>>> import mbuild
>>> mbuild.recipes.GraphenePore()
>>> exit()
```

5.4 Installation of GOMC

`GOMC` (https://github.com/GOMC-WSU/GOMC.git) is one of the two MC engines utilized in this study. Installation of `GOMC` requires a working c/c++ compiler; please consult the

user manual for detailed information: (https://gomc-wsu.github.io/Manual/software_requirements.html).

The GOMC source code will be cloned from GitHub. The commands are listed below.

```
# GOMC requires cmake for compilation, which should already be installed in this setup.
↪  However, it is not, add it with the following commands via conda:
$ conda activate slitpore37
$ conda install -c conda-forge cmake

# change into `software` directory
$ cd ~/software
# clone GOMC
$ git clone https://github.com/GOMC-WSU/GOMC.git
$ cd GOMC
$ chmod u+x metamake.sh
$ ./metamake.sh
# once compiled, the executable should be located in the bin directory
$ ls ./bin
# add the GOMC bin folder to our path, so we can find the executable no matter
# the directory (i.e., the GOMC_CPU_GCMC, GOMC_CPU_GEMC, GOMC_CPU_NVT, etc., files)
# note: the below would need to be completed every time you open a new terminal window.
$ LOC_GOMC="$(pwd)/bin"
```

The section provided above is one method to have access to the compiled GOMC executables that were just created. The section with "LOC_GOMC" will need to be repeated if the terminal session is closed.

The next codeblock is another way to access the GOMC executables if you want to reference the full path to said binaries. This does not require the "LOC_GOMC" steps above. This assumes you are currently located in the GOMC folder: "~/software/GOMC".

```
$ export PATH="${LOC_GOMC}:$PATH"

# obtaining the explicit path, "full_path_to_GOMC_bin_folder", to the GOMC executable
↪  using the 'pwd' command, assuming you are already in the GOMC directory.
$ cd bin
$ pwd
# Example 1 (GCMC executable path with executable file):
↪  "full_path_to_GOMC_bin_folder"/GOMC_CPU_GCMC ;
# Example 2 (general GOMC executable path with executable file) :
↪  "full_path_to_GOMC_bin_folder"/GOMC_XXX_aaaa
# Example 3 (terminal code to run GOMC with Y cores from the directory containing the
↪  in.conf file) :  "full_path_to_GOMC_bin_folder"/GOMC_XXX_aaaa +pY in.conf > out.dat
```

The full path is now listed in the terminal. This path plus the GOMC executable is required to run GOMC without setting up the PATH every time the user opens a new terminal window. Therefore, entering the explicit path to the GOMC bin folder with the GOMC

executable file is a means to run the GOMC software directly. Some examples are provided to show this method for directly running the GOMC software. Note: the in.conf is the existing configuration file, out.dat is the name of the output file, and the items in quotation marks ("") are user-specific paths obtained by the entering 'pwd' command in the above step (i.e., "full_path_to_GOMC_bin_folder").

5.5 Installation of CP2K

CP2K (https://www.cp2k.org/) software is used for the first-principles molecular dynamics simulations in this study. CP2K can either be installed using a package manager or directly from the source (https://github.com/cp2k/cp2k). For testing the simulations on your local machine, we recommend installing through the use of a package manager (apt or brew). For running simulations on a HPC system, we recommend installing from conda or from source. Please see https://www.cp2k.org/howto for instructions to install from source.

5.5.1 Installation using a package manager

On Linux machines, CP2K can be installed using the following command:

```
$ sudo apt install cp2k
```

On MacOS, CP2K can be installed using the Homebrew package manager (https://brew.sh). The following commands will install homebrew (if you haven't already done so) and cp2k:

```
# install homebrew
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
# install cp2k
$ brew install cp2k
```

5.5.2 Installation from conda

The easiest way to install CP2K on a HPC system is through conda. This can be done by running the following command:

```
# conda install in `slitpore37` environment
$ conda install -c conda-forge cp2k
```

For installation from source, please see https://www.cp2k.org/howto for installation instructions from source.

5.5.3 cp2kmdpy installation

Input files for CP2K will automatically be generated through the use of `cp2kmdpy` (https://cp2kmdpy.readthedocs.io). Activate your `slitpore37` environment and make sure that Pore-builder is installed (section 5.3). After that follow these steps to install the package:

```
# change into `software` directory
$ cd ~/software
# copy the `cp2kmdpy` package
$ git clone https://github.com/ramanishsingh/cp2kmdpy.git
$ cd cp2kmdpy
```

Within the home directory of `cp2kmdpy`, lines 4 and 11 of `runners.py` or lines 7 and 17 (shown below) `runners_mpi.py` should be modified according to how `CP2K` is called on your machine (the default method is `cp2k.popt`). For example, if called using `cp2k.sopt`, then all instances of `cp2k.popt` should be replaced with `cp2k.sopt`.

```
# lines 4 and 11 of `runners.py`
call("cp2k.popt -i {} -o {}".format(input_filename, output_filename), shell=True)

# lines 7 and 17 of `runners_mpi.py`
process = Popen("mpirun -n {} cp2k.popt -i {} -o {}".format(np, input_filename, output_filename),
    shell=True,
    universal_newlines=True,
    stdin=PIPE,
    stdout=PIPE,
    stderr=PIPE )
```

Once these files have been appropriately modified, the package can be installing through the following steps:

```
# install package through pip
$ pip install -e .
# change into `software` directory
$ cd ~/software
$ git clone https://github.com/ramanishsingh/mosdef_cp2k_writer.git
$ cd mosdef_cp2k_writer
$ pip install -e .
# Finally we need to install the ele package
$ pip install ele
```

5.6 Instructions to Run Molecular Simulations

Below are detailed instructions to run all molecular simulation and analysis steps contained with `mosdef_slitpore`. Please note that small differences in the final results may exist due

to different compute architectures, parallelization schemes, and random seeds used to run the simulations. Also note that analysis for signac projects may not successfully run until all jobs have been completed.

5.6.1 Running GOMC Simulations

The GOMC molecular coordinates (pdb), connectivity (psf), and force field files (inp) are constructed using the MoSDeF framework and located in their appropriate directories. GOMC's configuration files are created by running the `simbuild.py` python file for each set, or are already present in the directory (i.e., the ".conf" files already exist). The `simbuild.py` python files are employed to generate up to five duplicate data sets for each system. If there are multiple simulations per set, the simulation chemical potentials or other variables are changed for each simulation. The directions for running each type of simulation are provided in the `README` files, located in the GitHub repository. The `README` files include the specific order to run the simulations if the user wants to recreate all the simulations. All the simulation's starting points are all provided so that the user can recreate a particular part of any simulation. The GOMC engine simulates the saturation pressure of water, vapor pressure of water at specific chemical potentials, water adsorption in the graphene slitpore, water desorption from the graphene slitpore, and the condensed water structure due to the pore's interactions with the graphene surface. An example of building the molecular system and running the small-1.0 nm system with GOMC on a desktop computer is provided below.

```
# change into `mosdef_slitpore` directory
$ cd mosdef_slitpore
# change into `NVT_build` directory
$ cd simulations/nvt-pore/1x1x1.0nm_1-layer/gomc/NVT_build
# run python script to build input files
$ python NVT_build_pore_1x1.0nm_1-layer.py
# change into `set1` directory
$ cd ../set1
# Run python script to build configuration files
$ python simbuild.py
# change into `1r1` directory
$ cd 1r1
# run simulation
# see section 5.4 for instructions to get path to GOMC executable
$ "full_path_to_GOMC_bin_folder"/GOMC_CPU_NVT +p2 in.conf > out.dat
```

Once the simulation is complete, the analysis scripts can be run with the commands below. Note that all five simulation runs are needed to successfully run these analysis scripts. If you wish to analyze a single run, the "filepath_list" variable in the scripts can be edited.

```
# change into `mosdef_slitpore` directory
$ cd mosdef_slitpore
# change into analysis directory
```

```
$ cd simulations/nvt-pore/1x1x1.0nm_1-layer/gomc/analysis
# run number density script
$ python Get_No_density_for_all_runs.py
# run order parameter s script
$ python Get_order_parameter_s_for_all_runs_test.py
```

5.6.2 Running Cassandra Simulations

The Cassandra simulations are initialized, run, and managed using the `signac` framework which is installed as part of the `slitpore37` environment. For more information on `signac` please consult the user documentation for detailed information (https://docs.signac.io/en/latest/). Each directory in `mosdef_slitpore` containing `Cassandra` simulations contains a separate `signac` project. Each `signac` project contains multiple `jobs` that have unique `statepoints`, or conditions at which the simulation is run. For example, each `job` in the adsorption `signac` projects has a unique chemical potential `statepoint`. Each `signac` project can be initialized by running the following lines below. Specifically, the small-1.0 nm signac project will be demonstrated.

```
# change into `mosdef_slitpore` directory
$ cd mosdef_slitpore
# change into a cassandra signac project directory
$ cd simulations/nvt-pore/1x1x1.0nm_1-layer/cassandra
# initialize signac project
$ python init.py
# check for the `workspace` directory
$ ls
```

The project can be verified to be initialized by checking for the `workspace` directory, which contains a directory for each `job` in your project. Once initialized, the next step is to perform operations on each job. The project operations are contained in `project.py` and can be run from the root directory of your project:

```
# run `run_adsorption` operation for a single job
$ python project.py run -o run_simulation -n 1
```

Due to the length of the `Cassandra` simulations, it is highly advised that these operations are run on a high performance computing (HPC) system, which is supported with `signac-flow`. For more details and a list of HPC environments supported out of the box, please consult the documentation (https://docs.signac.io/projects/flow/en/latest/). Job operations can be submitted to an HPC system by running:

```
# submit a `run_adsorption` operation to a HPC system
$ python project.py submit -o run_simulation -n 1
```

Once the simulation operations are complete, analysis can be performed by running the Python scripts contained within the `analysis` directory contained in each project. The functions defining each simulation type (adsorption, desorption, and NVT) are contained in `mosdef_slitpore/utils/cassandra_runners.py`.

5.6.3 Running GROMACS Simulations

Similar to `Cassandra`, the `GROMACS` simulations are initialized, run, and managed with the `signac` framework. All projects are contained within the `nvt-pore` directory in `mosdef_slitpore`. A walkthrough of initializing and running the GROMACS simulations for the small-1.0 nm slitpore system is shown below. To initialize the signac project run the following:

```
# change into `mosdef_slitpore` directory
$ cd mosdef_slitpore
# change to project directory from root directory of `mosdef_slitpore`
$ cd simulations/nvt-pore/1x1x1.0nm_1-layer/gromacs
# initialize the signac project
$ python init.py
```

Once the project is initialized the `signac` job operations may be performed. Again, it is highly advised to submit the simulation operations to a HPC system.

```
# run the `initialize` operation for a single job
$ python project.py run -o initialize -n 1
# run the energy minimization operation for a single job
$ python project.py run -o run_em -n 1
# submit the NVT MD simulation to a HPC system
$ python project.py submit -o run_nvt -n 1
```

To analyze the NVT simulation trajectories, run the following python script:

```
# calculate the number density profile and s order-parameter
$ python analysis.py
```

5.6.4 Running LAMMPS Simulations

The LAMMPS simulations are run similarly to the `signac` workflow of GROMACS. A walkthrough of initializing and running the LAMMPS simulations for the small-1.0 nm slitpore is shown below.

```
# change into `mosdef_slitpore` directory
$ cd mosdef_slitpore
# change to project directory from root directory of `mosdef_slitpore`
$ cd simulations/nvt-pore/1x1x1.0nm_1-layer/lammps
# initialize the signac project
$ python init.py
```

Once the project is initialized the `signac` job operations may be performed. Again, it is highly advised to submit the simulation operations to a HPC system.

```
# run the `initialize` operation for a single job
$ python project.py run -o initialize -n 1
# submit the NVT MD simulation to a HPC system
$ python project.py submit -o run_nvt -n 1
```

To analyze the NVT simulation trajectories, run the following python script:

```
# calculate the number density profile and s order-parameter
$ python analysis.py
```

5.6.5 Running CP2K Simulations

`CP2K` simulations are also initialized, run, and managed with the `signac` framework. `CP2K` projects are contained within the `nvt-pore` directory in `mosdef_slitpore` directory. The following assumes that the instructions for installing CP2K in Section 5.5 have been completed.

5.6.6 Running `CP2K` simulations using `signac`

```
# change into `mosdef_slitpore` directory
$ cd mosdef_slitpore
# change to cp2k directory from root directory of `mosdef_slitpore`
$ cd simulations/nvt-pore/1x1x1.0nm_1-layer/cp2k/
#initialize the signac project
$ python init.py
```

`CP2K` simulations are computationally expensive and therefore cluster submission of the jobs is advised. If desired, simulations can be run on a local system.

On a local machine, the simulations can be run using the following command:

```
# copy the setter
$ python project.py run -o copy_setter -n 1
# create MD input files
$ python project.py run -o md_files -n 1
# run MD simulation
$ python project.py run -o run_md -n 1
```

For submission to the cluster, modify the `templates/script.sh` file to load all modules (used during CP2K installation) and the `slitpore37` conda environment. A sample `templates/script.sh` file is provided. Use the following commands to submit jobs to the cluster:

```
# submit the jobs
$ python project.py submit
# This step will make sure that required files are present in the workspace. Now
↪  molecular dynamics simulations can be laucnhed
$ python project.py submit -o run_md
```

Simulations can be restarted using the following command:

```
# restart MD
$ python project.py submit -o restart_md
```

For analyzing the trajectories and obtaining the order parameter plot and the number density profiles execute the following command:

```
# Analyze trajectories
$ python analysis.py
```

This command will create a directory named `data` which will contain the plots and data for number density and order parameter.

5.7 Removing the `conda` Environment

If you are finished running the simulations and would like to remove the `conda` environment and all relevant files, the following command can be run:

```
# Remove `slitpore37` environment
$ conda remove --name slitpore37 --all
```

References

1. MoSDeF-Anaconda Cloud.
URL https://anaconda.org/mosdef/

2. MoSDeF web site.
URL http://www.mosdef.org

3. Lindahl E, Hess B. GROMACS 3.0 : a package for molecular simulation and trajectory analysis. *Journal of Molecular Modeling*. 2001;7:306–317.

4. Abraham MJ, Murtola T, Schulz R, Páll S, Smith JC, Hess B, Lindahl E. Gromacs: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX*. 2015;1-2:19–25.

5. Hess B, Uppsala S, Lindahl E. GROMACS 4 : Algorithms for Highly Efficient , Load-Balanced , and Scalable Molecular Simulation. *Journal of Chemical Theory and Computation*. 2008;4:435–447.

6. Shirts MR, Klein C, Swails JM, Yin J, Gilson MK, Mobley DL, Case DA, Zhong ED. Lessons learned from comparing molecular dynamics engines on the SAMPL5 dataset. *Journal of Computer-Aided Molecular Design*. 2017;31(1):147–161.

7. ParmEd — ParmEd documentation. 2018.
URL http://parmed.github.io/ParmEd/html/index.html

8. Thompson MW, Matsumoto R, Sacci RL, Sanders NC, Cummings PT. Scalable Screening of Soft Matter: A Case Study of Mixtures of Ionic Liquids and Organic Solvents. *Journal of Physical Chemistry B*. 2019;123(6):1340–1347.

9. Summers AZ, Gilmer JB, Iacovella CR, Cummings PT, McCabe C. MoSDeF, a Python Framework Enabling Large-Scale Computational Screening of Soft Matter: Application to Chemistry-Property Relationships in Lubricating Monolayer Films. *Journal of Chemical Theory and Computation*. 2020;16(3):1779–1793.

10. Plimpton S. Fast parallel algorithms for short-range molecular dynamics. 1995.

11. Shah JK, Marin-Rimoldi E, Mullen RG, Keene BP, Khan S, Paluch AS, Rai N, Romanielo LL, Rosch TW, Yoo B, Maginn EJ. Cassandra: An open source Monte Carlo package for molecular simulation. *Journal of Computational Chemistry*. 2017;pp. 1727–1739.

12. MoSDeF-Cassandra Github repository.
URL https://github.com/MaginnGroup/mosdef_cassandra

13. Nejahi Y, Soroush Barhaghi M, Mick J, Jackman B, Rushaidat K, Li Y, Schwiebert L, Potoff J. GOMC: GPU Optimized Monte Carlo for the simulation of phase equilibria and physical properties of complex fluids. *SoftwareX*. 2019;9:20–27.

14. Kühne TD, Iannuzzi M, Del Ben M, Rybkin VV, Seewald P, Stein F, Laino T, Khaliullin RZ, Schütt O, Schiffmann F, Golze D, Wilhelm J, Chulkov S, Bani-Hashemian MH, Weber V, Borštnik U, Taillefumier M, Jakobovits AS, Lazzaro A, Pabst H, Müller T, Schade R, Guidon M, Andermatt S, Holmberg N, Schenter GK, Hehn A, Bussy A, Belleflamme F, Tabacchi G, Glöß A, Lass M, Bethune I, Mundy CJ, Plessl C, Watkins M, VandeVondele J, Krack M, Hutter J. CP2K: An electronic structure and molecular dynamics software package-Quickstep: Efficient and accurate electronic structure calculations. *The Journal of Chemical Physics*. 2020;152(19):194103.

15. Striolo A, Chialvo AA, Cummings PT, Gubbins KE. Water adsorption in carbon-slit nanopores. *Langmuir*. 2003;19(20):8583–8591.

16. mBuild Github repository.
URL https://github.com/mosdef-hub/mbuild

17. Klein C, Summers AZ, Thompson MW, Gilmer JB, McCabe C, Cummings PT, Sallai J, Iacovella CR. Formalizing atom-typing and the dissemination of force fields with foyer. *Computational Materials Science*. 2019;167(May):215–227.

18. Foyer Github repository.
URL https://github.com/mosdef-hub/foyer

19. mosdef_slitpore Github repository.
URL https://github.com/mosdef-hub/mosdef_slitpore

20. Berendsen HJC, Grigera JR, Straatsma TP. The Missing Term In Effective Pair Potentials. *Journal of Physical Chemistry*. 1987;91(24):6269–6271.

21. Lorentz HA. Ueber die Anwendung des Satzes vom Virial in der kinetischen Theorie der Gase. *annalen der physik*. 1981;248(1):127–136.

22. Berthelot D. Sur mélange de gaz. In: *COMPTES RENDUS DES SÉANCES DE L'ACADÉMIE DES SCIÉNCES*, pp. 1703–1855. 1898;.

23. Soroush Barhaghi M, Torabi K, Nejahi Y, Schwiebert L, Potoff JJ. Molecular exchange Monte Carlo: A generalized method for identity exchanges in grand canonical Monte Carlo simulations. *The Journal of chemical physics*. 2018;149(7):072318.

24. Rafferty JL, Siepmann JI, Schure MR. Retention mechanism for polycyclic aromatic hydrocarbons in reversed-phase liquid chromatography with monomeric stationary phases. *Journal of Chromatography A*. 2011;1218(51):9183–9193.

25. Bai P, Siepmann JI. Assessment and Optimization of Configurational-Bias Monte Carlo Particle Swap Strategies for Simulations of Water in the Gibbs Ensemble. *Journal of Chemical Theory and Computation*. 2017;13(2):431–440.

26. Hess B, Bekker H, Berendsen HJC, Fraaije JGEM. 3 LINCS : a linear constraint solver for molecular simulations. *Journal of Computational Chemistry*. 1997;1472:1463–1472.

27. Essmann U, Perera L, Berkowitz ML, Darden T, Lee H, Pedersen LG. A smooth particle mesh Ewald method. *The Journal of Chemical Physics*. 1995;103(19):8577–8593.

28. Bussi G, Donadio D, Parrinello M. Canonical sampling through velocity rescaling. *The Journal of chemical physics*. 2007;126(1):014101.

29. Shinoda W, Shiga M, Mikami M. Rapid estimation of elastic constants by molecular dynamics simulation under constant stress. *Physical Review B - Condensed Matter and Materials Physics*. 2004;69(13):16–18.

30. Ryckaert JP, Ciccotti G, Berendsen HJ. Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. *Journal of Computational Physics*. 1977;23(3):327–341.

31. Hockney RW, Eastwood JW. *Computer simulation using particles*. Bristol: Hilger, 1988. 1988.

32. Becke AD. Density-functional exchange-energy approximation with correct asymptotic behavior. *Physical review A*. 1988;38(6):3098.

33. Lee C, Yang W, Parr RG. Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density. *Physical review B*. 1988;37(2):785.

34. Grimme S, Antony J, Ehrlich S, Krieg H. A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu. *The Journal of Chemical Physics*. 2010;132(15):154104.

35. Goedecker S, Teter M, Hutter J. Separable dual-space Gaussian pseudopotentials. *Physical Review B*. 1996;54(3):1703.

36. Hartwigsen C, Gœdecker S, Hutter J. Relativistic separable dual-space Gaussian pseudopotentials from H to Rn. *Physical Review B*. 1998;58(7):3641.

37. Nosé S. A unified formulation of the constant temperature molecular dynamics methods. *The Journal of Chemical Physics*. 1984;81(1):511–519.

38. Hoover WG. Canonical Dynamics: Equilibrium Phase-Space Distributions. *Physical Review A*. 1985;9(4):253–257.

39. Martyna GJ, Klein ML, Tuckerman M. Nosé–Hoover chains: The canonical ensemble via continuous dynamics. *The Journal of chemical physics*. 1992;97(4):2635–2643.

40. Müller EA, Rull LF, Vega LF, Gubbins KE. Adsorption of water on activated carbons: a molecular simulation study. *The Journal of Physical Chemistry*. 1996;100(4):1189–1196.

41. Striolo A, Chialvo A, Gubbins K, Cummings P. Water in carbon nanotubes: Adsorption isotherms and thermodynamic properties from molecular simulation. *The Journal of chemical physics*. 2005;122(23):234712.

42. Graphene-Pore Github repository.
URL https://github.com/rmatsum836/Pore-Builder