Modelling Gas Adsorption inside Metal-Organic Frameworks using a Grand-Canonical Monte Carlo Algorithm

By
Arthur York

A THESIS

submitted to

Oregon State University

Honors College

in partial fulfillment of
the requirements for the
degree of

Honors Baccalaureate of Science in Computer Science
(Honors Scholar)

Presented May 21, 2021
Commencement June 2021

AN ABSTRACT OF THE THESIS OF

Arthur York for the degree of Honors Baccalaureate of Science in Computer Science presented on May 21, 2021. Title:
Modelling Gas Adsorption inside Metal-Organic Frameworks using a Grand-Canonical Monte Carlo Algorithm

Abstract approved:

Cory Simon

Metal-organic frameworks are promising novel materials for gas storage and separations because of their extremely high internal surface area and their modular structures based on metal nodes and organic linker molecules. Due to their modular design, it is possible to finely tune a structure for optimal storage of a specific gas but it is costly and time consuming to synthesize thousands of materials for experimental testing. Therefore it is beneficial to screen potential structures using high throughput computing and then create only the most promising materials for more thorough experimentation. The goal of this project is to create and test a software package - `PorousMaterials.jl` - for simulation of gas adsorption in metal-organic frameworks and other nano-porous materials. As the scope of research became broader, so did the functionality of `PorousMaterials.jl`. First, it was extended to allow for tracking and visualization of adsorbates within the crystal structure. Second, options for inferring and examining bonds were implemented. This culminated in a free and open source software package that provides a wide range of analysis for metal-organic frameworks and other nano-porous materials.

Key Words: Metal-Organic Frameworks, Monte Carlo Algorithm, Computational Simulation, Theoretical Chemistry, Applied Mathematics

Corresponding e-mail address: yorkar@oregonstate.edu

Modelling Gas Adsorption inside Metal-Organic Frameworks using a Grand-Canonical Monte Carlo Algorithm

By
Arthur York

A THESIS

submitted to

Oregon State University

Honors College

in partial fulfillment of
the requirements for the
degree of

Honors Baccalaureate of Science in Computer Science
(Honors Scholar)

Presented May 21, 2021
Commencement June 2021

Honors Baccalaureate of Science in Computer Science project of Arthur York presented on May 21, 2021.

APPROVED:

Cory Simon, Mentor, representing College of Biological, Chemical, and Environmental Engineering

David Roundy, Committee Member, representing Department of Physics

Kyriakos Stylianou, Committee Member, representing Department of Chemistry

Toni Doolen, Dean, Oregon State University Honors College

I understand that my project will become part of the permanent collection of Oregon State University Honors College. My signature below authorizes release of my project to any reader upon request.

Arthur York, Author

Contents

1 Introduction

Gasoline is extremely potent as a fuel source due to its high energy density, but its greenhouse gas emissions are cause for concern [1]. Natural gas is a prime alternative for gasoline in vehicle use during the transition to renewable energy sources because of lower emissions, large supply, and the ease with which existing infrastructure can be adapted to transport and use it [2]. However, storage of natural gas is difficult because it requires extreme conditions to be stored as either liquefied natural gas (around -162°C) or compressed natural gas (around 200 bar) [3]. To make natural gas vehicles viable, there needs to be a more efficient method for storing it for use in vehicles. Metal-organic frameworks are of interest here because they have been shown to increase storage of usable fuel at a much lower pressure of 65 bar [4].

Metal-organic frameworks (MOFs) are promising novel materials for gas separation and storage that are modular by design [5]. They are comprised of metal nodes and organic linkers that self-assemble into organized crystal structures with nano-pores when the component parts are combined in solution. The high internal surface areas formed by these nano-pores provide ideal adsorption sites for a variety of gaseous atoms and molecules. They can be used to improve gas storage without requiring extreme temperature and pressures, a useful tool for implementing in natural gas vehicles [4]. MOFs can also be used for simpler gas separation as is the case with CaSDB being more selective for Xe over Kr in used nuclear fuel [6, 7]. The linkers and nodes can be selected to curate the MOF for higher selectivity of a gas molecule, increasing the storage or separation capabilities of the MOF. This modularity means that there are thousands of potential MOFs that can be synthesized and tested, so traditional synthesis and evaluation is not viable for testing each MOF.

Computational simulation and modelling is a powerful tool for predicting the behaviour of complex systems [8]. It provides a means for rapidly testing thousands of candidates faster, cheaper, and more easily than conducting real world experiments. Computational modelling is highly applicable to a wide range of fields because it allows researchers to explore novel concepts that are difficult to tackle in traditional laboratory settings. Models can be developed for sociology, epidemiology, or chemistry (to name a few) and establish a strong basis for future research [9–11]. The scalability of these simulations in high throughput computation is an especially powerful factor because it removes the need to perform tedious and repetitive tasks in a wet lab while still providing important information [12]. Computational simulation is a perfect tool for examining large quantities of novel metal-organic frameworks because they can be processed more efficiently.

In addition to providing estimates for gas storage, computational modelling and analysis of these MOFs can allow researchers to better understand how gasses are adsorbing in the structure. These simulations track the positions of the adsorbates throughout the simulation, so it is possible to identify likely sites of adsorption [13]. Analysis of these structures *in silico* can also lead to a better understanding of how nano-porous materials are organized [14]. Computational analysis of these materials creates opportunities for research that experimental methods are incapable of at this time.

Creating a community dedicated to studying these materials using computation can enhance and inspire experimental work and drive the field forward. A free and open source software package capable of modelling gas adsorption in metal-organic frameworks and other nano-porous materials would prove useful for screening large pools of potential candidates and can narrow down options to the most promising materials. With thousands of synthesized and even more hypothetical MOFs, narrowing down the choices is a daunting yet important task [12]. Many existing software packages capable of running these simulations are proprietary, so they limit those able to study MOFs *in silico* to those with the funds to afford the licenses, and the methods used are hidden from researchers. A free and open source (FOSS) package built for a FOSS language opens the door to more researchers and improves the community of those studying these materials [15]. Having an open source package builds validity in the model because end users are able to understand the inner mechanisms and provide input on how to improve it.

This thesis explores how computational analysis of MOFs using `PorousMaterials.jl` can yield invaluable data for experimental researchers and propel the field forward [16]. It starts by examining the Buffon's needle problem because of its many structural similarities to this method of molecular simulation. Then there is a discussion of how the model is constructed and verified. The modelling algorithm is validated by producing simulated adsorption isotherms and comparing them to experimental data. The simulation is further expanded to help locate adsorption sites within MOFs. `PorousMaterials.jl` is lastly used for analyzing crystal structures by inferring and then analyzing their bonds to locate disjoint structures.

2 Buffon's Needle

2.1 The Buffon's Needle Problem

Buffon's needle is a probability problem originally discussed in the 18th century. The problem revolves around needles of a given length and a floor marked with infinite parallel, equidistant lines. Figure 1 shows an example of the floor in this problem. The goal is to determine the probability that a dropped needle will intersect one of those lines. This is a perfect problem to introduce computational modelling because there is an analytical solution that can be used to verify the model.

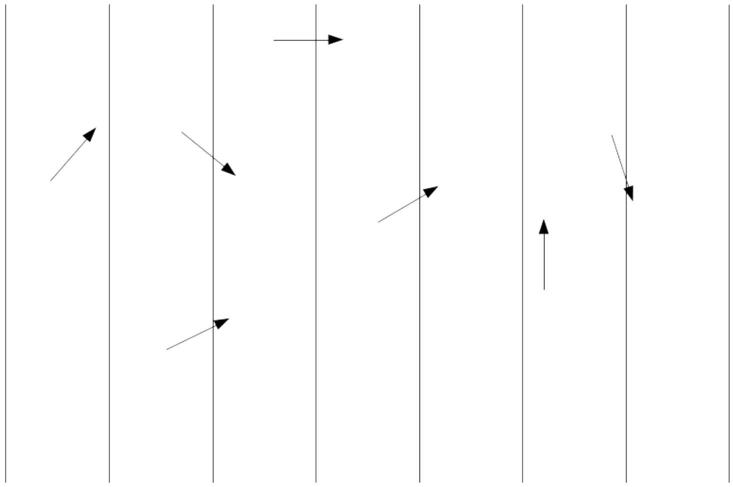

Figure 1: This is the floor for the Buffon's Needle problem. The vertical lines shown are the lines on the floor that have infinite vertical length. The arrows are examples of needles that have fallen on the floor. Some of the needles overlap the lines and some of them do not.

For the sake of simplicity, two assumptions are being made for the work done for this problem. The first assumption is that the length of the needle is less than the distance between the lines on the floor, eliminating the possibility that a needle could intersect two or more lines on the floor. The second assumption is that the location needles land on the floor, and the orientation of the needle as it falls are both uniformly distributed.

2.2 Applications to Computer Simulation

The simulation method for Buffon's needle is not required because the analytical approach simplifies to a solvable equation. However, this is a useful tool for showing how a simulated model can be developed when there is no exact solution present. This problem is a perfect primer for the simulations done with MOFs in `PorousMaterials.jl` because it has many parallels. The concept of simplifying the main problem through periodic boundaries is an important aspect of `PorousMaterials.jl` . MOFs are infinitely repeating crystalline structures, so finding a subsection to examine that will be representative of the whole structure is valuable. This problem also introduces the idea of exploring a model through examining the state space. Buffon's needle is simple because the state space is uniformly distributed, `PorousMaterials.jl` also uses an exploration of state space but more complicated methods must be used because not all state spaces occur with equal probability. Using a toy model such as Buffon's needle provides an opportunity to introduce more complex topics that are used extensively in `PorousMaterials.jl` .

2.3 Representation of the Needle and the Floor

For the analytical and the simulation approach it is important to define a representation for needles and the floor. The position and orientation of each needle was defined by the Cartesian coordinates of the center of the needle and the angle ($0 \to 2\pi$ in radians) between a vertical line and the needle. The floor will be represented as the distance between the parallel vertical lines. Every line is infinitely long in the y direction and they repeat *ad infinitum* in the x direction.

To make both models easier to work with, the representations can be simplified. Due to the rotational symmetry of a needle, it is not necessary to use the full range $0 \to 2\pi$ to represent the orientation of the needle. The needle rotates about its center, so an orientation of 0 is the same as an orientation of π. This means the range of angles can be restricted to be from $0 \to \pi$.

It is unwieldy to drop needles on an infinite plane, so the axes need to be bounded. The y coordinate of a needle has no influence over whether it intersects a line. With everything else held constant, a change in the y value of a needle will not change the outcome of the drop.

The lines repeat in the x direction and are evenly spaced a distance d apart. The space between two lines can be defined as a chunk, and each chunk is identical. Therefore, periodic boundaries may be applied because finding the probability that a needle intersects a line in a single chunk will be representative of the probability that a needle intersects a line when dropped anywhere on the floor. This restricts the possible x coordinates for a needle to be in $0 \leq x \leq d$.

With this information, the state space for needles can be constructed. The x-value exists from $0 \to d$ and the θ value exists from $0 \to \pi$. Equation 1 shows the state space for the problem.

$$R := \{(x, \theta) | 0 \leq x \leq d, 0 \leq \theta \leq \pi\} \tag{1}$$

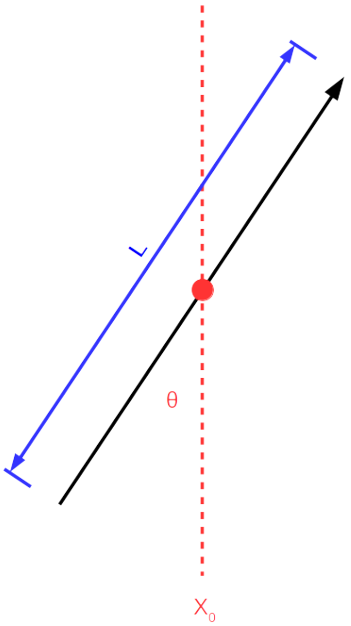

Figure 2: An needle with its parameters for length L, horizontal location x_0, and angle from vertical θ

2.4 Analytical Solution to Buffon's Needle

One of the benefits of this problem is that there is an exact analytical solution that can verify the simulation model. This analytical solution can be found by working with areas inside the state space that was defined for the needle in equation 1.

Dropping a needle on the floor is the same as uniformly sampling the state space because all positions and orientations of the needle are equally likely when it is dropped. Thus the fraction of the space that represents an intersection is the probability that a randomly dropped needle intersects a line. This can be calculated because the size of the state space is finite.

In one chunk, the needle can either intersect the line that is at $x = 0$ (left line) or the line at $x = d$ (right line). All needles have a center between 0 and d, so if an endpoint of the needle is less than 0 or greater than d then it must cross a line. The needle crosses the left line when $x - \frac{l}{2}sin(\theta) < 0$, and it crosses the right line when $x + \frac{l}{2}sin(\theta) > d$. Where the length of the needle l and the distance between the lines d is held constant.

These equations can be solved analytically to find the total area inside the state space that they satisfy. they can be re-arranged to be in terms of θ to make integration simpler. The equation for the left line becomes $x < \frac{l}{2}sin(\theta)$. Equation 2 goes through solving for the area under the lower curve.

$$\begin{aligned} A_{lower} &= \int_0^{\pi} (\frac{l}{2} sin(\theta)) d\theta \\ A_{lower} &= l \end{aligned} \tag{2}$$

The same process is done for the right line to find the area above the upper curve. The inequality for determining if a needle intersects the right line is rewritten $x > d - \frac{l}{2} sin(\theta)$. This is solved in equation 3.

$$\begin{aligned} A_{upper} &= \int_0^{\pi} (d - \frac{l}{2} sin(\theta)) d\theta \\ A_{upper} &= l \end{aligned} \tag{3}$$

The last step is to find the total area of the state space. This is done by taking a double integral over both axes. Equation 4 shows the calculation of the total area of the state space.

$$\begin{aligned} A_{total} &= \int_0^{d} \int_0^{\pi} d\theta dx \\ A_{total} &= d\pi \end{aligned} \tag{4}$$

The probability that a needle intersects a line when it is randomly dropped is the fraction of the space that represents an intersection. The probability that a randomly dropped needle intersects a line is $P_{intersection} = (A_{lower} + A_{upper})/A_{total} = \frac{2l}{d\pi}$.

2.5 Simulation Approach to Buffon's Needle

The simulation approach is to take uniformly distributed samples from the state space, and keep track of how many samples would intersect a line. A given sample intersects a line if it satisfies either $x + \frac{l}{2} sin(\theta) > d$ or $x - \frac{l}{2} sin(\theta) < 0$. Because the samples are uniform, they are representative of the entire state space as a whole. This model was written in Julia to maintain consistency with `PorousMaterials.jl` [17].

The code shown in code fragment 1 is the struct used to represent a single needle in the simulation. For each cycle of the simulation a needle is "thrown" onto the floor by sampling uniformly from the state space. It gets a random number between 0 and π for θ and a random number between 0 and d for x. The value for l is set by the specifications for the particular simulation.

```
struct Needle
    l::Float64
    x::Float64
    θ::Float64
end
```

Code Fragment 1: The struct used for representing a needle in the `julia` program

2.6 Testing the Validity of the Model

To compare these methods, the state spaces for these methods can be plotted side by side. Figure 3 shows the area in the state spaces that result in the needle intersecting a line. The blue in each plot highlights the region in the state space that results in a needle intersecting a line.

The analytical model provides the exact solution. If the results of the simulation match the analytical model then the simulation is accurate and provides relevant data. The results of the two models estimating needles of length 10 and a distance between lines of 30 are shown in figure 3. The blue shaded areas of both plots cover the same regions, so the simulation model is a fairly strong approximation for the analytical approach.

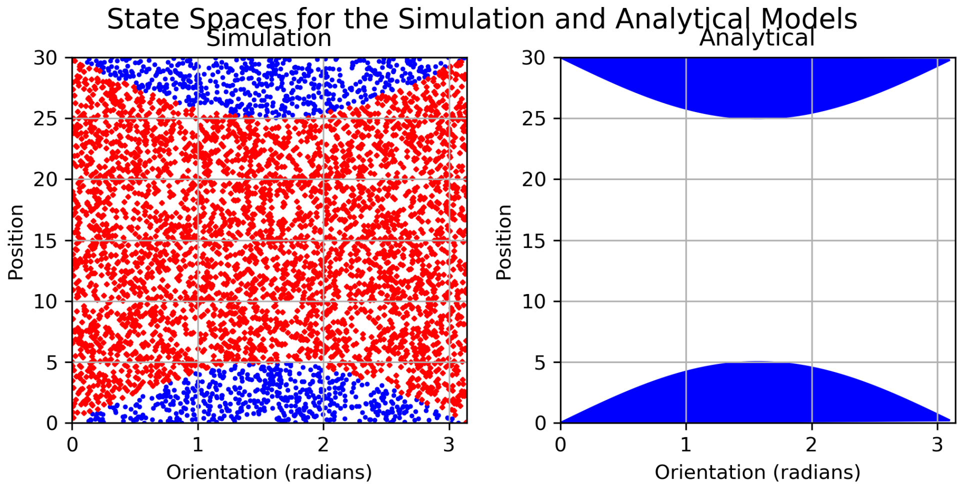

Figure 3: Comparison of the simulation and analytical approaches. The blue area in each graph represents a state that results in the needle intersecting a line. Red dots in the simulation plot represent samples that did not intersect a line. These are generated from a run where the distance between the lines is 30 and the length of the needle is 10.

One way of showing that the simulation is accurate is to compare directly with the exact solution. The number of throws in a simulation can be compared to the mean of the results to find how accurate the model is. Figure 4 shows the mean (as a point) and standard deviation (as error bars) as the number of throws increases. Once the number of throws reaches 10^5 the error bars are no longer visible. The simulated data from figure 3 was collected with 5,000 samples, where the standard deviation is around 0.02.

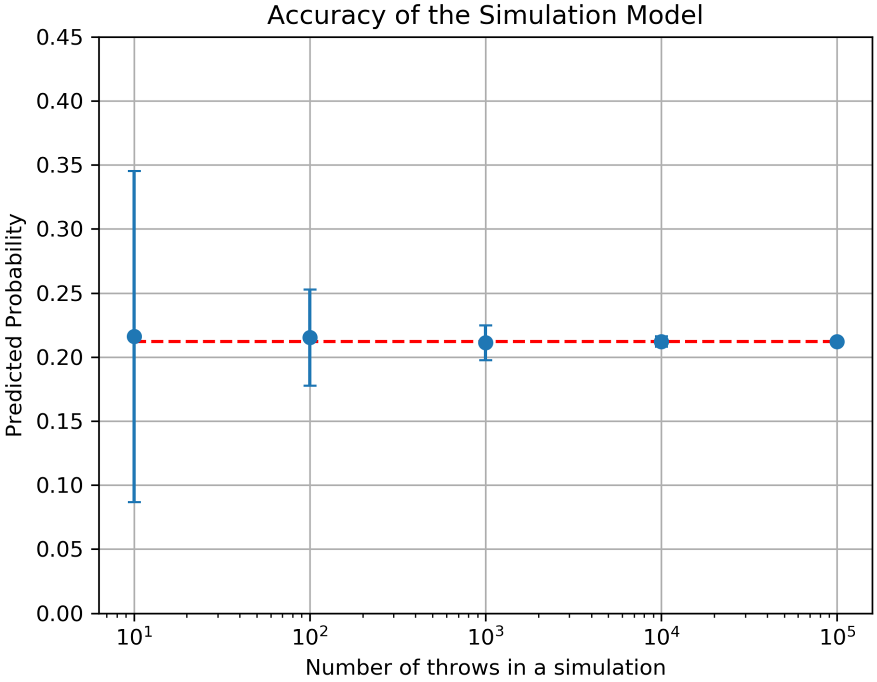

Figure 4: Mean and standard deviation over a varying number of throws. Each number of throws was tested with 100 simulations. The mean of the data is shown as the point and the standard deviation is represented with the error bars.

3 Modelling Gas Adsorption with a Grand Canonical Monte-Carlo Algorithm

3.1 Grand-Canonical Monte Carlo Simulation

`PorousMaterials.jl` uses a Metropolis Hastings algorithm for molecular simulations [16]. This algorithm is a more complex version of the simple Monte Carlo algorithm used in the Buffon's Needle problem discussed previously. By exploring the statespace of these problems the average statespace can show an approximate solution. A major difference between the Buffon's Needle problem and molecular simulation is the probability distribution of each state. Each state in the Buffon's Needle problem has an equal probability of occurring. This is not true of gasses adsorbing to porous structures.

The statespace for this molecular model is defined as the total number of adsorbate molecules in the system and their positions in 3D space, shown in equation 5. The order of the positions is not important because it has no impact on what is being represented. These states should not occur with equal probability in order to accurately simulate this process. For instance, one state with a pair of overlapping atoms and another state with the overlapping atom removed should not occur with the same probability. To favor realistic permutations, `PorousMaterials.jl` uses the the grand-canonical, μVT, ensemble of equations with a metropolis hastings algorithm to run grand-canonical Monte Carlo (GCMC) simulations. These explore feasible states based on the energy of the system and avoid improbable, energetically unfavorable, states.

$$R := \{N, \{\vec{x}_1, \ldots, \vec{x}_N\}\} \tag{5}$$

It is important to note that the GCMC simulation explores the statespace of the model to simulate it, and that cycles are not related through time. Given the state of the system after cycle x, cycle $x + 1$ does not represent a snapshot of the system after 1 unit of time. Cycle $x + 1$ represents another state that can be attainable through insertion, deletion, or translation of a single adsorbate, or no change from cycle x. Going from cycle x to $x + 1$ does not indicate a change in time in the system.

The simulation works by proposing one of three trial moves: insertion, deletion, or translation. The GCMC simulation uses acceptance rules to determine whether or not the simulation accepts or rejects this trial move based on the change in energy. The simulation will accept trial moves with a probability proportional to the increase in energy. Therefore large changes in energy will never be accepted, small changes have a chance of being accepted, and decreases in energy will always be accepted. The trial moves allow the model to explore the entire statespace, but the acceptance rules limit the spaces it can actually visit. At the end of the simulation, information about the visited states such as number of adsorbates and total energy can be averaged to calculate an approximate solution to the system.

3.2 Deriving the Acceptance Rules for GCMC

The grand-canonical Monte Carlo acceptance rules are derived from a Metropolis Hastings Algorithm and the partition function for the grand-canonical statistical mechanical μVT ensemble [18, 19]. The Metropolis Hastings algorithm defines the probability of going from the current state c to a new state n as:

$$acc(n|c) = min\left(\frac{\alpha(c|n)p(n)}{\alpha(n|c)p(c)}, 1\right) \tag{6}$$

Where $p(x)$ is a probability mass function for all states in R and $\alpha(t|s)$ defines the probability of proposing transition state t given the current state is s.

The partition function from the μVT ensemble can be used to define a calculable value that is proportional to the probability density of a given state as shown in equation 7. There is no function that describes the exact probability of each state, so a value proportional to their probability is the most accurate method for informing transitions between states. The simulation holds temperature T, volume V, and pressure P constant, while the number of adsorbates in the system N and their positions $(\vec{x}_1, \dots, \vec{x}_N)$ are allowed to fluctuate through the acceptance of GCMC proposals. The potential energy at a given state U will be calculated as a function of the positions of the adsorbates via their interactions with each other and the atoms in the host framework. The chemical potential μ is related to the pressure. The De Broglie wavelength, Λ, is present in the partition function but will be substituted later to cancel terms. $\beta = 1/(k_B T)$ where k_B is the Boltzmann constant.

$$p(N, \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N\}) \propto \frac{\Lambda^{-3N}}{N!} e^{\beta\mu N} e^{-\beta U(\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N)} \tag{7}$$

The particles within the system are indistinguishable because this simulation is designed to only handle one type of adsorbate at a time. The $N!$ term is required in equation 7 to prevent double counting permutations of the same set of particles. However, when calculating the guest-guest and guest-host energy of each asorbate, $U(\vec{x}_1, \dots, \vec{x}_N)$, the order of the particles is not taken into consideration. Therefore, the states are not counted multiple times, and this term can be removed to simplify this equation. This creates a new equation that is still proportional to equation 7 for use in the Metropolis Hastings Algorithm, but is easier to work with.

$$p(N, \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N\}) \propto \Lambda^{-3N} e^{\beta\mu N} e^{-\beta U(\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N)} \tag{8}$$

For simplifying the equation so as not to need the De Broglie wavelength, the following equivalence may be used when working with an ideal gas [18]:

$$\Lambda^3 = \frac{e^{\beta\mu}}{\beta P} \tag{9}$$

To determine the acceptance rule for each Monte-Carlo proposal, every component of equation 6 must be found for the state space transition from inserting, deleting, or translating an adsorbate molecule.

3.2.1 Insertion

Insertion into the system modifies the state space through the transition shown in equation 10. For the sake of reducing clutter in the equations below, these will be simplified to (N, x^N) for the initial state and $(N+1, x^{N+1})$ for the proposed state.

$$\{N, \{\vec{x}_1, \ldots, \vec{x}_N\}\} \rightarrow \{N+1, \{\vec{x}_1, \ldots, \vec{x}_N, \vec{x}_{N+1}\}\} \tag{10}$$

When choosing to insert a molecule, a random location will be uniformly chosen from within the simulation box $(1/V)$. The reverse will also be needed for finishing equation 6. When reversing this insertion, a random molecule will be uniformly chosen from all adsorbates currently in the system $(1/N+1)$. These combine to obtain:

$$\frac{\alpha((N, x^N)|(N+1, x^{N+1}))}{\alpha((N+1, x^{N+1})|(N, x^N))} = \frac{V}{N+1} \tag{11}$$

Equation 8 can then be inserted into $p(n)/p(c)$ for:

$$\frac{p(N+1, x^{N+1})}{p(N, x^N)} = \Lambda^{-3} e^{\beta\mu} e^{-\beta(U(x^{N+1}) - U(x^N))} \tag{12}$$

Equation 9 can further simplify this into:

$$\frac{p(N+1, x^{N+1})}{p(N, x^N)} = \beta P e^{-\beta(U(x^{N+1}) - U(x^N))} \tag{13}$$

Lastly, equations 13 and 11 can be used within equation 6 to achieve the probability of acceptance for an insertion proposal.

$$acc((N+1, x^{N+1})|(N, x^N)) = min\left(\frac{VP}{k_B T(N+1)} e^{-\beta(U(x^{N+1}) - U(x^N))}, 1\right) \tag{14}$$

3.2.2 Deletion

Deletion of a randomly chosen k^{th} molecule modifies the state space through the transition shown in equation 15. The current state will be represented by the same notation in insertion, and $(N-1, x^{N-1})$ will represent the proposed state with a randomly selected molecule deleted.

$$\{N, \{\vec{x}_1, \ldots, \vec{x}_{k-1}, \vec{x}_k, \vec{x}_{k+1}, \ldots, \vec{x}_N\}\} \rightarrow \{N-1, \{\vec{x}_1, \ldots, \vec{x}_{k-1}, \vec{x}_{k+1}, \ldots, \vec{x}_N\}\} \tag{15}$$

The proposed moves for deletion will be the opposite of insertion. When deleting a molecule, a random molecule will be uniformly chosen from the list of all adsorbates to remove $(1/N)$. The reverse of this action involves generating a random point uniformly chosen from within the simulation box $(1/V)$. These are the opposites of the rules for insertion, so it naturally follows that this will be the inverse of equation 11.

$$\frac{\alpha((N, x^N)|(N-1, x^{N-1}))}{\alpha((N-1, x^{N-1})|(N, x^N))} = \frac{N}{V} \tag{16}$$

Deletion will follow the same process for simplifying this equation as insertion, first creating $p(n)/p(c)$ and then substituting in equation 9. This achieves:

$$\frac{p(N-1, x^{N-1})}{p(N, x^N)} = \frac{1}{\beta P} e^{-\beta(U(x^{N-1})-U(x^N))} \tag{17}$$

Equations 16 and 17 can be substituted into equation 6 to get the acceptance rule for a deletion proposal.

$$acc((N-1, x^{N-1})|(N, x^N)) = min\left(\frac{k_B T N}{VP} e^{-\beta(U(x^{N-1})-U(x^N))}, 1\right) \tag{18}$$

3.2.3 Translation

Translation of a randomly chosen k^{th} molecule modifies the state space through the transition shown in equation 19. The notation from insertion and deletion will be upheld for the current state, and the new state will be shortened to (N', x'^N).

$$\{N, \{\vec{x}_1, \ldots, \vec{x}_k, \ldots, \vec{x}_N\}\} \rightarrow \{N, \{\vec{x}_1, \ldots, \vec{x}'_k, \ldots, \vec{x}_N\}\} \tag{19}$$

With translation, a random molecule will be chosen from the list of current adsorbates and then translated to a nearby position. When moving to a nearby position, V' can signify the volume around its current position from which a uniformly chosen point will be selected. Therefore, the probability of proposing a given translation from some state (N, x^X) is $1/(N * V')$. This same logic can be followed for reversing the translation to yield the same probability for reversing the action. This generates:

$$\frac{\alpha((N, x^N)|(N', x'^N)}{\alpha((N', x'^N)|(N, x^N))} = 1 \tag{20}$$

Because the number of adsorbates within the system stays constant between the current and proposed states, $p(n)/p(c)$ simplifies further without needing to substitute a value for the De Broglie wavelength.

$$\frac{p(N', x'^N)}{p(N, x^N)} = e^{-\beta(U(x'^N)-U(x^N))} \tag{21}$$

Equations 20 and 21 can be substituted into equation 6 to achieve the acceptance rule for translation proposals.

$$acc((N', x'^N)|(N, x^N)) = min\left(e^{-\beta(U(x'^N)-U(x^N))}, 1\right) \tag{22}$$

3.3 Grand-Canonical Monte Carlo implemented in `PorousMaterials.jl`

`PorousMaterials.jl` uses a grand-canonical Monte Carlo simulation to approximate the storage potential of adsorbate-MOF pairs. The simulation can be controlled through a variety of parameters that allow the user to approximate the scenario they are most interested in. When using the `μVT_sim` function, there are five required arguments. An example of this is shown in code fragment 2, and a further explanation of each is provided in table 1. There are many optional keyword arguments (shown in table 2) that allow the user to tune the length of the simulation (thus increasing the accuracy) as well as control other data collection options (i.e. adsorbate snapshot functionality).

The `μVT_sim` function is a series of proposed GCMC moves that are accepted or rejected based on the change in energy they apply to the system. At every iteration, the simulation will choose to propose an insertion, deletion, or translation by randomly selecting an option based on pre-assigned weights. It then proposes a new trial state based on the selected move. Because only one molecule is being modified for each proposal, the total system energy does not need to be recalculated. Instead, only the contribution of the molecule in question is used because it provides $U(proposed) - U(current) = \Delta E$. The list below explains how trial moves are generated, while retaining information about the current state, for each type of proposal.

- **Insertion:** A new molecule is placed within the simulation box and appended to the array of molecules.
- **Deletion:** A random molecule is selected in the array for deletion, but it is left in the array to preserve information about the current state.
- **Translation:** A random molecule is selected, its position is saved, and it is translated by a small amount to a nearby location.

After a trial move is generated the change in energy is calculated and the system randomly chooses whether it will accept or reject it. The change in energy is used within equation 14, 18, or 22 based on its proposal type. The term inside of the min function will generate a positive value for the probability of accepting that move. A value greater than 1 indicates that the proposed state is more energetically favorable and will always be taken. A value close to 0 means the proposal is extremely energetically unfavorable, likely from overlapping atoms, and will always be rejected. Any where in between shows that the proposed state is less energetically favorable than the current state, so it will be accepted with a probability defined in one of the acceptance rules. Travelling to less favorable states prevents the system from becoming trapped inside local minima and allows the simulation to explore more of the state space.

```
result = μVT_sim(xtal, molecule,
  ↪  temperature, pressure, ljff)
```

Code Fragment 2: Example of the `μVT_sim` with the required arguments

These proposals are grouped into cycles to prevent oversampling the system. These proposals have low acceptance rates so very few prompt a change. Oversampling can inadvertently skew the data if many proposals pass without the state changing. Each cycle contains $max(20, N)$ proposals.

The simulation starts by executing "burn cycles" where it explores the state space before samples are collected because it starts with 0 adsorbates. This sets up the simulation by reaching an energetically favorable position before sampling for data. Due to the low acceptance rate of the proposals, it takes many cycles before the system reaches a state that is indicative of the real world. These cycles do not contain information that is valuable to the user because they are held at a constant temperature and do not represent the MOF filling with gas. If these cycles were sampled, then the simulation would underestimate the amount of gas that a MOF can store because many of its data points are from the beginning of the simulation where no gas is present.

Table 1: Rquired Arguments for `μVT_sim`

Argument Name	Type	Purpose
`xtal`	`Crystal`	The crystal structure being tested. Read in from a `*.cif` or `*.cssr` file in using the `Crystal()` constructor from `PorousMaterials.jl`
`molecule`	`Molecule`	The adsorbate being tested inside the structure. Read in using the `Molecule(...)` constructor from `PorousMaterials.jl`
`temperature`	`Float64`	The temperature of the system in *Kelvin*
`pressure`	`Float64`	The pressure of the system in *bar*
`ljff`	`LJForceField`	The parameters used for atomic interactions. Read in from a `*.csv` file using the `LJForceField() constructor in PorousMaterials.jl`

3.4 Example Usage within `julia`

Code fragment 3 shows the basics of running a GCMC simulation using `PorousMaterials.jl` in `julia`. Lines 4 - 10 are reading in necessary information from files including the framework being tested (IRMOF-1.cssr), the forcefield to use (Dreiding forcefield), and the molecule being adsorbed (methane). The universal forcefield can be used to simulate almost any structure gas pair (because it contains atomic interaction information for all elements), but in this case the Dreiding forcefield will generate a more accurate simulation because it uses more precise information for a smaller set of

```julia
using PorousMaterials

# read in the metal-organic framework
structure = Crystal("IRMOF-1.cssr")
# atom labels are read in with numbers, these interfere with the
 ↪  simulation
strip_numbers_from_atom_labels!(structure)
# read in the forcefield being used - the Dreiding forcefield
ljforcefield = LJForceField("Dreiding")
# read in the adsorbate being used - Methane
molecule = Molecule("CH4")

# run a single GCMC simulation with a pressure of 50.0 bar
result = µVT_sim(structure, molecule, 298.0, 50.0, ljforcefield;
 ↪  n_burn_cycles=2000, n_sample_cycles=7000, verbose=true,
 ↪  eos=:PengRobinson)

# calculate an adsorption isotherm by running GCMC simulations with
 ↪  pressures from 1.0 bar to 80.0 bar
pressures = [1.0, 5.0, 10.0, 20.0, 30.0, 40.0, 50.0, 60.0, 70.0,
 ↪  80.0]
results = adsorption_isotherm(structure, molecule, 298.0, pressures,
 ↪  ljforcefield; n_burn_cycles=10000, n_sample_cycles=10000,
 ↪  verbose=true, eos=:PengRobinson)
```

Code Fragment 3: Running a GCMC simulation for a single pressure, and for a range of pressures in `julia` with `PorousMaterials.jl` atoms [20,21]. A single GCMC simulation run with optional arguments is shown on line 13. This models the adsorption of methane inside IRMOF-1 using the Dreiding forcefield file provided at 298.0°K and 50.0 bar. Because the model starts empty, the simulation has been set to perform 2,000 "burn cycles". The simulation has also been set to run 7,000 "sample cycles" where it collects information about the state of the system. Lastly this simulation will use the `PengRobinson` equations of state instead of treating the gas as ideal.

Code fragment 3 also shows a function call for calculating an adsorption isotherm. It takes the same arguments (and optional arguments) as the `μVT_sim` function except it takes an array of pressures instead of a single value. This allows the user to create an adsorption isotherm across a range of pressures with a single function call. The `adsorption_isotherm` function takes advantage of `pmap` within `julia`, so these different simulations can be run in parallel on powerful enough machines.

3.4.1 Verifying the Model

The results from the `adsorption_isothem` call in code fragment 3 can be used to construct an adsorption isotherm for methane in IRMOF-1 alongside experimental data. The results from this simulation can be compared to the results from experimental data to determine the accuracy of the model.

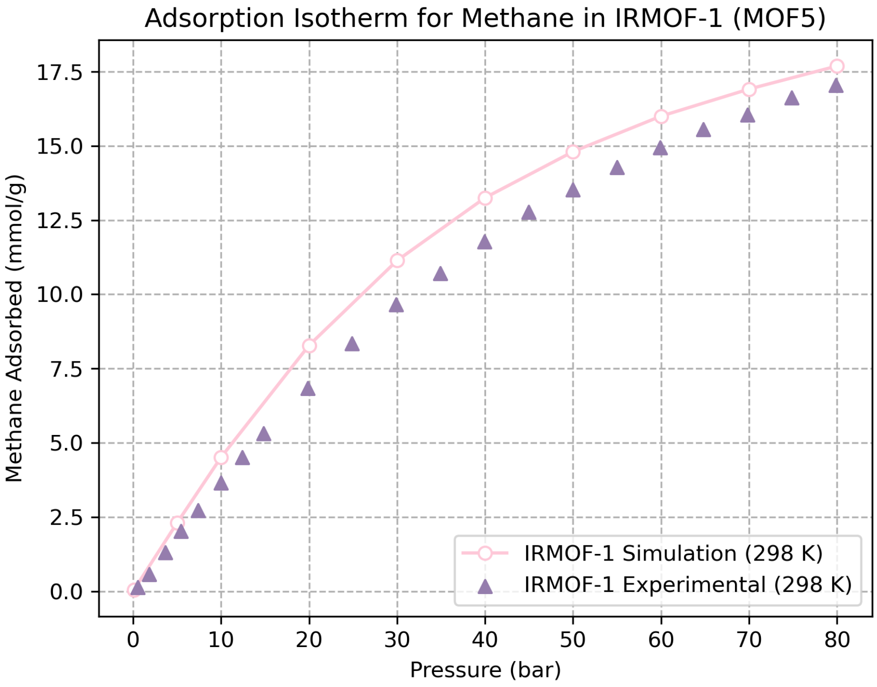

Figure 5: An adsorption isotherm for xenon in calcium sulfonyl dibenzoate with 10,000 burn cycles, 10,000 sample cycles, the Dreiding forcefield (for atomic interactions), 298.0K, and pressures ranging from 0.1 bar to 80.0 bar.

Figure 5 shows a comparison of simulation results to experimental data for methane in IRMOF-1 [22].

This shows that given the specialized Dreiding forcefield, this model can approximate the adsorption of methane in IRMOF-1. This is not a perfect match, but provides insight into how well a given MOF can perform. It can be useful for determining which MOFs are most promising for gas adsorption and should be investigated further.

This model assumes ideal and uniform conditions. For example, MOF synthesis involves mixing nodes and linkers in solution, and having the framework form in solution. This leaves solvent inside of the pores of the MOF, blocking the channels. Removing the solvent "activates" the MOF, and allows experimental tests to be run [5]. There are many methods for activation, and a MOF is not guaranteed to be entirely evacuated of solvents in its pores.

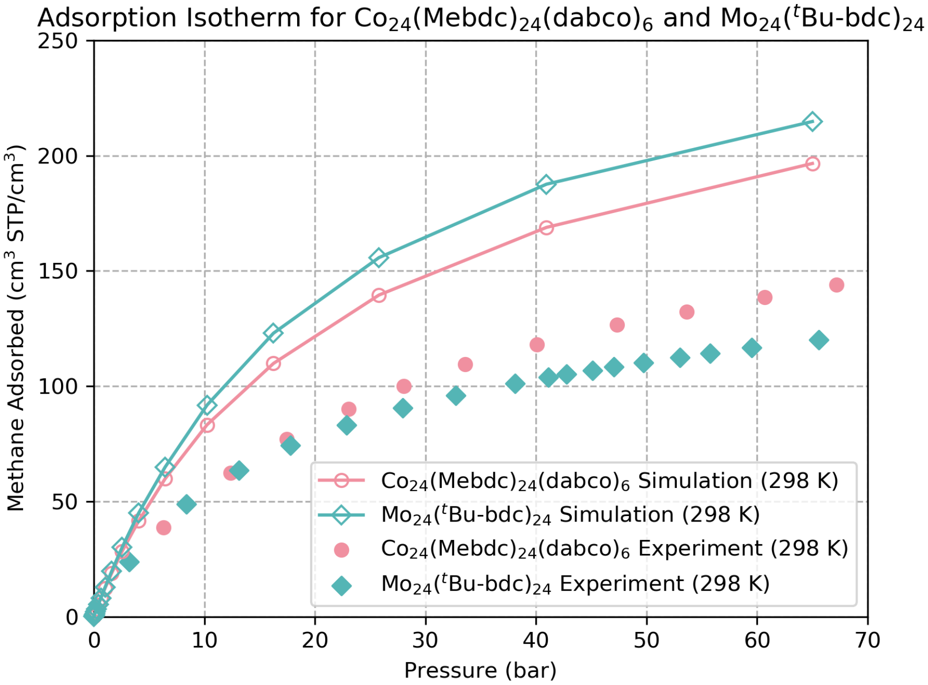

Figure 6: An adsorption isotherm highlighting the limitations of the model. The simulated uptake of methane is significantly larger than the experimental uptake.

Differences can occur between the simulation data and experimental data because `PorousMaterials.jl` assumes the porous structure has been perfectly activated and no solvent remains. Figure 6 shows an experimental and simulated adsorption isotherm that don't match due to issues relating to the activation of the porous material. The difference shown here is believed to be influence in large part by an imperfect removal of solvent, thus impacting the experimental data. The simulation data assumes a best case scenario within the system, so it will overestimate the amount of gas it can store if the material is not prepared carefully.

3.4.2 Flexible MOFs

Flexible MOFs such as Co(bdp) are potential materials for use in natural gas vehicles because not only do they increase energy storage potential over compressed and liquid natural gas systems due to their large internal surface areas, but they also release more of the material when the system goes to a low pressure, meaning more usable fuel [4]. Flexible MOFs have a higher usable capacity because they transition between a "closed" form at low pressures and an "open" form at high pressure [4]. They retain the same amount of gas, but when they pass a pressure threshold during desorption they collapse and expel all remaining gas in the system.

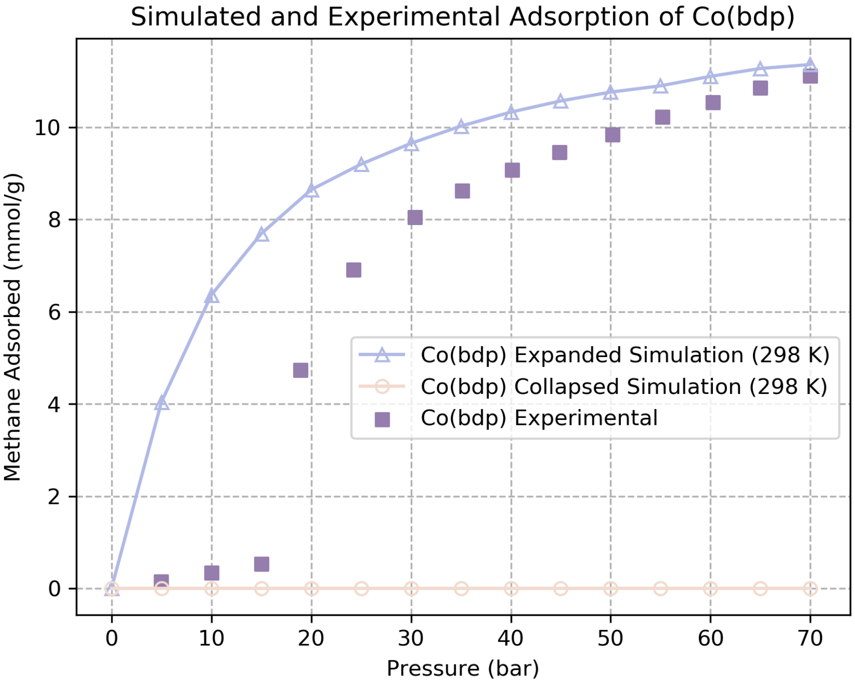

Figure 7: An adsorption isotherm for Co(bdp) showing experimental results alongside theoretical results using the collapsed and expanded crystallographic files.

`PorousMaterials.jl` assumes a rigid, non-moving structure meaning that it is not able to properly simulate flexible materials. While `PorousMaterials.jl` can approximate the "closed" and "open" forms of a flexible MOF fairly well, as seen in figure 7, it cannot simulate the transition between the two. The transition point is extremely important because it needs to happen in a reasonable pressure range that allows the user to extract most of the fuel they put in. Because `PorousMaterials.jl` cannot simulate this behaviour, it cannot make assumptions about when the transition occurs. This means it cannot serve as a guide for which materials are the most promising due to their flexible nature.

Table 2: Optional Keyword Arguments for `μVT_sim`

Argument Name	Type	default	Purpose
`molecules`	`Array`	[]	An array of molecules that can be pre-loaded into the simulation if the user does not want to start with an empty box
`n_burn_cycles`	`Int`	5000	The number of cycles that will run at the beginning of the simulation where data is not collected
`n_sample_cycles`	`Int`	5000	The number of data collection cycles that will be run after the burn cycles are completed
`sample_frequency`	`Int`	1	The number of sample cycles between each data collection (after burn cycles are completed). For example, 1 means data is collected every cycle, while 2 means every other cycle and so on.
`ewald_precision`	`Float64`	1e-6	The precision used for long range ewald summations
`eos`	`Symbol`	`:ideal`	The equations of state used within the simulation. Can either be 'ideal' or 'PengRobinson'
`write_adsorbate_snapshots`	`Bool`	false	Whether the simulation will track and output adsorbate positions to a `*.xyz` file
`snapshot_frequency`	`Int`	1	The number of sample cycles between each snapshot (after burn cycles are completed). Similar to `sample_frequency`.
`calculate_density_grid`	`Bool`	false	Whether the simulation will generate a density grid of where adsorbates are located during the simulation
`density_grid_dx`	`Float`	1.0	The size of voxels used in the density grid in *Angstroms*.
`density_grid_species`	`Symbol | nothing`	nothing	The atomic species within the adsorbate that will be tracked for the density grid. If `nothing` is chosen and there is only one unique species, the unique species will be used. Otherwise the simulation will error.
`density_grid_sim_box`	`Bool`	true	A `true` value means the density grid covers the entire simulation box after the crystal is replicated. A `false` value means the density grid only covers the original crystal passed in.
`verbose`	`Bool`	false	Whether the simulation will print out updates about the status of the simulation.
`autosave`	`Bool`	true	Whether the simulation autosaves its progress in the `gcmc_checkpoints` directory.
`result_filename_comment`	`String`	""	String that will be appended to the end of autosaved filenames.

4 Adapting the Monte Carlo Algorithm to track Adsorbate Positions

When studying gas adsorption in nano-porous materials, it is important to know where molecules are located in relation to the structure. X-ray diffraction can be used to get this information, but it has issues with non crystalline materials. For example, porous coordination cage molecules serve a similar purpose to metal-organic frameworks with respect to gas storage but their non-crystalline structure means x-ray diffraction cannot find the locations of individual molecules inside the pores. `PorousMaterials.jl` can provide an estimate because it already tracks the locations of the molecules in the pores of the crystal to run the simulation. These adsorbates were tracked using two separate methods: storing coordinates for individual atoms and tracking the number of atoms that appear within a given unit of space.

This information is extremely useful for designing optimal pore spaces. Understanding where gasses are likely to adsorb within the structure can guide research about tuning MOFs and other porous structures to increase their gas storage potential. This insight can be applied to functionalize or replace linkers in the hopes of generating stronger adsorption between the gas and the internal surface area of the structure.

4.1 Tracking Adsorbate Positions using `snapshots`

The simulation code for modelling gas adsorption can be extended to take “snapshots” of the molecule positions during the simulation. The user can specify how frequently they want snapshots to be taken in terms of sample cycles. For each snapshot, the location of every molecule in the system is output to a single `.xyz` file which is closed and ready for use at the end of the simulation.

These snapshots are useful for generating average positions for the molecules, but they are not indicative of molecules moving in the porous material. The simulation in `PorousMaterials.jl` is not time dependent. The GCMC moves are transitioning from one state space to another state space with a similar energy level. Therefore the snapshot functionality cannot be used to show the positions of the adsorbates at a particular time. Instead a snapshot shows the state of the system at a given sample cycle. Adsorbate tracking can give insight into where molecules will likely be located, but it cannot describe the motion of molecules within the system.

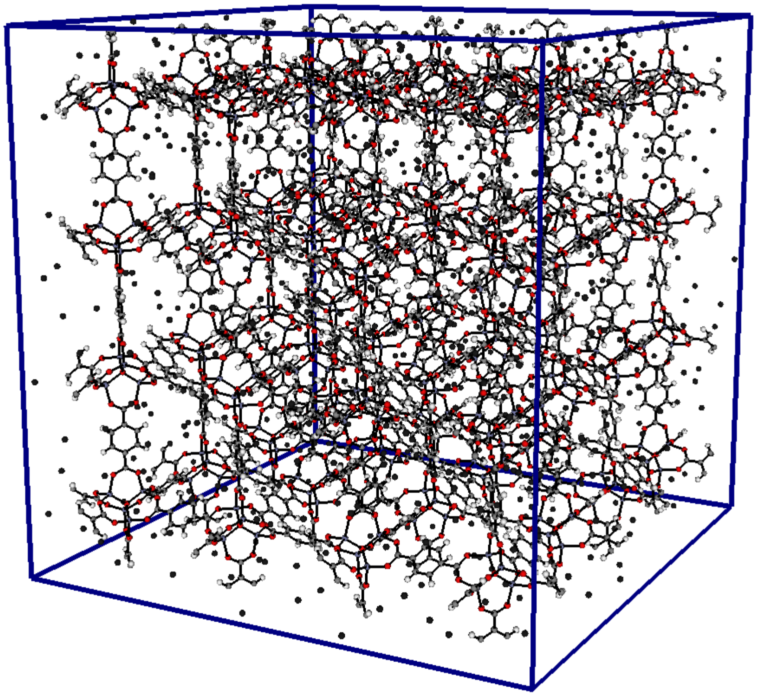

Figure 8: This is a single snapshot taken from a GCMC simulation for methane adsorbtion in IRMOF-1 at 298.0K and 50.0 bar. (Blue-Grey - Zinc, Red - Oxygen, Grey - Carbon, White - Hydrogen, Black - Methane)

Figure 8 is a snapshot of methane adsorbing inside IRMOF-1. It shows the locations of the methane molecules in the crystal structure at an arbitrary sample cycle. The simulation was run at 298.0K and 50.0 bar using the Dreiding forcefield [20]. There were 10,000 burn cycles, 10,000 sample cycles, and snapshots were taken every 1,000 sample cycles. This is one possible state space for the metal-organic framework under the given conditions. This visualization is a useful conceptual tool but does not provide conclusive quantitative data.

4.2 Tracking Adsorbate Positions using a `Grid`

Capturing snapshots is helpful for visualizing where the adsorbates are most likely to adsorb within the system, but lacks a convenient method for aggregating the data to make meaningful conclusions. The snapshot shows the position of each adsorbate at the end of a given sample cycle, but this is not enough to make broad statements about where gasses are likely to adsorb. The format of the snapshot is not conducive to large-scale analysis because one must sift through every entry and compare positions to determine where clusters are occurring. Another method would be to lay a grid over the crystal structure and instead of taking snapshots, each element of the grid can track the average number of adsorbates that appear in its unit of space.

This method is a considerable improvement over snapshots because it requires fewer computational resources and likely adsorption locations will naturally reveal themselves. Writing information to a file is a costly action when compared to updating an array element, so a density grid can collect information from more sample cycles throughout the simulation without sacrificing time. The grid is not tracking individual atoms, instead it has "bins" that count the number of times an atom is found in it every time the grid updates.

The density grid is implemented in the `μVT_sim` function by generating a `Grid` struct that is laid over the crystal such that each element of the density grid corresponds to a voxel in the simulation box where each voxel has the same dimensions. When a snapshot is taken, the simulation iterates through each molecule currently in the system, converts the fractional coordinate of the molecule to indices in the density grid using `xf_to_id`, and increments the counter at that index. At the end of all sample cycles, each element is divided by the total number of snapshots taken. Each element of the grid now contains the average number of adsorbates in the corresponding voxel.

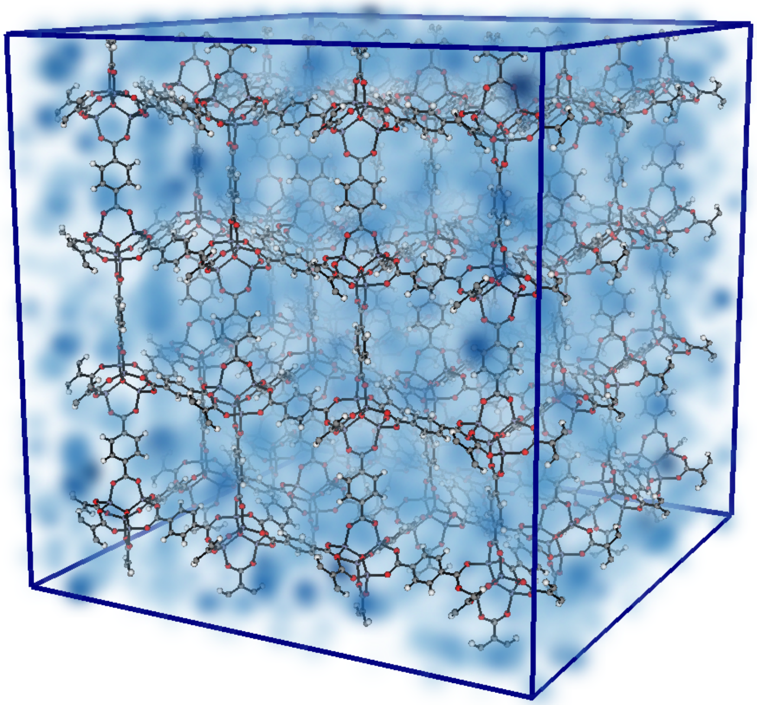

Figure 9: This is the density grid of methane molecules (modelled as spheres) represented as a cloud plot laid over IRMOF-1. The darker blue portions of the cloud have a higher probability of observing methane at that position. Generated from a GCMC simulation for methane adsorbtion in IRMOF-1 at 298.0K and 50.0 bar. (Blue-Grey - Zinc, Red - Oxygen, Grey - Carbon, White - Hydrogen).

Figure 9 shows a density grid generated under the same conditions as figure 8. The density grid ran for the same number of burn and sample cycles, however the `snapshot_frequency` was set to 1 for a more robust density grid. The voxels are no more than 0.5Å apart. This visualization produces more general estimates of where adsorbates are located through gradations in the cloud plot. A darker blue indicates more atoms were found there during the simulation, and it is an energetically favorable position for methane within IRMOF-1. The lighter portions of the cloud plot indicate that adsorbates were found there during the simulation, but not with the same frequency. These are likely positions that require higher energy states because `PorousMaterials.jl` explored them, but quickly accepted a proposal that deleted the offending molecule or moved it back into an energetically favorable location.

Compared to the previous method, this visualization is more rich in information. Instead of describing the state of the system at the end of one sample cycle, the density grid describes the average positions of adsorbates within the system. It becomes easier to differentiate between positions that are likely adsorption sites from positions where adsorbates have been observed. Because the system can jump to higher energy states to escape local minima, the system will not be in the perfect configuration at every cycle. The density grid is able to differentiate between these by averaging data across the entire simulation so positions that are rarely visited due to the increase in energy required appear as less likely areas for adsorption.

5 Generating Bonding Information

Bonds within metal-organic frameworks can be used to reveal the topology of these structures and better understand their form. The main benefit of having information about these bonds is that it shows isolated structures within the crystal file. Being able to view these isolated structures can be extremely valuable. They may reveal multiple structures woven together, or solvents present in the file that were not removed prior its use. Having control over bond information can also be extremely powerful for the formation of hypothetical structures because it enforces the intended topology when it is relaxed using molecular dynamics.

`PorousMaterials.jl` already has an extensive tool kit for interacting with MOFs and other frameworks through its `Crystal` struct, so it was expanded to store information about bonds. Storing bonding information as a graph provides countless, well established algorithms for searching and traversing the structure through neighboring atoms. This means the bonds only need to be stored by `PorousMaterials.jl`, and functions from other libraries can then be used to analyze and modify them.

5.1 Inferring Bonds in `PorousMaterials.jl`

`PorousMaterials.jl` is capable of reading in bonds from crystal source files, inferring them within the system, and writing them when outputting crystal files. It uses an undirected, unweighted graph data structure from `LightGraphs.jl` to represent the crystal where each vertex is an atom and each edge is a bond between two atoms. It can also take user defined bonding rules to determine where bonds exist within the crystal file that is supplied.

`PorousMaterials.jl` can infer bonds on any structure and store that information as a graph inside the `Crystal` struct. The first step is to define a set of bonding rules. These are built using the `BondingRule` struct within `PorousMaterials.jl` that defines a minimum and maximum distance for a bond to occur between two defined atom species. Wildcards can also be used in place of a specific species as a stand in for any type of atom. An array of bonding rules can be passed into the `infer_bonds!` function along with a structure, and a boolean that determines if it will check for bonds between atoms over the boundary of the simulation box. This will not work if the structure already has bonds present either because of bonding information read in from the `*.cif` file or if bonds have already been inferred, to avoid having overlapping sets of bonds.

The `infer_bonds!` function checks every pair of atoms in the system and determines if they are bonded by finding a bonding rule that applies to the given species and then comparing the distance between them to the defined minimum and maximum distances allowed under that rule. The order of the `BondingRules` in the array matters because `infer_bonds!` will find and use the first rule that matches the species of a given pair. If they are not bonded under this rule, it will not check the remaining rules. This is there to ensure that wildcard rules are obeyed. For example, when using the rules outlined in table 3, and working with a hydrogen and oxygen atom 1.3 Å apart, it should not classify them as bonded. If it stops after the first rule that matches their species (H-Any), then they will not be bonded. However, if it instead attempts to use every rule listed it will classify them as bonded under the final rule (Any-Any).

Using a graph is beneficial because there are many algorithms built into `julia` and `LightGraphs.jl` for quick traversal and analysis. Graphs are perfect tools for tasks like this, because each vertex is paired with an atom so one can evaluate a single atom and the atoms it is directly bonded to. One example of how a graph is easily applicable is the `connected_components` function within `LightGraphs.jl`. Once bonds within a structure are inferred, the `connected_components` can be used to find isolated groups of atoms within the original file read in. These groups can be separate structures, solvents, or something else entirely.

5.2 Removing Solvent Atoms from Crystal Files

A time consuming component of simulating metal-organic frameworks is the need to prepare or "clean" the crystal files. Many crystallographic crystal files on the Cambridge Crystallographic Data Centre still have atoms from the solvents present in a low level symmetry group. Before simulations can be run, these must be removed. It is time consuming to run trial and error deletions of atoms by hand. An automated process that can detect and extract the proper crystal structure can increase the number of potential structures for testing.

The crystal structure CaSDB can be found on the Cambridge Crystallographic Data Centre in its "dirty" form under the identifier `KAXQIL` [23]. This is a simple example where the solvent atom (oxygen) is easy to locate in its lower symmetry form. However it is a useful example of how solvent removal can be automated to pave the way for large scale simulation using experimentally derived crystal structure files. This can be refined and scaled up to automate the cleaning of batches of simulation files before they can be used.

Figure 10 shows the crystal structure CaSDB from the Cambridge Crystallographic Data Centre without any modification aside from a conversion to P1 symmetry and a replicated unit cell. The visualization shows how the solvent atoms have been isolated and can be removed from the structure. The main crystal structure is colored red and the solvent atoms present in the original crystal file are colored blue. This was achieved by inferring bonds in the structure across periodic boundary conditions using the bonding rules shown in table 3 to generate a graph of the crystal.

By assuming that CaSDB is a single structure, that it has more connected atoms than any of the solvent groups, and that no solvent is bonded to the crystal, it can be isolated and extracted from its original form. The largest connected component of the graph representation of the structure must be CaSDB under these assumptions, and it can be removed to form a separate crystal using functionality within `PorousMaterials.jl` . The remaining connected components can also be separated and stored to prove through visualization that structure has been properly "cleaned."

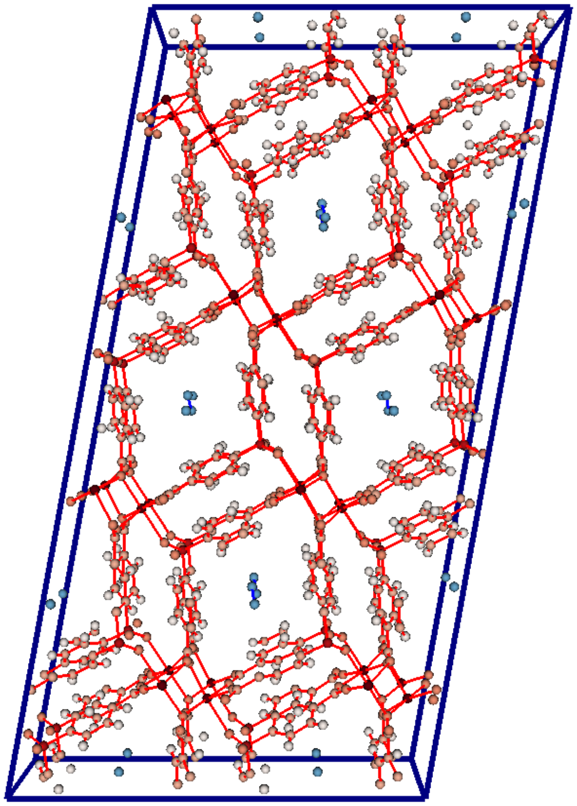

Figure 10: This is the CaSDB structure file from the Cambridge Crystallographic Data Centre (identifier: KAXQIL) with the solvent atoms still present. The red atoms and bonds represent the "cleaned" structure that can be used for simulation. The blue atoms and bonds represent the solvents present in the raw data file that must be removed.

Table 3: Bonding Rules for CaSDB

Species 1	Species 2	Min. Bonding Distance	Max Bonding Distance
H	Any	0.4 Å	1.2 Å
Ca	Any	1.2 Å	2.5 Å
Any	Any	0.4 Å	1.9 Å

5.3 Separating Interpenetrated Metal-Organic Frameworks

Some metal-organic frameworks are interpenetrated, meaning that there are disjoint structures woven through each other to make up the greater structure. These are not always labeled as interpenetrated in large databases, but it is valuable information to have. Having an automated tool for scanning databases and identifying interpenetrated structures for further study can cut down on time spent searching for them. This can also reveal the level of interpenetration and separate the components for further study of its non-interpenetrated forms.

This is useful for analyzing interpenetrated metal-organic frameworks because it provides a simple way for identifying the separate structures. All one needs to do to isolate them is infer bonds in the structure across periodic boundaries and then find the connected components of the bond graph. This yields a collection of subgraphs that represent the disparate structures. From here, the subgraphs can guide the creation of individual crystals for each interpenetrated layer from the original file. These can then be written to individual `*.cif` files to be examined on their own, or to showcase their disjoint nature through visualization.

Figure 11 shows a $2 \times 2 \times 2$ unit cell of NiPyC2 where the disjoint structures are colored separately. The interpenetrated layers only revealed themselves when the unit cell was replicated by 2 along each axis. The bonds were inferred wrapping across the periodic boundaries of the unit cell using the bonding rules shown in table 4. The graph of the bonds contained two connected components, so the structure was separated into two sub-structures using the vertices in each subgraph to determine which atoms were present. Finally the two crystal structures were written to separate `*.cif` files and then visualized in separate colors side-by-side to show how the interpenetrated layers of NiPyC2 are woven together.

Table 4: Bonding Rules for NiPyC2

Species 1	Species 2	Min. Bonding Distance	Max Bonding Distance
H	Any	0.4 Å	1.2 Å
N	Ni	1.2 Å	2.5 Å
O	Ni	1.2 Å	2.5 Å
Any	Any	0.4 Å	1.9 Å

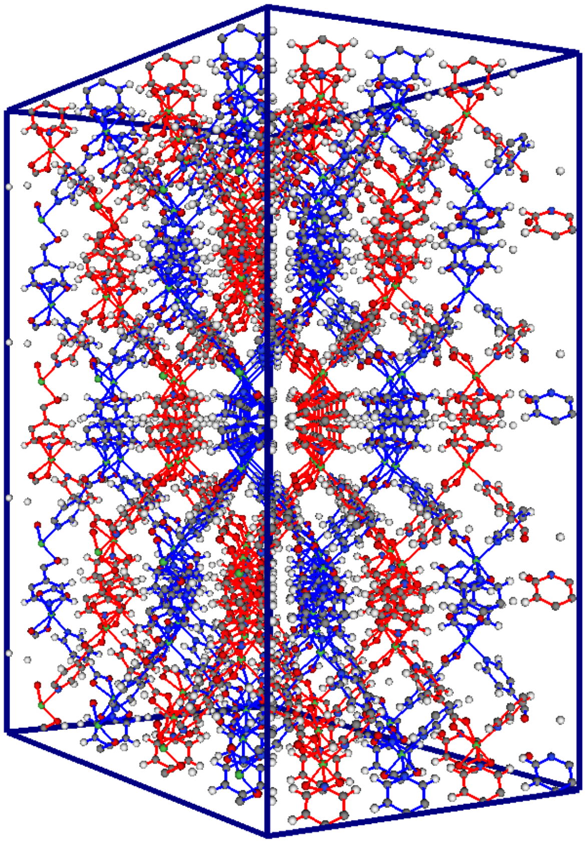

Figure 11: NiPyC2 is an example of an interpenetrated MOF. In its crystal file there are two distinct structures that are not connected. By calculating the bonds in `PorousMaterials.jl` it is possible to separate the crystal based on the connected components of the bond graph. Then these components can be colored (red and blue) to show the two distinct structures.

6 Conclusion

`PorousMaterials.jl` is a powerful, open source package for `julia` that is capable of modelling and analyzing complex porous crystalline structures. The powerhouse of the software is its ability to model gas adsorption in nano-porous structures and generate adsorption isotherms that match experimental data. It also provides unique analysis through the adsorbate tracking functionality to predict high probability locations for adsorption. Lastly `PorousMaterials.jl` can analyze and manipulate the structures through inferred bonds and the `Crystal` structure.

`PorousMaterials.jl` is the perfect tool to complement experimental research because it can guide the synthesis of materials through high throughput computation of a batch of potential MOFs. Many synthesized and hypothetical MOFs lack reliable adsorption data for a range of gasses making it difficult to know where to begin examining them [24]. `PorousMaterials.jl` can easily be scaled up with enough computing power to produce hypothetical adsorption isotherms for many distinct materials. These can be well known MOFs, or hypothetical variations developed *in silico*. This simulation can produce more data faster than traditional experimental methods, so it can be invaluable for prioritizing promising areas of research.

Not only can this simulation estimate adsorption, but it can predict where these materials can be located inside the structure. This data provides insight into modifying linker molecules to provide the optimal environment for attracting and holding adsorbates. It is a perfect companion to producing hypthetical structures because it provides a simple, efficient way to analyze how modifications to the structure alter the adsorption within it.

The potential to examine bonds provides another layer of analysis for the structure. The bonding information can reveal interesting aspects of the topology such as woven layers or stray solvents. It also allows users to quickly modify structures through replacing or functionalizing certain linkers while retaining the desired topology by enforcing bonds. Bonding information is another lens through which these materials can be examined.

`PorousMaterials.jl` is an extensive software package that can provide expansive functionality in a single, free and open source software package written in a user friendly language. From experimental crystal structure files, it can identify solvents or separate layers woven together and clean them for use in simulations. It then has the power to modify these structures and create new hypothetical MOFs for further investigation. These crystal structures then model adsorption through the grand-canonical Monte Carlo simulation to understand how well a given MOF stores a certain gas. The adsorption sites within the structure can then be studied to guide the creation of another round of hypothetical MOFs in the quest to find the optimal structure. `PorousMaterials.jl` is a single free and open source software package that can manipulate and model nano-porous materials quickly, making it ideal for exploring novel MOFs and optimizing known structures.

References

[1] M. Q. Wang and H. S. Huang, "A full fuel-cycle analysis of energy and emissions impacts of transportation fuels produced from natural gas," tech. rep., United States, Argonne National Lab, 2000.

[2] S. Yeh, "An empirical analysis on the adoption of alternative fuel vehicles: The case of natural gas vehicles," *Energy Policy*, vol. 35, pp. 5865–5875, 2007.

[3] C. M. Simon, J. Kim, D. A. Gomez-Gualdron, J. S. Camp, Y. G. Chung, R. L. Martin, R. Mercado, M. W. Deem, D. Gunter, M. Haranczyk, D. S. Sholl, and R. Q. Snurr, "The materials genome in action: identifying the performance limits for methane storage," *Energy & Environmental Science*, vol. 8, pp. 1190–1199, 2015.

[4] J. A. Mason, J. Oktawiec, M. K. Taylor, M. R. Hudson, J. Rodriguez, J. E. Bachman, M. I. Gonzalez, A. Cervellino, A. Guagliardi, C. M. Brown, P. L. Llewellyn, N. Masciocchi, and J. R. Long, "Methane storage in flexible metal–organic frameworks with intrinsic thermal management," *Nature*, vol. 527, pp. 357–361, 2015.

[5] H. Furukawa, K. E. Cordova, M. O'keeffe, and O. M. Yaghi, "The chemistry and applications of metal-organic frameworks," *Science*, vol. 341, no. 6149, pp. 1230444–1230444, 2013.

[6] D. Banerjee, C. M. Simon, A. M. Plonka, R. K. Motkuri, J. Liu, X. Chen, B. Smit, J. B. Parise, and M. H. amd Praveen K. Thallapally, "Metal–organic framework with optimally selective xenon adsorption and separation," *Nature Communications*, vol. 7, p. n.p., 2016.

[7] D. Banerjee, A. J. Cairns, J. Liu, R. K. Motkuri, S. K. Nune, C. A. Fernandez, R. Krishna, D. M. Strachan, and P. K. Thallapally, "Potential of metal–organic frameworks for separation of xenon and krypton," *Accounts of Chemical Research*, vol. 48, pp. 211–219.

[8] A. Sturluson, M. T. Huynh, A. R. Kaija, C. Laird, S. Yoon, F. Hou, Z. Feng, C. E. Wilmer, Y. J. Colon, Y. G. Chung, D. W. Siderius, and C. M. Simon, "The role of molecular modelling & simulation in the discovery and deployment of metal-organic frameworks for gas storage and separation," *ChemRxiv*. https://doi.org/10.26434/chemrxiv.7854980.v2.

[9] B. Edmonds and D. Hales, "Computational simulation as theoretical experiment," *Journal of Mathematical Sociology*, vol. 29, pp. 209–232, 2005.

[10] O. Silva, "Black death–model and simulation," *Journal of Computational Science*, vol. 17, pp. 14–34, 2016.

[11] J. Carney, D. Roundy, and C. Simon, "Statistical mechanical model of gas adsorption in a metal–organic framework harboring a rotaxane molecular shuttle," *Langmuir*, vol. 36, pp. 13112–13123, 2020.

[12] B. J. Sikora, C. E. Wilmer, M. L. G. b, and R. Q. Snurr, "Thermodynamic analysis of xe/kr selectivity in over 137,000 hypothetical metal–organic frameworks," *Royal Society of Chemistry*, vol. 3, pp. 2217–2223, 2012.

[13] G. R. Lorzing, B. A. T. Aeri J. Gosselin, A. H. P. York, A. Sturluson, C. A. Rowland, G. P. A. Yap, C. M. Brown, C. M. Simon, and E. D. Bloch, "Understanding gas storage in cuboctahedral porous coordination cages," *Journal of the American Chemical Society*, vol. 141, no. 30, pp. 12128–12138, 2019.

[14] A. Sturluson, M. T. Huynh, A. H. P. York, and C. M. Simon, "Eigencages: Learning a latent space of porous cage molecules," *ACS Central Science*, vol. 4, pp. 1663–1676, 2018.

[15] K. Carillo and C. Okoli, "The open source movement: A revolution in software development," *Journal of Computer Information Systems*, vol. 49, pp. 1–9.

[16] C. Simon, A. H. York, Á. Sturluson, M. T. Huynh, A. S. Rosen, C. Laird, , and M. Piibeleht, "Simonensemble/porousmaterials.jl:bonds, symmetry and flexible file paths."

[17] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM*, vol. 59, no. 1, pp. 65–98, 2017.

[18] C. M. Simon, *Computation assisted discovery of nanoporous materials for gas storage and separations*. PhD thesis, 2016.

[19] H. Tijms, *Probability: A Lively Introduction*. Cambridge University Press, 2018.

[20] S. L. Mayo, B. D. Olafson, and W. A. Goddard, "Dreiding: a generic force field for molecular simulations," vol. 94, no. 26, pp. 8897–8909, 1990.

[21] A. K. Rappe, C. J. Casewit, K. S. Colwell, W. A. Goddard, and W. M. Skiff, "Uff, a full periodic table force field for molecular mechanics and molecular dynamics simulations," *Journal of the American Chemical Society*, vol. 114, no. 25, pp. 10024–10035, 1992.

[22] J. A. Mason, M. Veenstra, and J. R. Long, "Evaluating metal-organic frameworks for natural gas storage," *Chemical Science*, vol. 5, pp. 32–51.

[23] D. Banerjee, Z. Zhang, A. M. Plonka, J. Li, and J. B. Parise, "A calcium coordination framework having permanent porosity and high co2/n2 selectivity," *Cyrstal Growth and Design*, vol. 12, pp. 2162–2165.

[24] A. Sturluson, A. Raza, G. D. McConachie, D. Siderius, X. Fern, and C. M. Simon, "A recommendation system to predict missing adsorption properties of nanoporous materials," *ChemRxiv*. https://doi.org/10.26434/chemrxiv.14205929.v3.