Efficient Implementation of Monte Carlo Algorithms on Graphical Processing Units for Simulation of Adsorption in Porous Materials

Zhao Li[ab]*, Kaihang Shi[ac], David Dubbeldam[d], Mark Dewing[b], Christopher Knight[b], Álvaro Vázquez-Mayagoitia[b]*, Randall Q. Snurr[a]*

[a] Department of Chemical and Biological Engineering, Northwestern University, 2145 Sheridan Road, Evanston, Illinois 60208, United States

[b] Computational Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Lemont, Illinois 60439, United States

[c] Department of Chemical and Biological Engineering, University at Buffalo, The State University of New York, Buffalo, New York 14260, United States

[d] Van 't Hoff Institute for Molecular Sciences, University of Amsterdam, Science Park 904, 1098XH Amsterdam, the Netherlands

Email:

Zhao Li: zhaoli2023@u.northwestern.edu

Álvaro Vázquez-Mayagoitia: vama@alcf.anl.gov

Randall Q. Snurr: snurr@northwestern.edu

Abstract

We present enhancements in Monte Carlo simulation speed and functionality within an open-source code, gRASPA, which uses graphical processing units (GPUs) to achieve significant performance improvements compared to serial, CPU implementations of Monte Carlo. The code supports a wide range of Monte Carlo simulations, including canonical ensemble (NVT), grand canonical, NVT Gibbs, Widom test particle insertions, and continuous-fractional component Monte Carlo. Implementation of grand canonical transition matrix Monte Carlo (GC-TMMC) and a novel feature to allow different moves for the different components of metal-organic framework (MOF) structures exemplify the capabilities of gRASPA for precise free energy calculations and enhanced adsorption studies, respectively. The introduction of a High-Throughput Computing (HTC) mode permits many Monte Carlo simulations on a single GPU device for accelerated materials discovery. The code can incorporate machine learning (ML) potentials, and this is illustrated with grand canonical Monte Carlo simulations of CO_2 adsorption in Mg-MOF-74 that show much better agreement with experiment than simulations using a traditional force field. The open-source nature of gRASPA promotes reproducibility and openness in science, and users may add features to the code and optimize it for their own purposes. The code is written in CUDA/C++ and SYCL/C++ to support different GPU vendors. The gRASPA code is publicly available at https://github.com/snurr-group/gRASPA.

Introduction

Graphical processing units (GPU) have been extensively used in physics-based simulations. For those simulations that focus on molecular systems with classical mechanics, parallelization is usually done when evaluating pairwise interactions. Molecular-level simulations include molecular dynamics (MD) simulations, which integrate Newton's equations of motion through time, and Monte Carlo (MC) simulations, which use a Markov chain for the evolution of the

system. Between MD and MC, parallelization in MD simulations is more common. Many biological systems are simulated using MD, ranging from protein folding[1] to the study of Alzheimer's disease.[2,3] These studies often use large system sizes with thousands to millions of particles, where the benefits of parallelization are most apparent.[4] However, in MC, especially for studying adsorption in crystalline materials having periodic unit cells, researchers usually consider smaller system sizes, often with only a few thousand particles, which benefit less from parallelization and GPUs.

Algorithmic differences between MC and MD also contribute to parallelization being more common in MD simulations than in MC. Although both classical MC and MD simulations evaluate pairwise energies, MD moves every particle in the system at each time step. Conventional MC, on the other hand, typically uses single-molecule moves, which means that only the energy change of a single molecule is needed at each MC step. For a system with N = 1000 atoms, $N * (N - 1)/2$ = 499,500 pairs of energy evaluations are needed at each MD step, while only $N - 1$ = 999 pairwise interactions are considered for each MC step. This makes it more challenging to efficiently parallelize an MC simulation. Efficiently parallelized MD codes such as NAMD[5], AMBER[6] and GROMACS[7] are widely used in the biology community, and LAMMPS[8] is popular for MD simulations of various material systems.

Another factor leading to the larger number of parallelized codes for MD than for MC is that MC algorithms (and codes) tend to be more application-specific. MD simulations integrate the classical equations of motion, regardless of the type of system. MC simulations, on the other hand, use a much wider range of ensembles and move types, and MC moves can be invented specifically for the system of interest. For example, for computational studies for adsorption,

open ensembles like grand canonical Monte Carlo (GCMC) and Gibbs Monte Carlo are widely used. GCMC uses insertion and deletion moves to mimic the molecular transfers between phases in a chemical equilibrium. For adsorption systems, GCMC relies on an implicit bulk phase reservoir and only simulates the adsorbed phase, whereas Gibbs Monte Carlo explicitly simulates both the bulk and the adsorbed phases and utilizes Gibbs particle transfer moves to allow the phases to reach equilibrium. Similar to most other MC moves, GCMC insertion and deletion moves and Gibbs particle transfer moves are single-molecule moves. In addition to different ensembles, special Monte Carlo moves, especially biased MC moves,[9] can be applied to enhance the MC sampling. For example, configurational-bias Monte Carlo[10] (CBMC) was invented for efficiently sampling chain molecules in a variety of MC moves, while energy-bias insertion moves[11] were invented for boosting the efficiency of simulations of adsorption in narrow pores. These moves speed up simulations for specific applications such as conformational sampling or gas adsorption but also hinder the generalization of MC codes.

Despite these difficulties, MC codes that benefit from parallelization and GPUs have appeared in recent years. For example, HOOMD-blue is a Python package that enables GPU acceleration for MD and MC simulations.[12] HOOMD-blue uses GPU parallelization for rigid body molecular dynamics and hard-particle MC simulations, which are well suited for studying the self-assembly of colloidal systems.[13] Another example is the GPU-Optimized Monte Carlo code (GOMC),[14] which parallelized the energy evaluations. GOMC features multi-particle moves such as the force-bias multi-particle method.[15] By using these multi-particle MC moves, more pairs are evaluated for each move, making the parallelization more beneficial. Kim et al.[16–18] developed an in-house MC code to run multiple GCMC calculations on the GPU. Recently, they used their

code to screen metal-organic frameworks for methane adsorption.[19] Their code uses tabulated energy data for accelerating GCMC simulations on the GPU, shifting the simulations from computation-intensive to memory-intensive. Although these advances in using parallelization for MC simulations have greatly accelerated computational discoveries in specific research fields, these GPU-enabled parallelization strategies, such as hard-particle MC or force-bias moves, are seldom applied in adsorption or phase equilibrium studies. The code by Kim et al.[16–18] is not open-source and appears to be custom-designed for certain applications, such as methane storage[19] and CO_2 adsorption.[20] Besides the codes mentioned above, there are other Monte Carlo codes that exploit CPU parallelization or other efficiency optimization strategies and aim at generalization of functionalities, such as Cassandra[21] and Towhee.[22]

In this work, we develop a GPU MC code, gRASPA (pronounced "gee raspa"), which is particularly focused on simulations of the adsorption of guest molecules in zeolites and metal-organic frameworks (MOFs). It can also be used for simulating vapor-liquid equilibria and other phase equilibrium problems. The gRASPA code is written in CUDA/C++ with the C++ 20 standard, and it includes the basic features of RASPA-2,[23] a widely used serial CPU code designed for simulating molecular adsorption and diffusion in flexible nanoporous materials. The gRASPA code can perform various Monte Carlo moves, such as translation and rotation moves[9] and swap (insertion/deletion) moves[9] using configurational-bias Monte Carlo (CBMC)[10], as well as continuous-fractional component MC[24] (CFC MC) and CFC with CBMC[25] (CBCFC). The gRASPA code reduces the overhead of GPU calculations by minimizing data transfers between the CPU and the GPU and reusing the GPU pointers and allocated memories. We demonstrate the efficiency of the gRASPA code through benchmarking with RASPA-2 and RASPA-3[26], a

recent MC simulation program developed for better output formatting, code readability, and simulation performance compared to RASPA-2.

In gRASPA, we also incorporated new features that take advantage of the GPU architecture and are not available in RASPA-2,[23], such as an option to use machine learning (ML) potentials. We developed a MC move that combines the ML potential with CBMC and tested its applicability for argon and CO_2 adsorption in Mg-MOF-74.[27] In addition, new features such as semi-flexible framework moves, which allow for movements of certain portions of the framework or extra-framework ions, and transition-matrix Monte Carlo[28,29] (TMMC) are also included. In addition to offloading calculations to Nvidia devices via CUDA, we also translated gRASPA to SYCL/C++ for users wanting to perform calculations on non-Nvidia GPUs or even field programmable gate arrays. The gRASPA code is lightweight and can be easily deployed to run dozens of Monte Carlo simulations on one graphic card at the same time, dramatically increasing the throughput while still maintaining a fast speed. Finally, we pushed the desired throughput further and developed a high-throughput computing mode of gRASPA that can run hundreds to thousands of simulations on one graphic card. This mode can significantly benefit researchers interested in screening materials for applications such as carbon capture and water harvesting. The gRASPA codes are open source and publicly available at https://github.com/snurr-group/gRASPA.

Methods

General Design

As Nejahi et al. pointed out in their papers about GOMC,[14,30] simulations performed on the GPU suffer greatly from memory transfers between the CPU and the GPU. Although the GPU can

perform massively parallelized calculations quickly, the atomic data, including atom positions, charges, and atom types, are normally prepared and stored on the CPU and transferred to the GPU whenever they are involved in a calculation.

Our gRASPA implementation focuses on reducing the memory transfers and their latencies between the CPU (host) and the GPU (device). This is done by storing most simulation data on the GPU instead of the CPU. Because of this, graphic cards that have higher memory bandwidth usually have better performance when performing calculations. At the beginning of the simulation, atomic data are read from the input and transferred to the GPU. A list of 10 million random numbers is pre-generated using the C++ standard random library on the CPU and then transferred and stored on the GPU for use during the entire simulation. This random number list can be extended when needed. The trial positions for the MC moves are generated on the GPU, and a trial translation, for example, consumes three random numbers for the displacements in the x, y, and z directions. Thus, we combine three random numbers into a double3 variable that is built-in in CUDA. This allows for much easier and more efficient use of the random numbers on the GPU. All system parameters, such as the number of molecules for each species, inverse temperature, and the transition matrices for transition matrix Monte Carlo simulations, are stored on the CPU.

In MC simulations, temporary storage of data is needed. Monte Carlo simulations require spaces to hold both the trial (new) and current (old) positions since the fate of the trial configuration will be determined based on the acceptance criterion of the move. If the move is rejected, the trial configuration must be discarded, and the current configuration must be retained. One can see that declaring new pointers, allocating new spaces on the memory, and freeing them for each move

would be very inefficient.[14] Thus, we reuse these pointers and allocations so that the new allocation of memory is minimized and, if possible, eliminated during the simulations.

The MC moves are classified into three categories based on the parallelization of the energy evaluations. The moves that only involve one trial configuration are considered single particle moves and generalized into one function. These include translation, rotation, non-CBMC swap moves (including both insertion and deletion moves), and semi-flexible framework moves such as linker rotations. The second types are the CBMC-based moves. These include the swap moves, reinsertion moves, identity swap moves, and particle transfer moves in the Gibbs ensemble. These moves share the same CBMC backbone. Currently, the code only supports rigid adsorbate molecules, but MC moves for flexible molecules will be available in the near future. Finally, system-wide moves such as (constant total-volume) volume perturbations of the simulation boxes in the NVT-Gibbs ensemble or volume moves in the constant-pressure, constant-temperature ensemble are classified as the third type of move since these moves change the configuration of every atom in a system and thus involve the calculation of total energies. Having these three generalized categories of moves allows us to experiment with different ways of parallelization more easily.

General Energy Evaluation

Both pairwise and non-pairwise energy interactions are considered in gRASPA. Parallelized energy calculations are performed using blocks and threads. In parallel computing and programming with NVIDIA GPUs, a "CUDA block" refers to a group of threads that can cooperate and synchronize within the same block while executing a parallel task. Threads within a block can communicate with each other but cannot talk to threads from another block. Pinned

memory, also known as page-locked memory, is a type of memory that cannot be swapped to disk and remains in physical RAM. Because of this feature, GPU's direct memory access engine can access the data on pinned memory directly without waiting for the operating system to load into physical memory, thus reducing memory transfer overhead. However, the downside of pinned memory is that it reduces the amount of memory available on the CPU for other processes. To minimize memory transfer and its latency while not abusing this feature, we allocate pinned memories for the data that needs to go back and forth between the CPU and GPU. This includes an array of floating point values for energy evaluations and an array of Boolean variables for indicating overlaps between pairwise interactions. If the simulation needs information on atom positions or partial charges on the CPU, for example for machine learning potentials, the memory related to these variables will also be allocated as pinned memory.

During the energy evaluations, the various contributions to the energies are tracked separately to provide additional information to the user. Energies are reported as the sum of van der Waals (vdW), short-range Coulombic and long-range Coulombic interactions, each divided among intra-framework, framework-adsorbate, and adsorbate-adsorbate interactions.

Pairwise Interactions

In gRASPA, each thread handles one or more than one distance pair between two atoms for the energy calculation during a move, similar to the method documented by Mick et al.[31] For each block, which is the bundle of threads, we perform parallel reduction (or summation) within the block on the energies each thread gives using the GPU's cache memory (or shared memory). This not only increases the utility of the graphic card but also minimizes the amount of data transfer between the device and the host. During the evaluation, threads that have super-high

repulsive pairwise interaction energies or have a very short pairwise distance are marked as "overlapped." The user can set the threshold energy and threshold distance. The overlap Boolean variable will then be written into the pre-allocated array on the GPU and then copied to the CPU via the pinned memory. If there is an overlap, and the move is translation, rotation, or moves that do not use CBMC, then the whole move is stopped. If the move involves CBMC, then the energy evaluation step of the overlapped trial is skipped, and CBMC cannot select that trial. If there is no overlap for the single trial move or for the CBMC trial, then the block sums are copied to the CPU via the pinned memory, and the pairwise energy for the move is the sum of the block sums.

To illustrate this implementation, Scheme 1 shows pseudo-codes for a non-CBMC insertion move running serial on the CPU vs. running on the GPU through gRASPA. We can see that instead of straight-forwardly looping over the atoms in the new molecule and atoms in the surroundings, for the GPU parallelization, one must unroll the for loops by first grouping pairwise interactions into threads, then grouping threads into CUDA blocks. Then, for each thread in each CUDA block, it loops over a number of pairwise interactions, and for each pairwise interaction, the thread solves for the index of the atom in the new molecule and of the surrounding atoms and calculates the distance and energy. If an overlap is found, instead of exiting the move completely, since threads are executed in parallel, an overlap flag is used for the CUDA block. Once every thread in a CUDA block finishes the calculation, the overlap flag is synchronized. For every energy summed over on each thread, shared memory is used to perform parallel reduction to generate a CUDA block sum. Once every CUDA block has finished calculation, the overlap flag first gets transferred to the CPU from the GPU. If the flag

reports an overlap, then the move is discarded. If not, the CUDA block energies are transferred to the CPU and then summed up to generate the total energy for this move.

Scheme 1. A non-CBMC insertion move on serial CPU vs. GPU

Algorithm 1: Single Particle Insertion Move on CPU

```
Initialize New Positions for Atoms in New Molecule;
for each Atom in New Molecule do
    for each Atom in Surrounding do
        Calculate distance and energy;
        if (energy ≥ threshold)
        or
        (distance ≤ overlap distance) then
            Overlap! Exit and Abandon Move;
        end
        E += energy;
    end
end
PACC = exp(-E / kBT) * f * V / (NMOL+1);
if RANDOMNUMBER ≤ PACC then
    Update Position,
    Number of Molecules,
    and TOTAL ENERGY;
end
```

Algorithm 2: Single Particle Insertion Move on GPU

```
Initialize New Positions for Atoms in New Molecule on
 GPU;
Determine Number of CUDA Blocks and Number of
 Threads for each Block;
parallel for each Block in CUDA Blocks do
    parallel for each Thread in Block do
        for each Interaction in Thread do
            Get AtomA from New Molecule;
            Get AtomB from Surrounding;
            Calculate distance and energy;
            if (energy ≥ threshold) or (distance ≤ overlap
             distance) then
                Overlap Flag = TRUE;
            end
            THREAD ENERGY[Thread] += energy;
        end
        BLOCK ENERGY[Block] += THREAD
         ENERGY[Thread];
    end
end
Transfer Overlap Flag to CPU;
if Overlap then
    Exit and Abandon Move;
end
Transfer BLOCK ENERGY to CPU;
E = Σ[BLOCK ENERGY];
PACC = exp(-E / kBT) * f * V / (NMOL+1);
if RANDOMNUMBER ≤ PACC then
    Update Position on GPU;
    Update NMOL and SYSTEM ENERGY on CPU;
end
```

Coulombic Interactions

For long-range Coulombic interactions, the real-space part and the Fourier part of the energies are calculated using the Ewald method.[9] For the single-particle and CBMC moves, each thread handles the energy difference computation and the change in the structure factors of each k-point for the Fourier part of the Ewald summation. The new structure factors are then stored in buffer storage and are updated if the move is accepted. If the move is accepted, the structure factors are adopted by swapping GPU pointers between the old storage and the buffer storage for the structure factors.

For MC moves that involve single particle movements, including translation, rotation, insertion/deletion, re-insertion, and Gibbs particle transfer, since only a single molecule has been moved, the difference in Fourier energies and structure factors is calculated. In this case, each thread handles the calculation of structure factors for one k-point to increase the amount of work each thread has and to reduce the usage of the GPU.

For the initial and final stages of the simulation, where lack of energy drift needs to be verified on the GPU, and for MC moves such as NVT-Gibbs, which need the energy of the whole system, we use a CUDA block to calculate the structure factor of a k-point and parallelize over the atoms in the system.

For the Fourier part of the Ewald summation, the intra-molecular and self-exclusion energies are crucial. Since single-molecule moves only change a small number of atoms, the intra-molecular and self-exclusion energies are calculated before the simulation starts, stored on the CPU, and used when a swap move is performed. This avoids the need to calculate this energy every time a molecule is swapped into or out of the system, thus eliminating the need to launch a CUDA kernel to calculate this for only one molecule. However, when calculating the total energy of the system, the self-exclusion and intra-molecular energies for all the molecules in the system are re-calculated.

Results

The simulations were performed on a local GPU workstation with an RTX 3090 GPU and an AMD Threadripper 3960X 3.8 GHz 24-core/48-thread processor CPU for all our test cases. Although the test cases were conducted on the local machine, the source code, the compilation, the setup files, and examples have also been prepared for supercomputer clusters such as Perlmutter of the National Energy Research Scientific Computing Center (NERSC) and Quest of Northwestern University.

1. Benchmark Results for SPC/E Water

As a first test, we calculated the reference energies for the four configurations of SPC/E water from the NIST reference calculations.[32] The results summarized in Table 1 show that the gRASPA code is able to reproduce the energies of the given configurations. Details about the calculations are provided in the SI.

Table 1. Energies (in Kelvin) for the four configurations of SPC/E water. E_{self} and E_{Intra} represent the Coulombic energy of an atom with itself and between atoms in the same molecule, respectively. Values from the NIST reference calculations (Ref. 32) are shown for comparison.

Configuration	Configuration 1		Configuration 2		Configuration 3		Configuration 4	
Code	NIST	gRASPA	NIST	gRASPA	NIST	gRASPA	NIST	gRASPA
$E_{vdW}/k_B\ (K)$	111992	111992	43286	43286	14403.3	14403.3	25025.1	25025.1
$E_{Tail}/k_B\ (K)$	-4109.19	-4109.19	-2105.61	-2105.61	-1027.3	-1027.3	-163.091	-163.091
$E_{Real}/k_B\ (K)$	-727219	-727219	-476902	-476902	-297129	-297129	-171462	-171462
Number of Wave Vectors	831	831	1068	1068	838	838	1028	1028
$E_{Fourier}/k_B\ (K)$	44677	44677	44409.4	44409.7	28897.4	28897.5	22337.2	22323.8
$E_{self}/k_B\ (K)$	-11581958	-11582033	-8686468	-8686525	-5790979	-5791017	-2895489	-2895508
$E_{Intra}/k_B\ (K)$	11435363	11435437	8576522	8576578	5717681	5717719	2858841	2858859
$E_{self+Intra}/k_B\ (K)$	-146595	-146596	-109946	-109947	-73298	-73298	-36648	-36649
$E_{Total}/k_B\ (K)$	-721254	-721255	-501259	-501259	-328153	-328153	-160912	-160912

2. GCMC Simulation of CO_2 Adsorption in MFI Zeolite

In this example, we used the adsorption of CO_2 in MFI zeolite at 298 K as a test case to demonstrate the efficiency of gRASPA compared to RASPA-2, which has undergone extensive testing previously. The RASPA convention, adopted by RASPA-2[23], RASPA-3[33], and gRASPA, uses the number of cycles instead of steps for simulations, and each cycle consists of N steps, where N equals the maximum of 20 and the number of molecules in the system at the beginning of the cycle. Here, we performed 30,000 cycles for initialization of the system and 30,000 cycles for gathering the averages. Each move was chosen with equal probability among translation, rotation, reinsertion, and swap moves. For a swap move, an insertion or deletion is chosen with an equal probability. The force field parameters are summarized in Table S3.

Figure 1 shows that the three codes generate consistent results. Regarding the simulation time, we can see that gRASPA is 4 to 5 times faster than the single-core RASPA-3. RASPA-3 is already faster than RASPA-2, and our gRASPA code pushes this limit further, showing a 19-fold acceleration in computational efficiency compared to RASPA-2. When evaluating the short-range pairwise interaction energies, the gRASPA code, by default, performs summation, or more technically speaking, reduction on four different values: framework-adsorbate vdW, adsorbate-adsorbate vdW, framework-adsorbate short-range Coulombic, and adsorbate-adsorbate short-range Coulombic interactions. This means four parallel reductions must be performed to calculate the sum of these different types of interactions correctly. For long-range interactions, there are two reductions for the framework-adsorbate and adsorbate-adsorbate long-range interactions. We designed a special version of gRASPA, referred to as “gRASPA-fast” in Figure 1, which disables the separate reporting of the energies for individual interaction types. Using

this Fast Option, only the total energy, which is the sum of the six different types, is returned. Although this gRASPA-fast version disables some functionalities, it does not change the simulation result, such as the number of molecules or the trajectory of the Markov chain; it simply reduces the number of reductions that must be performed for the energy calculations. Thus, the computation time is reduced from 228.3 to 188.7 seconds for CO_2 adsorption in MFI at 298 K and 10^4 Pa, a 20% performance improvement compared to the default gRASPA implementation.

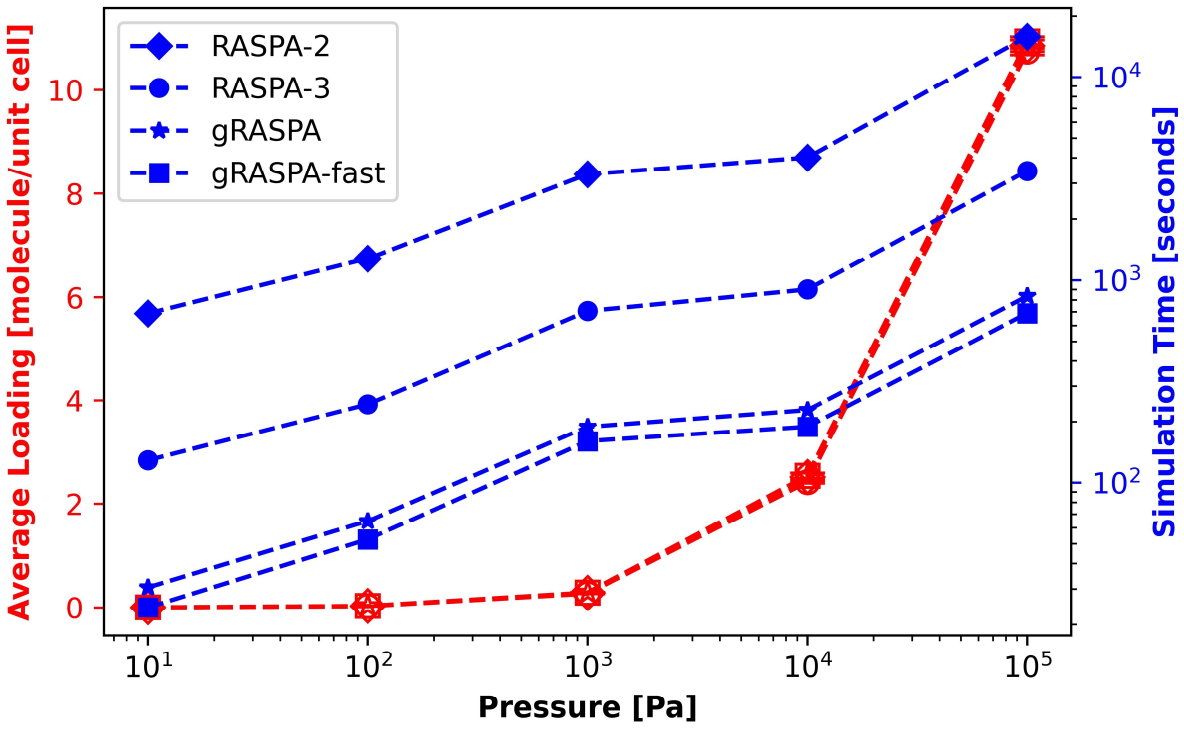


Figure 1. Simulation time and average loading comparisons for GCMC simulation of CO_2 adsorption in MFI zeolite using 8 unit cells at 298 K using different MC software. "gRASPA-fast" denotes the special version of gRASPA that disables the energy type separation. Data in

this figure is summarized in Table S4a. For the error bars reported in the table, we show the 95% confidence interval, which is two times the standard deviation of the block averages in the number of molecules.

We also investigated the GPU usage and benchmarked gRASPA's performance for simulations with only canonical moves (translation, rotation, and with/without reinsertion). The numbers are summarized in Table S4b for systems with 2.5 CO_2 molecules per unit cell (and varying numbers of unit cells), which is the loading from the GCMC simulation shown in Figure 1 at 10^4 Pa. Table S4b shows that as the simulation size increases, the simulation time, GPU memory and GPU utility increase. When the size of the simulation increases by 16 times from 8 unit cells to 128 unit cells, the number of MC steps per second only decreases by half.

3. Use of Nvidia Multi-Process Service (MPS)

Nvidia MPS is a binary-compatible implementation of the CUDA API that utilizes the multiple hardware queues to enable CUDA kernels from multiple processes to be offloaded to the same GPU without changing the code or re-compiling the executable. It is especially useful since the systems of interest for adsorption simulations are typically small, and a single gRASPA simulation underutilizes the GPU. We tested the performance of the CO_2-MFI simulations using gRASPA and Nvidia MPS on one GPU at 298 K and 10^4 Pa using 5000 MC cycles. The simulation uses the same MFI structure and force field parameters as those in Sec. 2. As comparisons, a RASPA-2 and RASPA-3 simulation were also performed on a CPU core with the same simulation condition and number of cycles. Table 2 summarizes the performance metrics. Here, we call the number of simulations performed on a single GPU simultaneously the throughput. We can see from the table that by utilizing MPS, it is possible to have high

throughput with some sacrifice in the speed of each simulation: for example, with only one simulation on a GPU, it takes 17.9 seconds. Using MPS and running two simulations, we double the throughput but at the cost of making each simulation 3.3 seconds slower. This creates an 18% speed decrease for each simulation, but the throughput is doubled; thus, it is very profitable to increase the throughput further. To quantify this competing relationship between the throughput and the speed ratio, we call the product of number of simulations performed concurrently and the speed of each simulation compared to serial RASPA-3 the performance index (PI) of gRASPA for the current application: $PI = N_{sim} * Speed\ ratio$, where $Speed\ ratio$ is defined to be the ratio of simulation time between serial RASPA-3 and gRASPA: $Speed\ ratio = Time_{RASPA-3}/Time_{gRASPA}$.

Table 2. Speed comparisons of the gRASPA code using Nvidia-MPS versus single-CPU-core RASPA-2 and RASPA-3 for CO_2 adsorption in MFI zeolite at 298 K and 10^4 Pa using 5000 MC cycles.

Number of Simulations	GPU Time [secs]	GPU Time (gRASPA-fast) [secs]	RASPA-2-Serial [secs/simulation]	RASPA-3-Serial [secs/simulation]
1	17.9	15.2	329.4	75.6
2	21.2	16.4		
3	23.2	17.7		
5	29.6	20.9		
8	41.6	25.3		
10	49.8	29.1		
12	60.1	32.9		
15	76.5	38.7		
20	106.9	49.2		
24	135.8	57.8		

Figure 2 shows that the Fast version (gRASPA-fast) gives a better PI than the gRASPA default option. For the GPU performances, the default option reaches maximum performance at ten simulations, while the Fast Option has not reached a maximum even at 24 simulations. Figure 2 also shows the CPU baseline performance from RASPA-3 which represents the performance by running N serial CPU simulations independently. The CPU baseline outperforms the GPU PI after 15 simulations, meaning the GPU simulations are slower than serial CPU RASPA-3 if more than 15 simulations are performed simultaneously on one graphic card. However, the CPU performance is still below the Fast-Option GPU performance even at 24 simulations. Thus, for high-throughput screening studies where the requirement on the level of details for different types of energies is low, the user can switch to the Fast Option to take advantage of both the speed and the throughput. After initial screenings of materials, the user can switch back to the default option of the gRASPA code for better interpretability of thermodynamic properties and statistical averages. Another possible strategy is to run initialization and equilibration cycles for MOFs using MPS and the Fast version of the code, then run the production cycles to gather detailed adsorption properties using the default option of the code.

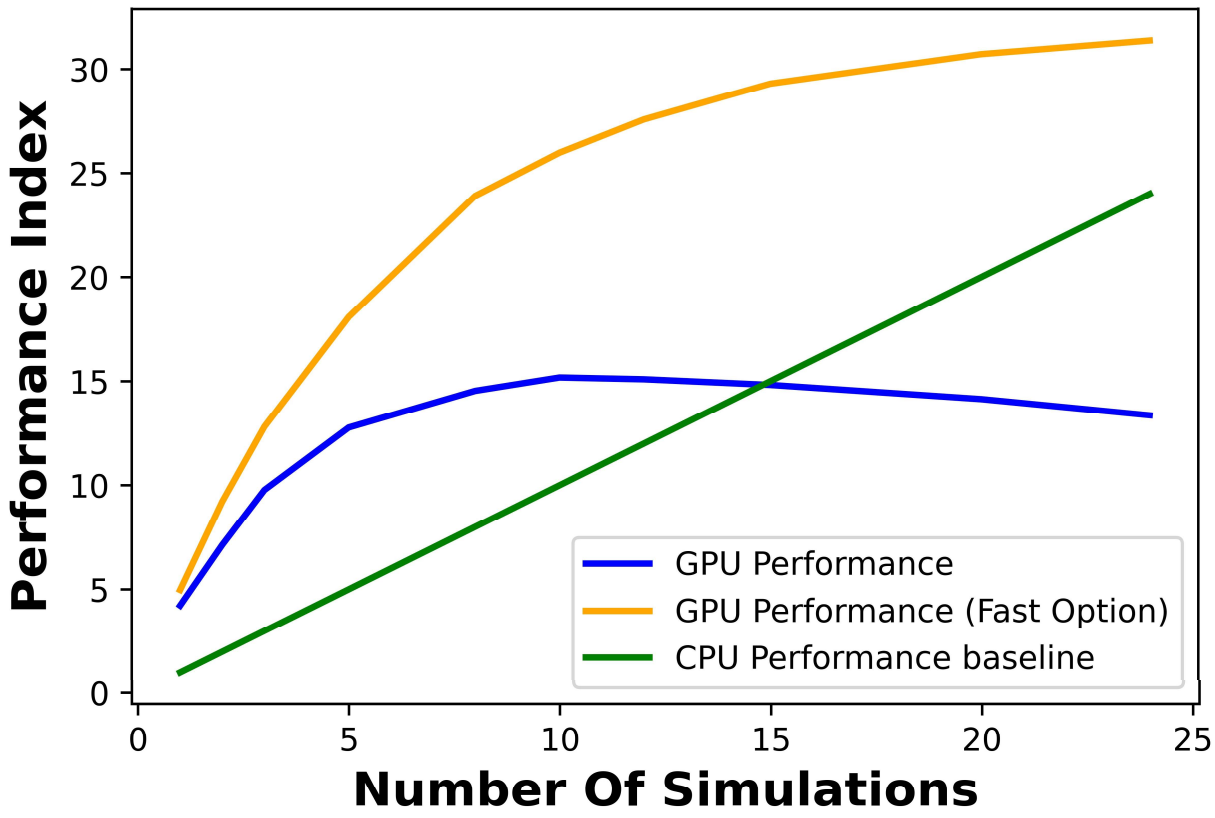


Figure 2. Performance index, defined by the speed ratio per simulation compared to serial RASPA-3 times the number of simulations running concurrently via Nvidia-MPS, versus the number of simulations running concurrently. The green line shows the baseline performance of RASPA-3 using a single CPU core. Since RASPA-3 is serial, there are no diminishing returns when multiple simulations are performed concurrently on multiple CPU cores.

4. Pure Component and Mixture Simulations for Separating CO_2/CH_4 in a Mixed Ligand Framework

Besides simulating single component adsorption, the code can also simulate mixture adsorption via GCMC or Gibbs Monte Carlo. Here, we present GCMC simulations for CO_2/CH_4 separation in the MOF $Zn_2(NDC)_2(DPNI)$ synthesized by Ma et al.,[35] where NDC is 2,6-naphthalenedicarboxylate and DPNI is N,N‘-di(4-pyridyl)-1,4,5,8-naphthalenetetracarboxydiimide. Bae and co-workers conducted GCMC simulations using the MuSiC[36] code at 296 K for an equimolar mixture of CO_2 and CH_4 in this MOF, and they predicted that it is a promising material for CO_2/CH_4 separations, especially for natural gas purification.[37] We conducted the mixture calculation using the same system as Bae et al.[37] For the single component and mixture calculations, the fugacity coefficients for each species were calculated using the Peng-Robinson equation of state.[38] The fugacity coefficients are summarized in Table S5. For the molecular representations of CO_2 and CH_4, we used the TraPPE model[39] and Goodbody et al.[40] parameters, respectively. The Lennard-Jones parameters for the framework atoms were from the DREIDING force field,[41] and the partial charges were taken from Bae et al.[37] Tables S6 and S7 summarize the parameters used. 10 million initialization steps were used to equilibrate the system, and 10 million production steps were used to generate the averages. Each step randomly chooses a move from translation, rotation (just for CO_2), reinsertion, swap (insertion or deletion), and identity change move (for mixture simulation) with equal probabilities. The excess loadings for CO_2 and CH_4 are reported. The binary selectivity is defined as $(\frac{x_A}{y_A})/(\frac{x_B}{y_B})$, where x_i and y_i are the mole fractions for component i in the adsorbed and bulk phases, respectively. We report the binary selectivity for CO_2. As a comparison, we also present the results at the same pressure using RASPA-2.[23] For these RASPA-2[23] simulations, 20,000 initialization and 20,000 production cycles were used, while the other simulation settings were the same as gRASPA. Other details about the simulation setup are summarized in the SI.

Figure 3a shows that the excess loadings for the single-component CO_2 and CH_4 adsorption calculated from gRASPA are within the error bars of those calculated with RASPA-2. Figure 3b shows that the loadings from the mixture simulations from the two codes are also in good agreement. The only difference can be observed for the selectivity, but selectivity is highly sensitive to small differences in the loadings of individual species, especially in the low pressure region where the loading of the species in the denominator of the selectivity equation is small. In our case, although the CH_4 loadings between the two codes agree well, the tiny difference gets magnified and becomes noticeable in the selectivity.

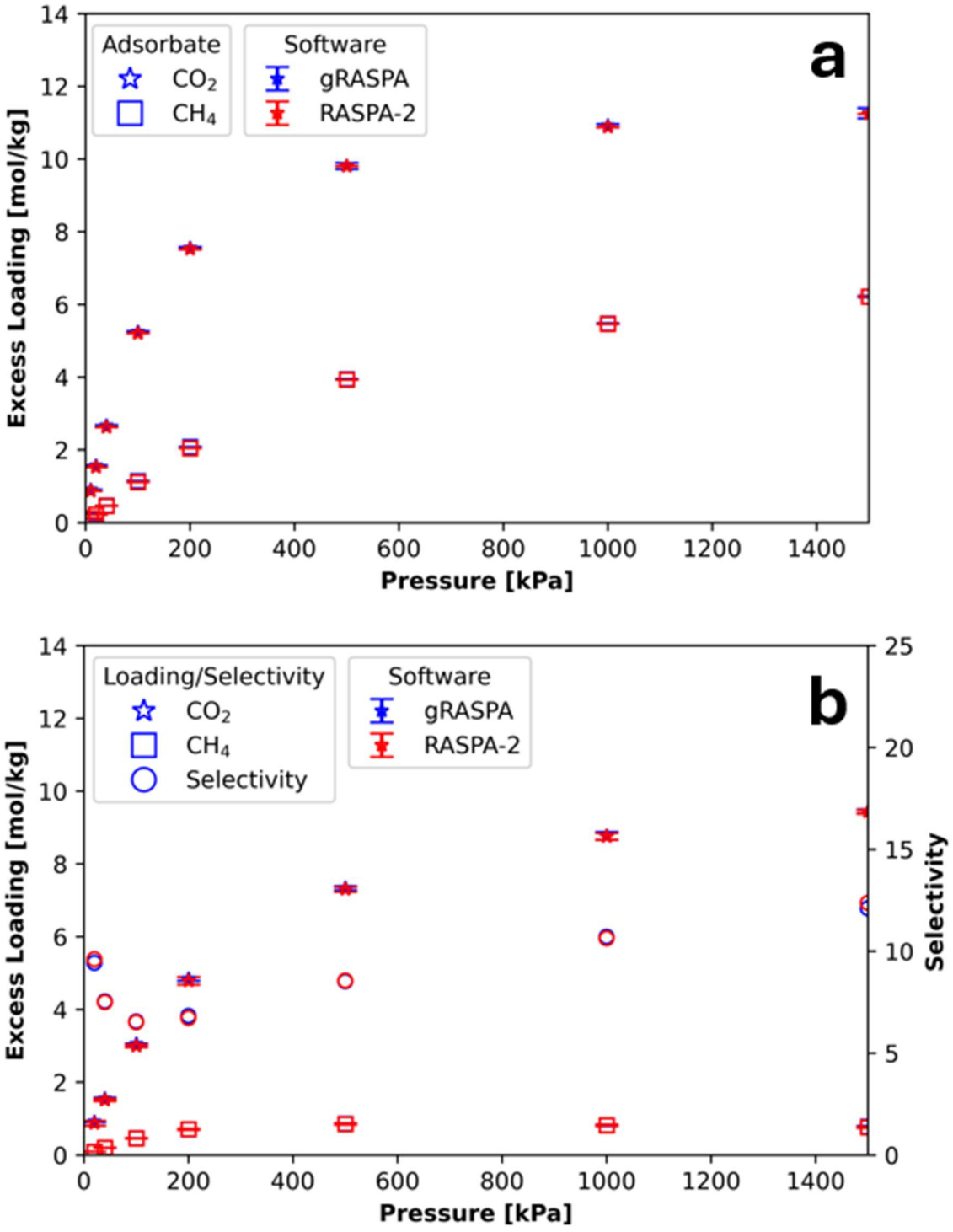


Figure 3. GCMC simulations for (a) single-component and (b) equimolar mixtures of CO_2 and CH_4 in the MOF $Zn_2(NDC)_2(DPNI)$ at 296 K. Stars are the CO_2 loadings, squares are CH_4 loadings, and circles are

the calculated selectivity of CO_2 over CH_4 from the excess loadings. Blue results are from gRASPA, red results are from RASPA-2.

5. Gibbs Ensemble and Constant-Pressure, Constant-Temperature Monte Carlo Simulations

5.1 Gibbs Ensemble Monte Carlo for Phase Equilibrium Calculations

Besides GCMC, another popular MC algorithm, Gibbs Ensemble Monte Carlo (GEMC)[42] in either the canonical ensemble (NVT) or constant-pressure, constant-temperature ensemble (NPT), is also implemented in gRASPA for single components. The Gibbs ensemble allows for the direct determination of the phase equilibrium of fluids from a single simulation by explicitly simulating the two phases in two simulation boxes. In the canonical ensemble, GEMC fixes the total volume of the two boxes, the total number of molecules in the two boxes, and the temperature of both systems. During the simulation, in addition to the thermal equilibration moves in both boxes (e.g., translation and rotation), the two boxes also experience volume exchange moves and particle transfer moves in the NVT ensemble. The former move changes the volumes of both boxes while keeping the total volume of the two boxes fixed, and the latter move selects a particle in one box and attempts to transfer it to the other box. Using this method, the Siepmann group has developed the widely-used TraPPE models[39,43] for a range of molecules by fitting the GEMC results to experimental values. For NPT-Gibbs, instead of the volume exchange move, the two boxes experience the NPT volume move independently, which randomly chooses a box and randomly perturbs the volume of the box. Here, we try to reproduce the vapor-liquid equilibrium for CO_2 using GEMC and the TraPPE-UA[39] model and compare the results against the experimentally reported VLE data[44] and transition matrix Monte Carlo simulations from NIST.[45] In addition to the Gibbs ensemble

simulations, we also performed single-box NPT (NPT MC) simulations for both the vapor and liquid phases as an additional validation.

For the NVT-Gibbs Monte Carlo simulations, we used 10,000 initialization cycles and 10,000 production cycles to generate statistical averages. For each cycle, N steps are performed, where $N = \max\{20, N_{Box_1}, N_{Box_2}\}$ and N_{Box_1} and N_{Box_2} are the numbers of molecules in the two boxes, respectively. For each step, a move is randomly chosen from translation, rotation, reinsertion, Gibbs particle transfer, and Gibbs volume moves with probabilities equal to 1:1:1:1:0.1. A cutoff of 15.0 Å for vdW and 15.0 Å for short-range Coulombic interaction was used for CO_2. Tail corrections were used. The Lennard-Jones parameters of the pseudo-atoms in the CO_2 molecule are summarized in Table S8a. Table S8b provides the initial setup of the CO_2 NVT-Gibbs simulations, the average densities, and the timing benchmarks. Note that some of these calculations were performed on an Intel i9-14900KF CPU and an Nvidia GeForce RTX 4090 GPU, while some others were done on L40S and A100 GPUs. The GPU used for the simulations is labeled in Table S8b.

The NIST computational data for CO_2 also provided the equilibrium pressure for each temperature, and we used these equilibrium pressures[45] for NPT-Gibbs simulations. The NPT-Gibbs simulations used 10,000 initialization cycles and 10,000 production cycles. For each step, a move is randomly chosen from translation, rotation, reinsertion, Gibbs particle transfer, and volume change moves with probabilities equal to 1:1:1:1:0.1. As noted above, as a complement to Gibbs Monte Carlo, we also ran NPT MC simulations for the vapor and liquid phases separately. These simulations used 25,000 initialization and 25,000 production cycles. Each MC step used a move that was randomly chosen from translation, rotation, reinsertion, and volume moves with probabilities equal to 1:1:1:0.1. Other details of the calculations are provided in the SI. All three sets of simulations (NVT Gibbs, NPT Gibbs, and single-box NPT) used the same force field parameters and cutoffs. Tables S8c and S8d provide the detailed initial

setup, the average densities, and the timing benchmarks of the NPT-Gibbs calculations and NPT MC, respectively.

Figure 4 summarizes the VLE curves for CO_2 simulated by gRASPA using NVT-Gibbs, NPT-Gibbs, single-box NPT MC, and simulation results from NIST using TMMC simulations with the same force field. [45] Experimental values reported by NIST [44] are also plotted as reference. Our calculated vapor and liquid densities are in good agreement with the NIST simulated and experimental values. This example demonstrates the ability of gRASPA to perform reliable Monte Carlo simulations in constant-pressure and constant-temperature Gibbs ensembles.

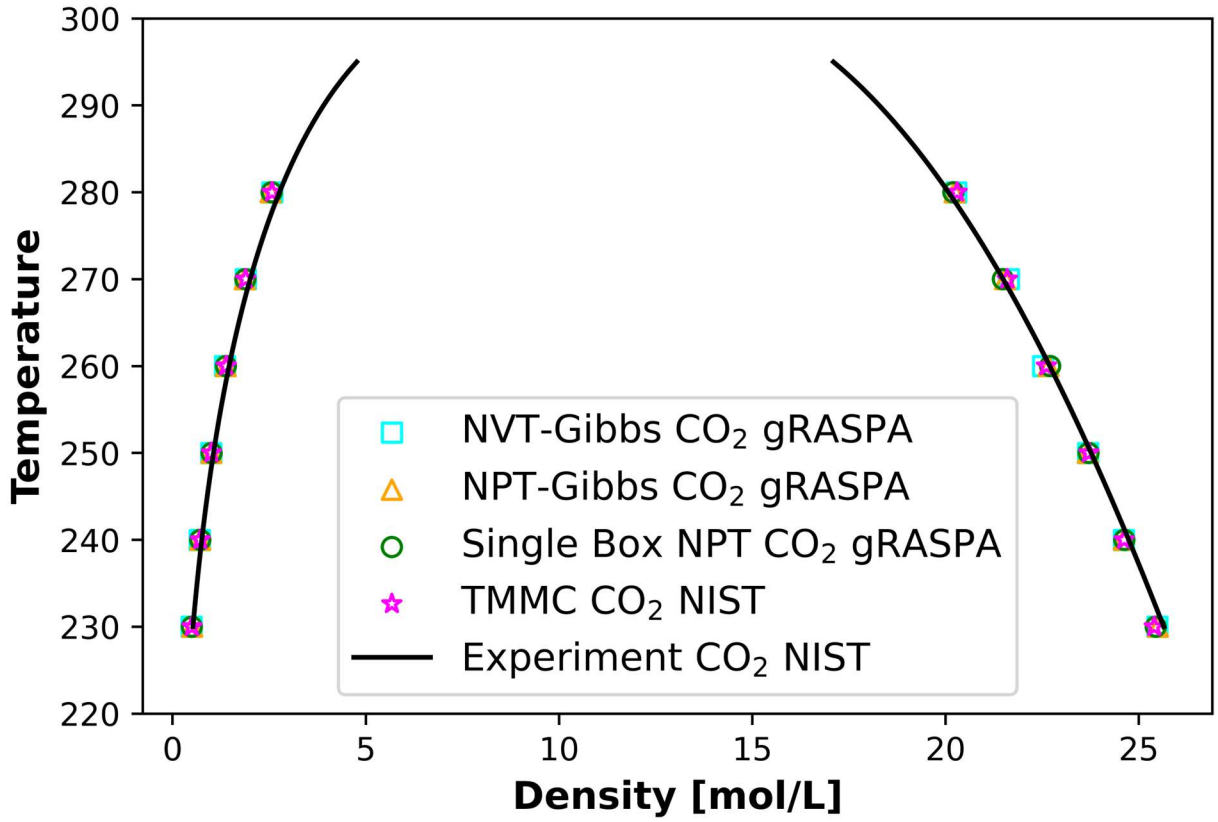


Figure 4.Vapor-liquid equilibrium (VLE) curves for CO_2 using the TraPPE-UA model taken from Ref. 39, while the simulated TMMC CO_2 results are from Ref. 45, and experimental CO_2 data replotted from Ref. 44.

5.2 Timing Benchmarks and Profiling for NPT MC

Single-box NPT simulations provide a useful way to benchmark a code's performance, because the cost of the energy calculations depends on the number of atoms, and the number of atoms is fixed in single-box NPT MC (unlike in GCMC or Gibbs MC). In addition, NPT MC can be used to measure the performance of the calculation of the total energy of the system since a volume change move is used. Here, we used Nsight compute, a profiling software by Nvidia to profile the performance for creating 700 CO_2 molecules and performing 2,000 MC steps in an NPT MC simulation. The results including the average execution time for each CUDA function are shown in Table S8e and S8f. We can see that the total energy calculations take much more time than any other CUDA function. Besides the top three functions in terms of averaged time elapsed in that table, namely "TotalVDWRealCoulomb", "TotalFourierEwald", and "TotalFourierEwald_CalculateEnergy", for the total energy calculations, the only other function that is involved in calculating the total energy is calculating the total intra-molecular and self-exclusion energy, which is a part of the total Fourier energy.

Comparing the short-range calculations between CBMC trial moves and non-CBMC moves shows that the "Calculate_Multiple_Trial_Energy_VDWReal" function, which handles CBMC trial energy calculation (15.35 μseconds, 8 trial positions and 8 trial orientations used), is faster than "Calculate_Single_Body_Energy_VDWReal," which handles non-CBMC moves (15.59 μseconds). This indicates that non-CBMC moves, such as translation and rotation, under-utilize

the GPU. While the CBMC trial energy move needs almost eight times more calculations compared to the non-CBMC move, it is on average 0.2 μsecond faster due to its highly parallelized nature.

Initialization of the wave vectors prior to the Fourier energy difference calculation, namely "Initialize_WaveVector_Reinsertion" and "Initialize_WaveVector_General", also involves some overhead, and they are ranked the 5th and 6th in Table S8e among all the CUDA functions doing preparations. This is because of the overhead of serial initialization for wave vectors as the later wave vectors depend on the previous ones for each atom. This is a potential part of the code that could be further optimized. It is also worth noting that the Fourier energy difference calculation "Fourier_Ewald_Diff", the function responsible for calculating the Fourier energy difference for single-particle MC moves, is the fastest CUDA function among all energy calculation functions. Despite the latency in preparing the wave vectors for the Fourier part, which are "Initialize_WaveVector_Reinsertion" and "Initialize_WaveVector_General", the actual calculation can be highly parallelized.

The sections above demonstrate functionalities of gRASPA that exist in RASPA-2 and other CPU Monte Carlo codes.[23] Below, we demonstrate the development and the usage of new functionalities that take advantage of GPU parallelization.

6. Machine Learning (ML) Potential for Modeling Gas Adsorption in Metal-Organic Frameworks

Most molecular dynamics and Monte Carlo simulations rely on empirical force fields due to their computational efficiency. For vdW interactions, it is typical to assume pairwise interactions

between the atoms using the Lennard-Jones equation. Electrostatic interactions are commonly modeled using fixed point charges on the atoms. In addition to these non-bonded interactions, bonding terms are included for bond stretching, bond angle bending, and dihedral angles. Density-functional theory (DFT), which provides an approximate solution to the Schrödinger equation, is expected to be more accurate for calculating energies and forces than empirical force fields in many cases. There have been works that try to utilize first-principles calculations in MC simulations. For example, work by Fetisov et al.[46] developed the First-Principles Monte Carlo (FPMC) method in the CP2K package[47] to simulate the reaction equilibrium of a mixture of nitrogen and oxygen at high temperature and pressure to mimic the effect of atmospheric lightning strikes. They found that FPMC shows good agreement with the simulation results parametrized to experimental data. However, the computational cost of applying DFT in molecular simulation is tremendously high,[48] as it scales cubically with the number of electronic degrees of freedom.[49]

Recent advances in machine learning (ML) force fields or ML potentials provide a new route to strike a balance between computational expense and accuracy.[50,51] The general idea of a ML potential is to use a ML model, such as a neural network,[52] to learn the mathematical mapping from the atomic environment to the atomic forces and system energy by training the ML model using high-fidelity quantum mechanical data. Substituting the classical force field with such a ML potential model in molecular simulation allows for modeling molecular and material systems at larger spatial and temporal scales with ab initio level accuracy. Since the seminal work on ML potentials around 2010,[52,53] ML potentials have been extensively studied and implemented in MD simulations.[54]

Compared to the popularity of ML potentials in MD, there are very few applications of ML potentials in Monte Carlo simulations to date. This wide difference in application can be understood from the following factors. Most ML potentials are parameterized to give the energy of the entire system, but most MC algorithms move one molecule at a time and thus only need the (change in) energy of a single molecule to accept or reject each move. By assuming pairwise additivity, the computational cost of evaluating the energy difference before and after a Monte Carlo move is reduced in simulations using a classical force field because only the pairwise energies related to the moved molecule (instead of the total energy of the configuration) are required. However, for MD simulations, the total energy of the system and the forces on all atoms are evaluated at every step. This makes MD better suited for ML potentials with many-body features. However, MC is the most popular method for predicting adsorption properties and vapor-liquid equilibria. Especially in GCMC, which mimics an open system where the number of adsorbate molecules changes during the simulation using swap moves,[9,34] MC moves are more direct and intuitively understandable compared to MD in the grand canonical ensemble.[55–57]

Attempts to implement ML potentials in Monte Carlo simulations, specifically GCMC for modeling gas adsorption in MOFs, appeared very recently. Current implementations of GCMC with an ML potential largely rely on a customized Python script[58] and the GCMC functionality in the LAMMPS simulation software.[59,60] While LAMMPS implements the GCMC algorithm with standard MC moves, such as translation, rotation and exchange moves (insertion and deletion), advanced biased moves, such as configurational bias and energy bias moves, are missing in LAMMPS. These biased MC moves are essential for efficient simulations of large molecules and for systems with strong specific interactions, such as water adsorption in MOFs.

gRASPA works with ML potential models that are developed using either TensorFlow[61] or PyTorch.[62] To demonstrate gRASPA's compatibility with TensorFlow, we implemented a simple neural network ML potential developed by Li-Chiang Lin and co-workers (referred to as the "Lin model" hereafter).[63] The Lin model was designed particularly for adsorption systems. The model takes transformed pair distances as input and predicts the adsorption energy between adsorbate molecules and a framework. They found that their model can predict Henry's constants of adsorbates such as CO_2 and H_2O in Mg-MOF-74.[27] In addition to the Lin model, we also implemented a state-of-the-art ML potential model, the Allegro model, in gRASPA to demonstrate the compatibility with PyTorch. The Allegro model[64] is an equivariant neural network interatomic potential for predicting system energy and atomic forces based on the local atomic environment in the simulation box. The Allegro model was able to reproduce the properties of ab initio MD, such as the radial distribution function for lithium thiophosphate.[64] Due to its localized atomic features, the Allegro model demonstrated exceptional scalability to large systems through parallel computation. Recently, Allegro's scalability was illustrated by a nanoseconds-long MD simulation for a 44-million atom structure of a complete, all-atom, explicitly solvated HIV capsid.[65]

We used both the Lin model[63] and the Allegro model[64] to calculate adsorption isotherms of argon and CO_2 in Mg-MOF-74. The code incorporates the ML potential into every MC move used for GCMC simulations, including CBMC. We benchmarked the accuracy and speed of the two models and point out the bottleneck that limits the usage of ML potentials in MC simulations.

6.1 General Setup

We implemented a hybrid scheme for modeling gas adsorption[66] where the host-guest interactions are modeled using a ML potential while guest-guest interactions are still modeled using a classical force field since the TraPPE force field is well-tuned for capturing the phase equilibrium of adsorbates.[43] Adopting this hybrid modeling scheme is helpful to effectively reduce the required amount of training data for the ML potential. In this case, only configurations with one adsorbate molecule are needed for producing training data; otherwise, training data should contain MOF structures at multiple loadings of adsorbate molecules so as to enable the ML potential to predict the entire adsorption isotherm accurately.

For non-CBMC moves, such as translation moves, the classical energies are evaluated first. This includes the vdW, short-range and long-range Coulombic interaction energies for framework-adsorbate and adsorbate-adsorbate pairs. Then, the classical framework-adsorbate energies are discarded and re-evaluated using the ML potential. Although the classical framework-adsorbate energies are unused for the acceptance criteria, they are used for determining whether the trial positions overlap with the framework. If there is an overlap when evaluating the classical framework-adsorbate energies, the expensive ML evaluation can be skipped.

For CBMC moves, such as the swap insertion move, the classical vdW and real part of the Coulombic energies are used to select which of the trial positions is chosen for the adsorbate molecule. Here, we denote the Rosenbluth weight as W_r (see SI for detailed formula). The framework-adsorbate (FA) vdW and real part of the Coulombic energies for the selected trial configuration are $E_{vdW,FA}$ and $E_{Coulomb-rea\ ,FA}$. Once one of the trial configurations is selected, the Fourier part of the Coulombic energy of the adsorbate molecule for the selected trial

configuration $E_{Coulomb-Four\quad,FA}$ is calculated, and its contribution to the Rosenbluth weight is added to the current value of W_r (for more information about the Rosenbluth weight, please refer to the Monte Carlo moves section in the SI):

$$W_r = W_r * \exp\left(-\beta E_{Coulomb-Fourier,FA}\right) \tag{1}$$

The ML potential is then calculated for the selected trial configuration, and the ML-corrected Rosenbluth weight $W_{r,ML}$ is calculated as follows:

$$W_{r,ML} = W_r * \exp\left(-\beta \Delta E_{ML}\right) \tag{2}$$

where β is the inverse temperature ($1/k_B T$, where k_B is Boltzmann's constant), and ΔE_{ML} is the difference in energy between the ML potential and the classical interaction for the framework-adsorbate:

$$\Delta E_{ML} = E_{ML} - \left(E_{vdW,FA} + E_{Coulomb-real,FA} + E_{Coulomb-Fourier,FA}\right) \tag{3}$$

Once $W_{r,ML}$ is calculated, it is plugged into the acceptance rules for CBMC moves, including the swap insertion, swap deletion, and reinsertion moves, to determine the fate of the move:

$$P_{acc}^{Insertion} = \min\left(1, \frac{W_{r,ML,New}\beta V}{N+1}\frac{f}{\langle W_{IG}\rangle}\right), \tag{4}$$

$$P_{acc}^{Deletion} = \min\left(1, \frac{N}{W_{r,ML,Old}\beta V}\frac{\langle W_{IG}\rangle}{f}\right),$$

$$P_{acc}^{Reinsertion} = \min\left(1, \frac{W_{r,ML,New}}{W_{r,ML,Old}}\right),$$

where $W_{r,ML,New}$ and $W_{r,ML,Old}$ are the ML-corrected Rosenbluth weights for the new and old configurations, V is the volume of the simulation box, f is the imposed fugacity of the GCMC

simulation, and $\langle W_{IG} \rangle$ is the averaged Rosenbluth weight of an isolated molecule in the gas phase evalulated using the classical force field. For a rigid molecule, $\langle W_{IG} \rangle$ is set to 1.[10,67]

6.2 Lin model

For the implementation of the Lin model, every adsorbate-framework atom pair type is considered. For each type of pairwise interaction, the nine smallest distances are sorted in ascending order. Then, for each of the nine smallest distances, the raw pairwise distance r is transformed into six features, i.e., $\exp(-r)$, $1/r$, $1/r^4$, $1/r^6$, $1/r^8$, and $1/r^{10}$. For example, for CO_2 adsorption in Mg-MOF-74, there are eight guest-host atom pair types, i.e., C-Mg, C-O, C-C, C-H, O-Mg, O-O, O-C, and O-H, where the first atom type is from the guest molecule (CO_2) and the second is from the MOF material. For each type of pair, say C-Mg, we pick the first nine smallest distances $\{r_1, r_2, \dots, r_9\}$ in ascending order. Then for each distance, we calculate the six distance features, e.g., for distance r_1, we have $\{\exp(-r_1), 1/r_1, 1/r_1^4, 1/r_1^6, 1/r_1^8, 1/r_1^{10}\}$. Therefore, for each atom type pair, we have $9 \times 6 = 54$ features. Because there are eight unique pairs, the total number of input features for a single CO_2 configuration is $54 \times 8 = 432$. An illustration of this featurization process is available in Figure S4. With these features calculated, the model can then make predictions. Since it is a shallow model with only five hidden layers (see SI for more details), we performed the predictions on the CPU instead of the GPU to avoid latencies in loading data to and from the GPU.

6.3 Allegro model

The implementation of the Allegro model[64] in LAMMPS[8] is based on the neighbor list for each atom in each subdomain. The use of subdomains is a technique for handling large systems of atoms using message-passing interface (MPI) processes by dividing the simulation box into

parts. Then, each CPU core can handle a subdomain efficiently. Atoms from neighboring subdomains are stored as "ghost" atoms for each subdomain. In gRASPA, to reproduce the effect of the ghost atoms, we generate 26 (3 x 3 x 3 − 1) replica cells that are exact copies of the central cell and surround the central cell, where the central cell is the framework structure used for generating the training data for ML potential. A detailed description of the featurization process for using the Allegro model is summarized in the SI.

6.4 Results

We implemented the Lin model and the Allegro model in gRASPA for two test cases: (1) Ar adsorption in Mg-MOF-74 at 77 K, where the model was trained using classical force fields for all interactions. This case was chosen to check if ML potential models can reproduce the adsorption isotherm of reference classical simulations; (2) CO_2 adsorption in Mg-MOF-74 at 313 K, where guest-host interaction energies were predicted by the ML potential at ab initio accuracy. This case highlights the superiority of the ML potential in modeling challenging gas adsorption systems where classical force fields fail to match experimental data.

6.4.1 Ar adsorption in Mg-MOF-74

In this test case, we generated data to train the ML model using an NVT MC simulation at 80,000 K using classical force fields with only one Ar molecule in the unit cell of Mg-MOF-74. Using such a high temperature was intended to generate a training data set containing diverse configurations. ML potentials were trained to regress the classical Ar binding energy for a given Ar configuration. Once the ML models were validated, we performed GCMC simulations using gRASPA to test the performance of the ML potential models in reproducing the classical force field. Details for training data generation, ML training, and GCMC simulations including the

classical force fields are available in Section S5.1. Simulation and ML training input files are also available in our github repository: https://github.com/snurr-group/gRASPA/tree/main/Examples.

In this simple case, both the Lin and Allegro models can regress the classical force field well. The mean absolute errors (MAE) for the Lin and Allegro models are 1.64 kJ/mol and 0.77 kJ/mol (equivalently, 17.00 meV and 7.98 meV), respectively, based on 1,000 testing points. Parity plots showing the performance of both models on testing data are available in Figures S1 and S2. As shown in Figure 5, the simulated adsorption isotherms of Ar in Mg-MOF-74 at 77 K using both ML potential models agree quantitatively with the reference classical simulations within the statistical error of the simulations. The consistent results confirm the correct implementation of the ML potential functionalities in gRASPA and the validity of the Lin and Allegro models for simple Lennard-Jones systems.

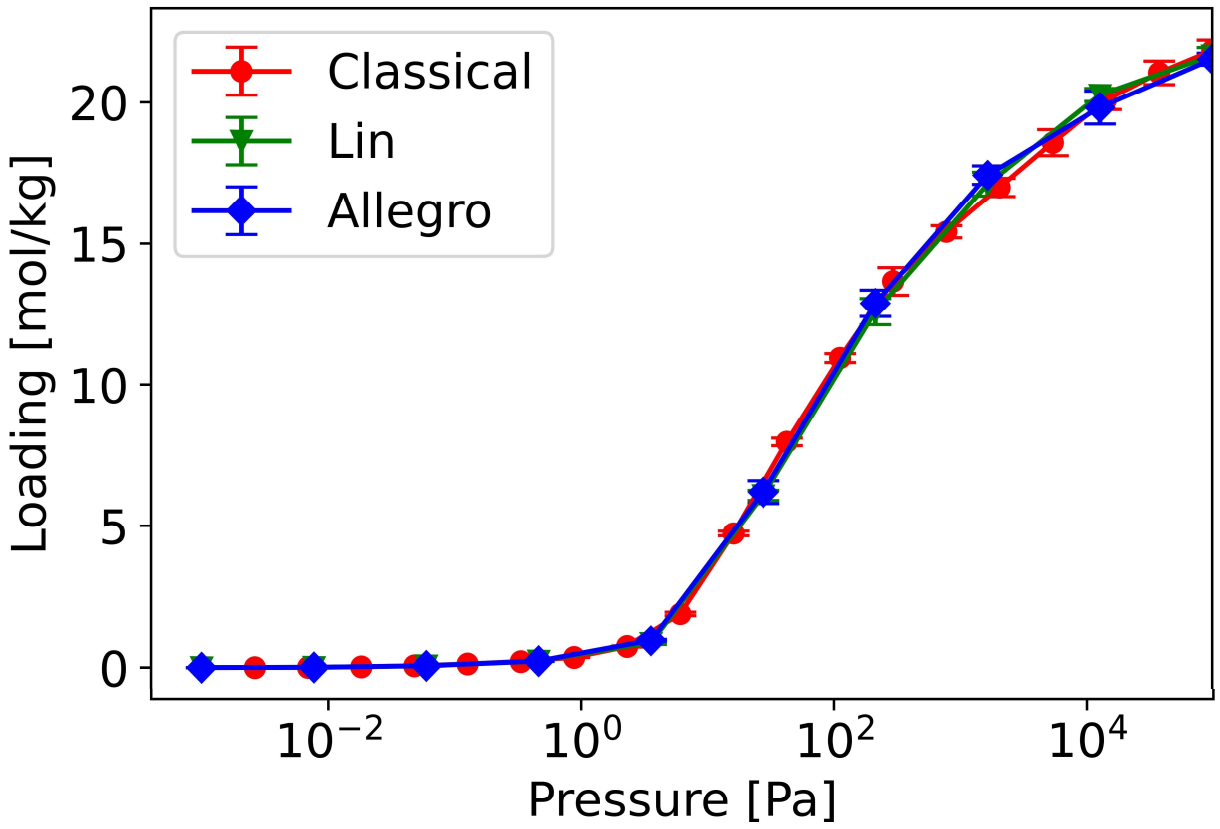


Figure 5. Simulated adsorption isotherms for Ar in Mg-MOF-74 at 77 K. A reference adsorption isotherm was generated using the classical force field shown in Table S9 and is shown in red. Adsorption isotherms predicted by the Lin and Allegro models are shown in green and blue, respectively.

6.4.2 CO_2 adsorption in Mg-MOF-74

In the presence of open metal sites in MOFs, classical force fields, such as the Universal Force Field (UFF)[68] and DREIDING[41], typically fail to reproduce the strong binding energies of adsorbate molecules at low pressure. CO_2 adsorption in Mg-MOF-74 is a well-known example of this kind. In previous studies, tailored analytical force fields were developed to capture the strong interactions of CO_2 with the open Mg site.[69] In our work, without restricting the potential energy

surface to a specific, analytical form, we used ML to model the complex potential energy surface of a single CO_2 molecule interacting with the MOF to a high accuracy. We generated training data of CO_2 binding energies in the unit cell of Mg-MOF-74 using DFT (see details in the SI). Following the ML architecture, the Lin model was trained with only the energy data, while the Allegro model was trained using both the energy and force labels with equal weights, as recommended in the original work.[64,70] We found that training with additional force information benefited the overall accuracy of the Allegro model but also increased the training time. If the training time permits, we suggest training the Allegro model using both energy and force data, even though only the output energy is useful in GCMC simulations. With both the Lin and Allegro models ready, we performed GCMC simulations using gRASPA to predict CO_2 adsorption in Mg-MOF-74 at 313 K and compared simulated results to experimental values. Details of training data generation, ML training, and GCMC simulations are available in Section S5.2, and the necessary input files are also provided in the SI as well as the GitHub repository at https://github.com/snurr-group/gRASPA/tree/main/Examples.

Figure 6 shows the comparison among simulated adsorption isotherms and experimental data. The Lin and Allegro models agree with each other very well, as expected, since they were trained on the same DFT data. Both the Lin and Allegro models predict adsorption isotherms that are in much better agreement with the experimental data than the simulations using a classical potential. The ML potential simulations especially outperform the classical simulations at low pressures (<50,000 Pa), where strong interactions between CO_2 molecules and the open Mg sites dominate the adsorption. Deviations of ML predicted adsorption loadings from the experimental data may be attributed to limitations of the DFT functionals[58] or possible defects in the

experimental samples. These results show the great promise of ML potentials for simulating challenging adsorption systems where classical force fields fall short.

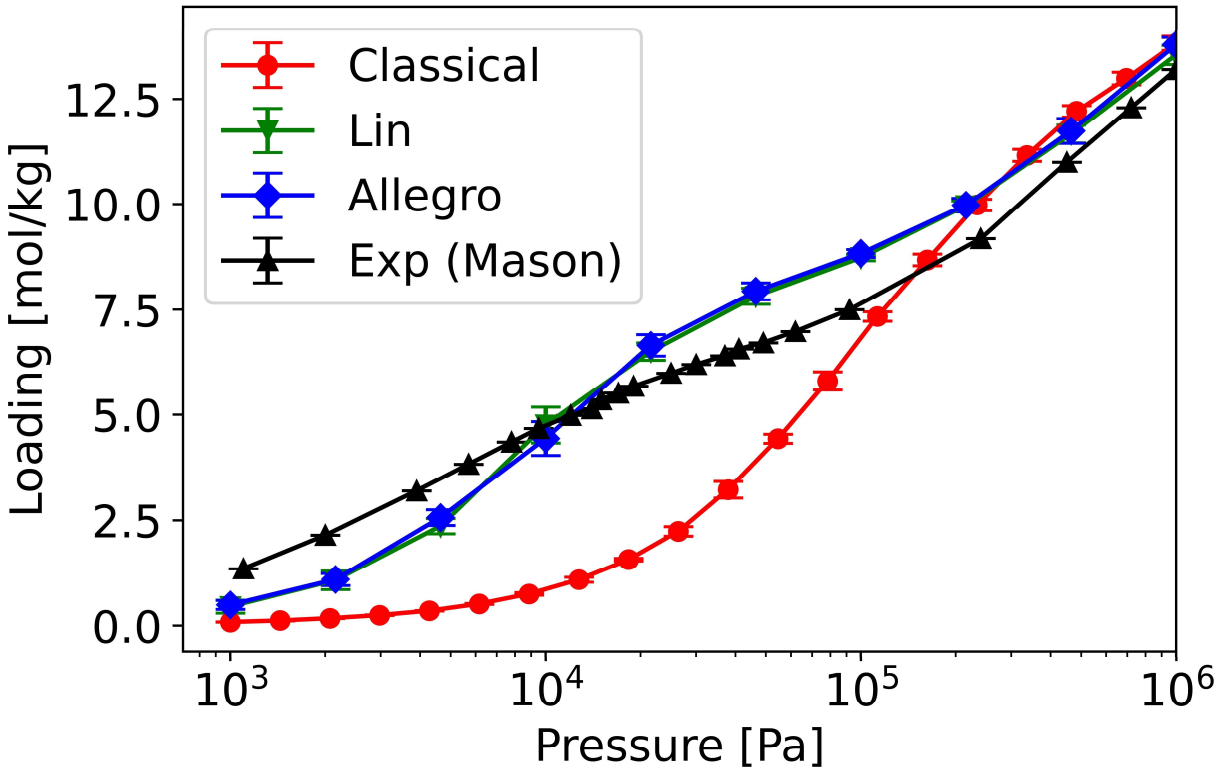


Figure 6. Adsorption isotherms for CO_2 in Mg-MOF-74 at 313 K. Simulated adsorption isotherms using the classical force field, the Lin model, and the Allegro model are shown in red, green, and blue, respectively. The experimental isotherm from Mason et al. (Ref. 71) is shown in black as a comparison.

6.4.3 Benchmarking simulation time

Finally, we benchmarked the simulation time using the ML potentials compared to classical force fields in gRASPA. For the Lin model, we first measured the time for calculating 10,000 energies. We wrote a small C++ program to call the model prediction 10,000 times on the same input features. We found that by running the model on the CPU instead of loading it and running it on the GPU, the model prediction is 4 times faster (Table S10). We also benchmarked the time for performing 10,000 Monte Carlo steps using both ML potentials for predicting argon adsorption in Mg-MOF-74 at 77 K and 100 Pa. The results are shown in Table 3. In this table, we divided the time into three categories: classical calculation time (pairwise distances, LJ), feature preparation time (sorting pairwise distances, generating neighbor lists), and prediction time (time spent by the ML model). It is worth noting that for the Lin model, when we further decomposed the preparation time for the features, we found that sorting the nine smallest pairwise distances between argon and the MOF is the most time-consuming step. It took 0.67 seconds using the "std::sort" function with CPU parallelism via the keyword "stdpar=multicore" during the compilation. Thus, we can recommend that for future development of fast ML potential models for MC simulations, developers should take the performance of their model as well as the feature preparation into consideration to develop a model that is both accurate and cost-effective to be deployed in MC simulation software.

Table 3. Benchmarking the performance of 10,000 Monte Carlo steps using the Lin and Allegro models for argon adsorption in MgMOF-74 at 77 K and 100 Pa. The Lin model prediction is performed on the CPU, while the Allegro model prediction is performed on the GPU.

Model	Classical Time [s]	Preparation Time [s]	Prediction Time [s]	Total Time [s]
Lin	0.43	0.91	1.44	2.78
Allegro	0.43	0.69	316.11	317.23

In summary, the Allegro model shows much lower MAE than the Lin model (Figures S6 and S7) thanks to Allegro's equivariant architecture for describing the detailed local environment of non-spherical molecules. Although, in general, the Allegro model would be recommended due to its state-of-the-art accuracy, training of an Allegro model is more time-consuming (around tens of hours on an Nvidia A100 graphic card) compared to the Lin model (several minutes on a single-core CPU). In addition, due to its simplicity, the Lin model executes much faster than the Allegro model during GCMC simulations (Table 3). Thus, the Lin model could be used to generate preliminary results.

7. Transition-Matrix Monte Carlo in the Grand Canonical Ensemble (GC-TMMC)

GC-TMMC is a powerful tool for obtaining relative free energies and relative probabilities of observing different states in phase coexistence[72] and studying adsorption phase equilibrium. It was originally proposed by Fitzgerald et al.[28,29] and then further developed by Errington et al.[72] for efficient implementation in GCMC using an additional bias that helps the system sample the less-probable states with higher frequencies. Recently, Siderius et al.[73] have used it to study the adsorption of CO_2 in IRMOF-1 and argon in carbon nanotubes, and Shen et al.[74] extended the GC-TMMC method to mixture simulations.

Here, we implemented GC-TMMC and tested it on TraPPE CO_2 vapor-liquid equilibrium[39] and then on ethane adsorption in hypothetical MOF #667[75] at 179 K.[76] We calculated the free energy versus density or loading for both systems and compared them against the results reported by NIST and by Li et al.,[76] respectively. The force field parameters are reported in Table S11 and Table S12 for CO_2 and for ethane in MOF #667, respectively.

In GC-TMMC simulations, for each Monte Carlo move, the collection matrix C is updated according to the acceptance probability for the Monte Carlo move:

$$C(Old \rightarrow New) = C(Old \rightarrow New) + P_{Acc}(Old \rightarrow New) \tag{5}$$
$$C(Old \rightarrow Old) = C(Old \rightarrow Old) + 1 - P_{Acc}(Old \rightarrow New)$$

where old and new represent the old and the new macrostates for the attempted Monte Carlo move, and P_{Acc} is the acceptance probability of the move. We defined macrostates by the number of molecules in the system, N. In the grand canonical ensemble, there are three possible directions in the macrostate space for a Monte Carlo move, which are +1 for insertion moves, 0 for canonical ensemble moves such as translation and rotation that do not change the number of molecules, and -1 for deletion moves. We denote the new macrostate as N', so N' can be either $N-1, N,$ or $N+1$. Then, the probability in the transition matrix for $N \rightarrow N'$ can be derived from the elements in the collection matrix C:

$$P_{N \rightarrow N'} = \frac{C(N \rightarrow N')}{C(N \rightarrow N-1) + C(N \rightarrow N) + C(N \rightarrow N+1)} \tag{6}$$

From the probabilities in the transition matrix, the probability for each macrostate π can be calculated as

$$\ln\Pi(N+1;\ \mu, V, T) = \ln\Pi(N;\ \mu, V, T) + \ln\left(\frac{P_{N \rightarrow N+1}}{P_{N+1 \rightarrow N}}\right), \tag{7}$$

and each macrostate has a related bias $\eta = -\ln\Pi(N; \mu, V, T)$ that helps the GC-TMMC simulation sample the less probable states more frequently. The bias for each macrostate was updated every 1 million steps,

and a total of 100 million MC steps were performed. More details about GC-TMMC can be found in the work of Hatch, Siderius, Shen, and Errington.[72,73,77]

To sample the free energies with higher efficiency, we adapted a divide-and-conquer approach suggested by Siderius et al.[73] by dividing the space of the macrostates (number of molecules in the simulation box) into different ranges and running a separate simulation for each range. Each simulation samples only the number of molecules within its range, and all insertion or deletion moves that try to move out of the range are rejected. This naturally works with the Nvidia-MPS discussed in Section 2.

We used the same setup of the GC-TMMC simulation for bulk CO_2 using the TraPPE model[39] as reported by NIST. Thus, we used a 30 x 30 x 30 $Å^3$ cubic box and performed the GC-TMMC simulations at temperatures between 230 K and 300 K. More details of this set of simulations can be found in the SI. We first located the equilibrium fugacity where the probabilities of observing the gas and liquid phases are equal. Then, the average density of a phase α is defined by:

$$\langle \rho \rangle = \frac{1}{V} \frac{\sum_{N \in \alpha} N \Pi(N; \mu, V, T)}{\sum_{N \in \alpha} \Pi(N; \mu, V, T)}, \tag{8}$$

where N is the number of molecules, and the summation is over the values of N that fall within the range of phase α, and V is the volume of the simulation box. In addition to equilibrium loadings, the equilibrium pressures can also be calculated from the grand potential Ω. The grand potential for phase $\alpha, \Omega_\alpha = -(\ln(\Sigma_{N \in \alpha} \Pi(N; \mu, V, T)/\Pi(0; \mu, V, T))$.[73] The pressure for phase α, $p_\alpha = \frac{k_B T}{V}(\ln(\Sigma_{N \in \alpha} \Pi(N; \mu, V, T)/\Pi(0; \mu, V, T))$. At phase equilibrium for phases α and β, the grand potentials are equal: $\Omega_\alpha = \Omega_\beta$. These pressures are also summarized in Table 4. From

Table 4, we can see that the densities of the two phases, as well as the equilibrium pressures calculated by gRASPA, are very close to the NIST values, showing that gRASPA can generate the vapor-liquid equilibrium of TraPPE CO_2.

Table 4. Summary of equilibrium vapor-liquid densities of TraPPE CO_2. The values outside and inside the parentheses are generated from gRASPA and obtained from NIST (Ref. 45), respectively.

T (K)	$\rho_{vap}\left(\frac{mol}{L}\right)$	$\rho_{liq}\left(\frac{mol}{L}\right)$	Equilibrium Fugacity (Bar)	Equilibrium Pressure (Bar)
230	$5.000*10^{-1}$ ($5.015*10^{-1}$)	$2.540*10^{1}$ ($2.551*10^{1}$)	$7.812*10^{0}$	$8.588*10^{0}$ ($8.625*10^{0}$)
240	$7.173*10^{-1}$ ($7.199*10^{-1}$)	$2.462*10^{1}$ ($2.464*10^{1}$)	$1.102*10^{1}$	$1.245*10^{1}$ ($1.248*10^{1}$)
250	$1.006*10^{0}$ ($1.009*10^{0}$)	$2.371*10^{1}$ ($2.371*10^{1}$)	$1.499*10^{1}$	$1.744*10^{1}$ ($1.748*10^{1}$)
260	$1.389*10^{0}$ ($1.390*10^{0}$)	$2.261*10^{1}$ ($2.270*10^{1}$)	$1.977*10^{1}$	$2.378*10^{1}$ ($2.381*10^{1}$)
270	$1.891*10^{0}$ ($1.896*10^{0}$)	$2.160*10^{1}$ ($2.158*10^{1}$)	$2.531*10^{1}$	$3.156*10^{1}$ ($3.165*10^{1}$)
280	$2.575*10^{0}$ ($2.582*10^{0}$)	$2.030*10^{1}$ ($2.029*10^{1}$)	$3.167*10^{1}$	$4.112*10^{1}$ ($4.123*10^{1}$)

We then simulated ethane adsorption in hypothetical MOF #667 at 179 K as the next test case. The saturation loading of ethane at 179 K in MOF #667 is 289 molecules per unit cell (375 cm^3/cm^3). So, we took the range of number of molecules from zero to 320 and divided the range into five individual simulations and ran them in parallel using Nvidia-MPS. Each simulation handled a range of 64 ethane molecules. Additional details about the simulations are summarized in the SI.

Figure 7 shows that there are three local minima in the free energy profile. They correspond to the one stable and two metastable loadings at the given pressure on the "canonical" isotherm, which was obtained in previous work by performing Widom test particle insertions at various loadings,[76] as shown in Figure S12. From the adsorption isotherm in Figure S12, there are three plateaus and two steps. At 27,500 Pa, three solutions exist on the stable and metastable regions along the canonical isotherm, having loadings of 41.5 cm^3/cm^3 (32 molecule/uc), 140.2 cm^3/cm^3 (108 molecule/uc), and 358.4 cm^3/cm^3 (276 molecule/uc). We can see that the loadings for the circled points on the canonical isotherms in Figure S12 and the local minima in the free energy profile in Figure 7 match well, validating our implementation of GC-TMMC in gRASPA.

Such free energy profiles provide a rapid way to obtain the adsorption isotherm since one can easily access the free energy profiles via histogram reweighting:

$$\ln \Pi_{N,\mu'} = \ln \Pi_{N,\mu} + \frac{N(\mu' - \mu)}{k_B T}, \tag{9}$$

where μ and μ' are the current and desired chemical potential. By plugging in the definition of fugacity $f = \frac{W_{IG} k_B T N}{V \langle W_r \rangle}$, where W_{IG} is the ideal chain Rosenbluth weight, V is the volume of the

simulation box, and $\langle W_r \rangle$ is the averaged Rosenbluth weight, and the definition of excess chemical potential $\mu_{ex} = -k_B T ln \langle W_r \rangle$, equation 9 can be rewritten as

$$\ln \Pi_{N,f'} = \ln \Pi_{N,f} + N \ln \left(\frac{f'}{f} \right), \tag{10}$$

where f and f' are the current and desired fugacities. The free energy profile at different pressures yields the loadings of the most probable states at these pressures, and the free energies can provide insights into the adsorption system.

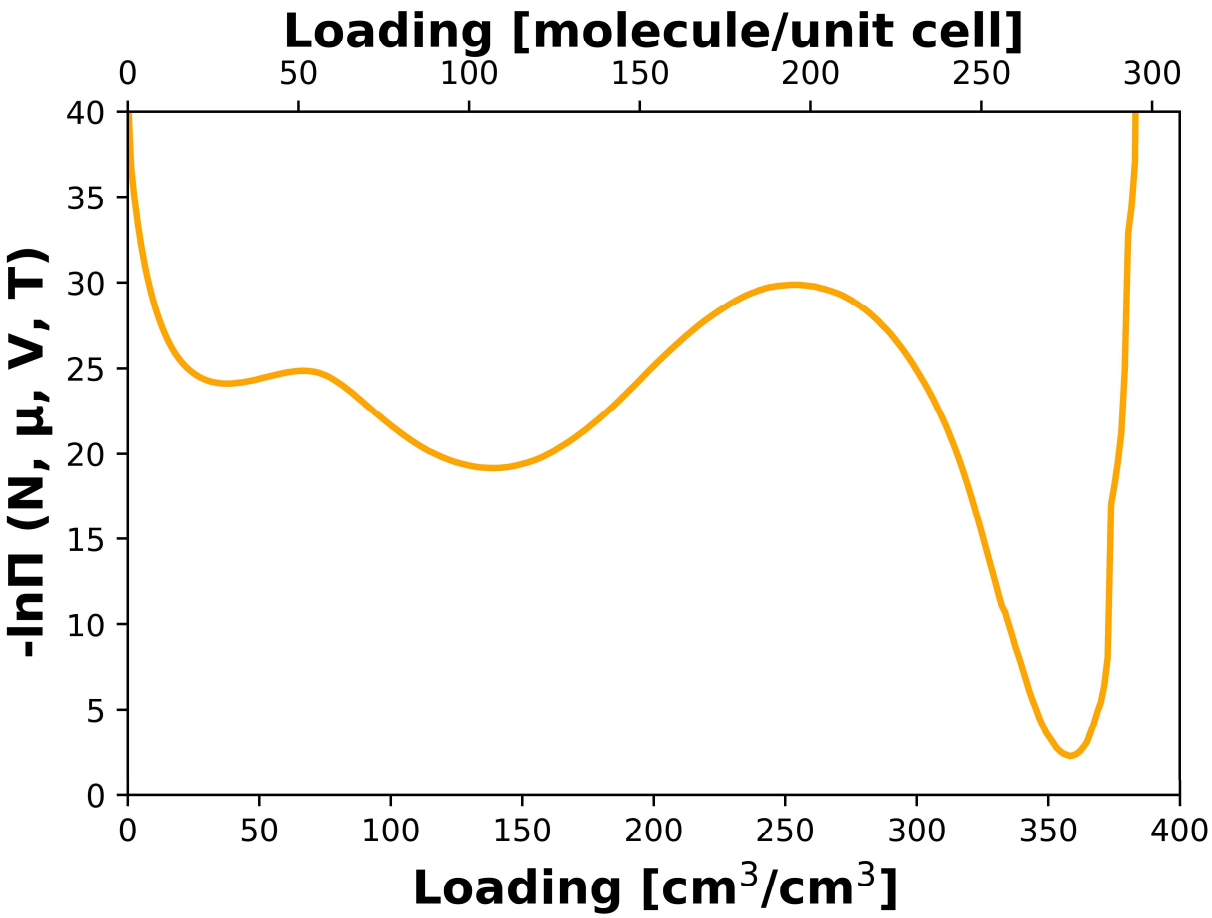


Figure 7. Free energy profile for ethane in MOF #667 at 179 K and 27,500 Pa calculated through GC-TMMC in gRASPA. The three local minima are 40.3 cm^3/cm^3 (31 molecules/uc), 142.8 cm^3/cm^3 (110 molecules/uc), and 358.4 cm^3/cm^3 (276 molecules/uc).

8. Framework Semi-Flexibility Move

In the original CPU RASPA convention, the framework is considered one component and one "molecule." Although this convention is straightforward and intuitive, it imposes some limitations on MOFs because it does not take advantage of their modular construction from metal nodes and organic linkers. For example, this convention makes it difficult to incorporate a MC move that moves only a part of the framework, such as rotating one or more linkers or rotating a functional group on a linker or on a node. To overcome this problem, in the gRASPA code we enabled the separation of MOF components to make the framework more modular. In the simulation input file, the user can specify the parts of the MOF to be separated into different components. Each framework component can then be assigned different MC moves.

Using this capability, we considered para-xylene adsorption in NU-2000[78] at 298 K and 3800 Pa. We used the same LJ parameters and partial charges as used by Idrees et al.[78] (Table S13). We used a cutoff of 12 Å for both the vdW and the real part of the Coulombic interactions. The vdW potentials were shifted so that they reach zero at the cutoff. Although the original LJ parameters and partial charges were derived for a fully flexible framework model, just to show how our semi-flexible move works, we did not consider the bonding, angle, and dihedral terms for this example. For p-xylene, translation, rotation, reinsertion, and swap (insertion and deletion) moves were attempted with equal probabilities. We also included a linker rotation move that rotates a randomly chosen linker around its linker axis to a random angle. Similar to a translation or rotation move in the canonical ensemble, the acceptance probability (P_{acc}) of this move is

$$P_{acc} = \min\{1, \mathrm{e}^{-\beta\Delta E}\}, \tag{11}$$

where β is the inverse temperature, and ΔE is the energy difference between the newly rotated and the original state of the selected linker. We used 100 million MC steps for the simulation, which roughly equals 3.3 to 5 million MC cycles in the RASPA terminology. We compared the results with and without linker rotation (i.e., in a fully rigid NU-2000 framework) using the same number of MC steps.

Table 5 shows the simulated result with the linker rotations compared to using a fully rigid framework model. We can see from Table 5 that the linker rotation move yields a much higher loading of p-xylene than the fully rigid model. Compared to the experimental saturation loading, which is 1.88 mol/kg, the simulations with the linker rotation move are in excellent agreement.

Table 5. Comparison of p-xylene adsorption in NU-2000 at 298 K and 3800 Pa between the semi-flexible model of NU-2000 where linker rotation move is used and the fully rigid model.

	Loading [molecule/uc]	Loading [mol/kg]
Linker Rotation	29.0	1.88
Fully Rigid	19.8	1.29

Figure 8 shows that the linker rotation move has a big effect on the framework structure and the adsorbed p-xylene configurations. It allows for denser packing of p-xylene molecules in the channels. In the rigid framework, the p-xylene molecules cannot efficiently utilize the space in the channels. Thus, the rigid representation of NU-2000 leads to a lower loading than the semi-flexible framework simulation. The results show that utilizing just a linker rotation move, which focuses on one type of motion of the

framework, can lead to the same conclusion as in the original work with NU-2000,[78] where the authors developed a fully flexible framework model for NU-2000.

As expected, the linker rotation move makes the simulation slower. The simulation with linker rotation moves took 6.5 hours, while the rigid framework simulation took 3.2 hours for the same number of MC steps. This is mainly because there are more pX molecules for the linker rotation simulation, and intra-host non-bonded interaction energies must be considered. In addition, the energy calculation takes longer because the linkers of NU-2000 are also subject to MC moves. However, with GPU acceleration, the current gRASPA simulation time is much less than that using RASPA-2. Thus, by incorporating a modular framework representation and semi-flexible framework moves, gRASPA can facilitate the development of molecular models and new force fields that take advantage of the modular nature of MOFs.

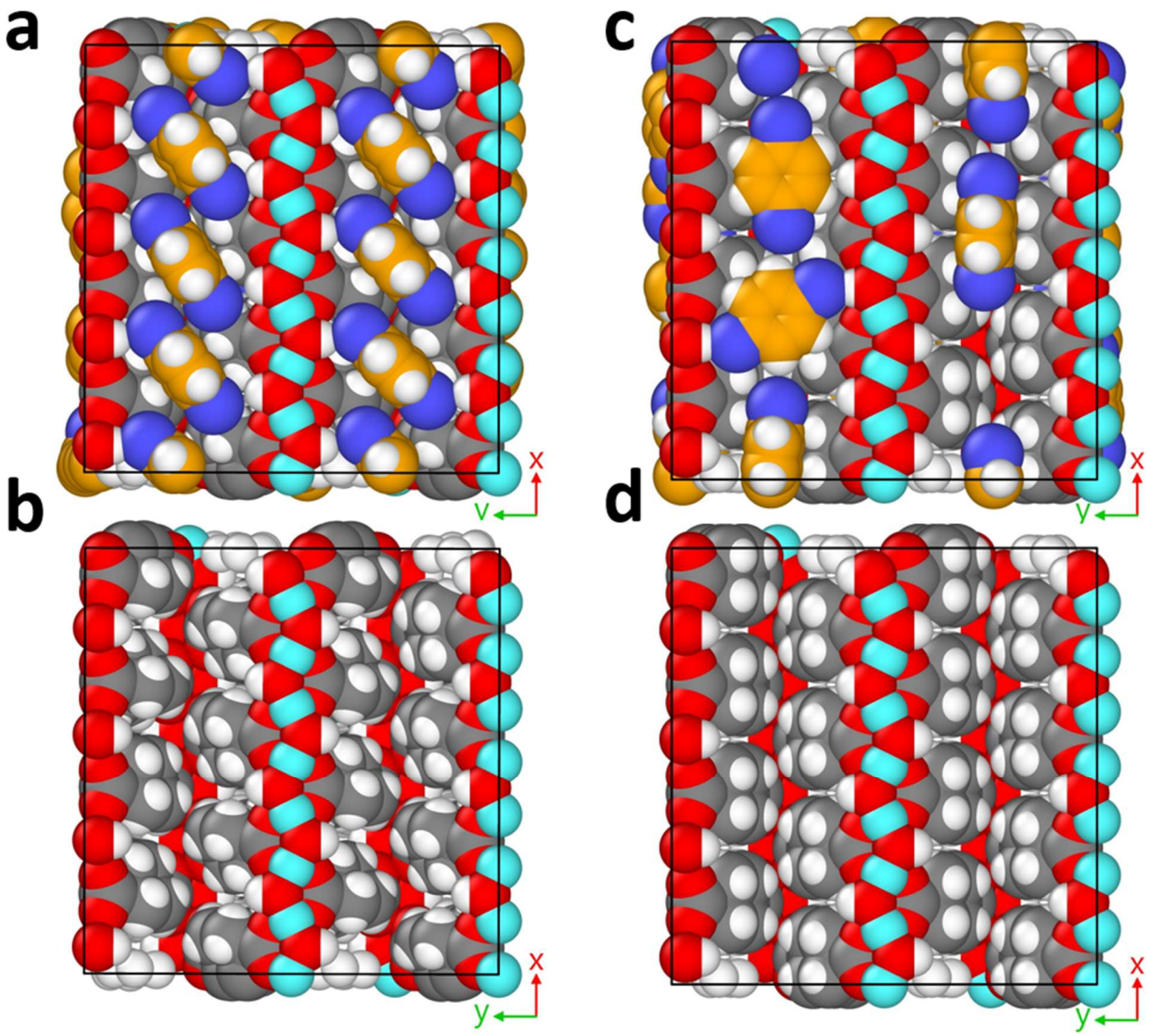


Figure 8. Comparison between semi-flexible (a, b) and fully rigid (c, d) framework simulations for p-xylene adsorption in NU-2000 at 298 K and 3800 Pa. Periodic boundary conditions were applied to the snapshots to wrap all atoms into the simulation box. The red, white, gray, cyan, yellow, and blue pseudo-atoms are oxygen, hydrogen, carbon, aluminum, carbon on the benzene ring, and methyl group, respectively. To show the linker rotation of the framework more clearly, in b and d, we deleted the p-xylene molecules from the snapshots.

9. High-Throughput Calculations using CUDA Blocks

By default, each gRASPA simulation utilizes multiple CUDA blocks for computation (see also Method section). To enable a high-throughput calculation (HTC) on a single graphics card, we can assign one independent MC simulation to each block on a GPU. Since every block contains a group of threads, these threads can be used for parallel evaluation of pairwise interactions and the Fourier part of the Ewald summation for the simulation. Here, we used 128 threads per block and ran different numbers of concurrent MC simulations to compare the speed and test the optimal operation condition of this type of simulation. We call this special version of the code gRASPA-HTC. Note that this gRASPA-HTC version is different from gRASPA or gRASPA-fast versions. gRASPA and gRASPA-fast benefit from the use of Nvidia MPS. gRASPA simulations using MPS are parallel processes, and each one uses one CPU core and offloads heavy calculations to the GPU. In this way, MPS is limited by the number of cores on the CPU. For our case, its limit is 24 simulations simultaneously. However, the block-based gRASPA-HTC runs the whole GCMC simulation on the GPU. This includes random selection of particles and moves, preparation of trial positions, and the Metropolis algorithm for accepting or rejecting a move. This naturally increases the throughput of simulations beyond 24. Similar ideas have been implemented by Kim et al.;[17] however, their work is not open-source, and their simulations rely on tabulating the energy calculations, including the Lennard-Jones, the real part and the Fourier part of the Ewald summations.[17] Our code performs real-time calculations of pairwise interactions and Ewald summation, aiming for higher accuracy.

We tested this HTC mode of gRASPA on an MC simulation of bulk methane. To perform a head-to-head comparison, we ran the HTC mode of gRASPA and a single-thread RASPA-2 simulation starting from the same initial configuration but different random seeds with 400 methane molecules in a 30 x 30 x 30 $Å^3$ cubic box at 95 K. Each simulation ran for 1,000 MC cycles, and each cycle only performed translation moves. The force field parameters are summarized in Table S14. The speed comparison in

Table 6 shows that the time required for the simulations remains nearly constant, whether executing a single simulation or up to 50 simulations simultaneously on a single graphics card using gRASPA's HTC mode. However, as the number of concurrent simulations surpasses approximately 250 to 500, the graphics card becomes saturated, causing the simulation time to increase. To show the ability of the code to calculate isotherms, we performed a GCMC simulation of bulk methane at 298 K in a 30 x 30 x 30 $Å^3$ cubic box at fugacities from 1 bar to 1000 bar. For the fugacities, 500 values were selected linearly in the $\log_{10}$ space between 1 and 1000 bar. For reference, we also conducted RASPA-2 GCMC simulations in this fugacity range. The results are shown in Figure 9 and show excellent agreement between the gRASPA-HTC code and RASPA-2.

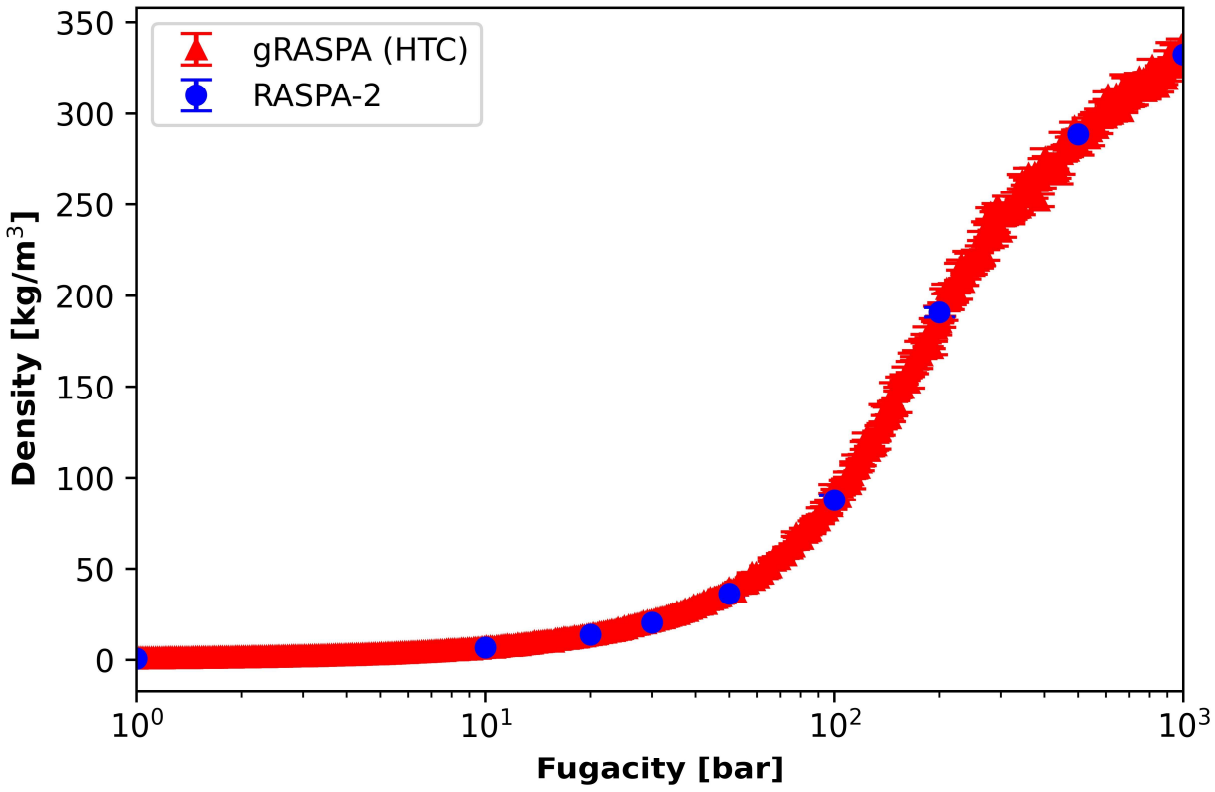


Figure 9. GCMC isotherms of bulk methane at 298 K simulated via RASPA-2 (blue circles) and gRASPA using the HTC mode (red triangles).

We also tested the performance of the HTC mode of gRASPA versus the normal RASPA-2 for CO_2 adsorption in MFI zeolite at 298 K and 10^4 Pa. This simulation used translation, rotation, and swap (insertion and deletion) moves. Note that we did not use CBMC for this example. We used a 12.8 Å cutoff for the Lennard-Jones interactions and a 12.0 Å cutoff for the short-range part of the Coulombic interactions. Ewald summation was used to calculate the Fourier part of the Coulombic interactions. We used 10^{-6} for the Ewald precision. The performance is summarized in Table 7, which shows that using the same amount of time, the methane case performed 400,000 MC steps while the CO_2 case performed 40,000 steps. This is because compared to running methane simulations, CO_2 adsorption in MFI simulations is more complicated and involves Ewald summation. For the CO_2 adsorption case, when running 50 GCMC simulations concurrently, each simulation is at least three times faster than a single-core RASPA-2 simulation. When running 500 simulations concurrently on a single graphics card, the performance per simulation is 1.5 times slower than that of a single-core RASPA-2 simulation, but this trade-off translates to a remarkable throughput gain of 500 times. Executing these 500 concurrent GCMC simulations on one graphic card took 36.1 seconds, while executing these 500 simulations on CPUs would require at least 21 CPU chips, assuming each is equipped with 24 cores, such as the AMD Threadripper processor used in this study, and assuming each RASPA-2 simulation takes one CPU core. Such an undertaking is typically accomplished by submitting CPU jobs to a large, centralized computing cluster. Using the HTC mode of gRASPA, researchers with limited computational resources (for example, just a laptop with an RTX 3090 GPU) can explore the adsorption space as quickly as someone with access to a large CPU cluster, expanding access to computational materials discovery. People with access to supercomputers can also benefit from the HTC mode of gRASPA. For example, to screen the CoRE-MOF 2019 database[79] for CO_2 capture, a mere 25 RTX 3090 GPUs, each handling 500 MOFs, are sufficient to concurrently compute the adsorption properties of all 14,142 MOF structures within the

database, a task easily accomplished by the multi-GPU nodes available on modern clusters. Furthermore, the projected release of new generations of Nvidia GPUs and technologies such as NVLink and PCIe connections between GPUs are expected to amplify the speed of gRASPA's HTC mode. Despite the maximized throughput of the HTC mode, in terms of speed of a single GCMC simulation, it is slower and less capable than the gRASPA base code. For the CO_2 case at 298 K, 10^4 Pa with 40,000 MC steps and no CBMC, the base gRASPA code takes 5 seconds to finish, compared to the values in Table 7, which is 6.2 seconds for one simulation. Also, the HTC mode currently only performs non-CBMC moves, while CBMC is important for many GCMC simulations. The user can use the HTC mode to quickly explore materials with fewer compute resources, then refine the calculated result using the non-HTC code.

Table 6. Run times for multiple concurrent simulations using CUDA blocks in the HTC mode of gRASPA for bulk methane at 95 K. The system contains 400 methane molecules, and 1000 MC cycles (equal to 400,000 MC steps) are performed. As a comparison, the same simulation with RASPA-2 using a single core took 6.73 seconds.

Number of concurrent simulations	gRASPA-HTC time [seconds]
1	6.00
2	6.01
5	6.01
10	6.06
20	6.04
50	6.09
100	10.52
200	15.26
250	20.20
500	35.28
1000	65.35
5000	311.43

Table 7. Run times for multiple concurrent simulations using CUDA blocks in the HTC mode of gRASPA for CO_2 adsorption in MFI zeolite at 298 K and 10^4 Pa for 40,000 MC steps. As a comparison, the same simulation with RASPA-2 using a single core took 20.81 seconds.

Number of concurrent simulations	gRASPA-HTC time [seconds]
1	6.24
2	6.26
5	6.30
10	6.30
20	6.27
50	6.42
100	10.7
200	15.57
400	26.03
500	36.15
1000	66.71

Conclusions

We have developed an open-source Monte Carlo simulation code, gRASPA, that runs on GPUs and shows substantial speed-ups compared to serial, CPU implementations of Monte Carlo. The utilization of Nvidia MPS significantly enhances the throughput of gRASPA simulations on a graphics card, with the Fast version displaying much better scalability for high-throughput screening. The additional HTC mode of gRASPA expands the limit of high-throughput materials discovery by allowing users to run a large number of GCMC simulations on a single GPU device. In addition to improved speed for MC simulations of adsorption, the code can integrate ML force fields for improved accuracy. We demonstrated that GCMC simulations with ML potentials trained on DFT data show improvements in adsorption isotherm predictions for CO_2 adsorption in Mg-MOF-74 compared to classical force fields. GC-TMMC is implemented in

gRASPA for calculating free energy profiles, which allows complete adsorption isotherms to be obtained very quickly. The code also allows users to specify different components of MOF structures (for example, nodes and linkers) and to incorporate different Monte Carlo moves for different components, which we demonstrated for simulations of p-xylene adsorption in NU-2000 with rotation moves for the MOF linkers.

Beyond these features, the code supports other MC simulations, including NVT-Gibbs Monte Carlo, Widom test particle insertions, and continuous-fractional component (CFC) Monte Carlo, and we have demonstrated its use for vapor-liquid equilibrium simulations. We plan to continue adding features and enhance the performance of the code, and since the code is open-source other users may add their own capabilities.

Associated Content

- Details about the timings of RASPA-2, RASPA-3, and gRASPA, training of ML potential and hyperparameters tuning, force field parameters used, and other supporting figures and tables (PDF)
- The default code of gRASPA is available at https://github.com/snurr-group/gRASPA.
- gRASPA-fast version, gRASPA translated to SYCL, and gRASPA-HTC are available as releases at https://github.com/snurr-group/gRASPA/releases.
- Simulation input and force field files for the examples in this work are available at https://github.com/snurr-group/gRASPA/tree/main/Examples.
- gRASPA documentation is available at https://zhaoli2042.github.io/gRASPA-mkdoc/.

Notes

The authors declare no competing financial interest.

Acknowledgment

This work was supported by the U.S. Department of Energy, Office of Basic Energy Sciences, Division of Chemical Sciences, Geosciences and Biosciences under Award No. DE-SC0023454. This research was supported in part through the computational resources and staff contributions provided for the Quest high performance computing facility at Northwestern University which is jointly supported by the Office of the Provost, the Office for Research, and Northwestern University Information Technology. Some of this work was supported by the U.S. Department of Energy, Office of Basic Energy Sciences, under award DE-FG02-17ER16362, as part of the Computational Chemical Sciences Program. This research used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 using NERSC award BES-ERCAP0023154. This research also used computational resources provided by the Center for Computational Research at the University at Buffalo.[80] Z.L. thanks Dr. Daniel W. Siderius and Dr. Harold W. Hatch from the National Institute of Standards and Technology for helpful discussion on details of Ewald summation for reproducing the reference energies. Z.L. also

thanks Dr. Thang Pham, Jiayang Liu, Kunhuan Liu, Maytham Alzayer, and Dr. Filip Formalik for their helpful discussions. Special thanks go to Dr. Thang Pham for help building the GPU workstation in the Snurr lab.

References

(1) Lindorff-Larsen, K.; Piana, S.; Dror, R. O.; Shaw, D. E. How Fast-Folding Proteins Fold. *Science* **2011**, *334* (6055), 517–520. https://doi.org/10.1126/science.1208351.

(2) Nguyen, P. H.; Ramamoorthy, A.; Sahoo, B. R.; Zheng, J.; Faller, P.; Straub, J. E.; Dominguez, L.; Shea, J.-E.; Dokholyan, N. V.; De Simone, A.; Ma, B.; Nussinov, R.; Najafi, S.; Ngo, S. T.; Loquet, A.; Chiricotto, M.; Ganguly, P.; McCarty, J.; Li, M. S.; Hall, C.; Wang, Y.; Miller, Y.; Melchionna, S.; Habenstein, B.; Timr, S.; Chen, J.; Hnath, B.; Strodel, B.; Kayed, R.; Lesné, S.; Wei, G.; Sterpone, F.; Doig, A. J.; Derreumaux, P. Amyloid Oligomers: A Joint Experimental/Computational Perspective on Alzheimer's Disease, Parkinson's Disease, Type II Diabetes, and Amyotrophic Lateral Sclerosis. *Chem. Rev.* **2021**, *121* (4), 2545–2647. https://doi.org/10.1021/acs.chemrev.0c01122.

(3) Wang, Y.; Bunce, S. J.; Radford, S. E.; Wilson, A. J.; Auer, S.; Hall, C. K. Thermodynamic Phase Diagram of Amyloid-β (16–22) Peptide. *Proceedings of the National Academy of Sciences* **2019**, *116* (6), 2091–2096. https://doi.org/10.1073/pnas.1819592116.

(4) Shaw, D. E.; Adams, P. J.; Azaria, A.; Bank, J. A.; Batson, B.; Bell, A.; Bergdorf, M.; Bhatt, J.; Butts, J. A.; Correia, T.; Dirks, R. M.; Dror, R. O.; Eastwood, M. P.; Edwards, B.; Even, A.; Feldmann, P.; Fenn, M.; Fenton, C. H.; Forte, A.; Gagliardo, J.; Gill, G.; Gorlatova, M.; Greskamp, B.; Grossman, J. P.; Gullingsrud, J.; Harper, A.; Hasenplaugh, W.; Heily, M.; Heshmat, B. C.; Hunt, J.; Ierardi, D. J.; Iserovich, L.; Jackson, B. L.; Johnson, N. P.; Kirk, M. M.; Klepeis, J. L.; Kuskin, J. S.; Mackenzie, K. M.; Mader, R. J.; McGowen, R.; McLaughlin, A.; Moraes, M. A.; Nasr, M. H.; Nociolo, L. J.; O'Donnell, L.; Parker, A.; Peticolas, J. L.; Pocina, G.; Predescu, C.; Quan, T.; Salmon, J. K.; Schwink, C.; Shim, K. S.; Siddique, N.; Spengler, J.; Szalay, T.; Tabladillo, R.; Tartler, R.; Taube, A. G.; Theobald, M.; Towles, B.; Vick, W.; Wang, S. C.; Wazlowski, M.; Weingarten, M. J.; Williams, J. M.; Yuh, K. A. Anton 3: Twenty Microseconds of Molecular Dynamics Simulation before Lunch. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*; SC '21; Association for Computing Machinery: New York, NY, USA, 2021; pp 1–11. https://doi.org/10.1145/3458817.3487397.

(5) Anderson, J. A.; Lorenz, C. D.; Travesset, A. General Purpose Molecular Dynamics Simulations Fully Implemented on Graphics Processing Units. *Journal of Computational Physics* **2008**, *227* (10), 5342–5359. https://doi.org/10.1016/j.jcp.2008.01.047.

(6) Case, D. A.; Aktulga, H. M.; Belfon, K.; Ben-Shalom, I.; Brozell, S. R.; Cerutti, D. S.; III, T. E. C.; Cruzeiro, V. W. D.; Darden, T. A.; Duke, R. E.; Giambasu, G.; Gilson, M. K.; Gohlke, H.; Goetz, A. W.; Harris, R.; Izadi, S.; Izmailov, S. A.; Jin, C.; Kasavajhala, K.; Kaymak, M. C.; King, E.; Kovalenko, A.; Kurtzman, T.; Lee, T.; LeGrand, S.; Li, P.; Lin, C.; Liu, J.; Luchko, T.; Luo, R.; Machado, M.; Man, V.; Manathunga, M.; Merz, K. M.; Miao, Y.; Mikhailovskii, O.; Monard, G.; Nguyen, H.; O'Hearn, K. A.; Onufriev, A.; Pan, F.; Pantano, S.; Qi, R.; Rahnamoun, A.; Roe, D. R.; Roitberg, A.; Sagui, C.; Schott-Verdugo, S.; Shen, J.; Simmerling, C. L.; Skrynnikov, N. R.; Smith, J.; Swails, J.; Walker, R. C.; Wang, J.; Wei, H.; Wolf, R. M.; Wu, X.; Xue, Y.; York, D. M.; Zhao, S.; Kollman, P. A. *Amber 2021*; University of California, San Francisco, 2021.

(7) Abraham, M. J.; Murtola, T.; Schulz, R.; Páll, S.; Smith, J. C.; Hess, B.; Lindahl, E. GROMACS: High Performance Molecular Simulations through Multi-Level Parallelism from Laptops to Supercomputers. *SoftwareX* **2015**, *1–2*, 19–25. https://doi.org/10.1016/j.softx.2015.06.001.
(8) Plimpton, S. Fast Parallel Algorithms for Short-Range Molecular Dynamics. *Journal of Computational Physics* **1995**, *117* (1), 1–19. https://doi.org/10.1006/jcph.1995.1039.
(9) Allen, M. P.; Tildesley, D. J. *Computer Simulation of Liquids: Second Edition*; Oxford University Press, 2017.
(10) Siepmann, J. I.; Frenkel, D. Configurational Bias Monte Carlo: A New Sampling Scheme for Flexible Chains. *Molecular Physics* **1992**, *75* (1), 59–70. https://doi.org/10.1080/00268979200100061.
(11) Snurr, R. Q.; Bell, A. T.; Theodorou, D. N. Prediction of Adsorption of Aromatic Hydrocarbons in Silicalite from Grand Canonical Monte Carlo Simulations with Biased Insertions. *J. Phys. Chem.* **1993**, *97* (51), 13742–13752. https://doi.org/10.1021/j100153a051.
(12) Anderson, J. A.; Glaser, J.; Glotzer, S. C. HOOMD-Blue: A Python Package for High-Performance Molecular Dynamics and Hard Particle Monte Carlo Simulations. *Computational Materials Science* **2020**, *173*, 109363. https://doi.org/10.1016/j.commatsci.2019.109363.
(13) Schönhöfer, P. W. A.; Sun, K.; Mao, X.; Glotzer, S. C. Rationalizing Euclidean Assemblies of Hard Polyhedra from Tessellations in Curved Space. *Phys. Rev. Lett.* **2023**, *131* (25), 258201. https://doi.org/10.1103/PhysRevLett.131.258201.
(14) 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. https://doi.org/10.1016/j.softx.2018.11.005.
(15) Moučka, F.; Rouha, M.; Nezbeda, I. Efficient Multiparticle Sampling in Monte Carlo Simulations on Fluids: Application to Polarizable Models. *The Journal of Chemical Physics* **2007**, *126* (22), 224106. https://doi.org/10.1063/1.2745293.
(16) Kim, J.; Rodgers, J. M.; Athènes, M.; Smit, B. Molecular Monte Carlo Simulations Using Graphics Processing Units: To Waste Recycle or Not? *J. Chem. Theory Comput.* **2011**, *7* (10), 3208–3222. https://doi.org/10.1021/ct200474j.
(17) Kim, J.; Smit, B. Efficient Monte Carlo Simulations of Gas Molecules Inside Porous Materials. *J. Chem. Theory Comput.* **2012**, *8* (7), 2336–2343. https://doi.org/10.1021/ct3003699.
(18) Kim, J.; Martin, R. L.; Rübel, O.; Haranczyk, M.; Smit, B. High-Throughput Characterization of Porous Materials Using Graphics Processing Units. *J. Chem. Theory Comput.* **2012**, *8* (5), 1684–1693. https://doi.org/10.1021/ct200787v.
(19) Lee, S.; Kim, B.; Cho, H.; Lee, H.; Lee, S. Y.; Cho, E. S.; Kim, J. Computational Screening of Trillions of Metal–Organic Frameworks for High-Performance Methane Storage. *ACS Appl. Mater. Interfaces* **2021**, *13* (20), 23647–23654. https://doi.org/10.1021/acsami.1c02471.
(20) Lin, L.-C.; Berger, A. H.; Martin, R. L.; Kim, J.; Swisher, J. A.; Jariwala, K.; Rycroft, C. H.; Bhown, A. S.; Deem, M. W.; Haranczyk, M.; Smit, B. In Silico Screening of Carbon-Capture Materials. *Nature Mater* **2012**, *11* (7), 633–641. https://doi.org/10.1038/nmat3336.
(21) Shah, J. K.; Marin-Rimoldi, E.; Mullen, R. G.; Keene, B. P.; Khan, S.; Paluch, A. S.; Rai, N.; Romanielo, L. L.; Rosch, T. W.; Yoo, B.; Maginn, E. J. Cassandra: An Open Source Monte Carlo Package for Molecular Simulation. *Journal of Computational Chemistry* **2017**, *38* (19), 1727–1739. https://doi.org/10.1002/jcc.24807.
(22) Martin, M. G. MCCCS Towhee: A Tool for Monte Carlo Molecular Simulation. *Molecular Simulation* **2013**, *39* (14–15), 1212–1222. https://doi.org/10.1080/08927022.2013.828208.
(23) Dubbeldam, D.; Calero, S.; Ellis, D. E.; Snurr, R. Q. RASPA: Molecular Simulation Software for Adsorption and Diffusion in Flexible Nanoporous Materials. *Molecular Simulation* **2016**, *42* (2), 81–101. https://doi.org/10.1080/08927022.2015.1010082.

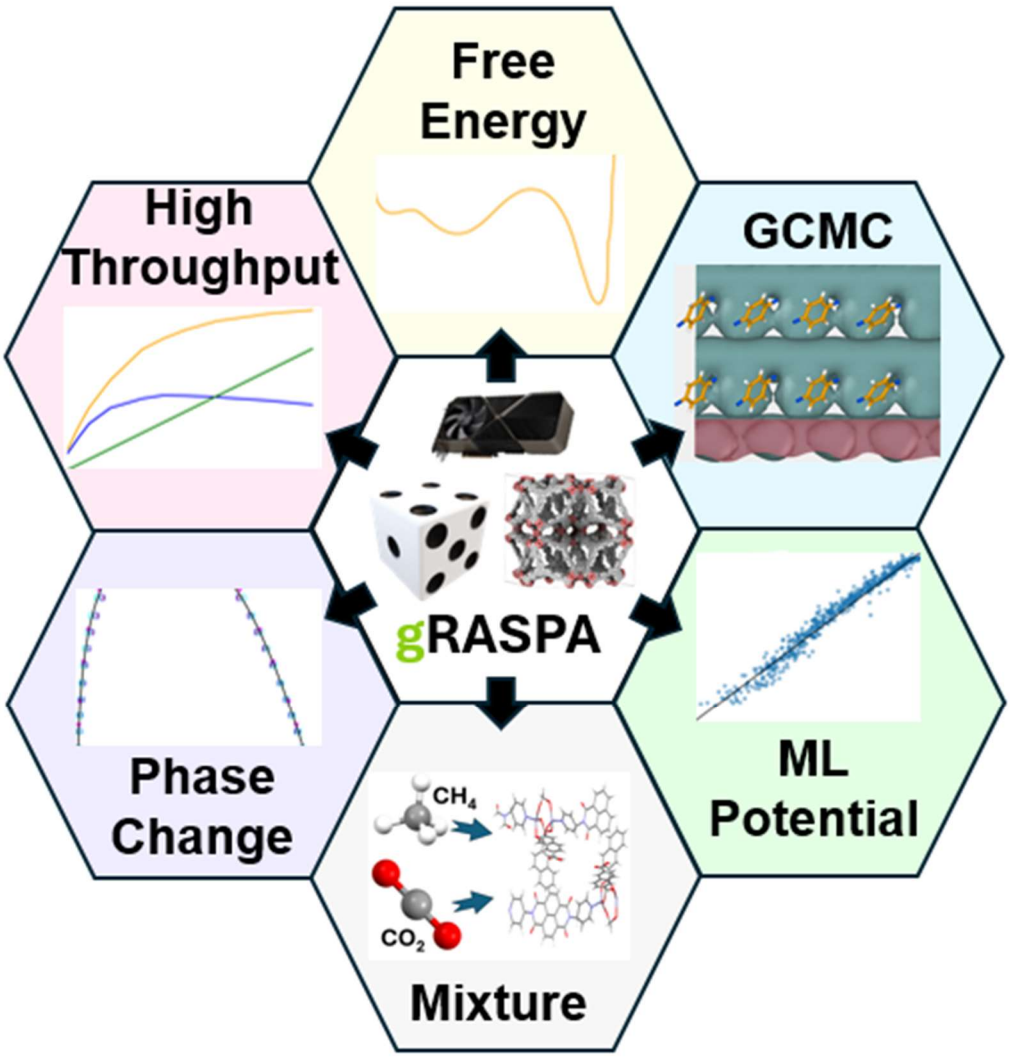
(24) Shi, W.; Maginn, E. J. Continuous Fractional Component Monte Carlo: An Adaptive Biasing Method for Open System Atomistic Simulations. *J. Chem. Theory Comput.* **2007**, *3* (4), 1451–1463. https://doi.org/10.1021/ct7000039.
(25) Torres-Knoop, A.; Balaji, S. P.; Vlugt, T. J. H.; Dubbeldam, D. A Comparison of Advanced Monte Carlo Methods for Open Systems: CFCMC vs CBMC. *J. Chem. Theory Comput.* **2014**, *10* (3), 942–952. https://doi.org/10.1021/ct4009766.
(26) Y.A. Ran; S. Sharma; S.R.G. Balestra; Z. Li; S. Calero; T.J.H. Vlugt; R.Q. Snurr; D. Dubbeldam. RASPA3: A Monte Carlo Code for Computing Adsorption and Diffusion in Nanoporous Materials and Thermodynamics Properties of Fluids. *Submitted*.
(27) Rosi, N. L.; Kim, J.; Eddaoudi, M.; Chen, B.; O'Keeffe, M.; Yaghi, O. M. Rod Packings and Metal−Organic Frameworks Constructed from Rod-Shaped Secondary Building Units. *J. Am. Chem. Soc.* **2005**, *127* (5), 1504–1518. https://doi.org/10.1021/ja045123o.
(28) Fitzgerald, M.; Picard, R. R.; Silver, R. N. Canonical Transition Probabilities for Adaptive Metropolis Simulation. *EPL* **1999**, *46* (3), 282. https://doi.org/10.1209/epl/i1999-00257-1.
(29) Fitzgerald, M.; Picard, R. R.; Silver, R. N. Monte Carlo Transition Dynamics and Variance Reduction. *Journal of Statistical Physics* **2000**, *98* (1), 321–345. https://doi.org/10.1023/A:1018635108073.
(30) Nejahi, Y.; Barhaghi, M. S.; Schwing, G.; Schwiebert, L.; Potoff, J. Update 2.70 to "GOMC: GPU Optimized Monte Carlo for the Simulation of Phase Equilibria and Physical Properties of Complex Fluids." *SoftwareX* **2021**, *13*, 100627. https://doi.org/10.1016/j.softx.2020.100627.
(31) Mick, J.; Hailat, E.; Russo, V.; Rushaidat, K.; Schwiebert, L.; Potoff, J. GPU-Accelerated Gibbs Ensemble Monte Carlo Simulations of Lennard-Jonesium. *Computer Physics Communications* **2013**, *184* (12), 2662–2669. https://doi.org/10.1016/j.cpc.2013.06.020.
(32) SPC/E Water Reference Calculations - 10Å Cutoff, 2021. https://www.nist.gov/mml/csd/chemical-informatics-group/spce-water-reference-calculations-10a-cutoff.
(33) Ran, Y. A.; Sharma, S.; Balestra, S. R. G.; Li, Z.; Calero, S.; Vlugt, T. J. H.; Snurr, R. Q.; Dubbeldam, D. RASPA3: A Monte Carlo Code for Computing Adsorption and Diffusion in Nanoporous Materials and Thermodynamics Properties of Fluids. *The Journal of Chemical Physics* **2024**, *161* (11), 114106. https://doi.org/10.1063/5.0226249.
(34) Frenkel, D.; Smit, B. *Understanding Molecular Simulation: From Algorithms to Applications*, 1st ed.; Academic Press, Inc.: USA, 1996.
(35) Ma, B.-Q.; Mulfort, K. L.; Hupp, J. T. Microporous Pillared Paddle-Wheel Frameworks Based on Mixed-Ligand Coordination of Zinc Ions. *Inorg. Chem.* **2005**, *44* (14), 4912–4914. https://doi.org/10.1021/ic050452i.
(36) Gupta, A.; Chempath, S.; Sanborn, M. J.; Clark, L. A.; Snurr, R. Q. Object-Oriented Programming Paradigms for Molecular Modeling. *Molecular Simulation* **2003**, *29* (1), 29–46. https://doi.org/10.1080/0892702031000065719.
(37) Bae, Y.-S.; Mulfort, K. L.; Frost, H.; Ryan, P.; Punnathanam, S.; Broadbelt, L. J.; Hupp, J. T.; Snurr, R. Q. Separation of CO2 from CH4 Using Mixed-Ligand Metal−Organic Frameworks. *Langmuir* **2008**, *24* (16), 8592–8598. https://doi.org/10.1021/la800555x.
(38) Peng, D.-Y.; Robinson, D. B. A New Two-Constant Equation of State. *Ind. Eng. Chem. Fund.* **1976**, *15* (1), 59–64. https://doi.org/10.1021/i160057a011.
(39) Potoff, J. J.; Siepmann, J. I. Vapor–Liquid Equilibria of Mixtures Containing Alkanes, Carbon Dioxide, and Nitrogen. *AIChE Journal* **2001**, *47* (7), 1676–1682. https://doi.org/10.1002/aic.690470719.
(40) Goodbody, S. J.; Watanabe, K.; MacGowan, D.; Walton, J. P. R. B.; Quirke, N. Molecular Simulation of Methane and Butane in Silicalite. *J. Chem. Soc., Faraday Trans.* **1991**, *87* (13), 1951–1958. https://doi.org/10.1039/FT9918701951.

(41) Mayo, S. L.; Olafson, B. D.; Goddard, W. A. DREIDING: A Generic Force Field for Molecular Simulations. *J. Phys. Chem.* **1990**, *94* (26), 8897–8909. https://doi.org/10.1021/j100389a010.
(42) Panagiotopoulos, A. Z. Direct Determination of Phase Coexistence Properties of Fluids by Monte Carlo Simulation in a New Ensemble. *Molecular Physics* **1987**, *61* (4), 813–826. https://doi.org/10.1080/00268978700101491.
(43) Martin, M. G.; Siepmann, J. I. Transferable Potentials for Phase Equilibria. 1. United-Atom Description of n-Alkanes. *J. Phys. Chem. B* **1998**, *102* (14), 2569–2577. https://doi.org/10.1021/jp972543+.
(44) Lemmon, E.; McLinden, M.; Friend, D. *NIST Chemistry WebBook, NIST Standard Reference Database Number 69ch.*; Gaithersburg, MD: National Institute of Standards and Technology.
(45) SAT-TMMC: Liquid-Vapor Coexistence Properties - TraPPE Carbon Dioxide (LRC), 2022. https://www.nist.gov/mml/csd/chemical-informatics-group/sat-tmmc-liquid-vapor-coexistence-properties-trappe-carbon-0.
(46) Fetisov, E. O.; Kuo, I.-F. W.; Knight, C.; VandeVondele, J.; Van Voorhis, T.; Siepmann, J. I. First-Principles Monte Carlo Simulations of Reaction Equilibria in Compressed Vapors. *ACS Cent. Sci.* **2016**, *2* (6), 409–415. https://doi.org/10.1021/acscentsci.6b00095.
(47) Hutter, J.; Iannuzzi, M.; Schiffmann, F.; VandeVondele, J. Cp2k: Atomistic Simulations of Condensed Matter Systems. *WIREs Computational Molecular Science* **2014**, *4* (1), 15–25. https://doi.org/10.1002/wcms.1159.
(48) Taherivardanjani, S.; Elfgen, R.; Reckien, W.; Suarez, E.; Perlt, E.; Kirchner, B. Benchmarking the Computational Costs and Quality of Vibrational Spectra from Ab Initio Simulations. *Advanced Theory and Simulations* **2022**, *5* (1), 2100293. https://doi.org/10.1002/adts.202100293.
(49) Jia, W.; Wang, H.; Chen, M.; Lu, D.; Lin, L.; Car, R.; E, W.; Zhang, L. Pushing the Limit of Molecular Dynamics with Ab Initio Accuracy to 100 Million Atoms with Machine Learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*; SC '20; IEEE Press: Atlanta, Georgia, 2020; pp 1–14.
(50) Formalik, F.; Shi, K.; Joodaki, F.; Wang, X.; Snurr, R. Q. Exploring the Structural, Dynamic, and Functional Properties of Metal-Organic Frameworks through Molecular Modeling. *Advanced Functional Materials n/a* (n/a), 2308130. https://doi.org/10.1002/adfm.202308130.
(51) Friederich, P.; Häse, F.; Proppe, J.; Aspuru-Guzik, A. Machine-Learned Potentials for next-Generation Matter Simulations. *Nat. Mater.* **2021**, *20* (6), 750–761. https://doi.org/10.1038/s41563-020-0777-6.
(52) Behler, J.; Parrinello, M. Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces. *Phys. Rev. Lett.* **2007**, *98* (14), 146401. https://doi.org/10.1103/PhysRevLett.98.146401.
(53) Bartók, A. P.; Payne, M. C.; Kondor, R.; Csányi, G. Gaussian Approximation Potentials: The Accuracy of Quantum Mechanics, without the Electrons. *Phys. Rev. Lett.* **2010**, *104* (13), 136403. https://doi.org/10.1103/PhysRevLett.104.136403.
(54) Zhang, L.; Han, J.; Wang, H.; Car, R.; E, W. Deep Potential Molecular Dynamics: A Scalable Model with the Accuracy of Quantum Mechanics. *Phys. Rev. Lett.* **2018**, *120* (14), 143001. https://doi.org/10.1103/PhysRevLett.120.143001.
(55) Çağin, T.; Pettitt, B. M. Molecular Dynamics with a Variable Number of Molecules. *Molecular Physics* **1991**, *72* (1), 169–175. https://doi.org/10.1080/00268979100100111.
(56) Çagin, T.; Pettitt, B. M. Grand Molecular Dynamics: A Method for Open Systems. *Molecular Simulation* **1991**, *6* (1–3), 5–26. https://doi.org/10.1080/08927029108022137.
(57) Lo, C.; Palmer, B. Alternative Hamiltonian for Molecular Dynamics Simulations in the Grand Canonical Ensemble. *The Journal of Chemical Physics* **1995**, *102* (2), 925–931. https://doi.org/10.1063/1.469159.

(58) Goeminne, R.; Vanduyfhuys, L.; Van Speybroeck, V.; Verstraelen, T. DFT-Quality Adsorption Simulations in Metal–Organic Frameworks Enabled by Machine Learning Potentials. *J. Chem. Theory Comput.* **2023**, *19* (18), 6313–6325. https://doi.org/10.1021/acs.jctc.3c00495.
(59) Liu, S.; Dupuis, R.; Fan, D.; Benzaria, S.; Bonneau, M.; Bhatt, P.; Eddaoudi, M.; Maurin, G. Machine Learning Potential for Modelling H2 Adsorption/Diffusion in MOFs with Open Metal Sites. *Chem. Sci.* **2024**. https://doi.org/10.1039/D3SC05612K.
(60) Tayfuroglu, O.; Zorlu, Y.; Kocak, A. Modeling Gas Adsorption and Mechanistic Insights into Flexibility in Isoreticular Metal–Organic Frameworks Using High-Dimensional Neural Network Potentials. ChemRxiv November 14, 2023. https://doi.org/10.26434/chemrxiv-2023-rk6sp.
(61) Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; Kudlur, M.; Levenberg, J.; Monga, R.; Moore, S.; Murray, D. G.; Steiner, B.; Tucker, P.; Vasudevan, V.; Warden, P.; Wicke, M.; Yu, Y.; Zheng, X. TensorFlow: A System for Large-Scale Machine Learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*; 2016; pp 265–283.
(62) Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv December 3, 2019. https://doi.org/10.48550/arXiv.1912.01703.
(63) Yang, C.-T.; Pandey, I.; Trinh, D.; Chen, C.-C.; D. Howe, J.; Lin, L.-C. Deep Learning Neural Network Potential for Simulating Gaseous Adsorption in Metal–Organic Frameworks. *Materials Advances* **2022**, *3* (13), 5299–5303. https://doi.org/10.1039/D1MA01152A.
(64) Musaelian, A.; Batzner, S.; Johansson, A.; Sun, L.; Owen, C. J.; Kornbluth, M.; Kozinsky, B. Learning Local Equivariant Representations for Large-Scale Atomistic Dynamics. *Nat Commun* **2023**, *14* (1), 579. https://doi.org/10.1038/s41467-023-36329-y.
(65) Kozinsky, B.; Musaelian, A.; Johansson, A.; Batzner, S. Scaling the Leading Accuracy of Deep Equivariant Models to Biomolecular Simulations of Realistic Size. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*; SC '23; Association for Computing Machinery: New York, NY, USA, 2023; pp 1–12. https://doi.org/10.1145/3581784.3627041.
(66) Achar, S. K.; Wardzala, J. J.; Bernasconi, L.; Zhang, L.; Johnson, J. K. Combined Deep Learning and Classical Potential Approach for Modeling Diffusion in UiO-66. *J. Chem. Theory Comput.* **2022**, *18* (6), 3593–3606. https://doi.org/10.1021/acs.jctc.2c00010.
(67) Esselink, K.; Loyens, L. D. J. C.; Smit, B. Parallel Monte Carlo Simulations. *Phys. Rev. E* **1995**, *51* (2), 1560–1568. https://doi.org/10.1103/PhysRevE.51.1560.
(68) Rappe, A. K.; Casewit, C. J.; Colwell, K. S.; Goddard, W. A.; Skiff, W. M. UFF, a Full Periodic Table Force Field for Molecular Mechanics and Molecular Dynamics Simulations. *J. Am. Chem. Soc.* **1992**, *114* (25), 10024–10035. https://doi.org/10.1021/ja00051a040.
(69) Lin, L.-C.; Lee, K.; Gagliardi, L.; Neaton, J. B.; Smit, B. Force-Field Development from Electronic Structure Calculations with Periodic Boundary Conditions: Applications to Gaseous Adsorption and Transport in Metal–Organic Frameworks. *J. Chem. Theory Comput.* **2014**, *10* (4), 1477–1488. https://doi.org/10.1021/ct500094w.
(70) Batzner, S.; Musaelian, A.; Sun, L.; Geiger, M.; Mailoa, J. P.; Kornbluth, M.; Molinari, N.; Smidt, T. E.; Kozinsky, B. E(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials. *Nat Commun* **2022**, *13* (1), 2453. https://doi.org/10.1038/s41467-022-29939-5.

(71) Mason, J. A.; Sumida, K.; Herm, Z. R.; Krishna, R.; Long, J. R. Evaluating Metal–Organic Frameworks for Post-Combustion Carbon Dioxide Capture via Temperature Swing Adsorption. *Energy Environ. Sci.* **2011**, *4* (8), 3030–3040. https://doi.org/10.1039/C1EE01720A.
(72) Errington, J. R. Direct Calculation of Liquid–Vapor Phase Equilibria from Transition Matrix Monte Carlo Simulation. *J. Chem. Phys.* **2003**, *118* (22), 9915–9925. https://doi.org/10.1063/1.1572463.
(73) Siderius, D. W.; Shen, V. K. Use of the Grand Canonical Transition-Matrix Monte Carlo Method to Model Gas Adsorption in Porous Materials. *J. Phys. Chem. C* **2013**, *117* (11), 5861–5872. https://doi.org/10.1021/jp400480q.
(74) Shen, V. K.; Errington, J. R. Determination of Fluid-Phase Behavior Using Transition-Matrix Monte Carlo: Binary Lennard-Jones Mixtures. *The Journal of Chemical Physics* **2005**, *122* (6), 064508. https://doi.org/10.1063/1.1844372.
(75) Colón, Y. J.; Gómez-Gualdrón, D. A.; Snurr, R. Q. Topologically Guided, Automated Construction of Metal–Organic Frameworks and Their Evaluation for Energy-Related Applications. *Crystal Growth & Design* **2017**, *17* (11), 5801–5810. https://doi.org/10.1021/acs.cgd.7b00848.
(76) Li, Z.; Turner, J.; Snurr, R. Q. Computational Investigation of Hysteresis and Phase Equilibria of N-Alkanes in a Metal-Organic Framework with Both Micropores and Mesopores. *Commun Chem* **2023**, *6* (1), 1–12. https://doi.org/10.1038/s42004-023-00889-3.
(77) Hatch, H. W.; Siderius, D. W.; Errington, J. R.; Shen, V. K. Efficiency Comparison of Single- and Multiple-Macrostate Grand Canonical Ensemble Transition-Matrix Monte Carlo Simulations. *J. Phys. Chem. B* **2023**, *127* (13), 3041–3051. https://doi.org/10.1021/acs.jpcb.3c00613.
(78) Idrees, K. B.; Li, Z.; Xie, H.; Kirlikovali, K. O.; Kazem-Rostami, M.; Wang, X.; Wang, X.; Tai, T.-Y.; Islamoglu, T.; Stoddart, J. F.; Snurr, R. Q.; Farha, O. K. Separation of Aromatic Hydrocarbons in Porous Materials. *J. Am. Chem. Soc.* **2022**, *144* (27), 12212–12218. https://doi.org/10.1021/jacs.2c03114.
(79) Chung, Y. G.; Haldoupis, E.; Bucior, B. J.; Haranczyk, M.; Lee, S.; Zhang, H.; Vogiatzis, K. D.; Milisavljevic, M.; Ling, S.; Camp, J. S.; Slater, B.; Siepmann, J. I.; Sholl, D. S.; Snurr, R. Q. Advances, Updates, and Analytics for the Computation-Ready, Experimental Metal–Organic Framework Database: CoRE MOF 2019. *J. Chem. Eng. Data* **2019**, *64* (12), 5985–5998. https://doi.org/10.1021/acs.jced.9b00835.
(80) Center for Computational Resaerch, University at Buffalo.

TOC


Free Energy
High Throughput
GCMC
gRASPA
Phase Change
ML Potential
Mixture
CH4
CO2