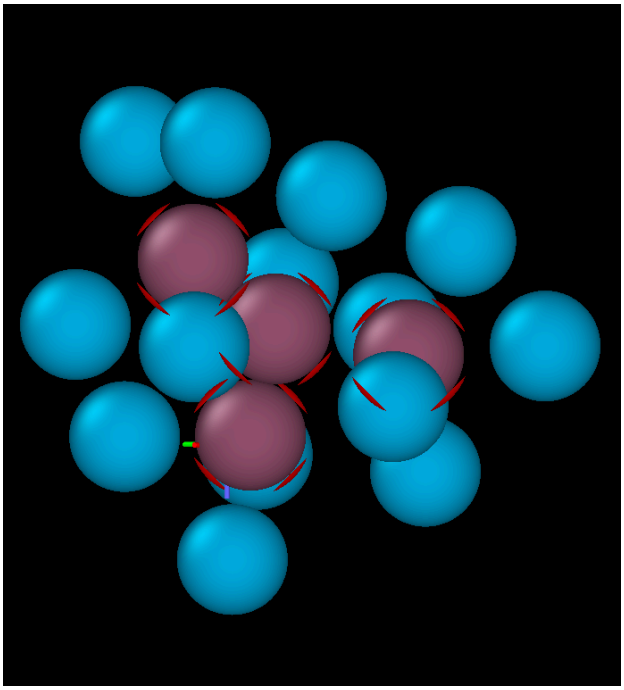
Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Structural Engineering

Kevin Cornolis

Identifying intermediate structures during water and hydrate phase transitions with machine learning

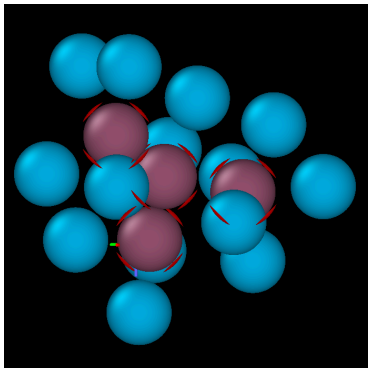
Master's thesis in Engineering and ICT
Supervisor: Senbo Xiao
June 2023



NTNU
Norwegian University of
Science and Technology

Kevin Cornolis

Identifying intermediate structures during water and hydrate phase transitions with machine learning



Master's thesis in Engineering and ICT
Supervisor: Senbo Xiao
June 2023

Norwegian University of Science and Technology
Faculty of Engineering
Department of Structural Engineering

NTNU
Norwegian University of
Science and Technology

Department of Structural Engineering
Faculty of Engineering
NTNU- Norwegian University of Science and Technology

ACCESSIBILITY

Open

MASTER THESIS 2023

SUBJECT AREA: Nanomechanics and Machine learning	DATE: 11.06.2023	NO. OF PAGES:99

TITLE:

Identifying intermediate structures during water and hydrate phase transitions with machine learning

Indentifisere midlertidige bindinger under vann og hydrat faseoverganger ved maskin læring

BY:

Kevin Cornolis

SUMMARY:
Water molecules can take different shapes and can form different structures such as ice. It also can form new substances when mixed with other compounds, like Hydrates. However, how the individual water molecule goes from being free-flowing in its liquid state to participating in forming solid structures is largely a random event and yet to be investigated in detail. Does there exist a transitional phase between water and ice or water and hydrate? Does there exist an intermediate structure that explains this transitional phase?

This Thesis attempted to discover these possible intermediate water structures during the transition from water to ice or hydrate. This was done by utilizing different machine learning techniques, such as DBSCAN and k-means clustering.

A Molecular Dynamics simulation was done, and the trajectory data from this simulation was used to find intermediate structures. By using different methods to explore the data, and using the mentioned machine learning techniques, three different experiments were conducted to find potential intermediate structures.

The most promising structures were then revealed in OVITO to see how they behaved during the MD simulation and see if they could be classified as an intermediate structures. For the first time, Y-shape structures formed by water molecules were identified in this work, which primarily validated the existence of intermediate structures in a water phase transition.

RESPONSIBLE TEACHER: Senbo Xiao

SUPERVISOR: Senbo Xiao

CARRIED OUT AT: Department of Structural Engineering

NTNU

Kunnskap for en bedre verden

TKT4950 - Master Thesis

Identifying intermediate structures during water and hydrate phase transitions with machine learning

Author:
Kevin Cornolis

Date 11.06.2023

Abstract

Water molecules can take different shapes and can form different structures such as ice. It also can form new substances when mixed with other compounds, like Hydrates. However, how the individual water molecule goes from being free-flowing in its liquid state to participating in forming solid structures is largely a random event and yet to be investigated in detail. Does there exist a transitional phase between water and ice or water and hydrate? Does there exist an intermediate structure that explains this transitional phase?

This Thesis attempted to discover these possible intermediate water structures during the transition from water to ice or hydrate. This was done by utilizing different machine learning techniques, such as DBSCAN and k-means clustering.

A Molecular Dynamics simulation was done, and the trajectory data from this simulation was used to find intermediate structures. By using different methods to explore the data, and using the mentioned machine learning techniques, three different experiments were conducted to find potential intermediate structures.

The most promising structures were then revealed in OVITO to see how they behaved during the MD simulation and see if they could be classified as an intermediate structure. For the first time, Y-shape structure formed by water molecules were identified in this work, which primarily validated the existence of intermediate structures in water phase transition.

Keywords: water, ice, hydrate, intermediate structure, molecular dynamics, machine learning, DBSCAN, k-means, OVITO

Sammendrag

Vannmolekyler er i stand til å komme i forskjellige former og kan danne forskjellige strukturer, som is. Vann har også evnen til å danne nye stoffer når det blander seg med andre forbindelser, som hydrater. Men hvordan individuelle vannmolekyler går fra å være fritt flytende i sin væskeform til å delta i dannelse av solide bindinger, er fortsatt en tilfeldig hendelse som trenger en detaljert undersøkelse. Finnes det en midlertidig binding som forklarer denne overgangsperioden?

Denne oppgaven vil forsøke å finne disse mulige midlertidige bindingene som forekommer i overgangen mellom vann og is, eller hydrat. Dette ble forsøkt ved bruk av forskjellige maskinlæringsmetoder som DBSCAN og k-means-gruppering.

En molekylær dynamikk-simulering ble gjennomført, og banedataene fra denne simuleringen ble brukt til å finne midlertidige bindinger. Ved å bruke forskjellige metoder for å utforske datasettet, og ved å bruke nevnte maskinlæringsmetoder, ble tre forskjellige eksperimenter gjennomført for å finne midlertidige bindinger.

De mest lovende bindingene ble presentert i OVITO for å se hvordan de oppførte seg i MD-simuleringen, for å vurdere om de kunne klassifiseres som midlertidige bindinger. For første gang har Y-formede strukturer dannet av vannmolekyler blitt identifisert i dette prosjektet, noe som primært bekrefter eksistensen av midlertidige bindinger i overgangsfasen mellom vann og is.

Preface

This Thesis has been written as the final project for the Master of Science. The work was conducted at the Department of Structural Engineering at the Norwegian University of Science and Technology (NTNU) during the spring semester of 2023. The author have a background from the study program Engineering and ICT, with a specialisation in structural engineering.

Acknowledgments

I want to express my gratitude towards my supervisor, Professor Senbo Xiao. I highly appreciate his advice, interest and availability beyond what can be expected.

I also want to thank Rui Ma for providing me with the Molecular Dynamics (MD) simulation data of ice and hydrate nucleation growth, that built the foundation of this Master Thesis.

Kevin Cornolis

Table of Contents

List of Figures

1 Introduction

Water is the most abundant natural resource on this planet [1]. It can take many different shapes and can be in different forms, such as liquid, ice, and gas[2]. It also can form new substances when mixed with other compounds, like a hydrate[3].

Currently, the investigation of water phase transition at the nanoscale remains an active area of research, particularly concerning the discovery of previously unexplored water structures [4][5]. To comprehensively comprehend the transition mechanism of water, it is crucial to identify the intermediate structures that emerge during the process. These intermediate structures serve as significant indicators of the free energy pathways, attracting scientific interest across various fields aiming to elucidate phase transitions [6][7][8].

This master's thesis centers around the investigation of intermediate water structures during the transition from water to ice and hydrate. The author specifically employed diverse machine-learning techniques to analyze high-resolution trajectories obtained from molecular dynamics simulations. The primary objective was to develop effective methods for recognizing water and hydrate structures and identifying the characteristic intermediate structures that play a crucial role in water and hydrate phase transitions.

1.1 Motivation

The reason for performing this master thesis is for the simple fact that hydrate and ice raise challenges for mankind's activities and made structures[9][10]. For example, ice formation causes a lot of challenges for a structure like a windmill. Ice formations make it difficult to operate the windmills and will cause less power to be generated[9]. Likewise, another water-based structure, hydrate, also can be highly harmful to infrastructures [10]. In deep sea oil pipes, the conditions are just right for hydrates to form inside the pipes. When the hydrates form, they can clog up the pipe, meaning that the oil from the rigs will have difficulty to move through the pipe[10]. By understanding how ice and hydrates form, especially the fundamentals like their intermediate structures in the phase transition, this information could be used to help prevent the formation of ice and hydrates. This could be done by finding materials or coatings that could prevent the nucleation of ice and hydrates.

1.2 Previous work in the field

While numerous studies have explored intermediate structures in various materials such as proteins and polymers[6][7][8], there is a scarcity of research focusing on intermediate water structures during the phase transition of water. This lack of investigation is primarily attributed to the relatively straightforward structure of water (compared to proteins and polymers) and the fleeting existence of potential intermediate structures. Consequently, this project places a high priority on using machine learning techniques to identify any potential intermediate water structures based on rational data derived from molecular dynamics simulations. The verification of these identified structures will be subject to future investigations.

1.3 Scope of this research

This section presents the objective of the thesis research and an overview of the methodology used to achieve the objectives.

1.3.1 Objective

The objective of this thesis is to: Use machine learning techniques to try to identify intermediate structures during ice and hydrate freezing nucleation, and get a better understanding of how

machine learning further can be used to identify intermediate structures in an optimal way.

1.3.2 Methodology

This section describes the methodology of how the goal of this thesis was tried to be accomplished.

STEP1 - MD-simulation:
Molecular Dynamics (MD) simulation was performed using the simulation package LAMMPS [11], which reproduces the phase transition of water to ice and hydrate. The water model utilized in the simulations was the mW water model specialized in capturing the hydrogen bonding in water [12]. The resulting MD simulation trajectories were then used at the raw data for water intermediate structure identification.

STEP2 - Water recognition:
To find the intermediate structures the following assumption was made: Intermediate structures would only be observable in the water layer of the MD simulation. Therefore the machine learning algorithm DBSCAN was used to filter out all molecules that did not belong to the water layer. Boundary conditions were also introduced to prevent the artificial "teleportation" effect of atoms in the simulation trajectories due to the periodic boundary condition of the simulation systems.

STEP3 - Experiments:
Three different experiments were conducted to try and find intermediate structures. These experiments had different approaches on how to find intermediate structures.

STEP4 - Properties:
Each experiment would then utilize a form of average bond distance analysis and an average volume distance analysis to describe the intermediate structures with properties (bond distance and volume) as such, patterns in the structures could be analyzed.

STEP5 - Analysing the results:
Finally, the k-means clustering algorithm would be used to find patterns in the intermediate structures. One sample from each cluster would be chosen from each experiment, and analyzed by looking at bonding patterns, comparing them to each other, and comparing them to ice and hydrate structures.

2 Theory

This chapter will present the necessary theory needed to understand this paper.

2.1 What is an intermediate structure

An intermediate structure is a molecular structure that forms between the initial stage and final stage of nucleation. In the case of this paper, it is the structure that forms between the initial stage and final stage of water-freezing nucleation (water turning into ice). These intermediate structures are primarily hypothetical.

2.2 The water molecule

To understand intermediate structures we first have to understand the building blocks of these intermediate structures, which is the water molecule.

Water, with the chemical formula H_2O [2], is a familiar and essential compound. It consists of two hydrogen atoms covalently bonded to a central oxygen atom [13]. The molecular structure of water gives rise to its unique properties and behavior.

Water molecules exhibit polarity as seen in figure 1, meaning they have a separation of electric charge. This polarity arises from the unequal sharing of electrons between the oxygen and hydrogen atoms in a water molecule. The oxygen atom is more electronegative than the hydrogen atoms, meaning it attracts the shared electrons more strongly [14].

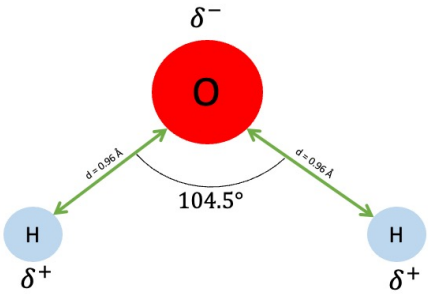


Figure 1: Model of a simple H20 molecule. The red atom is oxygen, the light blue is hydrogen. The oxygen atom in water is more electronegative than the hydrogen atoms, resulting in a partial negative charge (δ-) on the oxygen atom and partial positive charge (δ+) on the hydrogen atoms. This polarity leads to the formation of hydrogen bonds between water molecules. The average distance between the oxygen atom and hydrogen atom is depicted in the figure, and the average angle between the hydrogen bond is also depicted [14].

The polarity of water has several important consequences. First, it gives water the ability to form hydrogen bonds [15]. The positive hydrogen atom of one water molecule can be attracted to the negative oxygen atom of another water molecule, forming a relatively weak but significant bond. These hydrogen bonds also give water unique properties such as high boiling point, high specific heat, and high surface tension [2].

Furthermore, water's polarity allows it to dissolve many substances[2]. Water can surround and separate charged particles or polar molecules, facilitating their dispersion and creating solutions. This property is crucial for many biological processes, as water is the primary medium for chemical reactions in living organisms [15]. These properties will however not be investigated in greater detail as they are not needed for understanding this report.

In its liquid state, the water molecule adopts a bent or V-shaped geometry. The distance between the hydrogen atoms and the oxygen atom is approximately 0.96 Å (angstroms), and the angle

between the two hydrogen atoms is around 104.52 degrees[13] as seen in figure 1. These values reflect the average geometry of water molecules in the liquid phase and are influenced by the electrostatic interactions between the partially charged atoms[13].

When water transitions to its gaseous state as water vapor, the molecular arrangement changes slightly. The distance between the hydrogen atoms and the oxygen atom increases to around 0.97 Å, while the angle between the hydrogen atoms remains similar to that of the liquid state[16]. These alterations in bond lengths and angles are a consequence of the weakened intermolecular forces and increased molecular motion in the gas phase[16].

In the solid state, as in the case of ice, water molecules form a highly ordered lattice structure due to the formation of extensive hydrogen bonding. In ice, the distance between the hydrogen atoms and the oxygen atom expands further to approximately 0.99Å[16]. The angle between the hydrogen atoms in ice can be approximated to the ideal tetrahedral angle of 109.5 degrees[16]. The slightly larger distances and angles in ice compared to liquid water result from the more rigid and fixed arrangement of water molecules within the crystal lattice[16].

These variations in bond distances and angles between the different states of water highlight the dynamic nature of water molecules and their response to changes in temperature and pressure. The ability of water to exist in all three states—solid, liquid, and gas—makes it a crucial compound for supporting life and enabling various natural processes on Earth [2].

2.3 Ice

Ice is water in its solid state. The molecular structure for a single H_2O is mostly the same other than the differences mentioned in section 2.1.

When water freezes, it undergoes a phase transition from the liquid state to the solid state. This transition occurs at a specific temperature known as the freezing point, which is 273.15 K (0 °C) at normal atmospheric pressure[17]. At this temperature, the thermal energy of the water molecules decreases to a point where the intermolecular forces, particularly hydrogen bonding, become strong enough to overcome the molecular motion and hold the water molecules in a fixed, ordered arrangement.

Homogeneous freezing nucleation occurs when supercooled liquid water droplets are spontaneously converted into ice without the presence of any external solid surfaces or impurities. Supercooling refers to the process of cooling a substance below its freezing point without it immediately transitioning into a solid state[18]. In the case of water, supercooling can occur when the temperature drops below 273.15 K, but the water remains in the liquid state[19].

As the temperature of supercooled water droplets reaches around 230 K (approximately -43 °C) at normal atmospheric pressure, homogeneous freezing nucleation becomes more likely. At this point, the thermal energy of the water molecules is significantly reduced, allowing the formation of stable hydrogen bonding networks between neighboring water molecules. These hydrogen bonds facilitate the arrangement of the water molecules into an ordered, crystalline structure characteristic of ice[20].

The process of homogeneous freezing nucleation is initiated by the formation of ice nuclei, which are clusters of water molecules that adopt the ice lattice structure. Once these ice nuclei form, they can grow and propagate, eventually transforming the entire supercooled water droplet into a solid ice particle[20].

It's important to note that supercooling and homogeneous freezing nucleation are complex phenomena influenced by various factors such as impurities, pressure, and cooling rate[21]. These factors can affect the stability and kinetics of ice formation in supercooled water, and they play a significant role in atmospheric processes like cloud formation and precipitation[22].

2.3.1 Ice structure

This paper will mainly focus on hexagonal ice (ice Ih), but there are other forms of ice structures. The molecular structure of ice is not the same as in water and water vapor. In the liquid and gaseous states, the water molecules don't form any strong bonds with each other, they rather interact with each other through weak intermolecular forces, such as hydrogen bonding. These interactions are relatively transient, allowing the molecules to move freely. This is not the case for ice. In the solid state of ice, each water molecule forms hydrogen bonds with four neighboring water molecules. Which results in highly ordered and stable arrangement.[16]. These arrangements/bonds give rise to a hexagonal lattice structure that forms the ice Ih (also called hexagonal ice and the most common form of ice in the biosphere), which often forms under atmospheric pressure [23] and is illustrated in figure 2.

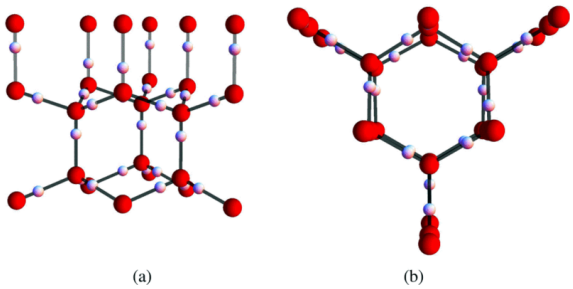


Figure 2: Crystal structure of hexagonal (Ih) water ice. Each oxygen (red sphere) forms two short, covalent bonds, and two long, hydrogen bonds with neighboring protons (white spheres). Oxygen atoms form an ordered lattice. (a) The structure is viewed perpendicular to the hexagonal symmetry axis. (b) Structure viewed along the hexagonal symmetry axis. [23]

In ice Ih, each water molecule is surrounded by four other water molecules, arranged tetrahedrally. Two of these molecules are hydrogen-bonded to the oxygen atom of the central molecule from above and below, while the remaining two are hydrogen-bonded to its two hydrogen atoms[23] which is illustrated in figure 2. This interconnected network of hydrogen bonds forms a stable three-dimensional structure[23].

The hexagonal lattice structure of ice leads to a significant amount of empty space between the water molecules as seen in figure 2. This arrangement causes ice to have a lower density than liquid water, which is an unusual property. Typically, substances become denser as they transition from the liquid to the solid state. However, due to the expansion caused by the hexagonal structure and the resulting open spaces, ice is less dense than water [24].

The distance between the two nearest water molecules in ice Ih is approximately 2.76 Å (angstroms) at 273.15 K (0 °C). As the temperature decreases, the distance between the water molecules in ice may slightly change. For example, at 213.15 K (-60 °C), the distance is approximately 2.755 Å. These small variations in the distance can be attributed to the thermal motion and the contraction of the lattice with decreasing temperature[25].

As mentioned at the beginning of this section, there are different forms of ice, known as ice phases. They have different molecular arrangements and densities. These differences arise from variations in temperature, pressure, and other factors. Each phase has a unique crystal structure and physical properties, adding to the complexity of ice's molecular arrangements[26].

2.4 Hydrates

Hydrates are complex crystalline solids that look like ice[27]. They are made up of water molecules and a guest molecule such as CO_2, Cl_2, halocarbons or various paraffin hydrocarbons[28]. Sometimes the guest molecules can be replaced by Nitrogen (N_2)[28]. Hydrates form during low temperatures, usually above the freezing point of water, but this temperature can increase if the pressure increases[28]. If the hydrate is warmed up, it will transform back to liquid water and the

original organic material [27].

As mentioned this paper will mainly focus on the methane hydrate.

2.4.1 Methane hydrate

Methane hydrate is made out of methane (CH_4) and water. It is a solid substance that closely resembles white ice in appearance[29].

To maintain stability, methane hydrate typically requires pressures of 35 bars or more, which is approximately 35 times atmospheric pressure, and temperatures between 273.15 to 277.15 kelvin (0 to 4 degrees Celsius)[29]. These conditions are commonly encountered in deep-sea regions with water depths ranging from 350 to 5000 meters. Within this range, the pressure and temperature conditions are conducive to the formation and stability of methane hydrate[29]. Bellow 5000m there is insufficient organic matter embedded in the deep-sea sediments, so there are little to no hydrates found beyond this point [29].

2.4.2 Hydrate molecular structure

Methane hydrate, like other hydrates, exhibits a solid molecular structure composed of water molecules interconnected through hydrogen bonds[28]. These bonds form a polyhedral cage-like structure, within which the guest molecule, such as methane, is encapsulated[28]. The presence of the guest molecule stabilizes the hydrate structure by occupying the void spaces in the water lattice[30].

Hydrates can be classified into three distinct molecular structure types: Type I, Type II, and Type H (which will not be discussed in this context). Each structure type is characterized by specific cage geometries formed by the water molecules[30] and is depicted in figure 3.

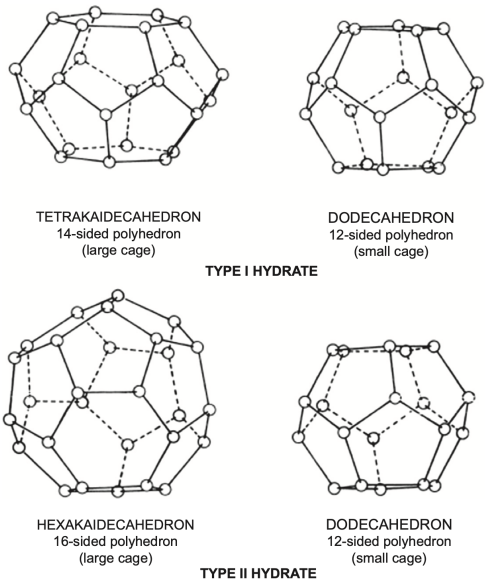


Figure 3: The polyhedral cages of Type I and Type II hydrates [30]

Type I hydrates exist in two variants: the small cage and the large cage. The small cage takes the form of a dodecahedron, a 12-sided polyhedron where each face is a regular pentagon. Conversely, the large cage is a tetrakaidecahedron comprising 14 sides, consisting of 12 pentagonal faces and 2 hexagonal faces. These cage structures can be observed in figure 3. Type I hydrates consist of approximately 46 water molecules, arranged within the cage structure[30].

Type II hydrates also manifest in two variants: the small cage and the large cage. The small cage of Type II hydrates resembles a dodecahedron, while the large cage adopts a hexakaidecahedral shape, featuring 16 sides comprised of 12 pentagonal faces and 4 hexagonal faces. The small and large cage variants of Type II hydrates are depicted in figure 3. The cage structure of Type II hydrates incorporates around 136 water molecules[30].

Methane hydrate conforms to a Type I structure and can be present within both the small and large cage structures. The specific cage structure in which methane hydrate forms depends on factors such as temperature, pressure, and the composition of the surrounding environment[30]. A methane hydrate is depicted in figure 4.

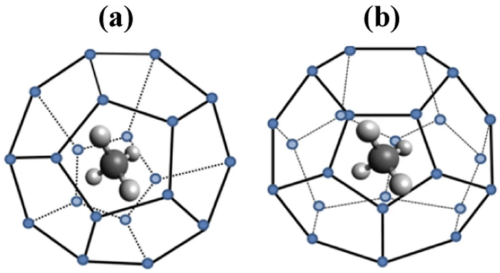


Figure 4: The molecular structure of methane hydrate, with the guest molecule (methane). (a) small cage, (b) large cage structure [31].

2.5 Molecular dynamics

Molecular dynamics (MD) is a powerful computational technique employed in the field of computational chemistry and physics. It involves solving the classical Newtonian equations of motion numerically to simulate the behavior, location, and trajectory of a system comprising of N particles. These particles represent atoms, molecules, or other entities of interest within the system[32][33].

The simulation is based on specific interatomic or intermolecular potential functions that describe the forces and interactions between the particles to allow the MD simulation to mimic the behavior of the real system under investigation[33].

To initiate an MD simulation, certain initial conditions, such as particle positions, velocities, and orientations, are specified. These conditions reflect the desired starting state of the system and can be based on experimental data, theoretical predictions, or random sampling techniques[33].

Boundary conditions are used to impose constraints imposed on the system during the simulation and to describe the environment. For example, periodic boundary conditions can be used to simulate an infinite system by creating periodic replicas of the simulation box[33].

To simulate the behavior of the particles, MD simulations integrate the equations of motion over discrete time steps. This provides a detailed atomistic view of the system's behavior. This is done by generating the trajectories that describe the positions, velocities, and energies of the particles as a function of time. By analyzing these trajectories a wide range of information can be extracted, such as temperature, pressure, and free energy, as well as dynamic properties like diffusion coefficients, reaction rates, and vibrational frequencies[33].

2.5.1 MW-model

To simulate water in an MD simulation, one usually starts by choosing a water model. The water model describes the interactions of water molecules with other molecules [34]. There are a lot of water models to choose from like, exp, SPC, SPCE, TIP3P, TIP4P, and TIP5P[12].

This paper will not focus on any of the mentioned models but will focus on the monatomic water model mW, simply called the mw model. This is a Coarse-Grained Model, which represents each

molecule as a single particle that interacts through anisotropic short-ranged potentials that favor "hydrogen-bonded" water structures[35]. The mW model uses the three-body Stillinger–Weber (SW) potential to account for angle-dependent inter-atomic interactions in water, featuring hydrogen bonding[34]. By using the three-body interactions among the water molecules, the mw model is able to capture the tetrahedron structures of the hydrogen bonding network, which is crucial in the studies of hydrate and ice[36]. This makes it a suitable candidate when one works with ice and hydrates.

When simulating hydrates, the mw model will describe the guest molecules as a "M" particle, especially methane in the hydrate. The "M" particle interacts with other particles through the two-body SW potentials[34].

2.6 Machine learning

Machine learning (ML) is a field in computer science that is rapidly evolving. It aims to develop algorithms and models capable of automatically learning patterns, making predictions, and extracting meaningful insights from empirical data[37], by also implementing statistical models to enable computers to learn and improve from experience without being explicitly programmed.[38].

While there are other types of ML algorithms and techniques beyond supervised, unsupervised, and reinforcement learning, these three categories are among the most commonly used and studied in the field. Each type serves different purposes and is suited to specific problem domains and data characteristics[37].

ML algorithms operate on large datasets, using statistical models and computational techniques to uncover patterns, relationships, and structures within the data. The algorithms analyze the data, identify patterns and trends, and generate models that can be used for various tasks, such as classification, regression, clustering, or anomaly detection[39].

Understanding the distinctions between these types of ML is crucial for selecting appropriate algorithms and methodologies to tackle real-world problems effectively. In this paper, the focus will mainly be on exploring and discussing the principles and applications of unsupervised learning, but an explanation of the most important fundamentals of supervised and reinforcement learning will be given, to better understand the choices made in the experiments later.

2.6.1 Supervised learning

In Supervised learning the algorithm is trained using a labeled data set, to predict outcomes or classify data accurately. This labeled dataset is often called a training dataset and is used by the algorithm to train itself to yield the desired outputs. This is done by feeding the algorithm with input data, it will then spit out an output. This output will be compared to the outputs contained in the training dataset, to validate the output of the algorithm. The model will use these inputs and outputs from the training dataset to make changes to its weights and learn over time to give the correct output that matches the output of the training dataset. The correctness or accuracy is measured using a loss function, and the model will adjust itself until its error is sufficiently minimized[40].

Supervised learning focuses on two types of problems: classification and regression[39].

Classification problems use an algorithm that is able to recognize specific entities within the dataset and attempts to find patterns on how these entities should be labeled. This gives it the ability to accurately assign test data into specific categories. A lot of different types of algorithms has emerged to handle these task, such as linear classifiers, support vector machines (SVM), decision trees, k-nearest neighbor, and random forest[39].

A simple example would be the image classification of animals in pictures. The algorithm is supposed to recognize cats, dogs, and cows from one another. It would receive a dataset containing pictures of the mentioned animals. The pictures of the dog would be labeled as a dog, the pictures

of the cat would be labeled cat, and so on. The algorithm would use this training dataset to learn a function that helps it map the right words (dog, cat, or cow) to the right picture. When fully trained, the algorithm can now be used to make predictions for a new dataset (unseen images of cats, dogs, and cows, that are not labeled). By inputting the image data into the model the algorithm would give out its predicted label as an output.

Regression analysis is used to understand the relationships between independent and dependent variables and is mostly used to make projections such as for sales revenue for a given business. Some popular algorithms are Linear regression, logistical regression, and polynomial regression[39].

Supervised learning is a powerful tool that can be used to classify and make predictions from data, but it has some drawbacks too. It can be time-consuming to train an algorithm, and it might require a lot of data to be able to be sufficient enough to make the right predictions. Datasets with human errors will result in the algorithm learning incorrectly and supervised learning needs some form of a reference (a dataset with the correct output for a given input) to cluster and classify a dataset (unlike unsupervised learning)[39].

2.6.2 Reinforcement learning

Reinforcement learning is quite similar to supervised learning, other than that the algorithm does not get a training dataset. In reinforcement learning the model has to learn by trial and error. To find the desired sequence of outcomes a reward system is often implemented. This helps the algorithm to learn and remember the right steps, to archive the preferable goal[41].

IBM's Watson system which won the game Jeopardy in 2011 is a famous example of a reinforcement learning algorithm. By using this method, the system knew when to do the right moves according to the circumstances [41].

2.6.3 Unsupervised learning

Unsupervised learning uses algorithms to cluster or/and analyze unlabeled datasets. Unlike supervised learning, there is no human intervention, and the algorithms are capable of finding hidden patterns or groupings in the data all by itself[41]. Unsupervised learning is usually deployed to perform one of three tasks: clustering, association, and dimensionality reduction[39]. This paper will only focus on the clustering tasks, as association and dimensionality reduction are not needed to understand the topic of this paper.

Clustering is a technique that groups unlabeled data based on their similarities (or differences). The clustering algorithms will process the raw unlabeled data, into cluster groups depending on their structures or/and patterns. There exist different clustering algorithms, but they can all be divided into different types. Some types are exclusive, overlapping, hierarchical probabilistic, and density clustering[41].

Exclusive clustering is referred to as "hard" clustering, which means that a datapoint is only able to be part of one cluster[39]. One of the most common exclusive cluster algorithms is the k-means clustering algorithm [41] (more about k-means clustering in section 2.6.5).

Overlapping clustering is "soft" clustering, which allows for a datapoint to be part of more than one cluster[39].

Hierarchical clustering is an algorithm that enables the grouping of similar data points into clusters based on their characteristics or proximity to one another. The algorithm is implemented by using either choosing agglomerative or a divisive approach.

Agglomerative is a "bottoms-up" approach[42]. Each data point is treated as an individual cluster. The clusters are iteratively merged based on their similarity, resulting in a hierarchical dendrogram, a tree-like diagram that visualizes the merging (or splitting) of data points at each iteration. [43].

The other method, divisive hierarchical clustering is a "top-down" approach[42]. It is not as popular

as the agglomerative approach, where all the data points start in a single cluster. The cluster is recursively split into smaller subclusters based on dissimilarity. This also forms a dendrogram [44].

The specific characteristics of the dataset and the desired clustering objectives are the prime deciders for which method to choose. Researchers often employ various distance metrics and linkage methods. The four most common methods used to find similarities or dissimilarities are:

- Ward's linkage: Where the sum of squared defines the distance between two clusters after they have merged[42].
- Average linkage: Where the mean distance between each cluster is the defining parameter[42]
- Complete (or maximum) linkage: Defines the maximum distance between two points in each cluster[42]
- Single (or minimum) linkage: Defines the minimum distance between two points in each cluster[42]

Different metrics can be used to calculate these distances, but the Euclidean distance is the most commonly used[39].

Probabilistic clustering is used to help solve density estimation or "soft" clustering problems. Like overlapping clustering, this model also allows a data point to be part of more than one cluster. The main difference is that in probabilistic clustering, the points are clustered based on the likelihood that they belong to a particular distribution[39]. One popular approach for probabilistic clustering is The Gaussian Mixture Model (GMM)[42]. Gaussian Mixture Models are classified as mixture models, which means that they are made up of an unspecified number of probability distribution functions. GMMs are primarily leveraged to determine which Gaussian, or normal, probability distribution a given data point belongs to. If the mean or variance is known, then we can determine which distribution a given data point belongs to. However, in GMMs, these variables are not known, so we assume that a latent, or hidden, variable exists to cluster data points appropriately. While it is not required to use the Expectation-Maximization (EM) algorithm, it is commonly used to estimate the assignment probabilities for a given data point to a particular data cluster[39].

Density-based clustering is a technique that clusters (as the name suggests) based on the density of data points in the feature space, rather than relying on distance or similarity measures. This means that density-based methods can uncover clusters of arbitrary shapes and effectively handle noise in the data[45]. One prominent density clustering algorithm is DBSCAN (Density-Based Spatial Clustering of Applications with Noise). This algorithm groups together neighboring data points within a predetermined distance, and with a minimum number of neighbors. A more detailed explanation of DBSCAN will be given in the section [**"ref**]sec:dbscan. Another notable algorithm is HDBSCAN (Hierarchical Density-Based Spatial Clustering), which extends DBSCAN by creating a hierarchy of clusters and automatically determining the optimal number of clusters[39].

2.6.4 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm that is able to discover clusters of arbitrary shapes and handle noise effectively compared to other popular algorithms such as k-means. This algorithm is able to differentiate between regions of high and low densities, allowing it to adapt to different density variations in the dataset.

DBSCAN is dependent on two variables: Epsilon ϵ and minimum points per cluster MinPts. ϵ is the maximum straight line distance tow points are allowed to have between each other to be considered neighbouring points. MinPts is the minimum number of neighbors required for a point to be considered a core point. Both variabels are set by the user [46].

DBSCAN worsk as follow:

1. The algorithm starts by randomly selecting a point in the dataset. It will then find all the neighbours within a distance ϵ. If the number of neighbors is greater than or equal to MinPts, the point is marked as a core point and becomes the starting point of a new cluster. This process is done for all the points in the data set. Points that do not qualify as core points is called non-core points[46].
2. It then selects one random core point and assigns it as the first cluster. The cluster will then add all core points that are within a distance ϵ into the cluster, the non-core points will be added later[46].
3. It will then use the newly added points and add all core points that are within a distance ϵ into the cluster, and continue this process until it cannot find any core points to add[46].
4. Then all non-core points that are within the distance ϵ of a core point in the cluster will be added to the cluster. When this is done, the first cluster is complete[46].
5. A new random core point that is not in a cluster is then selected, and the process is repeated until all core points are assigned a cluster[46].

All non-core points that were not assigned to a cluster are called outliers or noise points. In the end, most of the points, if not all of them will be assigned to a cluster, those that are not are classified as noise or outliers[46].

DBSCAN has some advantages over other algorithms. It requires minimal knowledge of the domain to determine the values of its input parameters, which is a very important problem for large data sets. It is able to discover arbitrary-shaped clusters compared to other algorithms. Lastly, it has good computational efficiency on large data sets [47].

2.6.5 K-means clustering

K-means clustering is a popular unsupervised machine learning algorithm used for partitioning a given dataset into distinct groups or clusters based on their similarity. The algorithm aims to minimize the within-cluster sum of squares, which measures the squared distances between data points and their assigned cluster centroids [42][44][48].

The k-means algorithm operates in the following manner:

1. Initialization: The algorithm begins by randomly selecting k random data points as the initial set of centroids. k is determined by the user and represents the desired number of clusters[49] as seen in figure 5 a to b.
2. Assignment: Each data point is then assigned to the cluster with the nearest centroid based on a distance metric, commonly Euclidean distance. This step involves calculating the distance between each data point and all cluster centroids and assigning the data point to the cluster with the minimum distance as seen in figure 5 c. distance[49].
3. Update: After the assignment step, the cluster centroids are updated and repositioned (figure 5 d. By recalculating the centroid of every cluster as the mean of all data points assigned to each cluster[49].
4. Iteration: Steps 2 and 3 are repeated iteratively until convergence (no further improvement) or until a maximum number of iterations is reached. Convergence occurs when the cluster assignments no longer change or change minimally between iterations [49], depicted in figure 5 d to f.

The final result of the k-means clustering algorithm is a set of k clusters, where each data point is assigned to one of the clusters based on its proximity to the corresponding centroid[49].

K-means clustering offers several advantages. It is computationally efficient and scalable, making it suitable for large datasets. The algorithm's simplicity and ease of implementation make it widely

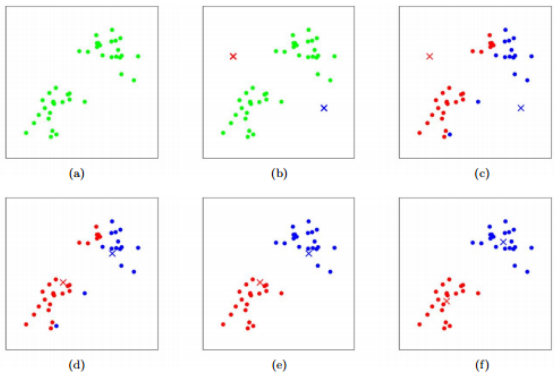


Figure 5: Visualisation of the k means algorithm, for k=2 [48].

accessible and applicable to various domains. K-means clustering is also highly interpretable, as the resulting clusters can be easily visualized and analyzed. Additionally, the algorithm can handle both numerical and categorical data, making it versatile for different types of datasets[39].

However, k-means clustering has some limitations. It requires the user to specify the number of clusters (k) in advance, which can be challenging if the optimal number of clusters is unknown. The algorithm is sensitive to the initial selection of centroids, leading to potential convergence to suboptimal solutions. K-means is also sensitive to outliers, as they can disproportionately influence the centroid calculation and clustering results[42].

2.6.6 Silhouette analysis

Silhouette analysis is a technique used for evaluating the quality of the k-means clustering results. It provides a quantitative measure of how well each data point fits within its assigned cluster, allowing for an assessment of the overall clustering structure[50][51][52].

The silhouette analysis process begins by computing:

- The average distance to other data points within the same cluster denoted as a
- The average distance to data points in the nearest neighboring cluster denoted as b

These distances are typically calculated using a distance metric such as Euclidean distance[50]. The silhouette coefficient, sc, for a data point is then defined as (b - a) divided by the maximum of either a or b, as seen in equation 1.

$$sc = \frac{(b-a)}{max(a,b)} \tag{1}$$

This coefficient ranges from -1 to 1, where values close to 1 indicate that the data point is well-clustered, with a substantial distance to other clusters. Values close to -1 indicate a potential misclassification, as the data point is closer to neighboring clusters than its assigned cluster. A silhouette coefficient close to 0 suggests that the data point resides on or near the decision boundary between clusters[50].

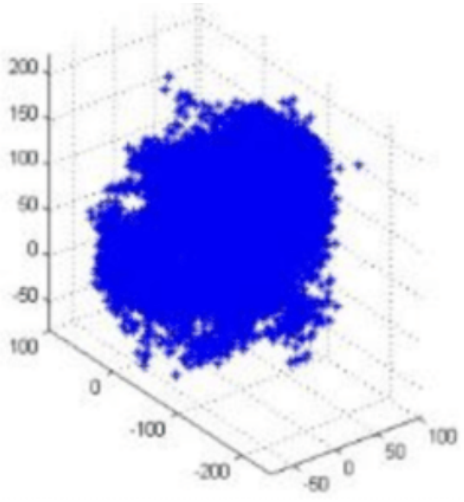
Examining the silhouette coefficients of all data points within a clustering solution gives insights into the overall quality and cohesion of the clusters. A higher average silhouette coefficient across all data points indicates a more appropriate and distinct clustering structure, while a lower average silhouette coefficient suggests potential issues, such as overlapping or poorly separated clusters[50].

Silhouette analysis provides several benefits for k-means clustering. First, it offers a numerical measure to evaluate the performance of k-means clustering, enabling researchers and practitioners

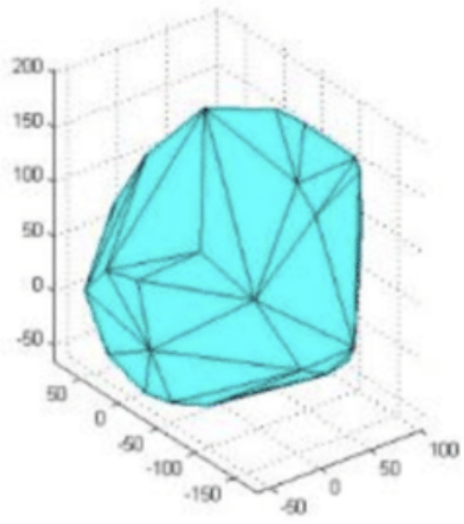
to compare different clustering solutions and select the one that yields the highest silhouette coefficient. Second, it helps in determining the optimal number of clusters, k, by evaluating the silhouette coefficients for different values of k and selecting the one that maximizes the overall cohesion and separation of data points. Lastly, silhouette analysis aids in identifying individual data points that may be misclassified or located near the decision boundaries, providing insights for potential cluster refinement or adjustments[51].

2.7 Convex Hull

The convex hull is the smallest convex polyhedron enclosing a given set of points [53]. In three-dimensional space, the convex hull encompasses all the points, forming a solid object with flat faces and straight edges [54] as seen in figure 6



(a) Point of view 1.



(d) Convex hull 1.

Figure 6: (a) The point cluster turned into a convexhull (d) [55].

Computing the convex hull in 3D involves determining the vertices and faces that define the

boundary of the polyhedron. Several algorithms have been developed to calculate the convex hull efficiently, but this paper will mainly focus on the QuickHull algorithm [56], only a quick overview will be given.

The process of computing the 3D convex hull typically involves the following steps:

Input: A set of points in 3d are received and represent the objects to be enclosed by the convex hull [56].

Convex hull construction: A face of hulls will be constructed by iterative selecting a set of points. This process continues until all points are enclosed or no more points can be added to the hull [56].

Face determination: The faces are determined by considering the orientation of each point with respect to the current set of faces. If a point lies inside the face, it gets removed. If a point lies outside, new faces are added to enclose the point [56].

Vertex determination: Lastly the extreme point on the faces are used to identify the vertices of the convex hull. These extreme points form the corners of each face [56].

The result is a convex hull in 3D as depicted in 6. The convex hull provides an enclosed structure for the given set of points [56].

The 3D convex hull has various applications in computational geometry, computer graphics, collision detection, robotics, and computer-aided design (CAD). It plays a crucial role in tasks such as finding the collision-free path for robots, calculating the enclosing volume of 3D objects, or generating 3D mesh models from point clouds [56].

2.8 Simple euclidean geometry

To understand some of the geometric calculations made in this paper, a simple explanation will be given for the most critical ones.

2.8.1 Euclidean distance

Euclidean distance is a metric used to find the shortest distance between two points (that is a straight line) in Euclidean space. It is a fundamental concept that is widely used in mathematics and computer science [57].

In Euclidean 3d-space, the distance, d, between a point $A = (x_1, y_1, z_1)$ and $B = (x_2, y_2, z_2)$

can be computed as the square root of the sum of the squared differences between their corresponding coordinates[57] as shown in formula 2:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{2}$$

A benefit of Euclidean distance is that its computationally efficient. The calculation involves straightforward arithmetic operations, such as subtraction and squaring, followed by a square root computation. These operations are fast and easily implemented in computer programs, making Euclidean distance an efficient distance measure for large datasets and real-time applications.

Euclidean distance also possesses several desirable mathematical properties. It satisfies the requirements of a metric, including non-negativity, symmetry, and triangle inequality [58]. These properties ensure that Euclidean distance provides consistent and reliable measures of separation between points, allowing for meaningful comparisons and reliable analysis.

Furthermore, Euclidean distance has been extensively studied and analyzed, leading to the development of various techniques and algorithms that leverage its properties. It serves as a foundation for numerous algorithms and methods, such as k-means clustering[42]. The availability of these estab-

lished techniques enhances the practical utility of Euclidean distance and facilitates its integration into existing frameworks and workflows.

Finally, Euclidean distance has broad applicability across different domains. It is particularly useful in spatial analysis, image processing, machine learning, and pattern recognition tasks[59]. Its ability to capture the straight-line distance between points makes it suitable for scenarios where the spatial arrangement or similarity between objects needs to be measured.

2.8.2 Angle between two points and the origin in 3D

To find the angle between two points and the origin, one can define two vectors. One that spans from the origin to point $a = (a_1, a2, a3)$, and one that spans from the origin to point $b = (b_1, b_2, b_3)$. To find the angle between these two vectors in a three-dimensional space, the dot product and trigonometric functions can be utilized.

Let's consider two vectors, A and B, with their respective components: $A = [a_1, a_2, a_3]$ and $B = [b_1, b_2, b_3]$.

Calculate the dot product of the two vectors using the formula:

$$A \cdot B = a_1 * b_1 + a_2 * b_2 + a_3 * b_3 \tag{3}$$

Next, find the magnitudes (lengths) of the vectors A and B using the formula:

$$||A|| = \sqrt{a_1^2 + a_2^2 + a_3^2} \tag{4}$$

$$||B|| = \sqrt{b_1^2 + b_2^2 + b_3^2} \tag{5}$$

Then by using the dot product and magnitudes, gives us the angle between the two vectors in radians [60]:

$$\theta = cos^{-1}((A \cdot B)/(||A|| * ||B||)) \tag{6}$$

To convert the angle from radians to degrees, the following formula can be used:

$$\theta_{degrees} = (\theta * 180)/\phi \tag{7}$$

The angle between two vectors can range from 0 to 180 degrees (or 0 to π radians).

It is important to be careful of some special cases. For instance, if either of the vectors A or B has zero magnitudes (i.e., it is the zero vector), the angle calculation is not meaningful, as the angle between a vector and the zero vector is undefined.

2.8.3 Shifting: Rotating a vector

There are different ways to rotate a vector in 3D space by an angle θ, but this paper will only focus on how to do this by using a rotation matrix, to rotate around the z-axis.

Using rotation matrices, assume the axis of rotation is represented by a unit vector: $U = (u_1, u_2, u_3)$.

One can construct a 3x3 rotation matrix R(θ) that represents the rotation around the axis by theta degrees. The rotation matrix is defined as:

$$\mathrm{R}= \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

To rotate the vector A, take the dot product with the rotation matrix, R(θ):

$$A_{rotated} = R(\theta) \cdot A \tag{8}$$

The resulting vector $A_{rotated}$ represents the rotated version of vector A by an angle θ around the z-axis [61].

3 Method

3.1 Molecular dynamics simulation

The MD simulation was conducted using the LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) package. The system consisted of 23,058 molecules, with 216 methane molecules and the remaining molecules being water molecules. The simulation duration was 50,000 picoseconds (ps), with a timestep of 500 ps. The simulation system was composed of three parts: a pre-built hexagonal ice lattice and a methane hydrate lattice on each end, with water molecules in the middle (as depicted in the figure 7). The trajectory file can be found in the attachment.

In the simulation, the system employed a periodic boundary condition, allowing for atomistic interactions between the top and bottom layers of the simulation box. The interactions between water molecules were described using the monatomic water model mW[12]. This model utilizes the three-body Stillinger-Weber (SW) potential to account for angle-dependent interatomic interactions in water, specifically capturing hydrogen bonding.

The simulation was performed at a temperature of 230 Kelvin (K). Throughout the simulation, various system properties were monitored and recorded. The final system snapshots at the conclusion of the simulation were collected and stored in a data file for further analysis and examination (figure 8).

3.2 Python

To perform the rest of the methods explained, various Python scripts where made to perform computational analysis. A README.txt file is attached to the attachments and describes the necessary procedures needed to run the scripts.

3.3 Data preparation using boundary conditions

The provided molecular dynamics (MD) simulation data contained essential information, which included the time step, atom count, simulation box dimensions, and the positional and velocity attributes (x, y, z) of each atom. A snip of this can be seen in figure 8.

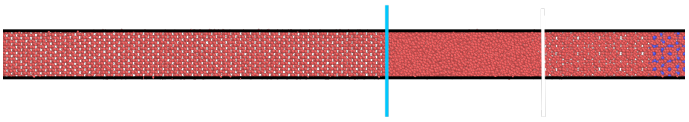
As seen the data was written to a .lammpstrj.txt file (LAMMPS trajectory file). That contained the mentioned information for 101 timeframes. Each frame beeing 500ns.

During the simulation, it was observed that a considerable number of atoms were in close proximity to the ends of the simulation box exhibiting teleportation behavior, rapidly transitioning from the uppermost region to the lowermost region (depicted in figure 9). This phenomenon of teleportation posed challenges in accurately analyzing individual atoms and their local environments. The environment is the neighboring atoms. Suddenly getting all new neighbors, made it impossible to compare a specific atom and how its neighboring atoms moved compared to each other when all the neighbors suddenly changed (see figure 11c). To address this issue, boundary conditions were implemented in the y and z directions of the simulation box.

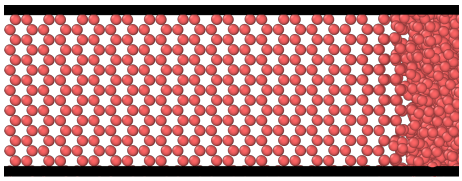
Specifically, the boundaries were defined as $< 4.5, 32 >$ in the y-direction and $< 7, 30 >$ in the z-direction. Consequently, the simulation box effectively contracted by approximately four atoms on each side. By imposing these limitations, the teleportation phenomenon was successfully eliminated, facilitating reliable analysis of atom trajectories and their interactions.

Furthermore, the simulation file was partitioned into 101 distinct files, with each file containing MD simulation data pertaining to a specific time step. To see the code for this section, please refer to the script provided in appendix A.

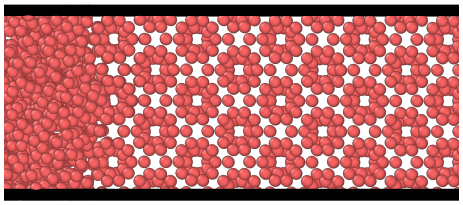
(a) The system in its initial phase



(b) A close-up of the system in its initial phase, with the hexagonal ice to the left of the blue line, and methane hydrate to the right of the white line. The water is between the blue and white lines.



(c) A close-up of hexagonal ice to the left and liquid water to the right



(d) A close-up of hydrate to the right and liquid water to the left

Figure 7: OVITO snapshot of the first frame (the initial phase of the MD simulation).

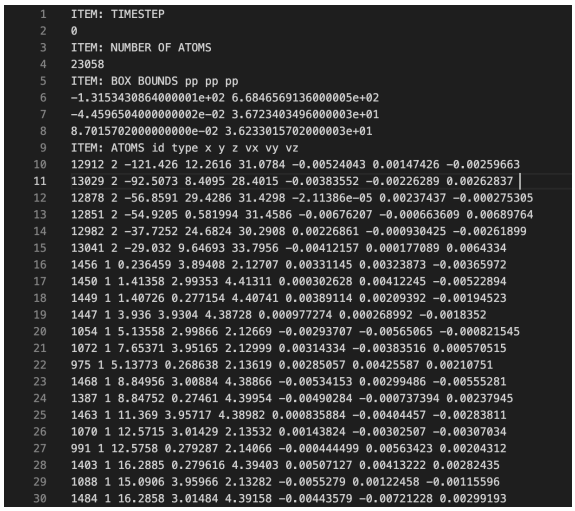


Figure 8: A snippet of the lammpstrj.txt. file used in this project. As seen the first and second line defines the timestep, the second and third the number of atoms, line five to eight defines the simulation box size, and the lines after defines the positions and velocities for a specific atom with a specific id and atom type (denoted by a number). The snip only shows the beginning of the file, this being the first frame. The other frames are written with the same setup in this same file.

3.4 Water, Ice and Hydrate differentiation with DBSCAN

After successfully addressing the teleportation issue, the subsequent task involved identifying the atoms being a part of liquid water, and not being a part of ice of hydrate

To accomplish this, a Python script was developed, utilizing the widely-used scikit-learn library, to implement the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm (see appendix). The script processed the simulation data, applying the DBSCAN algorithm to detect clusters corresponding to water, ice, and hydrate. Following the completion of the clustering process, the points associated with the second largest cluster, representing the water molecules in the water component, were extracted and saved in a separate file. This procedure was repeated for all simulation frames. The cluster containing water molecules was of particular interest, as it was assumed to be the only place to encompass potential intermediate structures during ice and hydrate nucleation.

The Python script for this part can be found in appendix B for experiment 1 and appendix F for experiments 2 and 3.

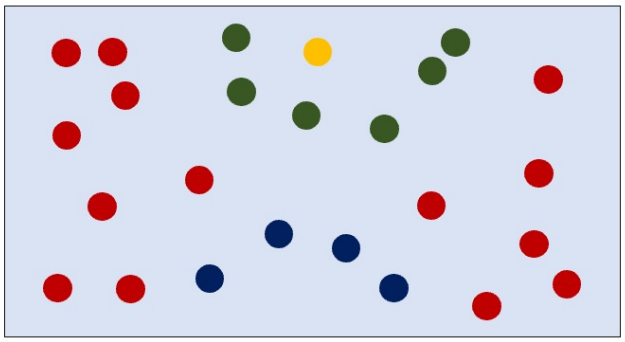
After identifying the molecules being a part of liquid water, three experiments were conducted to investigate and identify intermediate structures during the formation of ice and hydrate.

3.5 Experiment 1

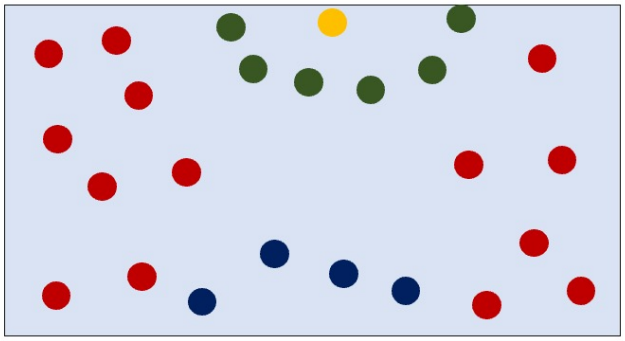
The initial method entailed employing the DBSCAN algorithm once more to explore potential intermediate structures. This involved investigating whether the clusters that corresponded to these structures exhibited any similarities or common characteristics.

3.5.1 Timeframe differences

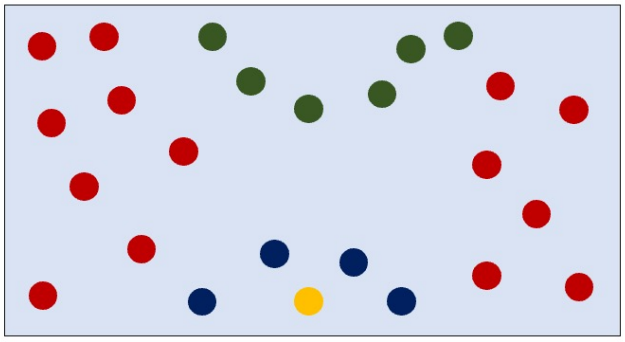
During the MD simulation, a notable observation was that each water molecule remained either in the liquid state throughout the entire simulation or became part of the ice or hydrate structure. With this understanding, an assumption was made that if a water molecule appeared within the water cluster in frame N (where N represents the frame number within the range of 1 to 101) but



(a) Different atoms in the simulation box.

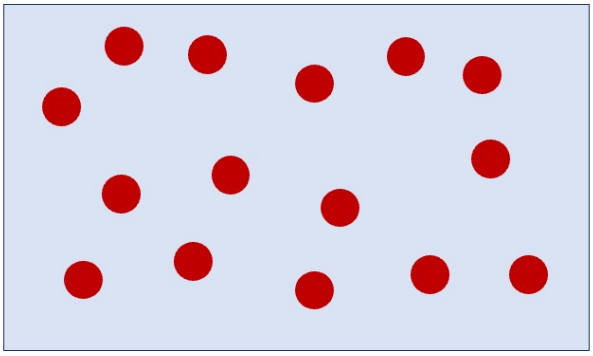


(b) As time goes all the atoms will move around.

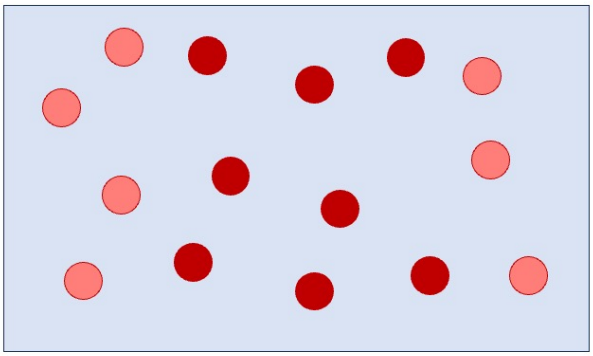


(c) Some of the atoms will "teleport" from the top to bottom or vice versa. Here you see that the yellow atom has teleported from the top to the bottom. In doing so it has gained new neighbours.

Figure 9: This figure illustrates how the teleportation phenomena take place, and how an atom (the yellow) goes from one neighborhood (the green atoms) to a new neighborhood (the blue atoms) when teleporting.



(a) Atoms in the liquid water region in frame N



(b) Atoms in the liquid water region in frame N+1. The light-shaded atoms are atoms that have disappeared from the liquid water region compared to frame N. The dark-shaded ones are the ones that remain. The light-shaded atoms will then be saved on a .txt file for further analysis.

Figure 10: This figure illustrates how some atoms disappear between frame N and frame N+1 in the liquid water region.

was absent in frame N+1, it likely underwent a transformation into ice, hydrate, or an intermediate structure. Hence, comparisons were made between frame N and frame N+1 to identify potential intermediate structures.

By tracking the absence of water molecules within the water cluster in frame N+1 compared to frame N, the objective was to identify instances where transitions occurred, indicating the formation of ice, hydrate, or intermediate structures. The water molecules that were absent in frame N+1 were then extracted and saved in a new file. This process was performed for all time frames of the simulation. The process is depicted in figure 10b.

The Python code used to implement this method can be seen in appendix B.

3.5.2 DBSCAN again

Subsequent to identifying the absence of water molecules, the DBSCAN algorithm was once again utilized to conduct a cluster analysis on this newly generated dataset. In this analysis, a value of 4 was chosen as the ϵ (epsilon) parameter, while the minimum number of points required to form a cluster (MPC) was set to 2. Additionally, in order to enhance the selection process, a criterion was implemented to exclude clusters comprising six or more molecules. This procedure was repeated for each frame interval, and the resulting clustering information was saved in a text file. To see the code please refer to appendix C.

3.5.3 Analyse the data

To analyze the data, two different approaches were employed to identify similarities. The first approach focused on examining the average bond distance between each molecule and the total bond distance within the molecule. Specifically, only small molecular bonds consisting of 3 to 5 molecules were considered (to see the code please refer to appendix D). Subsequently, k-means clustering was applied to identify similarities within this modified dataset (see appendix N for the code). The silhouette analysis technique was utilized to determine the optimal number of clusters. For each cluster size representing 3, 4, and 5 molecules, a representative sample was chosen. These samples were selected as the ones closest to the centroid of each cluster.

The second approach involved assessing the volume encompassed by each cluster and its corresponding density. The volume was determined by computing the convex hull of each cluster and subsequently calculating the volume of the convex hull. It is worth noting that a cluster needed to contain at least 4 atoms to construct a convex hull; therefore, clusters with 3 or fewer molecules were excluded from this analysis. To compute the density, the number of molecules within each cluster was divided by its convex hull volume. See appendix E for the Python script. Following this, k-means clustering was once again performed, and the silhouette analysis was utilized to identify the optimal number of clusters (See appendix N for code). Similarly, a representative sample was chosen from each cluster, specifically the samples closest to the centroid of each cluster.

3.6 Experiment 2

The second method employed for identifying intermediate structures focused on analyzing the rotational and translational movement of molecules within each timestep.

To evaluate the rotational movement, the angular displacement of each molecule was calculated by measuring the change in its orientation over the course of the timestep. This information was then used to determine if any molecules exhibited significant rotational motion, which could indicate the formation of intermediate structures.

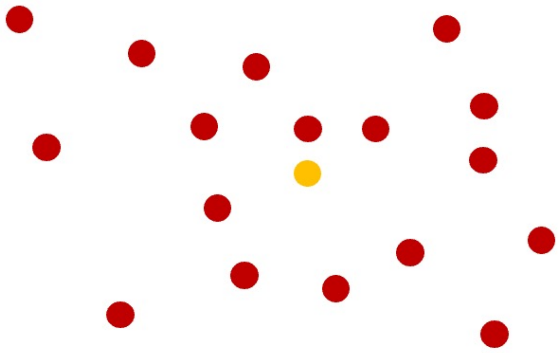
In addition, the translational movement of the molecules was examined by analyzing their displacements during the timestep. By measuring the distance traveled by each molecule and comparing it to a threshold value, it was possible to identify molecules that displayed considerable translational movement.

By combining the analysis of rotational and translational movement, potential intermediate structures could be inferred based on significant changes in both aspects of molecular motion. Further investigations, such as clustering or statistical analyses, could then be conducted to validate and explore these identified structures in more detail.

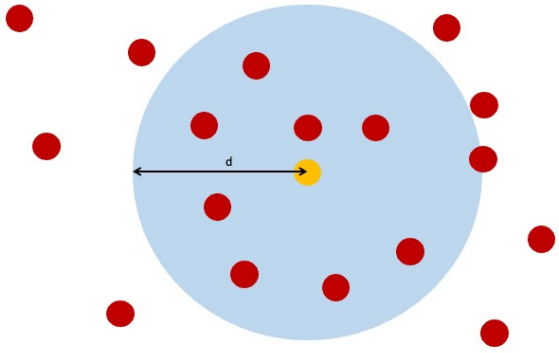
3.6.1 Neihgbours

The initial step involved identifying the neighboring molecules for each center molecule within the water cluster. A neighboring molecule was defined as one located within a distance of d from the center molecule, where d was set to 8 Angstroms. The process is shown in figure 11. The Euclidean distance was computed to measure the distance between the center molecule and potential neighboring molecules. This process was performed for all molecules within the water cluster identified in each frame of the simulation. The results of this analysis were recorded in a text file. See appendix G for the code.

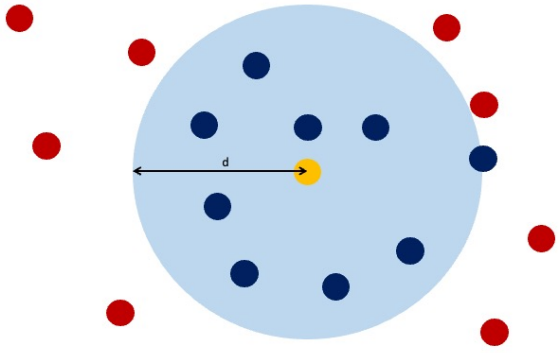
To facilitate the calculation of molecule displacement or shift during each frame, a new coordinate system was established for each neighborhood. In this new coordinate system, the center atom was positioned at the origin (0, 0, 0), and all other atoms were located relative to this center. To determine the new coordinates of the neighbors, the original coordinates of the center atom were subtracted from the original coordinates of the neighboring molecules. The resulting new coordinates were then saved in a text file. The code for this can be found in appendix H



(a) A center atom is picked, depicted as the yellow atom



(b) The distances from the atoms to the center atom are calculated using the Euclidean distance



(c) The atoms within a distance d, is selected as the neighbors (depicted in blue) for the center atom.

Figure 11: This figure depicts how the neighboring atoms within a distance d are found.

3.6.2 Rotational shift

Once the new coordinate system was established for each neighborhood, the shift of each atom was computed during each frame. This involved comparing the rotational and translational displacements between frame N and frame N+1 for the corresponding neighbors in each neighborhood.

To perform this calculation, a neighborhood was selected in frame N, and the corresponding neighborhood in frame N+1 was identified. Neighbors that did not have the same ID in both frames were discarded. Subsequently, a random atom was chosen, and the rotational displacement, represented by the angle alpha, was computed for this atom. The entire neighborhood was then shifted back based on their positions in frame N+1 using the calculated angle alpha. The displacement (Euclidean distance) from the atom's position in frame N+1 to its new position after the shift was then determined. Additionally, the (Euclidean) distance from each atom to the center in frame N+1 was calculated, and atoms with a distance greater than 4 were discarded. The average distance from the center to the atoms was computed, and the displacement lengths were summed for the remaining atoms. The entire process is visualized in figure 12 and figure 13.

Appendix I contains the code for this section

Using this modified data, k-means clustering was performed to identify similarities among the neighborhoods. The k-means algorithm utilized the average distance to the center for each atom in a neighborhood and the total displacements for all the atoms in each neighborhood. The silhouette analysis technique was employed to determine the optimal number of clusters in the k-means clustering results.

See appendix N for the k-means clustering code.

3.7 Experiment 3

In this third and final method, the concept of angular displacement was once again utilized to detect intermediate structures. The angular displacement of each molecule was compared to a reference value, denoted as theta, which represents the average angular value for hexagonal ice or hydrate structures.

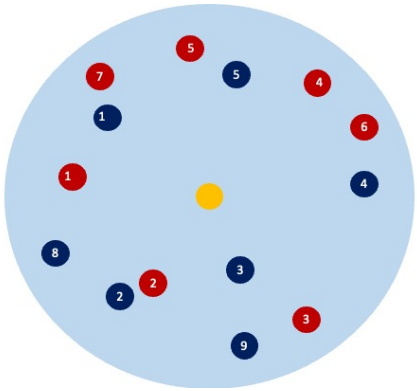
3.7.1 Neighbours

In this experiment, the neighboring atoms were also identified and placed in a new coordinate system, similar to the approach used for the neighborhoods in Experiment 2 (as described in section 3.6.1). The same procedure of establishing a new coordinate system with the center atom at the origin and determining the coordinates of the neighboring atoms relative to the center was applied to these neighboring atoms as well. This step allowed for a consistent reference frame and facilitated further analysis and comparisons between the neighboring atoms.

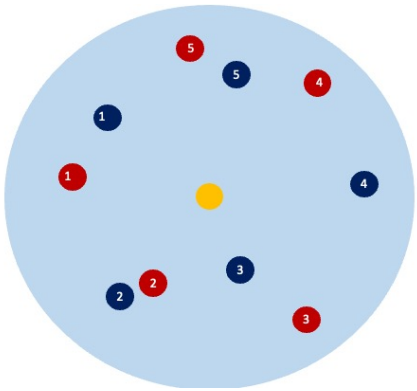
3.7.2 Angular displacement

With the new coordinate system established, the angular displacement was calculated using the same method as described in section 3.6.2. However, in this case, the angular displacement was calculated for all the atoms within each neighborhood. Subsequently, atoms with an angular displacement less than the reference value theta were discarded from each neighborhood.

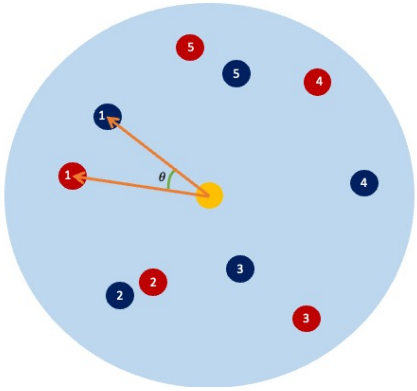
By comparing the angular displacement of each atom to the reference value θ, molecules that exhibited angular patterns significantly different from those observed in the desired intermediate structures were filtered out as depicted in figure 14. This step aimed to retain only the atoms within each neighborhood that displayed angular displacements indicative of potential intermediate structures, resembling those observed in hexagonal ice or hydrate formations.



(a) A center atom is picked, depicted as the yellow atom

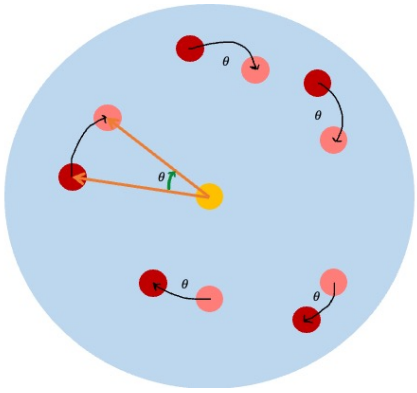


(b) A center atom (yellow) and its neighbors in frame N (blue) and its neighbors in frame N+1 (red), with their id's.

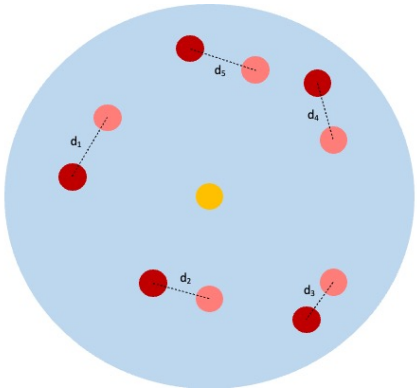


(c) A random atom is picked and the angular displacement is calculated for this atom. The angular displacement is calculated as the angle between two vectors (the orange ones) in a 3D space. Each vector spans from the center atom to its matching atom in frame N and frame N+1.

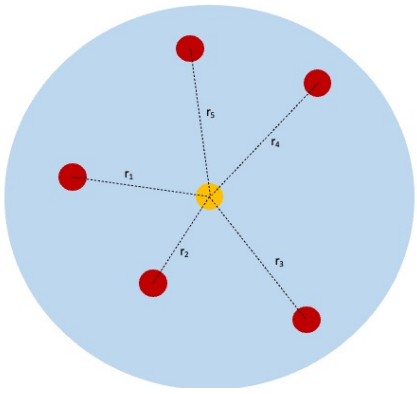
Figure 12: Process of calculating the angular displacement of an atom.



(a) Using the angle, theta, found in 12c, all the atoms are shifted back. They are shifted by rotating them at an angle theta on the Z-axis. Their new position is depicted in pink, their previous (frame N+1) position is depicted in red. The shifting was performed by using the steps and rotational matrix described in section 2.10.3 described.

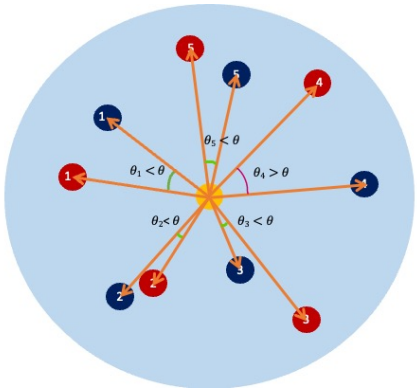


(b) The distance between their frame N+1 and the new rotated position is calculated for all atoms, depicted as d.

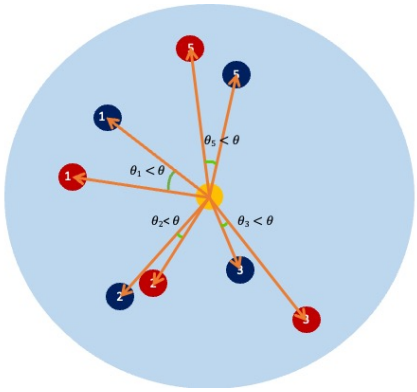


(c) Finally their average distance to the center is calculated from their frame N+1 position, by summing all distances and dividing it by the number of neighbors remaining.

Figure 13: This figure showcases how the atoms are shifted back, how the distance they have moved after getting shifted is calculated, and how the neighbour's average distance to the center atom is calculated.



(a) The angular displacement of all the neighbors are calculated



(b) The molecules with an angular displacement bigger than θ is discarded.

Figure 14: This figure depicts how the neighbors with an angular displacement bigger than θ are discarded.

The Python script for this process can be found in appendix K

This filtering process allowed for the selection of atoms with notable angular displacements, potentially representing the formation of intermediate structures.

Before the k means clustering a volume analysis was performed as in experiment 1 (see section 3.5.3 for more details), and an average angle and average bond length analysis was performed on the structures. The average angle and average bond length analysis were done by computing the average displacement angle for each neighbor, and calculating the average distance from each neighbor to the center atom. This can be seen in appendix L(average angle and average bond length analysis) and appendix M(volume analysis).

Finally, the k-means clustering would be performed. The Python script for the k means analysis could be found in appendix N

3.8 Final result analysis

After the k means analysis the structures closest to the centroids were picked out and plotted on a 3D graph. They were inspected and compared to each other to see if there were any similarities between one another, or if they showed similarities to ice and methane hydrate structures. Furthermore, the molecules in the structures analyzed in the original trajectory file (using OVITO) to see if these intermediate structures would transform to ice or hydrate during the simulations, if they did they would be classified as a potential intermediate structure, if not they would be classified as being liquid water.

3.9 Life cycle

The last thing that was done, was to look a the behavior of intermediate structures throughout the entire simulation. The molecules that showed the most promising intermediate structures were selected.

The life cycle analysis was done by selecting each molecule from the intermediate structure, then looking at all molecules that were within a distance of 3.5 Angstroms of these molecules (interacting molecules) by using the original trajectory file. Then a new trajectory file containing only the molecules from the intermediate structure and the interacting molecules was made. It was then simulated using OVITO and examined visually for features of interest.

The code for the life cycle analysis can be found in appendix O

4 Results

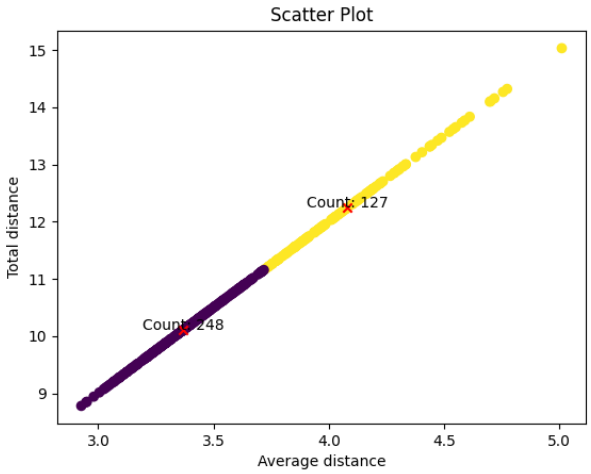
In this section, the results (potential intermediate structures) obtained from each method used in the research will be presented. These structures were carefully examined and analyzed to gain a deeper understanding of their characteristics. The purpose of this section is to provide a clear and comprehensive overview of the findings.

4.1 Experiment 1

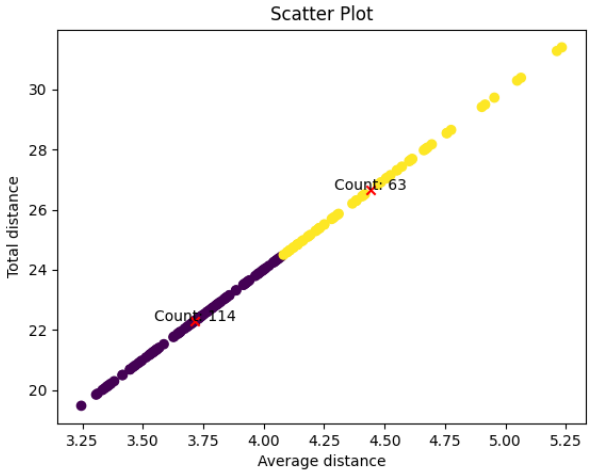
4.1.1 Average distance and total bond length analysis

This simple method involved the identification of a comprehensive set of 1239 potential intermediate structures. It is worth noting that the size of these structures, as measured by the number of atoms they comprised, exhibited a considerable range, spanning from a minimum of 3 atoms to a maximum of 75 atoms.

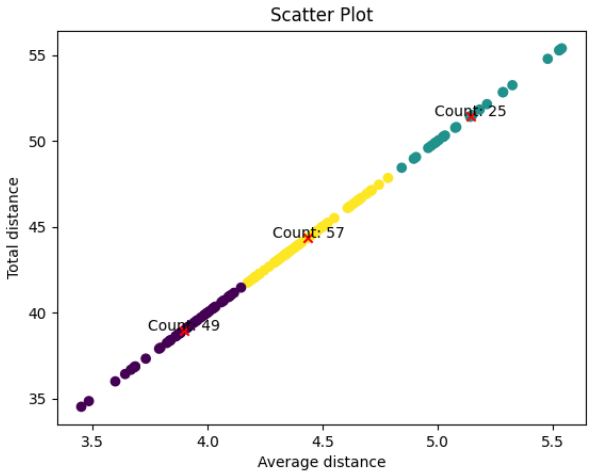
To facilitate a more focused investigation, as mentioned in section 3.4 a selection criterion was applied to isolate and analyze structures consisting solely of 3, 4, and 5 atoms. With the power of k-means clustering in combination with silhouette analysis, these similar structures were clustered together as seen in figure 15. Furthermore, one sample/structure was chosen from each cluster as mentioned in section 3.4.



(a) Clusters with centroid for structure with size 3

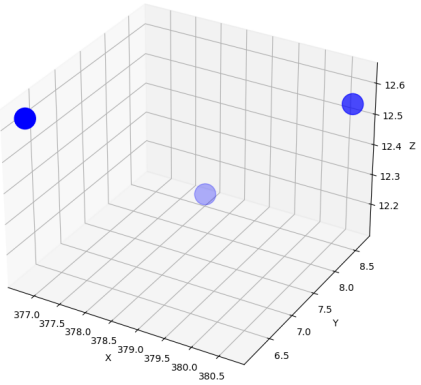


(b) Clusters with centroid for structure with size 4

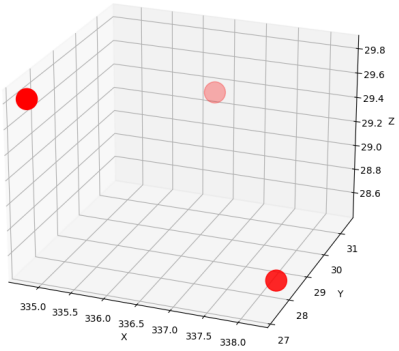


(c) Clusters with centroid for structure with size 5

Figure 15: Resulting clusters after k-means clustering, with their respective centeroids



(a) Potential intermediate structure from the first cluster.



(b) Potential intermediate structure from the second cluster.

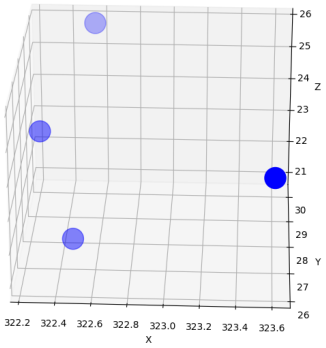
Figure 16: Potential intermediate structures of size 3.

Finally, a close look at each selected structure yields potential intermediate structures. Starting with the size 3 intermediate structures visualized in figure 16.

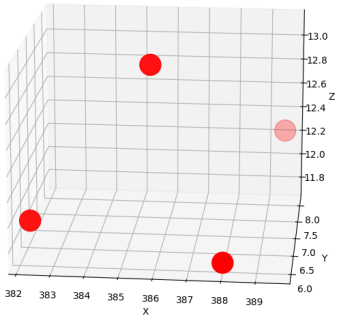
Upon further examination, it appears that the potential intermediate structures consisting of 3 atoms do not exhibit significant distinguishing characteristics amongst themselves. These structures, forming a triangular configuration in the plane, share similar bonding patterns and spatial arrangements. However, by considering their respective positions within the simulation and their proximity to the ice structure and hydrate, an interesting observation emerges.

Among the 3-atom structures, the one depicted in figure 16b is found in close proximity to the ice structure. This particular positioning suggests that this structure may indeed serve as an intermediate structure. The proximity to the ice structure implies a potential relationship or interaction between the two, indicating a potential role of this structure in the transformation or progression toward the formation of ice.

Upon scrutinizing the structures comprising 4 and 5 atoms, a more intricate and intriguing set of results emerges, providing valuable insights into the intermediate stages of the system under investigation. Let's delve into these findings in more detail.



(a) Potential intermediate structure from the first cluster.



(b) Potential intermediate structure from the second cluster.

Figure 17: Potential intermediate structures of size 4.

A closer examination of the potential intermediate structures with 4 atoms (visualized in figure 17) reveals interesting and distinct characteristics between two specific structures.

The intermediate structure in 17b (the red one) exhibits a wider and more flattened/square shape, setting it apart from the other structures. This distinctive morphology suggests that it may possess specific bonding patterns or spatial arrangements that differentiate it from the rest. Additionally, its flattened shape bears a resemblance to certain regions of a hexagonal ice structure, implying a potential connection between this structure and ice formation.

On the other hand, the structure in figure 17a forms a Y shape with its bonds, as seen in figure 18. This Y-shaped configuration shares a resemblance with specific segments of a hexagonal ice structure. This striking similarity strengthens the hypothesis that this structure might indeed serve as an intermediate structure for ice formation.

Furthermore, when considering the placement of both structures relative to the ice and hydrate structures within their respective frames, an intriguing pattern emerges. The structure with the y-bonds appears to be closely associated with the ice structure, indicating a potential intermediary role in the ice formation process. This observation further supports the hypothesis that the y-bond

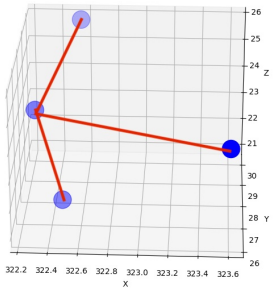


Figure 18: The red lines indicate the bonds between the molecules. One molecule is bonded to all the other molecules, making a "Y" shape with the bonds. This particular intermediate structure is the one depicted in figure 17a

structure represents an intermediate stage during the transition from ice to hydrate.

The analysis of the structures containing 5 atoms presents the most intriguing and diverse set of results thus far (figure 19). These structures exhibit distinctive characteristics, and their placement relative to ice and hydrate structures provides valuable insights into their potential roles as intermediates.

The first structure, depicted in blue (figure 19a), stands out due to its compactness and resemblance to a portion of a hexagonal ice structure. This compact and hexagonal-like shape suggests a potential connection to ice formation. Its similarity to a known ice structure supports the hypothesis that this structure may serve as an intermediate during the water-to-ice transformation.

The second structure, represented in black (figure 19b), displays an "arrow" structure, elongating it. This unique arrangement differentiates it from the rest and hints at possible distinct bonding patterns. However, upon considering the placement of these structures relative to ice and hydrate, no promising results emerge, limiting the potential significance of further analysis.

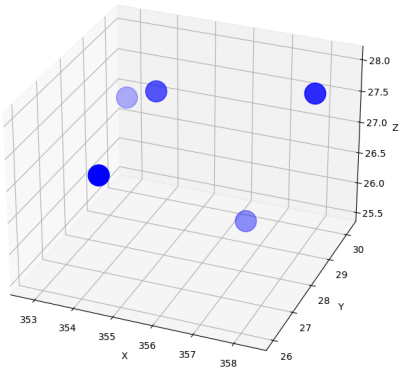
The last structure, in red (figure 19c), also exhibits an elongated structure. While visually intriguing, the analysis of its placement relative to ice and hydrate structures does not either yield any promising insights or correlations.

Given the lack of significant findings regarding the placement of these 5-atom structures in relation to ice and hydrate, it may be prudent to halt further analysis of these structures for now. However, it's important to note that this decision should be revisited if new information or insights emerge in future investigations.

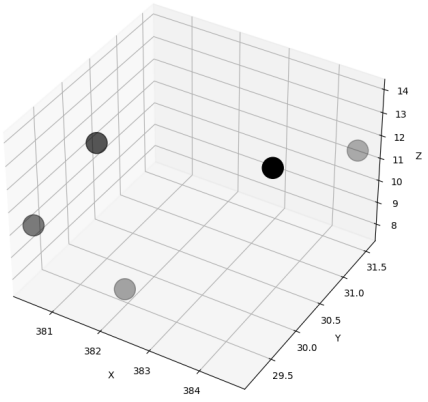
4.1.2 Convex hull

Through the implementation of the convex hull method, a total of 455 potential intermediate structures were identified. These structures were meticulously analyzed to gain further insights into their characteristics. For a more focused investigation, the selection was narrowed down to structures containing 4 and 5 atoms. With the help of k-means clustering patterns/similarities in the structures were determined. The resulting clusters can be seen in figure 20.

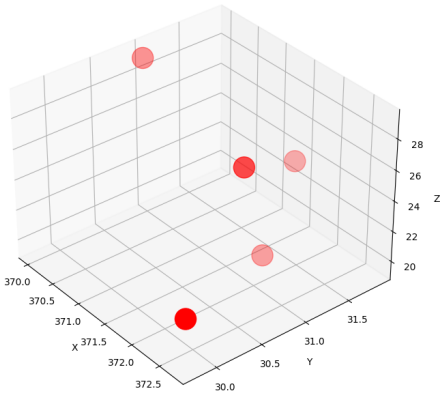
Plotting the volume of the structures on one axis and the density on another axis allows for the examination of their size and compactness. The volume represents the spatial extent occupied by the structure, while the density indicates the mass or number of atoms contained within a given volume. Analyzing these plots with the k-means clustering, yields trends and correlations of the 4 and 5-atom intermediate structures.



(a) Potential intermediate structure from the first cluster.

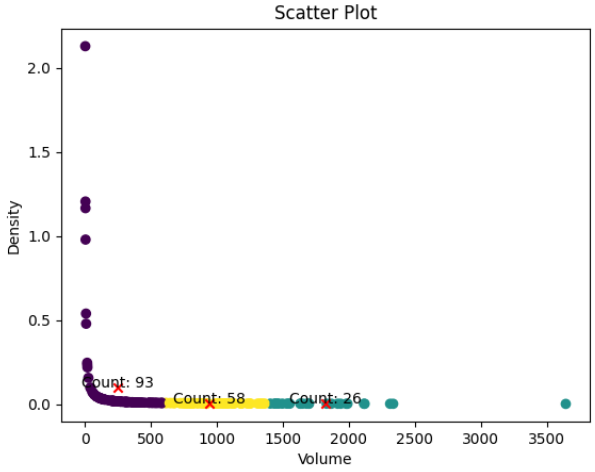


(b) Potential intermediate structure from the second cluster.

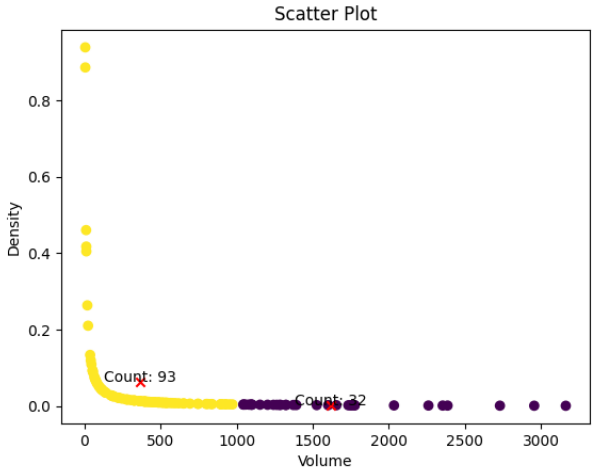


(c) Potential intermediate structure from the third cluster.

Figure 19: Potential intermediate structures of size 5.



(a) k-means clustering result for structure with size 4



(b) k-means clustering result for structure with size 5

Figure 20: Resulting clusters after k-means clustering, with their respective centeroids

By selecting the structures closest to each centroid, three distinct structures within the set of 4-atom intermediates were identified. These structures exhibit unique characteristics and proximity to their respective centroids, highlighting their potential significance. These structures can be seen in figure 21.

Upon closer examination of the second structure (black) (figure 21b), we observe a distinctive Y-shaped bond arrangement. This Y-shaped structure stands out compared to the other two structures (in figure 21), which exhibit more flat and trapezoid-like configurations.

The presence of the Y-shaped bond pattern in the black structure is intriguing and reminiscent of certain segments found in a hexagonal ice structure. This resemblance suggests a potential connection between the black structure and the formation of ice. Additionally, when considering the placement of these structures relative to ice, it becomes evident that the Y-shaped structure is the only one in close proximity to the ice structure.

The proximity of the Y-shaped structure to the ice structure indicates its potential role as an intermediate structure specifically related to ice formation. This finding aligns with the hypothesis that this particular structure may play a crucial role in the transition from water to ice.

Upon analyzing the structures containing 5 atoms (figure 22), we observe two very different structures that do not seem to resemble any ice or hydrate structures. However, when considering the positions of these structures relative to ice and hydrates, a compelling observation emerges.

Comparing the placement of these structures in relation to ice and hydrates, it becomes apparent that the black structure is in close proximity to hydrate structures. This observation suggests that the black structure may serve as an intermediate structure specifically involved in the formation of hydrates.

4.2 Experiment 2

A total of 29623 potential intermediate structures were found. As in experiment 1, only the intermediates with a size of 3, 4, and 5 were analyzed. The k means clustering yielded the results shown in figure 23.

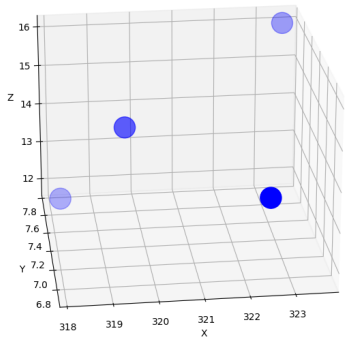
The size 3 structures seen in figure 24 are again very different and do not show any particularly notable features. Indicating that these structures are too simple to analyze properly by the purposed methods in this paper.

For the size 4 intermediates, the Y bond structure is again found as seen in figure 25b, figure 26a, and figure 26c. However, this time the Y bond structures are more likely to become a hydrate structure. All three are located close to the hydrates during the simulation.

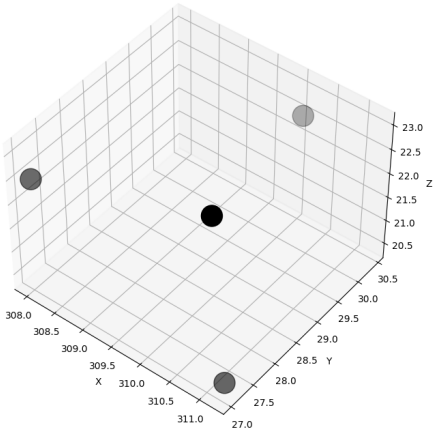
The blue (figure 25a) and cyan (figure 26c) structure also look like Y bond structures, but never turns into ice nor a hydrate during the simulation, which weakens the idea that these structures are intermediate structures.

The rest of the size 4 structures do not show any significant or reacquiring patterns or similarities to hexagonal ice or methane hydrate.

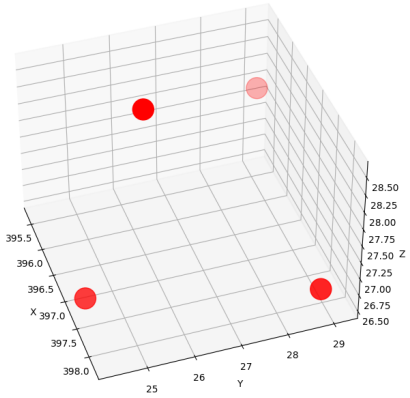
For the size 5 structures, all form different kinds of structures (as seen in figure 27), with hard-to-distinguish features. The black structure (figure 27b) is the only structure that turns into ice during the simulation. The other two size 5 structures are liquid water throughout the entire simulation.



(a) Potential intermediate structure from the first cluster.

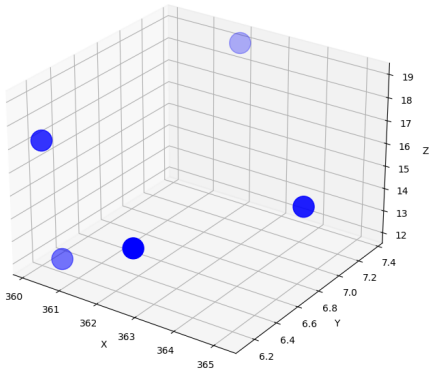


(b) Potential intermediate structure from the second cluster.

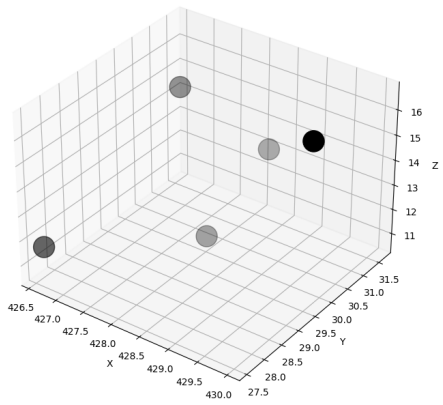


(c) Potential intermediate structure from the third cluster.

Figure 21: Potential intermediate structures of size 4.

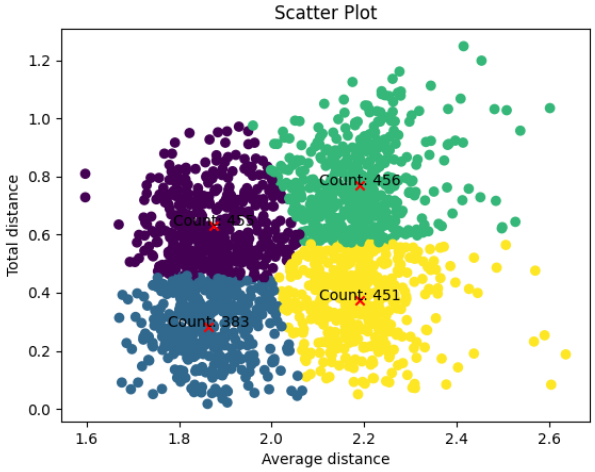


(a) Potential intermediate structure from the first cluster.

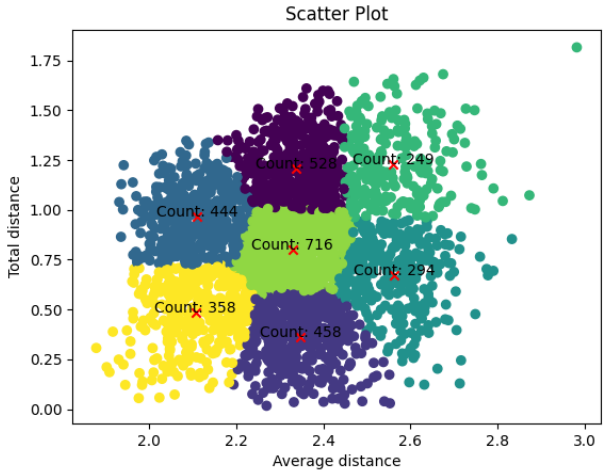


(b) Potential intermediate structure from the second cluster.

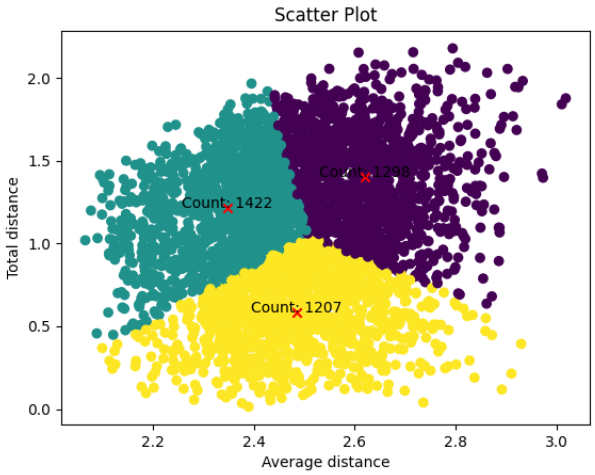
Figure 22: Potential intermediate structures of size 5.



(a) Clusters with centroid for structure with size 3

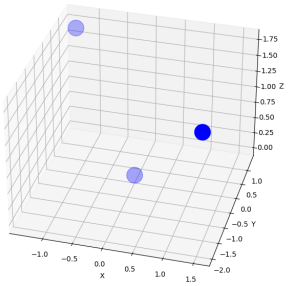


(b) Clusters with centroid for structure with size 4

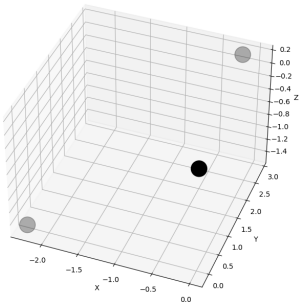


(c) Clusters with centroid for structure with size 5

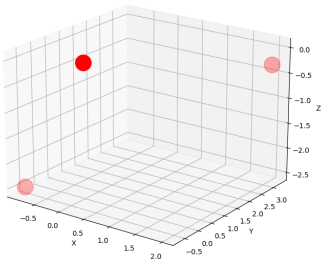
Figure 23: Resulting clusters after k-means clustering, with their respective centeroids



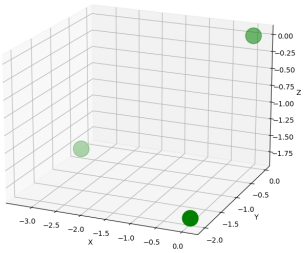
(a) Potential intermediate structure from the first cluster.



(b) Potential intermediate structure from the second cluster

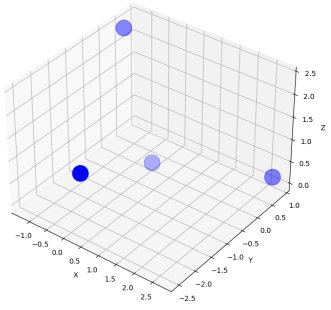


(c) Potential intermediate structure from the third cluster

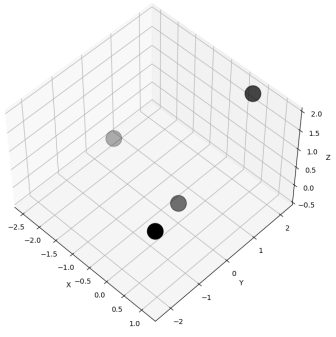


(d) Potential intermediate structure from the fourth cluster

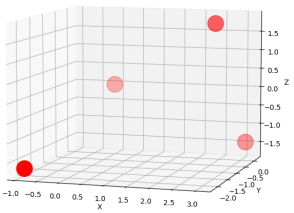
Figure 24: Potential intermediate structures of size 3



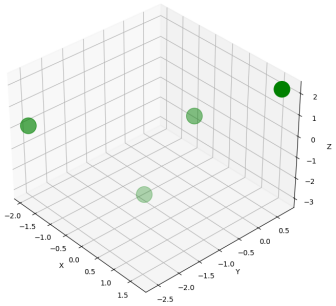
(a) Potential intermediate structure from the first cluster.



(b) Potential intermediate structure from the second cluster

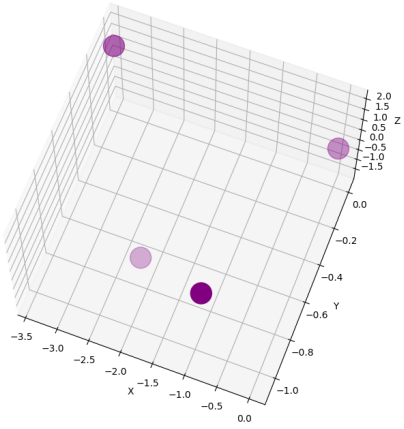


(c) Potential intermediate structure from the third cluster

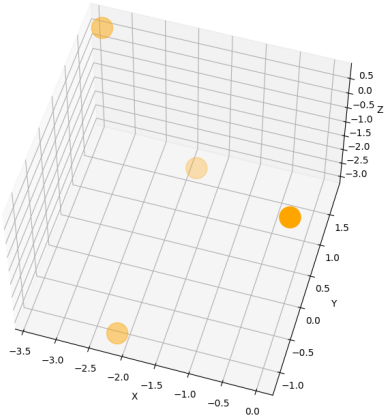


(d) Potential intermediate structure from the fourth cluster

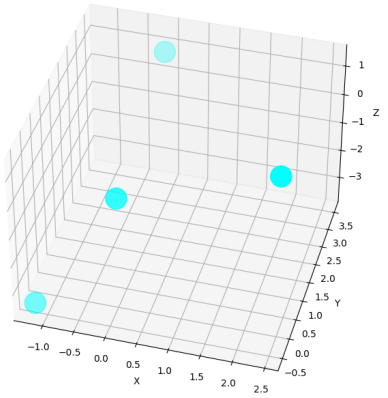
Figure 25: Potential intermediate structures of size 3



(a) Potential intermediate structure from the fifth cluster.

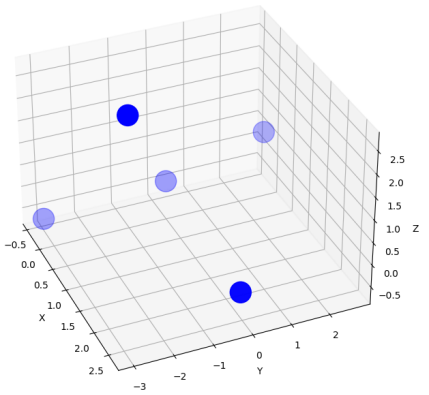


(b) Potential intermediate structure from the sixth cluster

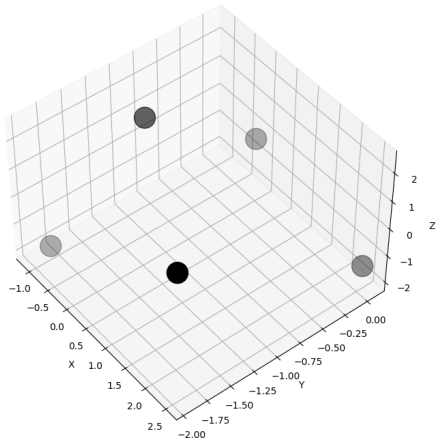


(c) Potential intermediate structure from the seventh cluster

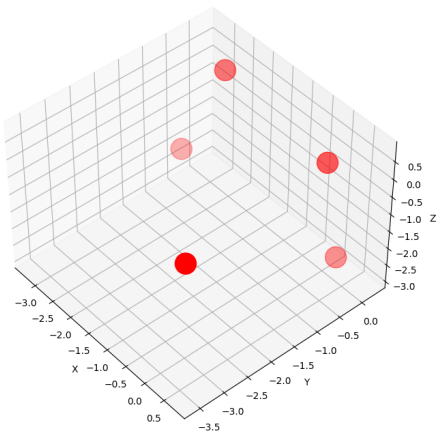
Figure 26: Potential intermediate structures of size 4



(a) Potential intermediate structure from the first cluster.



(b) Potential intermediate structure from the second cluster



(c) Potential intermediate structure from the third cluster

Figure 27: Potential intermediate structures of size 5

4.3 Experiment 3

4.3.1 Average distance and average angle analysis

For the method employed in this experiment, a total of 5755 potential intermediate structures were identified. Following the procedures outlined in section 3.6, the k-means algorithm provided the clusters depicted in figure 28.

Upon examining each cluster within the set of potential intermediate structures with a size of 3, we observe the structures in figure 29.

Upon closer examination of the structures within the size 3 cluster, we confirm the observation that these structures predominantly adopt a triangular arrangement. This recurring pattern suggests a limited range of bonding shapes given the small number of atoms involved.

Moreover, analyzing the positions of these structures during their respective frames provides valuable insights into their potential roles. Among these structures, one stands out, the red one (figure 29c) in its proximity to the ice structure. This finding implies that this particular structure has the potential to serve as an intermediate structure in the formation of ice.

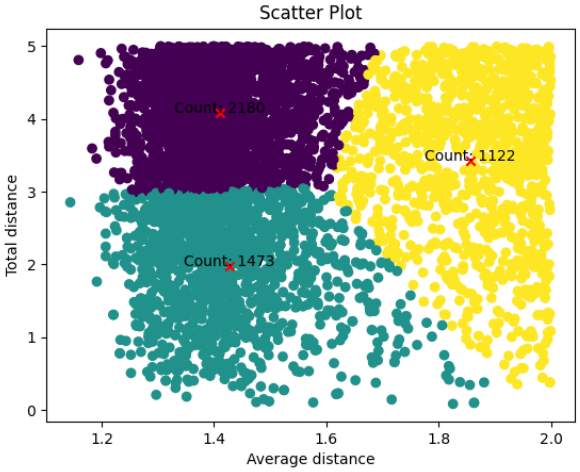
Although the structures within the size 3 cluster may share similarities and form a triangular configuration, the identification of a structure in close proximity to the ice structure highlights its significance in the ice formation process. Further analysis of this particular structure, including its bonding properties, energetic considerations, and dynamic behavior, will deepen our understanding of its role as an intermediate structure in ice formation.

Looking at the structures with a size of 4, we observe the structures showcased in figure 30.

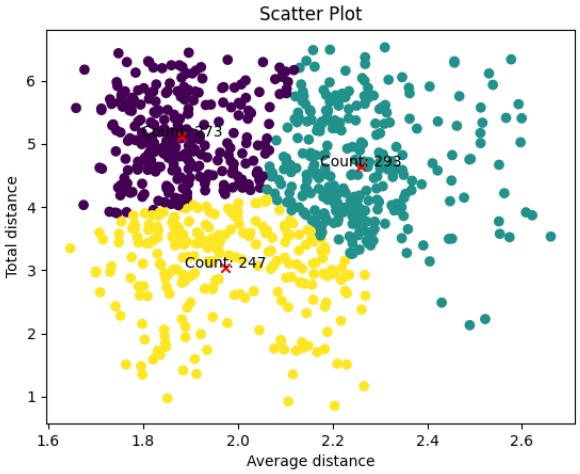
In the investigation of the structures with a size of 4, we once again encounter the presence of the Y-shaped bond structure (the blue one, figure 30a), which has been previously associated with potential ice formation. Furthermore, this structure is again found close to ice when its position relative to ice and hydrate is compared.

Inspecting the black structure (figure 30b), uncovers a flat square-like shape. The distinctive features of this structure, distinct from the Y-shaped configuration, suggest a different bonding pattern. This structure is found close to established hydrate structures, indicating that this might be an intermediate structure for hydrates.

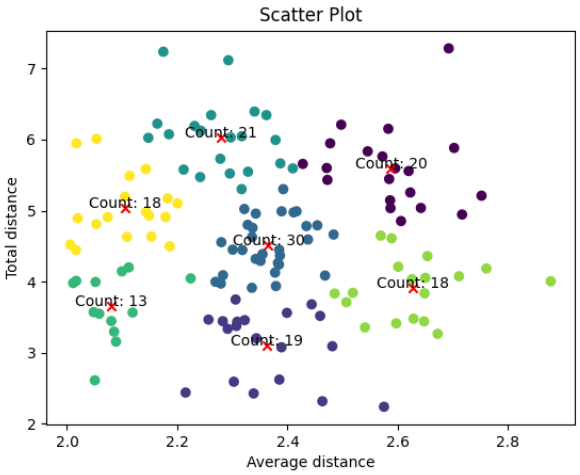
Conversely, the final structure, represented by the red structure (figure 30c), does not exhibit discernible features that can be readily associated with either ice or hydrates. Its lack of distinctive characteristics and absence of clear signs pointing towards hydrate or ice formation raises the possibility that this structure may simply be a representation of water molecules captured by the algorithm rather than a significant intermediate structure.



(a) Clusters with centroid for structure with size 3

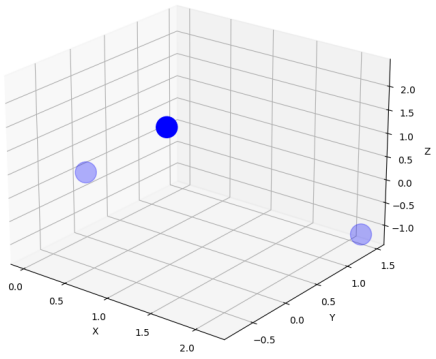


(b) Clusters with centroid for structure with size 4

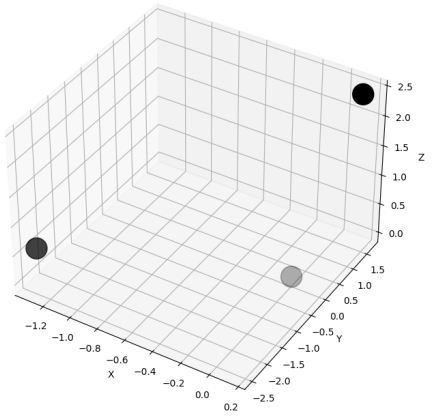


(c) Clusters with centroid for structure with size 5

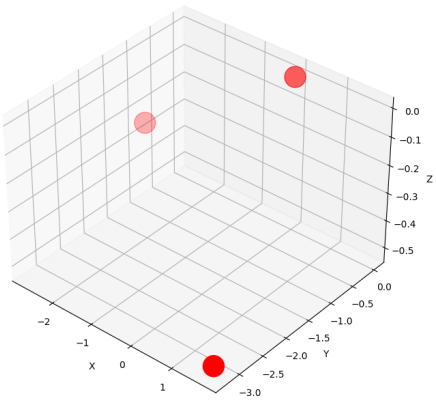
Figure 28: Resulting clusters after k-means clustering, with their respective centeroids



(a) Potential intermediate structure from the first cluster.

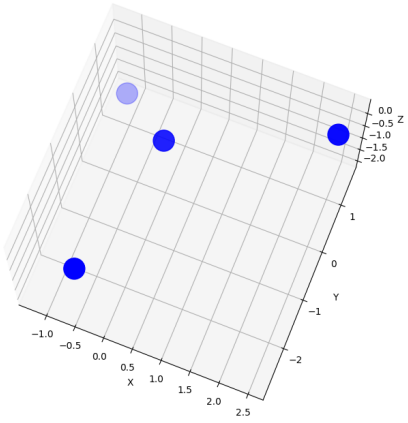


(b) Potential intermediate structure from the second cluster.

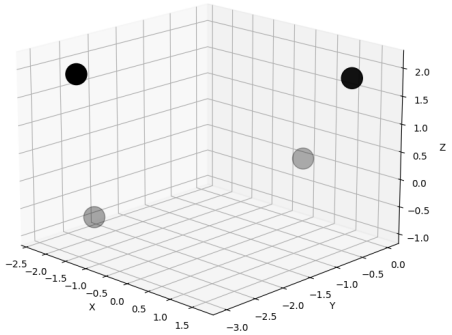


(c) Potential intermediate structure from the third cluster.

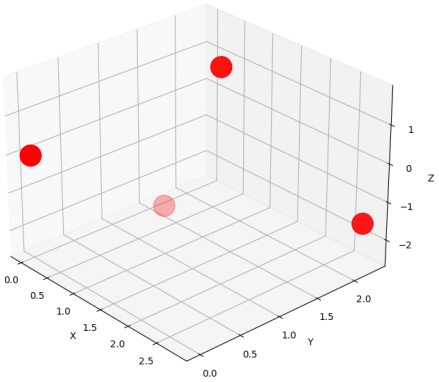
Figure 29: Potential intermediate structures of size 3.



(a) Potential intermediate structure from the first cluster.



(b) Potential intermediate structure from the second cluster.



(c) Potential intermediate structure from the third cluster.

Figure 30: Potential intermediate structures of size 4.

Finally, for structure size=5 we obtain these intermediates in figure 31, figure 32, and figure 33.

Analyzing the figures shows that the black, yellow, and green structures (in figure 31 and 32) lay close to hydrate structures indicating that they might be intermediate structures. The other structures do not show any potential to be intermediate structures.

4.3.2 Convex hull

For this method, we obtained 944 potential intermediates. The same procedures described for the other convex hull methods were also done this time and provided the clustering plot presented in figure 34.

From the clustering the two structures shown in figure 35 were extracted.

Only the biggest cluster was chosen since the smallest cluster did not have a significant amount of points to be considered valid enough. This was done for both the analysis for structures size 4 and size 5.

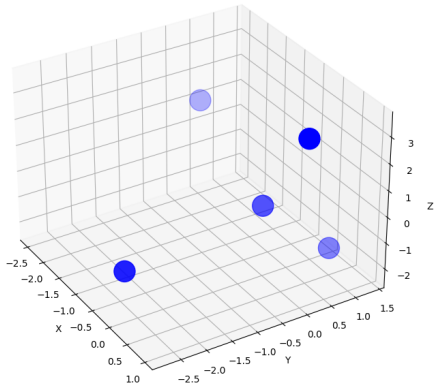
The size 4 structure looks like a Y bond structure, and as with previous results, these Y bond structures show potential to be an intermediate ice structure when its position is looked at relative to ice and hydrate.

For the size 5 structures, there is difficult to spot any distinguishing features, however, looking at its position for its given frame shows that this structure does not show any signs to be an intermediate structure.

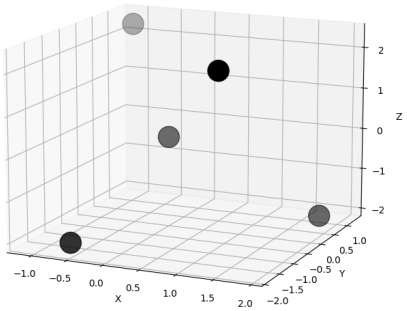
4.4 Life cycle analysis

Since the Y bond structure occurred in all the 4 sized structure analysis methods, a life cycle analysis was performed. From the life cycle analysis, only two of the four structures showed evidence of being an intermediate structure, as seen in figure 36 and figure 37.

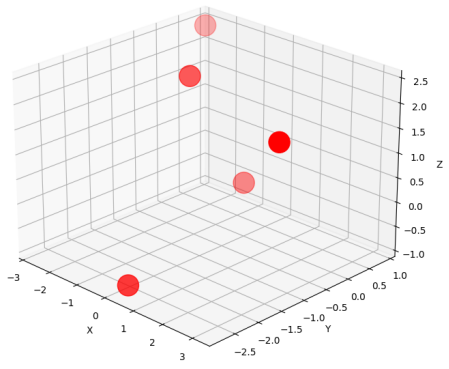
For the hydrate Y bond structures the results are of the opposite case. The intermediates from figure 25b, figure 26a and figure 26b. All three structures end opp splitting up, indicating that this is not an intermediate structure as seen in figure 38, figure 39, and figure 40.



(a) Potential intermediate structure from the first cluster.

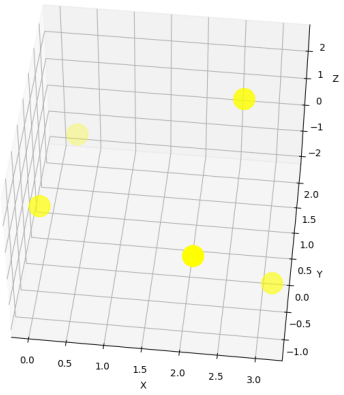


(b) Potential intermediate structure from the second cluster.

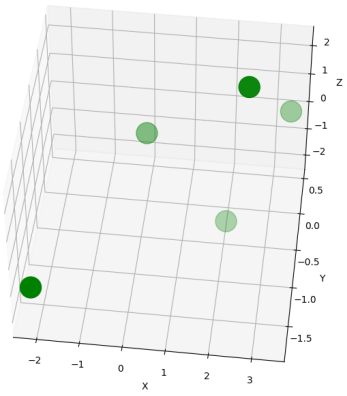


(c) Potential intermediate structure from the third cluster.

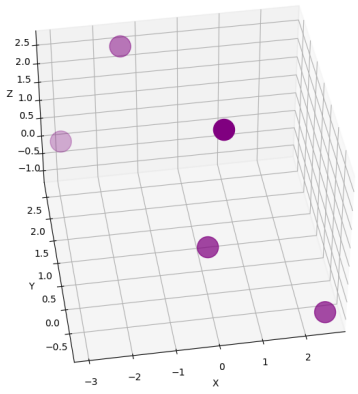
Figure 31: Potential intermediate structures of size 5.



(a) Potential intermediate structure from the fourth cluster.



(b) Potential intermediate structure from the fifth cluster.



(c) Potential intermediate structure from the sixth cluster.

Figure 32: Potential intermediate structures of size 5.

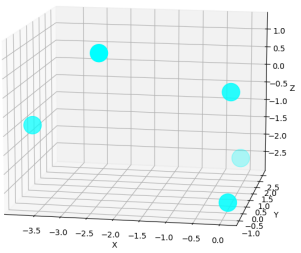
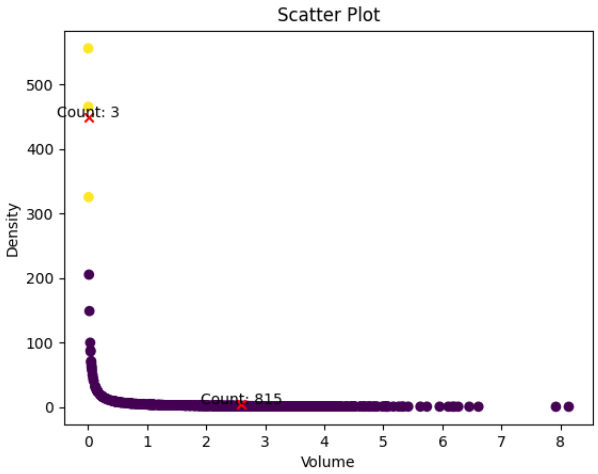
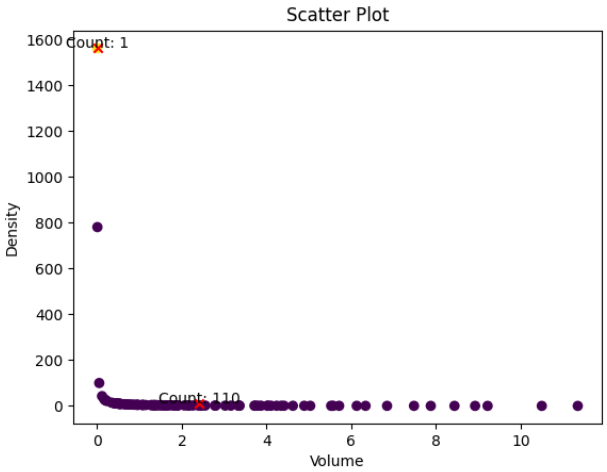


Figure 33: Potential intermediate structure from the seventh cluster.

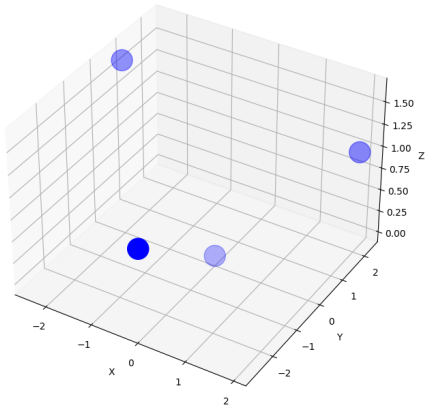


(a) k-means clustering result for structure with size 4

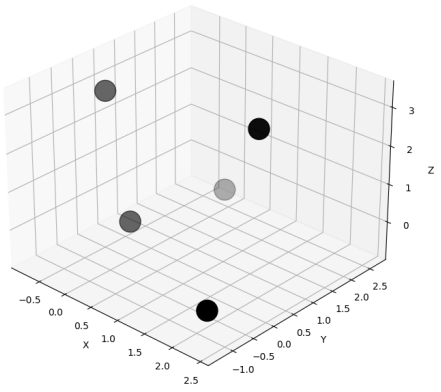


(b) k-means clustering result for structure with size 5

Figure 34: Resulting clusters after k-means clustering, with their respective centeroids

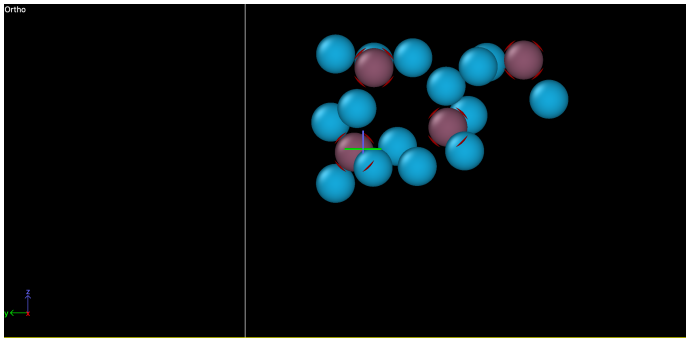


(a) Potential intermediate structure of size 4.

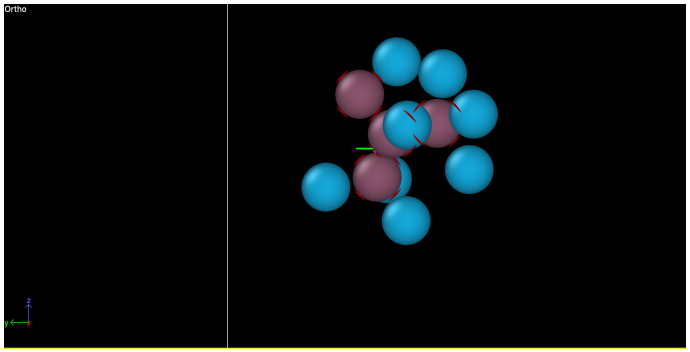


(b) Potential intermediate structure of size 5.

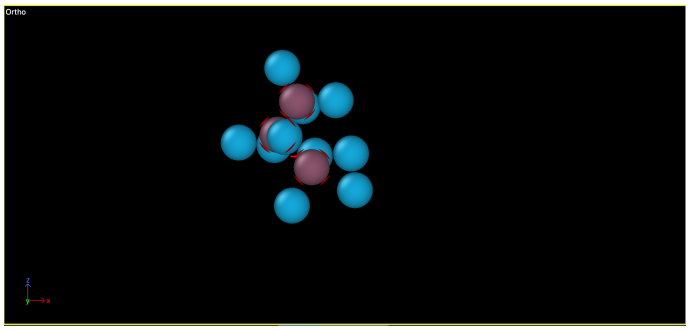
Figure 35: Potential intermediate structures of size 4 and 5.



(a) Snapshot when the molecules are in their water phase

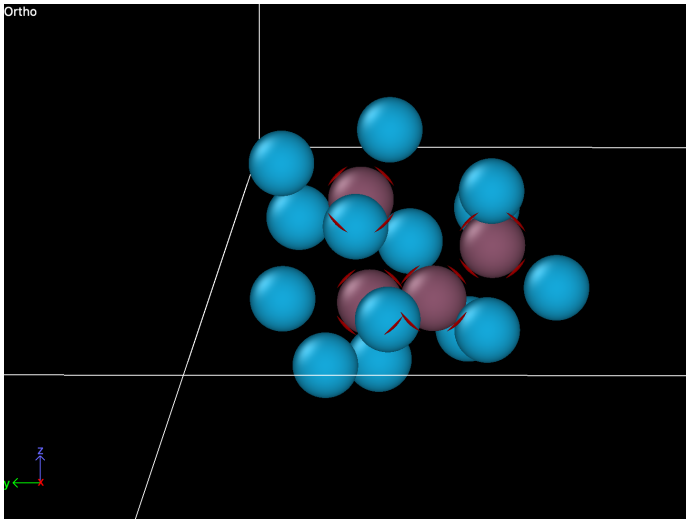


(b) Snapshot when the molecules have formed a Y bond structure

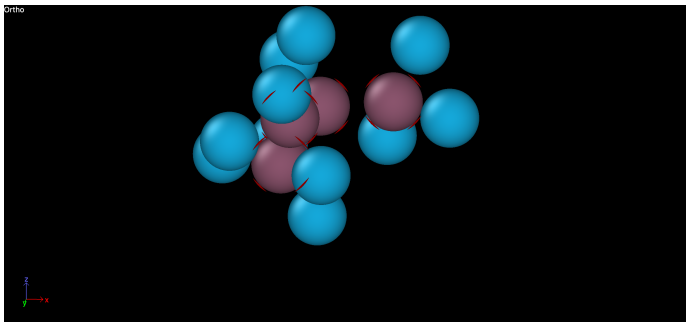


(c) Snapshot when the molecules have become a part of hexagonal ice

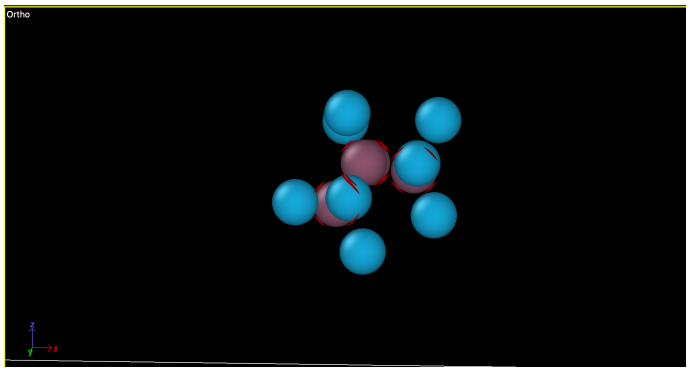
Figure 36: Snapshots of the life cycle of the molecules in the Y bond structure in the figure. The red molecules are the ones in the Y-bond structure, the blue are the ones that interact with the red ones during the life cycle. This is the intermediate from figure 17a.



(a) Snapshot when the molecules are in their water phase

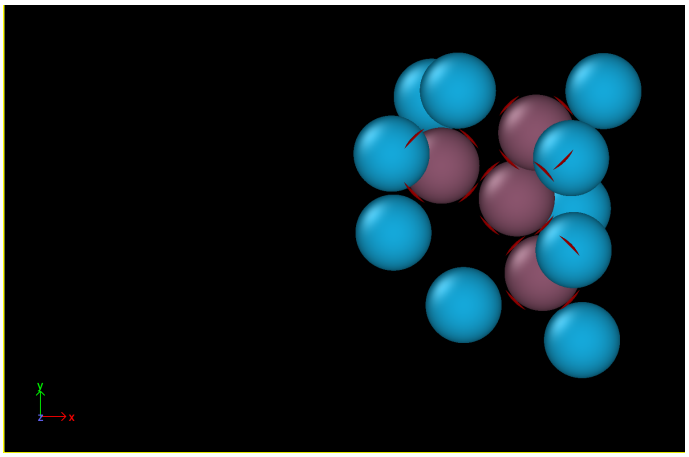


(b) Snapshot when the molecules have formed a Y bond structure

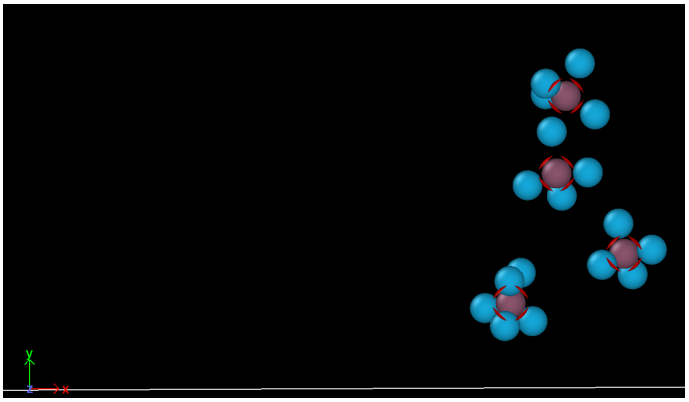


(c) Snapshot when the molecules have become a part of hexagonal ice

Figure 37: Snapshots of the life cycle of the molecules in the Y bond structure in the figure. The red molecules are the ones in the Y bond structure, the blue is the ones that interact with the red ones during the life cycle. This is the intermediate from figure 30a.

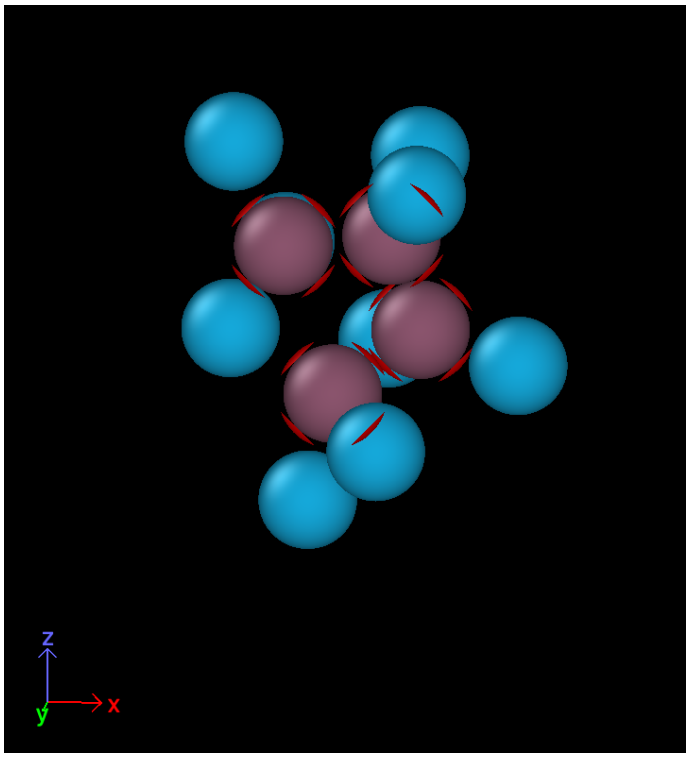


(a) The potential intermediate structure before it splits up

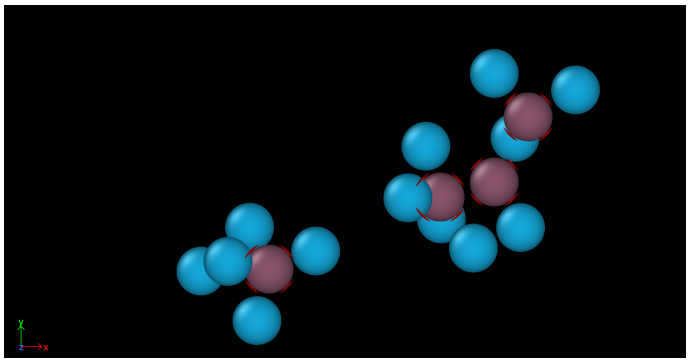


(b) The intermediate structure has been split up indicating that this is not an intermediate structure.

Figure 38: Life cycle analysis of hydrate depicted in figure 25b. The red molecules are the molecules that form the intermediate structure, the blue ones are molecules that interact with the red ones. This is the intermediate from figure 25b.

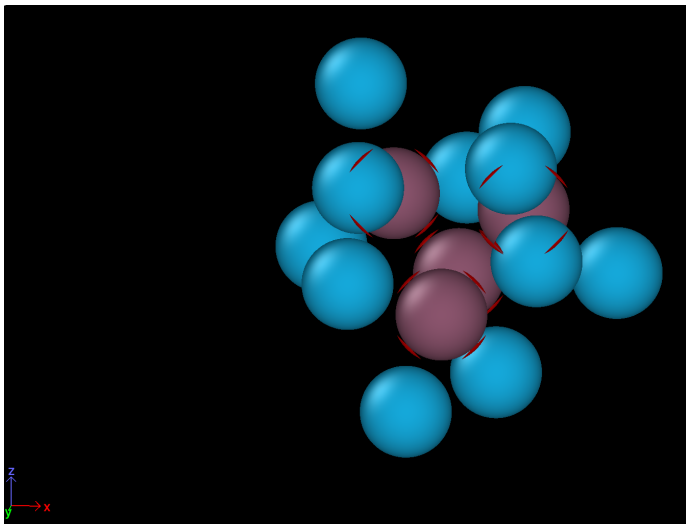


(a) The potential intermediate structure before it splits up

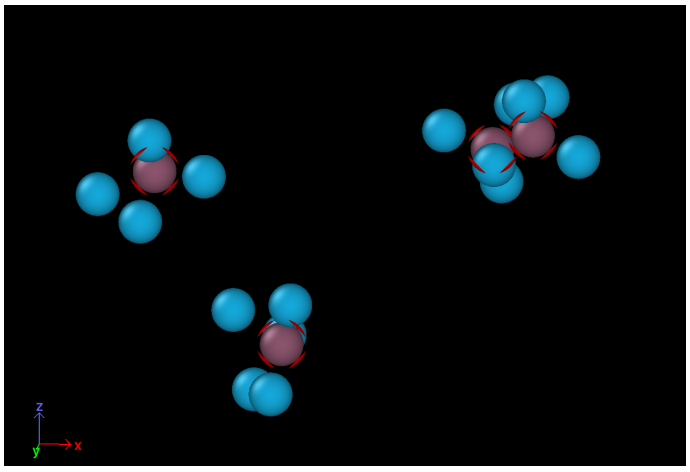


(b) The intermediate structure has been split up indicating that this is not an intermediate structure.

Figure 39: Life cycle analysis of hydrate depicted in figure 26a. The red molecules are the molecules that form the intermediate structure, the blue ones are molecules that interact with the red ones. This is the intermediate from figure 26a.



(a) The potential intermediate structure before it splits up



(b) The intermediate structure has been split up indicating that this is not an intermediate structure.

Figure 40: Life cycle analysis of hydrate depicted in figure 26b. The red molecules are the molecules that form the intermediate structure, the blue ones are molecules that interact with the red ones. This is the intermediate from figure 26b.

5 Discussion

In this section, we will mainly discuss the reasons why different approaches were made, why algorithms like DBSCAN and k-means were chosen, and what impact the different decisions has on the results. At the end of this section, the results will be evaluated and discussed.

5.1 MD-simulation

For the MD simulation, the mW water model was chosen. The primary reason for this comes from the fact that mw water models can correctly capture the tetrahedron structures of the hydrogen bonding network from the three-body interactions among the water molecules [34]. This is crucial for studies of hydrate and ice.

The conditions for the simulation were set at 230K at normal atmospheric pressure. As mentioned in section 2.3, in these conditions homogeneous freezing nucleation becomes more likely. The reduced thermal energy of the water molecules allows for the formation of stable hydrogen bonding networks between neighboring water molecules. This makes it easier to study the interactions between the molecules and is the reason for choosing these conditions.

5.2 Unsupervised learning

The choice of unsupervised learning over supervised learning and reinforcement learning in the context of identifying intermediate structures during ice and hydrate formations is indeed justified by several key reasons.

Firstly, the lack of known intermediate structures presents a significant challenge when applying supervised learning. Supervised learning relies on labeled training data, where the model learns from examples with predefined labels. In the absence of labeled data for intermediate structures, it becomes impractical to train a supervised learning model to recognize and identify these structures accurately.

Similarly, reinforcement learning, which involves learning through trial and error based on a reward system, faces challenges in the absence of a well-defined reward framework for identifying intermediate structures. Without a clear understanding of what constitutes an intermediate structure and how to measure its quality or relevance, establishing an effective reward system becomes highly challenging or even infeasible.

In contrast, unsupervised learning approaches, such as clustering algorithms and dimensionality reduction techniques, do not rely on labeled data or predefined reward systems. They allow for the exploration of patterns, similarities, and relationships within the data without explicit guidance or predefined labels. By leveraging unsupervised learning techniques, researchers can discover latent structures and patterns in the data, potentially revealing intermediate structures that were previously unknown.

Furthermore, the received dataset, lacking information about intermediate structures, would not serve as suitable training data for supervised or reinforcement learning approaches. The absence of labeled instances or a reliable reward system would hinder the effectiveness of these learning approaches, rendering them less viable for the specific task of identifying intermediate structures.

Considering the limitations and challenges associated with supervised learning and reinforcement learning, an unsupervised learning approach emerges as the most appropriate and optimal solution. By allowing the algorithm to autonomously uncover patterns (without any human interference) and relationships within the data, unsupervised learning enables the exploration and potential discovery of intermediate structures without the need for explicit labeling or a predefined reward system.

5.3 Boundary conditions

Boundary conditions were implemented to minimize the teleportation effect during the MD simulation to ensure reliable analysis. Boundary conditions were implemented in the MD simulation to minimize the "teleportation effect." The teleportation effect refers to situations where atoms undergo abrupt position changes, resulting in drastic modifications to their neighborhoods within the simulation. This effect poses challenges in tracking and analyzing the evolving neighborhoods of individual atoms, which was a fundamental part of experiments 2 and 3.

By introducing boundary conditions, the teleportation effect was mitigated, reducing the occurrences of atoms experiencing significant positional changes and acquiring entirely new neighbors. Consequently, the neighborhood analysis could be conducted more effectively and consistently, as the neighborhoods of atoms remained relatively stable throughout the simulation.

It is important to note that implementing boundary conditions led to a reduction in the data sample size by more than half. However, despite this reduction, the remaining number of samples was still considered sufficient for meaningful analysis. The decision to proceed with the reduced sample size was based on the judgment that it provided a satisfactory representation of the system under investigation.

However, it should be emphasized that for experiment 1, the introduction of boundary conditions was not necessary. Since the neighborhood analysis was not employed in this particular experiment, the application of boundary conditions, in this case, may have inadvertently resulted in the loss of potential intermediate structures. It is plausible that certain intermediate structures might have been missed due to the influence of the boundary conditions.

Additionally, it is worth considering the possibility that atoms that could potentially be part of intermediate structures may have been lost as a result of the boundary conditions. This means that the observed intermediate structures might not have been complete, as some atoms relevant to these structures might have been affected by the boundary effects. However, based on the analysis, it is reasonable to assume that this was not a significant concern, as most of the identified intermediate structures were located sufficiently far away from the boundaries. This observation enhances the validity and reliability of the results obtained from experiment 1.

By introducing the boundary conditions, the dataset loosed slightly over half of the atoms. The initial dataset, comprising approximately 23,000 atoms, was now reduced to approximately 11,000 atoms. Nonetheless, the remaining population of around 11,000 atoms still provided a substantial dataset for our analysis purposes.

In summary, the introduction of boundary conditions played a crucial role in minimizing the teleportation effect during the MD simulation, ensuring more reliable analysis for experiments 2 and 3. However, it is acknowledged that the use of boundary conditions in experiment 1 may have resulted in the potential loss of intermediate structures, which should be considered when interpreting the results of that specific experiment.

5.4 Why use DBSCAN?

The selection of DBSCAN as one of the clustering algorithms used was primarily driven by two key reasons, which make it particularly suitable for the analysis of the system under investigation.

Firstly, DBSCAN is well-suited for identifying clusters of arbitrary shapes and varying densities. Given that hexagonal ice, methane hydrate, and liquid water possess different shapes and densities, DBSCAN's ability to handle such diverse cluster characteristics makes it a reasonable choice. Unlike other clustering algorithms, DBSCAN is capable of detecting clusters that are not necessarily spherical or evenly distributed. This flexibility allows for a more accurate representation of the complex and diverse structural arrangements observed in the system.

Furthermore, DBSCAN is effective in identifying clusters that may be entirely surrounded by a different cluster, without being directly connected to it. This characteristic of DBSCAN helps

mitigate the issue of the so-called "single-link effect," where different clusters can be connected by a thin line of points. By specifying a minimum number of points (MinPts) required to form a cluster, DBSCAN reduces the impact of these thin connections and ensures more meaningful and distinct clustering results.

The parameter settings for DBSCAN, specifically epsilon and MinPts, were determined based on the knowledge of water molecules. Epsilon was chosen as 4, representing the maximum possible interaction length of a water molecule (rounded up from the actual value of 3.5 [62]). MinPts was set to 2, considering that a water molecule can bond with at least two other atoms, including itself.

To validate the effectiveness of DBSCAN in identifying liquid water clusters, the results were compared with the OVITO simulation. The clusters of water molecules identified by DBSCAN were visually inspected on three frames to confirm their correspondence with the expected liquid water regions. It is acknowledged that visual inspection is not an ideal method for verification and that a more reliable algorithm, such as the chill+ algorithm commonly used in the MD simulation community, would have been preferable. However, based on the positive results observed in the selected frames, it was assumed that DBSCAN would perform similarly for the remaining frames, given the consistent data format and relatively stable structural characteristics of ice, hydrate, and water across frames.

DBSCAN was also employed to cluster the atoms in experiment 1. The same reasons mentioned above, including the ability to handle arbitrarily shaped clusters and the lack of requirement to specify the number of clusters in advance, make DBSCAN a suitable choice for this experiment as well.

By utilizing DBSCAN, researchers can effectively analyze the system without the need for prior knowledge of the exact number of clusters or making assumptions about their shapes and densities. This flexibility and adaptability of DBSCAN contribute to its suitability for the clustering analysis of the given dataset.

5.5 Why use k-means

K-means clustering was selected for its simplicity, guaranteed convergence, and scalability to large datasets. It is also computationally efficient. Another important reason for selecting k means clustering was that the resulting clusters can be easily visualized and analyzed in a simple way. However, it has certain drawbacks that needed to be addressed. One limitation is the need to manually specify the number of clusters, which can be subjective and may not always be known in advance. Another challenge is that if one data axis has a significantly larger scalar value than the others, it can dominate the clustering process and result in suboptimal clusters.

To mitigate these disadvantages, two techniques were implemented. Firstly, data normalization was applied to standardize the data and ensure that all dimensions had a similar scale. This helped to alleviate the dominance of certain axes and improved the clustering performance. Secondly, silhouette analysis was employed to determine the optimal number of clusters for each analysis. Silhouette analysis was preferred over the elbow plot method because it eliminated the need for manual interpretation and decision-making. It automated the process of finding the optimal number of clusters, saving time and reducing the possibility of human error. The silhouette analysis provided a more objective and reliable criterion for selecting the optimal number of clusters, compared to visually analyzing elbow plots.

By incorporating data normalization and silhouette analysis, the k-means clustering method could be utilized effectively to identify intermediate structures in all three experiments. These techniques enhanced the clustering process and increased the reliability and objectivity of the results.

5.5.1 Intermediate structure selection process

The plots given by the k-means clustering served as valuable tools to depict and comprehend the intricate spatial arrangements and bonding characteristics inherent within the 3, 4, and 5-atom

structures. By examining and interpreting these plots, a comprehensive understanding of the structural properties and bonding tendencies of these smaller intermediates was achieved.

The decision to focus on the structure closest to the centroid was driven by the understanding that the centroid represents a central point that encapsulates the overall characteristics and tendencies of the structures within a given cluster. Thus, selecting the structure nearest to the centroid would yield an intermediate representation that most faithfully captures the collective features and traits exhibited by the structures in that cluster.

This careful selection process ensured that the chosen intermediate structures effectively represented the broader cluster from which they were derived. Consequently, the subsequent analysis and interpretation of these intermediate structures would provide valuable insights into the shared properties and behaviors of the structures within each cluster, enabling a more comprehensive understanding of their overall similarities and dissimilarities.

5.6 Other clustering algorithms

For the final analysis k-means clustering was chosen. However, there are other alternatives.

Overlapping clustering could have been used since this algorithm allows for a datapoint to be part of more than one cluster [39]. Which could have been able to find patterns between the different size clusters, something k-means clustering is not able to do. For the same reason, Probabilistic clustering could also have been used, by choosing The Gaussian Mixture Model (explained in section 2.6.3).

Hierarchical clustering is an algorithm that enables the grouping of similar data points into clusters based on their characteristics or proximity to one another [43]. This clustering technique could also have been used to find patterns in the data points. The resulting hierarchical dendrogram could help find patterns and relations in the intermediate structures.

By using different techniques new results will be gained to give better insight into identifying the characteristic intermediate structures that play a crucial role in water and hydrate phase transitions.

5.7 Experiment 1

This experiment was quite simple. By going from the assumption that if a molecule is absent from a water cluster in a frame but exists in the cluster in all previous frames, it should be safe to assume that this molecule is now, in fact, part of an intermediate structure, an ice structure or hydrate structure. Unfortunately, this is not exactly the case. As some molecules appear in the water cluster in frame N and be missing in frame n+i, they sometimes reappear in frame n+j (j¿i, and both in [0,100]). This undermines the validity of the results since the assumption is now not fully supported. This reappearance problem could be a result of choosing the wrong algorithm, meaning that DBSCAN is not the most optimal algorithm for this task. A better choice might have been using the chill+ algorithm. This algorithm has shown great potential in identifying ice, hydrates, and water with great accuracy by looking at different bond structures[63]. Another possibility might be that the assumption actually isn't true, but this does not seem like the case.

For this experiment the parameters for DBSCAN were set as $\epsilon = 4$Å, this is the maximum interaction distance for a water molecule rounded up from 3.5 Å. Having a larger ϵ value does not make sense since the atoms in these clusters would then not interact with each other, hence not performing any bonds. The reason to round up was done for simplicity reasons. However, keeping ϵ at 3.5 Å would have strengthened the validity of the results. The minPts was chosen as 2 since clusters containing one single molecule would not be an interest (since they don't interact with anything), but clusters containing two might be an intermediate structure, see appendix C for code.

5.8 Experiment 2

By using the shifting technique the idea was to see how a molecular neighborhood behaved over time to see if they showed any particular behavior of interest. However, this technique has a minor flaw that affects the results form this experiment. Since the shifting back (rotation) is being done according to the z-axis of a normal coordinate system, the shifting back is not being done correctly. The use of this method decreases the validity of the results of this experiment

A more accurate way would be to perform the rotation in a new coordinate axis where the center molecule still is in the origin, but the molecule used to compute the angular displacement should lay either on the x or y-axis. With this new coordinate axis in place, the rotation of all the neighboring molecules should be done according to the newly define z-axis. This will result in a more accurate shift back (rotation).

5.9 Experiment 3

Using the angular rotation to determine intermediate structures was done because water molecules within a crystal lattice do have a solid and fixed arrangement (as mentioned in section 2.2 and 2.3). It was therefore assumed that an intermediate structure of ice (and hydrate) maybe would share these properties. The solid arrangement would be determined by the angular displacement of a molecule. A big movement would mean that the molecule was in a liquid state, while a small displacement would mean a more solid arrangement that could possibly indicate an intermediate structure. This small angular displacement was seen as anything below 10 degrees. This number was chosen by looking at the angular displacement of the water molecules of ice and hydrate and choosing a max value that was fitted around 90 percent of the angular displacement of ice and hydrate.

5.10 Average distance and total bond length or angel analysis

The decision to use the average distance and total bond length for experiments 1 and 2, and average distance and angel for experiment 3 as parameters to find patterns in the different potential intermediate structures were used because these parameters can describe the shape, size, and relations between each molecule in a structure. Thereby give a set of data that can be compared to find patterns.

There are of course other ways to compare structures that might be more useful, like looking at the bonding forces or intermolecular pair forces like Lennard-Jones potential that might describe potential intermediate structures in a better way.

5.11 Convex hull analysis

The decision to use convex hull analysis was motivated by its ability to generate a 3D shape using a cluster of at least four points. This enabled the study to conduct shape analysis, which was crucial for the research objectives. Specifically, the volume of the convex hull was selected as a straightforward scalar parameter to facilitate the comparison of different cluster shapes and identify potential patterns. Additionally, the density of each convex hull was calculated to provide an additional scalar value for cluster comparison.

However, upon analyzing the resulting plots from the convex hull analysis, it became apparent that using volume and density as sole metrics for analysis might not be suitable. The negative exponential-looking plot did not exhibit a fitting pattern for analysis, hence the shape of the plot, and the application of k-means clustering did not demonstrate strong clustering capabilities in this particular context. Nevertheless, through the utilization of k-means clustering, certain patterns were observed, indicating that incorporating both volume and density metrics together could yield more meaningful insights for cluster analysis.

One limitation of the convex hull method is its inability to effectively analyze clusters consisting of only three atoms. However, in the context of this study, this limitation may not be significant since the results obtained from other analysis methods suggest that intermediate structures with three atoms are generally not of particular interest.

Overall, while the initial implementation of convex hull analysis provided some preliminary insights, the limitations, and challenges associated with using volume and density as the sole metrics necessitated further exploration and consideration of alternative approaches to enhance the cluster analysis.

5.12 Results

A lot of the results from experiment 1, 2, and 3 was not deemed plausible to become an intermediate structure, since they did not showcase any significant features or potential to become ice or hydrate (they never went from water to ice or hydrate during the simulation).

Especially the size 3 structures were hard to analyze. These structures were mostly overlooked because the analysis techniques that were used (average bond length and volume analysis) were not good enough to examine these structures. For a better understanding of these structures, investigation, and analysis in terms of their bonding properties, dynamics, and energetic considerations, can shed light on their potential role as an intermediate structure in the overall process of hydrate or ice formation.

The analysis of the 5-molecule structures highlights the diversity and complexity of potential intermediate structures. While some structures show compelling features with their placement relative to ice and hydrate, they do not provide substantial evidence to support their role as intermediates. This emphasizes the need for ongoing research to unravel the intricate mechanisms and dynamics involved in the transformation process from ice to hydrate. Like the size 3 structures, these structures could also be better understood with investigation and analysis in terms of their bonding properties, dynamics, and energetic considerations.

For the 4-molecule structures, only the Y-bond structure showed the potential of being an intermediate structure. However as with 3 and 5 molecular structures, to better understand the other structures investigation and analysis in terms of their bonding properties, dynamics, and energetic considerations should have been done. This could also provide more evidence to support the findings.

It should be mentioned that all clusters containing only two molecules were discarded since analyzing them in a reasonable way was not deemed possible with the analyzing methods that were used. Furthermore, clusters containing more than 5 molecules were neither analyzed. This was because of their more complex structures made it hard to see any specific patterns. To solve this the life cycle analysis was performed on the most promising intermediate structures to better understand how molecules interacted with each other. Another reason to not look at clusters bigger than 5 was the fact that a water molecule could maximally have 4 hydrogen bonds (making it 5 molecules in total).

5.12.1 Hydrates

The results from experiment 2 show that there are three promising results. The structure in figure 25b, figure 26a and figure 26b. All these structures pose the Y-bond feature and showed the potential of being a hydrate, but a life cycle analysis shows that this is not the case. As all the molecules end up breaking apart, which indicates that these are not intermediate structures.

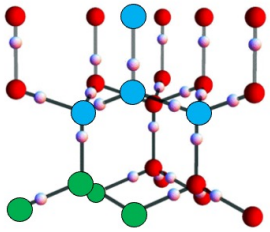
For the rest of the hydrate intermediates, there is not enough evidence to claim that one of the recognized structures from the experiments is indeed an intermediate structure for hydrate nucleation. The structures that were found are too dissimilar to each other to confidently say that they are intermediate structures for hydrates. However, this does not fully neglect the fact that these structures might be intermediate structures for hydrates. A hydrate can have more than one

intermediate structure, and the different results from the experiments show that there might be more than one intermediate structure for a hydrate.

5.12.2 Ice

Turning the attention to ice shows that there is a different story. The Y bond structure shows the most potential to be an intermediate structure, specifically an intermediate structure for ice. This structure showed up numerous times in the 4 atom-size clusters and was mostly located close to existing ice structures. The atoms in this structure also always ended up as part of hexagonal ice after some time was passed from the time step they were observed, further indicating that these Y-bond structures could be an intermediate structure for ice.

An interesting observation is that these Y bonds can be found in hexagonal ice, as seen in figure 41, which strengthens the case of these Y bond structures being intermediate structures.



(a)

Figure 41: The blue and green molecules represent two Y bond structures in hexagonal ice.

By studying the interactions and dynamics of atoms within the identified intermediate structures with a life cycle analysis, a deeper understanding of their stability and potential transition pathways could be obtained, contributing to the overall comprehension of the system under investigation.

The life-cycle simulation of the four Y bond structures showed mixed results, however, two of the four structures show signs of being an intermediate structure. As seen in figure 36 and figure 37, the molecules start out as water (figure 36a and figure 37a), forms the Y bond structure (figure 36b and figure 37b), and then stabilizes and becomes part of hexagonal ice (figure 36c and figure 37c). Both Y-bond structures take the same "positions" in hexagonal ice when comparing figure 36c and figure 37c, indicating that the Y-bond structure might be an intermediate structure for a certain part of hexagonal ice. It is however worth mentioning that these two results are not enough to say confidentially that this Y-bond structure is an intermediate structure, but that there is evidence to support this idea. Further investigation of the Y bond structure is needed to safely classify it as an intermediate structure by doing more experiments (or gathering more data/results) that support this idea.

It is worth mentioning that the intermediate structures that were found are found in a specific environment with specific conditions. The MD simulation that was performed was performed in an environment that allows for homogeneous freezing nucleation at 230 Kelvin at normal atmospheric pressure. At this temperature, water is supercooled. This means that homogeneous freezing nucleation becomes more likely. At this point, the thermal energy of the water molecules is significantly reduced, allowing the formation of stable hydrogen bonding networks between neighboring water molecules. These hydrogen bonds facilitate the arrangement of the water molecules into an ordered, crystalline structure characteristic of ice. This can be spotted in all the Y-bond structures, where the molecules form the mentioned order structure of a Y.

With different conditions, other intermediate structures may appear, that form different ordered structures. Finally the environment would also probably play a part in the formation of intermediate structures. There are no solid surfaces for ice to latch onto and form in this simulation. The introduction of a solid surface might influence the results. Therefore investigating and comparing intermediate structures in different conditions and environments will help with understanding the

formation of intermediates during ice-freezing nucleation. By doing this more data will be gathered to give a better understanding of the topic.

Finally, the timestep used in the MD simulation should be addressed. As this is an important factor in recognizing the intermediate structures. The few promising results indicate that a smaller timestep might be able to catch the nucleation process from liquid process to ice or hydrate more accurately. Looking at the life cycle analysis results also indicates that a smaller timestep would be able to capture how the Y molecules in the Y-bond structures interact with other molecules or other potential intermediate structures to form hexagonal ice (or methane hydrate).

6 Conclusion

The investigation of intermediate structures during the nucleation process of ice and hydrates represents a compelling and intriguing area of study. By employing various machine learning techniques, attempts were made to identify these intermediate structures during the phase transition of water to ice and hydrate.

The final results show significant promise with the Y bond-structures being a potential intermediate structure for hexagonal ice. These results confirm that there is a plausibility of the existence of an intermediate structure in the phase transition of water to hexagonal ice. It is thus worthy of the further verification of such water intermediate structures and the further understanding of their effect on the free energy pathway of phase transition. As mentioned in section 5, other methods should be investigated to provide a more sufficient way to analyze the potential intermediate structures.

The outcomes demonstrate considerable promise, particularly about the Y-bond structures, which exhibit potential as intermediate structures in the formation of hexagonal ice. These results provide empirical support for the plausibility of intermediate structures during the water-to-hexagonal-ice phase transition. Consequently, further investigation is warranted to verify the existence of such intermediate structures and to deepen our understanding of their impact on the free energy pathway of phase transitions. As mentioned in section 5, it is crucial to explore alternative methods that can offer more robust analyses of potential intermediate structures.

Additionally, revisiting the molecular dynamics (MD) simulations with smaller timeframes is essential to effectively capture subtle changes in the nucleation process of ice and hydrates. The chosen timeframe of 500ps falls short of providing sufficient insights into the molecular dynamics and temporal evolution of the system.

Furthermore, it is crucial to perform MD simulations under different conditions and environments, as these factors can influence the development and characteristics of intermediate structures during the nucleation process. Examining diverse conditions will contribute to a more comprehensive understanding of the influence of external factors on the formation of intermediate structures.

In summary, while there is still more work to be done, the application of machine learning has demonstrated its potential in recognizing intermediate structures. By implementing the aforementioned adjustments, the likelihood of identifying intermediate structures using machine learning techniques increases, as there is already evidence pointing towards the existence of such structures during ice and hydrate nucleation.

7 Further work

Based on the performed research it is believed that several of the following ideas should be done if further research is performed, and might serve as interesting topics to examine:

- Utilizing the chill+ algorithm instead of DBSCAN for identifying water, hydrate, and ice: Research indicates that the Chill+ algorithm has higher accuracy in identifying water, hydrate, and ice [63]. By employing the Chill+ algorithm in the identification process, potential errors associated with DBSCAN could be minimized. This would enhance the reliability of the identification process and contribute to more accurate results.
- Performing molecular dynamics (MD) simulations in a different environment: To gain deeper insights into the formation of ice and hydrates, it is recommended to conduct MD simulations within an environment featuring a solid surface conducive to ice formation. This investigation holds particular relevance for industrial applications, as it would facilitate a better understanding of the formation processes. The outcomes of such simulations could be utilized in developing coatings or structures that impede the development of intermediate structures, subsequently preventing ice formation on windmills or hydrate formation in deep-sea oil cables.
- Reducing the timestep in MD simulations to enhance nucleation process analysis: As highlighted in section 5.12.2 of the research, a decrease in the timestep utilized in MD simulations would enable a more precise analysis of the nucleation process. By capturing smaller temporal intervals, researchers can gain finer-grained insights into the mechanisms and dynamics of nucleation. This adjustment would contribute to a more comprehensive understanding of the process and potentially reveal crucial details that might have been missed with larger timesteps.
- Exploring Y bond structures: Given that the research identified Y bond structures as the most promising in terms of intermediate structures, further investigation into these structures is warranted. One interesting approach is to employ a supervised learning framework that focuses on identifying and analyzing these specific bond structures. By utilizing machine learning techniques, researchers can potentially uncover new insights and patterns associated with Y bond structures, ultimately enhancing our understanding of their role and implications.
- Using other types of clustering algorithms: Exploration of alternative clustering algorithms, such as overlapping, probabilistic, or hierarchical clustering techniques, holds the potential to illuminate additional intermediate structures that emerge during the phase transitions of water and hydrates. By incorporating these diverse clustering methodologies into the analysis, researchers can expand the scope of the investigation and potentially uncover novel intermediate structures that were not captured by the previous methods. These alternative algorithms provide a valuable avenue for further exploration, allowing for a more comprehensive understanding of the complex dynamics and structural transformations occurring during the phase transitions.

By pursuing these research directions, significant advancements can be made in the field. The proposed ideas have the potential to improve accuracy in identification methodologies, provide practical applications for industrial settings, refine simulation techniques for better analysis, and shed light on the characteristics of specific bond structures. Such endeavors would contribute to the scientific knowledge base and potentially lead to practical solutions for challenges related to ice and hydrate formation.

Bibliography

[1] Resources for the future. *Water Resources*.
URL: https://www.rff.org/topics/natural-resources/water/. (accessed: 08.06.2023).

[2] Steven S Zumdahl. *Water. Encyclopedia Britannica*.
URL: https://www.britannica.com/science/water.. (accessed: 08.06.2023).

[3] Ray Boswell and Timothy S. Collett. 'Current perspectives on gas hydrate resources'.
In: *Energy Environ. Sci.* 4 (4 2011), pp. 1206–1215. DOI: 10.1039/C0EE00203H.

[4] Ivana Durickovic et al. 'Water-ice phase transition probed by Raman spectroscopy'.
In: *Journal of Raman Spectroscopy* 42 (June 2011), p. 1408. DOI: 10.1002/jrs.2841.

[5] Christoph Salzmann et al. 'Structure and nature of ice XIX'.
In: *Nature Communications* 12 (May 2021). DOI: 10.1038/s41467-021-23399-z.

[6] Halim Kusumaatmaja and Apala Majumdar.
'Free energy pathways of a multistable liquid crystal device'.
In: *Soft Matter* 11 (24 2015), pp. 4809–4817. DOI: 10.1039/C5SM00578G.

[7] Roi Asor et al.
'Rapidly Forming Early Intermediate Structures Dictate the Pathway of Capsid Assembly'.
In: *Journal of the American Chemical Society* 142.17 (2020). PMID: 32233479,
pp. 7868–7882. DOI: 10.1021/jacs.0c01092. eprint: https://doi.org/10.1021/jacs.0c01092.

[8] Philipp Neudecker et al.
'Structure of an Intermediate State in Protein Folding and Aggregation'.
In: *Science (New York, N.Y.)* 336 (Apr. 2012), pp. 362–6. DOI: 10.1126/science.1214203.

[9] Linyue Gao et al. 'A field study of ice accretion and its effects on the power production of utility-scale wind turbines'. In: *Renewable Energy* 167 (2021), pp. 917–928. ISSN: 0960-1481.
DOI: https://doi.org/10.1016/j.renene.2020.12.014.

[10] Feng Wang et al. 'Anti-gas hydrate surfaces: perspectives, progress and prospects'.
In: *J. Mater. Chem. A* 10 (2 2022), pp. 379–406. DOI: 10.1039/D1TA08965J.

[11] Aidan P. Thompson et al. 'LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales'.
In: *Computer Physics Communications* 271 (2022), p. 108171. ISSN: 0010-4655.
DOI: https://doi.org/10.1016/j.cpc.2021.108171.

[12] Valeria Molinero and Emily B. Moore.
'Water Modeled As an Intermediate Element between Carbon and Silicon'.
In: *The Journal of Physical Chemistry B* 113.13 (2009). PMID: 18956896, pp. 4008–4016.
DOI: 10.1021/jp805227c. eprint: https://doi.org/10.1021/jp805227c.

[13] A.D. Buckingham.
'The hydrogen bond, and the structure and properties of H20 and (H20)2'.
In: *Journal of Molecular Structure* 250.2 (1991), pp. 111–118. ISSN: 0022-2860.
DOI: https://doi.org/10.1016/0022-2860(91)85023-V.

[14] Marry K. Campbell and Shawn O. Farrell. 'BIOCHEMISTRY 6th EDITION'. In:
Thomson Brooks/Cole, 2009. Chap. 2.1.

[15] Marry K. Campbell and Shawn O. Farrell. 'BIOCHEMISTRY 6th EDITION'. In:
Thomson Brooks/Cole, 2009. Chap. 2.2.

[16] Niels Bjerrum. 'Structure and Properties of Ice'. In: *Science* 115.2989 (1952), pp. 385–390.
DOI: 10.1126/science.115.2989.385.
eprint: https://www.science.org/doi/pdf/10.1126/science.115.2989.385.

[17] Steven S Zumdahl. *Water. Encyclopedia Britannica*.
URL: https://www.britannica.com/science/water/Structures-of-ice.. (accessed: 08.06.2023).

[18] Renaud G Vaysset A Geaymond O Pasturel A Schülli TU Daudin R.
'Substrate-enhanced supercooling in AuSi eutectic droplets'.
In: *Nature* 464 (2010), 1174–1177. DOI: doi:10.1038/nature08986.

[19] Harvard university. *Supercooling of Water*.
URL: https://sciencedemonstrations.fas.harvard.edu/presentations/supercooling-water.
(accessed: 02.06.2023).

[20] University of British Columbia Rolland Stull. *Nucleation of Ice Crystals*.
URL: https://geo.libretexts.org/Bookshelves/Meteorology_and_Climate_Science/Practical_Meteorology_(Stull)/073A_Precipitation_Processes/7.033A_Nucleation_of_Ice_Crystals.
(accessed: 02.06.2023).

[21] G. R. Wood and A. G. Walton. 'Homogeneous Nucleation Kinetics of Ice from Water'.
In: *Journal of Applied Physics* 41.7 (Nov. 2003), pp. 3027–3036. ISSN: 0021-8979.
DOI: 10.1063/1.1659359.
eprint: https://pubs.aip.org/aip/jap/article-pdf/41/7/3027/7946721/3027_1_online.pdf.

[22] Gabor Vali. '- Ice Nucleation — a review'. In: *Nucleation and Atmospheric Aerosols 1996*.
Ed. by Markku Kulmala and Paul E. Wagner. Amsterdam: Pergamon, 1996, pp. 271–279.
ISBN: 978-0-08-042030-1. DOI: https://doi.org/10.1016/B978-008042030-1/50066-4.

[23] Owen Benton, Olga Sikora and Nic Shannon. 'Electromagnetism on ice: classical and quantum theories of proton disorder in hexagonal water ice'.
In: *Physical Review B* 93 (Apr. 2015). DOI: 10.1103/PhysRevB.93.125143.

[24] sciencefacts. *Why Does Ice Float on Water*.
URL: https://www.sciencefacts.net/why-does-ice-float-on-water.html. (accessed: 11.06.2023).

[25] W. F. Kuhs* and M. S. Lehmann. 'The Structure of Ice Ih by Neutron Diffraction'.
In: *J. Phys. Chem.* 87 (1958), pp. 4312–4313.

[26] Lonsdale Kathleen Yardley. 'The structure of ice'.
In: *The Royal Society London* 247 (1958), pp. 424–434.
DOI: https://doi.org/10.1098/rspa.1958.0197.

[27] Ian Sutton. 'Chapter 3 - Hazards identification'.
In: *Process Risk and Reliability Management*. Ed. by Ian Sutton.
Oxford: William Andrew Publishing, 2010, pp. 79–190. ISBN: 978-1-4377-7805-2.
DOI: https://doi.org/10.1016/B978-1-4377-7805-2.10003-1.

[28] S.M. Masutani. 'Greenhouse Gas Hydrates in the Ocean'.
In: *Carbon Dioxide Utilization for Global Sustainability*.
Ed. by Sang-Eon Park, Jong-San Chang and Kyu-Wan Lee. Vol. 153.
Studies in Surface Science and Catalysis. Elsevier, 2004, pp. 487–494.
DOI: https://doi.org/10.1016/S0167-2991(04)80300-8.

[29] J.L. Stauber, A. Chariton and S. Apte. 'Chapter 10 - Global Change'.
In: *Marine Ecotoxicology*. Ed. by Julián Blasco et al. Academic Press, 2016, pp. 273–313.
ISBN: 978-0-12-803371-5. DOI: https://doi.org/10.1016/B978-0-12-803371-5.00010-2.

[30] John Carroll. 'Chapter 2 - Hydrate types and formers'.
In: *Natural Gas Hydrates (Fourth Edition)*. Ed. by John Carroll. Fourth Edition.
Boston: Gulf Professional Publishing, 2020, pp. 27–67. ISBN: 978-0-12-821771-9.
DOI: https://doi.org/10.1016/B978-0-12-821771-9.00002-1.

[31] Jana A.K Palodkar A.V. 'Modeling recovery of natural gas from hydrate reservoirs with carbon dioxide sequestration: Validation with Iġnik Sikumi field data.'
In: *Scientific Reports* 9.18901 (2019). DOI: https://doi.org/10.1038/s41598-019-55476-1.

[32] J. Polanski. '4.14 - Chemoinformatics'. In: *Comprehensive Chemometrics*.
Ed. by Steven D. Brown, Romá Tauler and Beata Walczak. Oxford: Elsevier, 2009,
pp. 459–506. ISBN: 978-0-444-52701-1.
DOI: https://doi.org/10.1016/B978-044452701-1.00006-5.

[33] Orozco M Gelpi J Hospital A Goñi JR.
'Molecular dynamics simulations: advances and applications'.
In: *Dovepress* 2015.8 (2015), pp. 37–47. DOI: 10.2147/AABC.S70333.

[34] Rui Ma et al.
'An interfacial gas-enrichment strategy for mitigating hydrate adhesion and blockage'.
In: *Chemical Engineering Journal* 453 (2023), p. 139918. ISSN: 1385-8947.
DOI: https://doi.org/10.1016/j.cej.2022.139918.

[35] Liam C. Jacobson, Waldemar Hujo and Valeria Molinero.
'Nucleation Pathways of Clathrate Hydrates: Effect of Guest Size and Solubility'.
In: *The Journal of Physical Chemistry B* 114.43 (2010). PMID: 20931990, pp. 13796–13807.
DOI: 10.1021/jp107269q. eprint: https://doi.org/10.1021/jp107269q.

[36] Liam C. Jacobson and Valeria Molinero. 'A MethaneWater Model for Coarse-Grained Simulations of Solutions and Clathrate Hydrates'.
In: *The Journal of Physical Chemistry B* 114.21 (2010). PMID: 20462253, pp. 7302–7311.
DOI: 10.1021/jp1013576. eprint: https://doi.org/10.1021/jp1013576.

[37] Thomas W. Edgar and David O. Manz. 'Chapter 6 - Machine Learning'.
In: *Research Methods for Cyber Security*. Ed. by Thomas W. Edgar and David O. Manz.
Syngress, 2017, pp. 153–173. ISBN: 978-0-12-805349-2.
DOI: https://doi.org/10.1016/B978-0-12-805349-2.00006-6.

[38] Michael Jordan and T.M. Mitchell.
'Machine Learning: Trends, Perspectives, and Prospects'.
In: *Science (New York, N.Y.)* 349 (July 2015), pp. 255–60. DOI: 10.1126/science.aaa8415.

[39] IBM. *Machine Learning*. URL: https://www.ibm.com/cloud/learn/machine-learning.
(accessed: 19.05.2023).

[40] Pádraig Cunningham, Matthieu Cord and Sarah Jane Delany. 'Supervised Learning'. In:
Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval.
Ed. by Matthieu Cord and Pádraig Cunningham.
Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 21–49. ISBN: 978-3-540-75171-7.
DOI: 10.1007/978-3-540-75171-7_2.

[41] Zoubin Ghahramani. 'Unsupervised Learning'. In:
Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures.
Ed. by Olivier Bousquet, Ulrike von Luxburg and Gunnar Rätsch.
Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 72–112. ISBN: 978-3-540-28650-9.
DOI: 10.1007/978-3-540-28650-9_5.

[42] Pádraig Cunningham, Matthieu Cord and Sarah Jane Delany. 'Supervised Learning'. In:
Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval.
Ed. by Matthieu Cord and Pádraig Cunningham.
Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 51–90. ISBN: 978-3-540-75171-7.
DOI: 10.1007/978-3-540-75171-7_2.

[43] Everitt. Brian. *Cluster Analysis / Brian S. Everitt. – 5th ed.* John Wiley Sons, Ltd, 2011.
ISBN: 978-0-470-97781-1.

[44] Anil K. Jain. 'Data clustering: 50 years beyond K-means'.
In: *Pattern Recognition Letters* 31.8 (2010). Award winning papers from the 19th International Conference on Pattern Recognition (ICPR), pp. 651–666. ISSN: 0167-8655.
DOI: https://doi.org/10.1016/j.patrec.2009.09.011.

[45] Joerg Sander. 'Density-Based Clustering'. In: *Encyclopedia of Machine Learning*.
Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010,
pp. 270–273. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_211.

[46] Erich Schubert et al.
'DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN'.
In: *ACM Trans. Database Syst.* 42.3 (2017). ISSN: 0362-5915. DOI: 10.1145/3068335.

[47] Kamran Khan et al. 'DBSCAN: Past, present and future'.
In: *The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014)*. 2014, pp. 232–238. DOI: 10.1109/ICADIWT.2014.6814687.

[48] Chris Piech. *K Means*. URL: https://stanford.edu/~cpiech/cs221/handouts/kmeans.html.
(accessed: 04.12.2022).

[49] Pasi Fränti and Sami Sieranoja.
'How much can k-means be improved by using better initialization and repeats?'
In: *Pattern Recognition* 93 (2019), pp. 95–112. ISSN: 0031-3203.
DOI: https://doi.org/10.1016/j.patcog.2019.04.014.

[50] Attila Lengyel and Zoltán Botta-Dukát. 'Silhouette width using generalized mean—A flexible method for assessing clustering efficiency'.
In: *Ecology and Evolution* 9.23 (2019), pp. 13231–13243.
DOI: https://doi.org/10.1002/ece3.5774.
eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/ece3.5774.

[51] Meshal Shutaywi and Nezamoddin N. Kachouie. 'Silhouette Analysis for Performance Evaluation in Machine Learning with Applications to Clustering'. In: *Entropy* 23.6 (2021). ISSN: 1099-4300. DOI: 10.3390/e23060759.

[52] Peter J. Rousseeuw.
'Silhouettes: A graphical aid to the interpretation and validation of cluster analysis'.
In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65.
ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(87)90125-7.

[53] Jack Sklansky. 'Finding the Convex Hull of a Simple Polygon'.
In: *Pattern Recogn. Lett.* 1.2 (1982), 79–83. ISSN: 0167-8655.
DOI: 10.1016/0167-8655(82)90016-2.

[54] R. TYRRELL ROCKAFELLAR. *Convex Analysis*. Princeton University Press, 1996.

[55] Andrés Serna and Flavio Prieto. 'Towards a 3D modeling of brain tumors by using endoneurosonography and neural networks'.
In: *Revista Ingenierías Universidad de Medellín* 16 (June 2017), pp. 129–148.
DOI: 10.22395/rium.v16n30a7.

[56] C. Bradford Barber, David P. Dobkin and Hannu Huhdanpaa.
'The Quickhull Algorithm for Convex Hulls'.
In: *ACM Trans. Math. Softw.* 22.4 (1996), 469–483. ISSN: 0098-3500.
DOI: 10.1145/235815.235821.

[57] Barrett O'Neill. 'Chapter 2 - Frame Fields'.
In: *Elementary Differential Geometry (Second Edition)*. Ed. by Barrett O'Neill.
Second Edition. Boston: Academic Press, 2006, pp. 43–99. ISBN: 978-0-12-088735-4.
DOI: https://doi.org/10.1016/B978-0-12-088735-4.50006-7.

[58] Fatima A. Merchant, Shishir K. Shah and Kenneth R. Castleman.
'Chapter Eight - Object Measurement'. In: *Microscope Image Processing (Second Edition)*.
Ed. by Fatima A. Merchant and Kenneth R. Castleman. Second Edition.
Academic Press, 2023, pp. 153–175. ISBN: 978-0-12-821049-9.
DOI: https://doi.org/10.1016/B978-0-12-821049-9.00017-4.

[59] Nisar Wani and Khalid Raza.
'Chapter 3 - Multiple Kernel-Learning Approach for Medical Image Analysis'.
In: *Soft Computing Based Medical Image Analysis*. Ed. by Nilanjan Dey et al.
Academic Press, 2018, pp. 31–47. ISBN: 978-0-12-813087-2.
DOI: https://doi.org/10.1016/B978-0-12-813087-2.00002-6.

[60] matematikk.org. *Vinkelen mellom to vektorer*.
URL: https://www.matematikk.org/artikkel.html?tid=192383&within_tid=154821.
(accessed: 11.06.2023).

[61] mathworks. *rotz*. URL: https://www.mathworks.com/help/phased/ref/rotz.html.
(accessed: 11.06.2023).

[62] Aleks Reinhardt and Jonathan P. K. Doye. 'Effects of surface interactions on heterogeneous ice nucleation for a monatomic water model'.
In: *The Journal of Chemical Physics* 141.8 (Aug. 2014). 084501. ISSN: 0021-9606.
DOI: 10.1063/1.4892804. eprint: https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/1.4892804/15482537/084501_1_online.pdf.

[63] Andrew H. Nguyen and Valeria Molinero. 'Identification of Clathrate Hydrates, Hexagonal Ice, Cubic Ice, and Liquid Water in Simulations: the CHILL+ Algorithm'.
In: *The Journal of Physical Chemistry B* 119.29 (2015). PMID: 25389702, pp. 9369–9376.
DOI: 10.1021/jp510289t. eprint: https://doi.org/10.1021/jp510289t.

Appendix

A frame_splitter_and_boundary_checker.py

```
# This .py file splits the original LAMMPS trajectory text
# files into seperate files containing only one frame
# It also gets rid of alle the molecules that does'nt fulfill
# the boundary condition

#This function splits the MD-simulation file into frames.
def frameSplitter(readfile,path):
    framenumber = 0
    f = open(readfile, "r")
    f2 = open(path+"/frame_"+str(framenumber)+".txt", "w")
    for line in f:
        #Closes the writing file and opens a new one to write te next frame on.
        if (line.strip('\n') == "ITEM: TIMESTEP"):
            f2.close()
            framenumber+=1
            f2 = open(path+"/frame_"+str(framenumber)+".txt", "w")
            f2.write(line)
        #Writes the lines to the file
        else:
           f2.write(line)
    f2.close()
    f.close()

def dataExtractor(path):
    #Goes through evry single frame
    for i in range(102):
        frame = path+"/frame_"+str(i)+".txt"
        f = open(frame, "r")
        f2 = open(path+"/temp", "w")
        atoms=0
        for j in range(9):
            line = f.readline()
            f2.write(line)
        for line in f:
                values=line.split()
                #Discards all molecules thats not within the boundary conditions.
                if (float(values[3])>4.5 and float(values[3])<32 and
                float(values[4])>7 and float(values[4])<30):
                    f2.write(line)
                    atoms+=1
        f.close()
        f2.close()
        f = open(frame, "w")
        f2 = open(path+"/temp", "r")
        #writes the atoms to a new file
        for k in range(3):
            line = f2.readline()
            f.write(line)
        f2.readline()
        f.write(str(atoms)+"\n")
        for line in f2:
            if line.strip("/n")!="":
                f.write(line)
```

```
        f.close()
        f2.close()

#The run function initializes the framSplitter function and
#dataExtractor function
def run (file, path):
    frameSplitter(file,path)
    dataExtractor(path)

#Path to the folder where the file is saved
path = "finalThree/growth"
#MD-simulation file
file = "finalThree/Ice-Amorphous-Hydrate-growth-500ps_230K.lammpstrj.txt"

#Remember to change the path, and file variable if you have saved the files
#elsewhere, or the file has a different name

#The function that runs the entire script
run (file, path)
```

B DBSCAN1.py

```
#imports
import numpy as np
import pandas as pd
from sklearn.cluster import DBSCAN


#This function uses DBSCAN to find the the water in frame n and frame n+1, then
# write the molecules that only apperar in frame n (and not in frame n+1) to
# a csv file
def dbscanClustering(frame1, frame2, writefile):
    with open(frame1,'r') as f:
        lines = f.readlines()
        lines = lines[9:]

    molecules = []
    for line in lines:
        values = line.split()
        values.pop(1)
        values.pop(-1)
        values.pop(-1)
        values.pop(-1)
        molecules.append(values)
    df = pd.DataFrame(molecules,columns=['id','x', 'y', 'z'])
    ogdf = df
    df2 = df.drop('id', axis=1)

    # Convert DataFrame to numpy array
    X = df2.values.astype(np.float)


    # create a DBSCAN object with the desired parameters
    eps = 4
    min_samples = 8
    dbscan = DBSCAN(eps=eps, min_samples=min_samples)

    # fit and predict the clusters
    dbscan.fit(X)
    clusters = dbscan.labels_

    # count the number of points in each cluster
    unique_clusters, counts = np.unique(clusters, return_counts=True)
    num_clusters = len(unique_clusters)


    # find the cluster with the largest number of points
    largest_cluster_idx = np.argmax(counts)
    largest_cluster_label = unique_clusters[largest_cluster_idx]
    largest_cluster_points = X[clusters == largest_cluster_label, :]

    df3 = pd.DataFrame(largest_cluster_points,columns=['x', 'y', 'z'])
    df['id'] = df['id'].astype(int)
    df['x'] = df['x'].astype(float)
    df['y'] = df['y'].astype(float)
    df['z'] = df['z'].astype(float)
    merged = pd.merge(df, df3, on=['x', 'y', 'z'])
```

```
with open(frame2,'r') as f:
    lines = f.readlines()
    lines = lines[9:]

molecules = []
for line in lines:
    values = line.split()
    values.pop(1)
    values.pop(-1)
    values.pop(-1)
    values.pop(-1)
    molecules.append(values)
df = pd.DataFrame(molecules,columns=['id','x', 'y', 'z'])
df2 = df.drop('id', axis=1)

# Convert DataFrame to numpy array
X = df2.values.astype(np.float)


# create a DBSCAN object with the desired parameters
eps = 4
min_samples = 8
dbscan = DBSCAN(eps=eps, min_samples=min_samples)

# fit and predict the clusters
dbscan.fit(X)
clusters = dbscan.labels_

# count the number of points in each cluster
unique_clusters, counts = np.unique(clusters, return_counts=True)
num_clusters = len(unique_clusters)


# find the cluster with the largest number of points
largest_cluster_idx = np.argmax(counts)
largest_cluster_label = unique_clusters[largest_cluster_idx]
largest_cluster_points = X[clusters == largest_cluster_label, :]

df3 = pd.DataFrame(largest_cluster_points,columns=['x', 'y', 'z'])
df['id'] = df['id'].astype(int)
df['x'] = df['x'].astype(float)
df['y'] = df['y'].astype(float)
df['z'] = df['z'].astype(float)
merged2 = pd.merge(df, df3, on=['x', 'y', 'z'])



ids_df1 = set(merged['id'])
ids_df2 = set(merged2['id'])
intermediates = list(ids_df2.difference(ids_df1))


matching_rows = ogdf[ogdf['id'].isin(intermediates)]

filtered_rows = matching_rows[
(matching_rows['x'] >= (307))
& (matching_rows['x'] <= 430)]
```

```
    filtered_rows.to_csv(writefile, index=False)

def run(path):
    for i in range (1,101):
        frame1 = path + 'frame_' + str(i) + '.txt'
        frame2 = path + 'frame_' + str(i+1) + '.txt'
        writefile = path + 'intermediates_' + str(i) + '_' + str(i+1) +
'.csv'
        dbscanClustering(frame1, frame2, writefile)
        print(i)

path = 'finalThree/growth/'

run(path)
```

C doubleDBSCAN.py

```
#imports
import numpy as np
import pandas as pd
from sklearn.cluster import DBSCAN

def potenialIntermediates(readfile,writefile,writefile2,largestcluster):
    df = pd.read_csv(readfile)
    df2 = df.drop('id', axis=1)

    # Convert DataFrame to numpy array
    X = df2.values.astype(np.float)

    # create a DBSCAN object with the desired parameters
    eps = 4 # maximum interaction distance between two water molecules
    min_samples = 2  #can be just a singel water molecule with no bonds
    dbscan = DBSCAN(eps=eps, min_samples=min_samples, n_jobs=1)

    # fit and predict the clusters
    dbscan.fit(X)
    clusters = dbscan.labels_

    # count the number of points in each cluster
    unique_clusters, counts = np.unique(clusters, return_counts=True)
    num_clusters = len(unique_clusters)

    # create a boolean mask to only keep clusters between 3 and 5 points.
    mask = (counts > 2)

    # get the indices of the points in clusters with more than one and less
    # than six points
    indices = np.where(mask)[0]
    point_indices = [np.where(clusters == i)[0] for i in unique_clusters[indices]]

    with open(writefile2,'w') as f:
        for cluster in point_indices:
            neighbours=[]
            for i in cluster:
                neighbours.append(list(df.iloc[i]))
            clustersize=len(neighbours)
            if clustersize>largestcluster:
                largestcluster=clustersize
            f.write(str(neighbours)+'\n')


    # create a new DataFrame with the selected points
    new_df = df.iloc[np.concatenate(point_indices)]


    new_df.to_csv(writefile, index=False)

    return largestcluster

def run(path):
    largestcluster=0
    for i in range(1,101):
        readfile = path + 'intermediates_' + str(i) + '_' + str(i+1) + '.csv'
```

```
        writefile = path + 'analyzed_intermediates_'+ str(i)
        + '_' + str(i+1) + '.csv'
        writefile2 = 'finalThree/method1/''analyzed_intermediates_'+ str(i)
        + '_' + str(i+1) + '.txt'
        largestcluster=potenialIntermediates(readfile, writefile,
        writefile2,largestcluster)
    print('largestcluster is: '+str(largestcluster))

path = 'finalThree/growth/'
run(path)
```

D method1_average_dist.py

```
#imports
import numpy as np
import os

#This function calculates the average distance from a neighbour to the center
# atom for an intermediate structure.
#It also calculates the sum of all neighbours to center distances in an
#intermediates structure.
#The variabel readfile contains the file to be analyzed,
# writefile is the file where intermeriates of each
# size found is written, and writefile2 contains all
# intermediate structures. The variable frame is the
# frame that is analyzed.
points=[]
numbers=[]
def average_distance(readfile, writefile,writefile2,frame):
    with open(readfile,'r') as f, open(writefile2,'a') as f3:
        for line in f:
            neighbours=np.array(eval(line),float)
            neighbours=neighbours.tolist()
            n = len(neighbours)
            total_distance = 0
            total_pairs = n * (n - 1) / 2
            atoms=[]
            for atom in neighbours:
                atoms.append(int(atom[0]))
            for i in range(n - 1):
                for j in range(i + 1, n):
                    id1, x1, y1 , z1 = neighbours[i]
                    id2, x2, y2, z2 = neighbours[j]
                    euclidean_distance = np.sqrt(
                    (x2 - x1) ** 2 + (y2 - y1) ** 2+(z2 - z1)**2)
                    total_distance += euclidean_distance
            average_distance = round(total_distance/total_pairs,4)
            info=[n,atoms,average_distance,round(total_distance,4),frame]
            info2=[n,atoms,frame]
            writefile = 'finalThree/method1/
____________intermediates_method1_n_1_c_'+str(n)+'.txt'
            f2 = open(writefile,'a')
            f2.write(str(info)+'\n')
            f2.close
            f3.write(str(info2)+'\n')

            if n==4:
                point=[average_distance,round(total_distance,4)]
                numbers.append(average_distance)
                points.append(point)


#average_distance('finalThree/method1/analyzed_intermediates_1_2.txt', '')

#This function initiates the script. The variable largest cluster is the
# number of molecules in the largest cluster that was found in the previouse step.
def run(largestcluster):
    for i in range(3,largestcluster+1):
```

```
        writefile = 'finalThree/method1/intermediates_method1_n_1_c_'
        +str(i)+'.txt'
        f = open(writefile,'w')
        f.close
    writefile2='finalThree/method1/intermediates_method1_n_1_allclusters.txt'
    f = open(writefile2,'w')
    f.close
    for i in range(1,101):
        readfile = 'finalThree/method1/
________analyzed_intermediates_'+str(i)+'_'+str(i+1)+'.txt'
        average_distance(readfile,writefile,writefile2,i)
    for i in range(3,largestcluster+1):
        if os.stat('finalThree/method1/
________intermediates_method1_n_1_c_'+str(i)+'.txt').st_size == 0:
            os.remove('finalThree/method1/
____________intermediates_method1_n_1_c_'+str(i)+'.txt')

run(75)
```

E convex_hull_method1.py

```
#imports
import numpy as np
from scipy.spatial import ConvexHull

#This function takes in a set of points, makes a convex hull out of these points,
# then calculte and returns the volume of this convex hull
def calculate_volume(points):
    hull = ConvexHull(points)

    # Get the vertices and simplices of the convex hull
    vertices = hull.points[hull.vertices]
    simplices = hull.simplices

    # Calculate the volume using the decomposition into tetrahedra
    volume = 0.0
    for simplex in simplices:
        tetrahedron = vertices[simplex]
        signed_volume = np.dot(tetrahedron[0],
        np.cross(tetrahedron[1]-tetrahedron[0], tetrahedron[2]-tetrahedron[0]))
        volume += signed_volume / 6.0

    return abs(volume)


#This function does a convexhull analysis. It takes in the file
# (readfile) to perform this analysis, then writes all the results too the writefile,
# the size 4 intermediates to writefile2, the size 5 intermediates to writefile3.
#Some intermediates can not be used to perform a convexhull analysis,
# these will be written to the errorfile.
def analyze(readfile,writefile,writefile2,writefile3,errorfile):
    with (open(writefile,'w') as f2, open(writefile2,'w') as f3,
    open(writefile3,'w') as f4, open(errorfile,'w') as f5):
        for i in range(1,101):
            print(i)
            readfile1=readfile+str(i)+'_'+str(i+1)+'.txt'
            with open(readfile1,'r') as f:
                for line in f:
                    points=[]
                    molecules=np.array(eval(line),float)
                    molecules=molecules.tolist()
                    ids=[]
                    for mol in molecules:
                        id=int(mol.pop(0))
                        points.append(mol)
                        ids.append(id)
                    if len(points)>3:
                        try:
                            calculate_volume(points)
                        except:
                            print(points)
                            f5.write(str(len(points))+': '+line)
                        else:
                            volume=calculate_volume(points)
                            density= len(ids)/volume
                            # i indicates the frame
                            info=[len(ids),ids,volume,density,i]
```

```
                                f2.write(str(info)+'\n')
                                if len(ids)==4:
                                    f3.write(str(info)+'\n')
                                if len(ids)==5:
                                    f4.write(str(info)+'\n')

'''
analyze('finalThree/method1/analyzed_intermediates_',
'finalThree/method1/convex_hull_all.txt',
'finalThree/method1/convex_hull_c_4.txt',
'finalThree/method1/convex_hull_c_5.txt',
'finalThree/method1/convex_hull_errors.txt')
'''
```

F DBSCAN2.py

```
#imports
import numpy as np
import pandas as pd
from sklearn.cluster import DBSCAN


def water(frame, writefile):
    # Read the file to be analysed
    with open(frame,'r') as f:
        lines = f.readlines()
        lines = lines[9:]

    molecules = []
    for line in lines:
        values = line.split()
        values.pop(1)
        values.pop(-1)
        values.pop(-1)
        values.pop(-1)
        molecules.append(values)
    # Store the values fram the txt file in a dataframe called df
    df = pd.DataFrame(molecules,columns=['id','x', 'y', 'z'])
    # copy df, but neglect the id's
    df2 = df.drop('id', axis=1)

    # Convert DataFrame to numpy array
    X = df2.values.astype(np.float)


    # create a DBSCAN object with the desired parameters
    eps = 4
    min_samples = 8
    dbscan = DBSCAN(eps=eps, min_samples=min_samples)

    # fit and predict the clusters
    dbscan.fit(X)
    clusters = dbscan.labels_

    # count the number of points in each cluster
    unique_clusters, counts = np.unique(clusters, return_counts=True)

    # find the cluster with the largest number of points
    largest_cluster_idx = np.where((unique_clusters != -1) &
    (counts == max(counts[unique_clusters != -1])))[0][0]
    largest_cluster_label = unique_clusters[largest_cluster_idx]
    largest_cluster_points = X[clusters == largest_cluster_label, :]

    # Create dataframes
    df3 = pd.DataFrame(largest_cluster_points,columns=['x', 'y', 'z'])
    df['id'] = df['id'].astype(int)
    df['x'] = df['x'].astype(float)
    df['y'] = df['y'].astype(float)
    df['z'] = df['z'].astype(float)

    # merge dataframe df and df3
    merged = pd.merge(df, df3, on=['x', 'y', 'z'])
```

```
    # Write the dataframe to a csv file
    merged.to_csv(writefile, index=False)

#The run function, this function loops through every fram en applies
#  the water function on them
def run(path):
    for i in range(1, 102):
        frame = path + '/frame_' + str(i) + '.txt'
        writefile = path + '2/water_' + str(i) + '.csv'
        water(frame, writefile)

#The path to the folder where everything can be found.
path = 'finalThree/growth'

#The run function initiate the entire script
run(path)
```

G water_neighbours.py

```
#imports
import pandas as pd
import numpy as np

#This function takes in a frame containg all molecules in the liquid state,
# then finds the neighbours of each molecule within a distance 4,
# and writes the results to a textfile writefile.

def neighbours(readfile, writefile):
    df = pd.read_csv(readfile)
    df2 = pd.read_csv(readfile)
    container =[]


    def neighbourhood(row):
        neighbours=[]
        x=float(row[1])
        y=float(row[2])
        z=float(row[3])
        point =np.array((x,y,z))
        molecule = [int(row[0]),x,y,z]
        neighbours.append(molecule)
        for atom in df2.itertuples(index=False):
            x=float(atom[1])
            y=float(atom[2])
            z=float(atom[3])
            point2 =np.array((x,y,z))
            distance = np.linalg.norm(point-point2)
            if (distance !=0 and distance <=4):
                molecule=[int(atom[0]),x,y,z]
                neighbours.append(molecule)
        container.append(neighbours)

    df.apply(neighbourhood, axis=1)

    with open(writefile,'w') as f:
        for element in container:
            f.write(str(element)+'\n')

#Initiates the entire script, path is the where the files that will be
# analyzed can be found.
def run(path):
    for i in range(1, 102):
        readfile = path + '/water_' + str(i) + '.csv'
        writefile = 'finalThree/method3.2/water_neighbours_' + str(i) + '.txt'
        neighbours(readfile, writefile)
        print(i)

path = 'finalThree/growth2'

run(path)
```

H new_coordinates.py

```
import numpy as np

# This function makes a new coordinate system according to the new
# center molecule
def coordinates(readfile, writefile):
    f = open(readfile, 'r')
    f2 = open(writefile, 'w')
    for line in f:
        neighbours=np.array(eval(line),float)
        neighbours=neighbours.tolist()
        if len(neighbours)!=1:
            center = neighbours.pop(0)
            centerX=center[1]
            centerY=center[2]
            centerZ=center[3]
            newCords=[]
            centerMol = [int(center[0]),0.0, 0.0, 0.0]
            newCords.append(centerMol)
            for molecule in neighbours:
                newX=molecule[1]-centerX
                newY=molecule[2]-centerY
                newZ=molecule[3]-centerZ
                newMol = [int(molecule[0]), round(newX,6), round(newY,6)
                ,round(newZ,6)]
                newCords.append(newMol)
            f2.write(str(newCords)+'\n')
    f.close()
    f2.close()

#initiates the entire script
def run (path):
    for i in range (1,102):
        readfile = path + '/water_neighbours_' + str(i) + '.txt'
        writefile = path + '/water_neighbours_' + str(i) + '_new_coordinates.txt'
        coordinates(readfile, writefile)
        print(i)

path1 = "finalThree/method3.2"

run(path1)
```

I rotater.py

```
#imports
import numpy as np
import pandas as pd
import math


#This function takes the coordinates of two molecules and calculates the
# rotatin angel between them
def calculate_rotation_angle_direction(x1, y1, z1, x2, y2, z2):
    # Vector A at t=0
    A0 = np.array([x1, y1, z1])
    # Vector A at t=1
    A1 = np.array([x2, y2, z2])

    # Normalize vectors
    A0_normalized = A0 / np.linalg.norm(A0)
    A1_normalized = A1 / np.linalg.norm(A1)

    # Compute cross product and rotation angle
    cross_product = np.cross(A0_normalized, A1_normalized)
    angle = math.acos(np.dot(A0_normalized, A1_normalized))

    # Determine rotation direction
    rotation_direction = 1 if np.dot(cross_product, A1_normalized) > 0 else -1

    return angle, rotation_direction

# This function performs the rotational shift in a neighbourhood.
def analyzer(readfile, readfile2, writefile):
    f = open(readfile, 'r')
    f3 = open(writefile, 'w')
    holder = []

    for line in f:
        neighbours=np.array(eval(line),float)
        neighbours=neighbours.tolist()
        f2 = open(readfile2, 'r')

        for line2 in f2:
            neighbours2=np.array(eval(line2),float)
            neighbours2=neighbours2.tolist()
            #Find the matching center atoms in both frames
            if neighbours[0][0]== neighbours2[0][0]:
                center1=neighbours.pop(0)
                center2=neighbours2.pop(0)
                columnNames=['id','x','y','z']
                df = pd.DataFrame(neighbours,columns=columnNames)
                df2 = pd.DataFrame(neighbours2,columns=columnNames)
                merged_df = pd.merge(df, df2, on='id', how='inner')
                x1 = merged_df.iloc[1]['x_x']
                y1 = merged_df.iloc[1]['y_x']
                z1 = merged_df.iloc[1]['z_x']
                x2 = merged_df.iloc[1]['x_y']
                y2 = merged_df.iloc[1]['y_y']
                z2 = merged_df.iloc[1]['z_y']
                theta, rotation_direction = calculate_rotation_angle_direction(x1,
```

```
                y1, z1, x2, y2, z2)
                #we want to rotate the atoms back so we multiply
                # with rotation_direction
                rotation_direction = rotation_direction*(-1)
                angle = round(np.degrees(theta),2)
                df3 = merged_df.drop(0)
                df4 = pd.DataFrame(columns=['id', 'x', 'y','z','angle'])
                center2.append(angle)
                df4.loc[len(df4)] = center2
                rotated = [[int(center2[0]),center2[1],center2[2]
                ,center2[3],center2[4]]]

                def rotate(row):
                    # Define the 3D vector
                    vector = np.array([row[4],row[5],row[6]])


                    # Define the rotation matrix
                    rot_z = np.array([
                        [np.cos(theta), -rotation_direction*np.sin(theta), 0],
                        [rotation_direction*np.sin(theta), np.cos(theta), 0],
                        [0, 0, 1]
                    ])  # 3x3 rotation matrix around z-axis

                    # Perform the rotation
                    vector_rotated = np.dot(rot_z, vector)
                    displacement =np.sqrt(
                    (row[4]-vector_rotated[0])**2+(row[5]-vector_rotated[1])**2
                    +(row[6]-vector_rotated[2])**2)
                    vr = [row[0],vector_rotated[0],vector_rotated[1],
                    vector_rotated[2],angle]
                    df4.loc[len(df4)] = vr
                    rotated.append(vr)


                df3.apply(rotate, axis=1)
                f3.write(str(rotated)+'\n')
                f2.close()
                break
            #holder.append(rotated)

f.close()
f3.close()
```

J method2_average_dist.py

```
#imports
import numpy as np
import pandas as pd

#This function calculates the average distance from a neighbour to the
# center atom for an intermediate structure.
# It also calculates the displacement from frame n to frame n+1
# in an intermediates structure.
#The variabel theta is largest angular displacemen that is allowed, and r is the
# maximum distance allowed between a neighbour and the center molecule.
def displacement(theta,r):
    writefile = 'finalThree/method2/distances.txt'
    writefile2 = 'finalThree/method2/avg_dist.txt'
    writefile3 = 'finalThree/method2/avg_dist_c3.txt'
    writefile4 = 'finalThree/method2/avg_dist_c4.txt'
    writefile5 = 'finalThree/method2/avg_dist_c5.txt'
    f3 = open(writefile, 'w')
    f4 = open(writefile2, 'w')
    f5 = open(writefile3, 'w')
    f6 = open(writefile4, 'w')
    f7 = open(writefile5, 'w')

    for i in range(1,101):
        readfile1 = 'finalThree/growth2/rotated_'+str(i)+'_'+str(i+1)+'.txt'
        readfile2 = 'finalThree/growth2/water_neighbours_'+str(i+1)
        +'_new_coordinates.txt'
        f = open(readfile1, 'r')
        for line in f:
            neighbours=np.array(eval(line),float)
            neighbours=neighbours.tolist()
            if neighbours[0][4]<=theta:
                f2 = open(readfile2, 'r')
                for line2 in f2:
                    neighbours2=np.array(eval(line2),float)
                    neighbours2=neighbours2.tolist()
                    if neighbours[0][0]== neighbours2[0][0]:
                        columnNames=['id','x','y','z', 'angle']
                        columnNames2=['id','x','y','z']
                        df = pd.DataFrame(neighbours,columns=columnNames)
                        df2 = pd.DataFrame(neighbours2,columns=columnNames2)
                        merged_df = pd.merge(df2, df, on='id', how='inner')
                        distances=[]
                        total_d=0
                        total_center_distance=0
                        ids=[]
                        data=[]
                        for index,row in merged_df.iterrows():
                            x1 = row[1]
                            y1 = row[2]
                            z1 = row[3]
                            x2 = row[4]
                            y2 = row[5]
                            z2 = row[6]
                            #displacement from previous frame
                            d = round(np.sqrt((x2 - x1)**2 + (y2 - y1)**2
                            + (z2 - z1)**2),4)
```

```
                distance_to_center = round(np.sqrt((x1 - 0)**2 +
                (y1 - 0)**2 + (z1 - 0)**2),4)
                atom =[int(row[0]),x2,y2,z2,d, distance_to_center]
                distances.append(atom)
                if distance_to_center<=r:
                    ids.append(int(row[0]))
                    total_center_distance+=distance_to_center
                    total_d+=d
            if len(ids)>0:
                average_center_distance=round(
                total_center_distance/len(ids),4)
                info=[len(ids),ids,average_center_distance
                ,round(total_d,4)]
                f4.write(str(info)+'\n')
                if len(ids)==3:
                    f5.write(str(info)+'\n')
                if len(ids)==4:
                    f6.write(str(info)+'\n')
                if len(ids)==5:
                    f7.write(str(info)+'\n')



            f3.write(str(distances)+'\n')
            f2.close()
            break
        print(i)
    f.close()
    f3.close()
    f4.close()
    f5.close()
    f6.close()
    f7.close()


displacement(10,4)
```

K angle.py

```
#imports
import numpy as np
import pandas as pd

#This function takes in an vector and returns the unit vector
def unit_vector(vector):
    """ Returns the unit vector of the vector."""
    return vector / np.linalg.norm(vector)

#This function calculates the angular displacement in a neighbourhood from frame n
# and frame n+1.
#The variabel readfile contains frame n, while readfile2 contains frame n+1
#the variable writefile is the name of the file where the angular displacement
# between two frames is saved.
#The variable theta is the the maximum angle that is allowed as an angular
# dispalcement for a molecule.
def analyzer(readfile, readfile2, writefile,theta):
    with open(readfile, 'r') as f, open(writefile, 'w') as f3:
        for line in f:
            neighbours=np.array(eval(line),float)
            neighbours=neighbours.tolist()
            f2 = open(readfile2, 'r')
            center1=neighbours.pop(0)
            for line2 in f2:
                neighbours2=np.array(eval(line2),float)
                neighbours2=neighbours2.tolist()
                center2=neighbours2.pop(0)
                if center1[0]== center2[0]:
                    columnNames=['id','x','y','z']
                    df = pd.DataFrame(neighbours,columns=columnNames)
                    df2 = pd.DataFrame(neighbours2,columns=columnNames)
                    merged_df = pd.merge(df, df2, on='id', how='inner')
                    #df3 = pd.DataFrame(columns=['id', 'x1', 'y1','z1','x2','y2'
                    ,'z2','d1','d2','angle'])
                    for i in range(6):
                        center2.append(0)
                    #df3.loc[len(df3)] = center2
                    data=[center2]
                    for index, row in merged_df.iterrows():
                        info=list(row)
                        x1 = row[1]
                        y1 = row[2]
                        z1 = row[3]
                        vector1 = [x1,y1,z1]
                        x2 = row[4]
                        y2 = row[5]
                        z2 = row[6]
                        vector2 = [x2,y2,z2]
                        unitvec1 = unit_vector(vector1)
                        unitvec2 = unit_vector(vector2)
                        angle_rad = np.arccos(np.clip(np.dot(unitvec1, unitvec2)
                        , -1.0, 1.0))
                        angle = round(np.degrees(angle_rad),2)
                        if angle<=theta:
                            euclidean_distance_1=np.sqrt(x1**2+y1**2+z1**2)
                            info.append(euclidean_distance_1)
```

```
                                euclidean_distance_2=np.sqrt(x2**2+y2**2+z2**2)
                                info.append(euclidean_distance_2)
                                info.append(angle)
                                data.append(info)
                        if len(data)>1:
                            f3.write(str(data)+'\n')
                            #df3.loc[len(df3)] = info

#The run function initiates the entire script
#The variable path is a string that conatins the path to the folder
# where the files that will be analyzed is contained, and where the analyzed
# files will be saved.
#The variable theta is the the maximum angle that is allowed as an angular
# dispalcement for a molecule.
def run (path, theta):
    for i in range (2,102):
        readfile = path + '/water_neighbours_' + str(i) + '_new_coordinates.txt'
        readfile2 = path + '/water_neighbours_' + str(i+1) + '_new_coordinates.txt'
        writefile = path+'/angle'+str(i)+'_'+str(i+1)+'.txt'
        analyzer(readfile, readfile2, writefile, theta)
        print(i)


path1 = "finalThree/method3.2"

run(path1, 10)
```

L method3_average_dist_and_angle.py

```
#imports
import numpy as np

#This function calculates the average distance from a neighbor to the center
# molecule for an intermediate structure.
# It also calculates the average angular displacement for
# an intermediates structure.
#The variable path contains information about where the folder containing
# the files that should be analyzed are contained.
def analyse(path):
    writefile = path + 'intermediates_method_3_avg_dist_and_angle_all.txt'
    writefile2 = path + 'intermediates_method_3_avg_dist_and_angle_c_3.txt'
    writefile3 = path + 'intermediates_method_3_avg_dist_and_angle_c_4.txt'
    writefile4 = path + 'intermediates_method_3_avg_dist_and_angle_c_5.txt'
    with open(writefile,'w') as f, open(writefile2,'w') as f2
    , open(writefile3,'w') as f3, open(writefile4,'w') as f4:
        for i in range(1,100):
            readfile='finalThree/method3.2/angle'+str(i)+'_'+str(i+1)+'.txt'
            with open(readfile,'r') as f5:
                for line in f5:
                    total_distance=0
                    total_angle=0
                    molecules=np.array(eval(line),float)
                    molecules=molecules.tolist()
                    if len(molecules)>2:
                        center=molecules.pop()
                        ids=[int(center[0])]
                        for atom in molecules:
                            ids.append(int(atom[0]))
                            total_distance+=atom[7]
                            total_angle+=atom[9]
                        average_disance=round(total_distance/(len(ids)-1),4)
                        average_angle=round(total_angle/(len(ids)-1),4)
                        info=[len(ids),ids,average_disance,average_angle,i]
                        f.write(str(info)+'\n')
                        if len(ids)==3:
                            f2.write(str(info)+'\n')
                        if len(ids)==4:
                            f3.write(str(info)+'\n')
                        if len(ids)==5:
                            f4.write(str(info)+'\n')

analyse('finalThree/method3.2/')
```

M convex_hull_method3.py

```
import numpy as np
from scipy.spatial import ConvexHull

#This function takes in a set of points, makes a convex hull out of these points,
# then calculte and returns the volume of this convex hull
def calculate_volume(points):
    hull = ConvexHull(points)

    # Get the vertices and simplices of the convex hull
    vertices = hull.points[hull.vertices]
    simplices = hull.simplices

    # Calculate the volume using the decomposition into tetrahedra
    volume = 0.0
    for simplex in simplices:
        tetrahedron = vertices[simplex]
        signed_volume = np.dot(tetrahedron[0],
        np.cross(tetrahedron[1]-tetrahedron[0], tetrahedron[2]-tetrahedron[0]))
        volume += signed_volume / 6.0

    return abs(volume)


#This function does a convexhull analysis. It takes in the file (readfile)
# to perform this analysis, then writes all the results too the
# writefile, the size 4 intermediates to writefile2,
# the size 5 intermediates to writefile3. Some intermediates can
# not be used to perform a convexhull analysis, these will
# be written to the errorfile.
def analyze(readfile,writefile,writefile2,writefile3,errorfile):
    with (open(writefile,'w') as f2,
    open(writefile2,'w') as f3, open(writefile3,'w') as f4,
     open(errorfile,'w') as f5):
        for i in range(1,101):
            print(i)
            readfile1=readfile+str(i)+'_'+str(i+1)+'.txt'
            with open(readfile1,'r') as f:
                for line in f:
                    points=[]
                    molecules=np.array(eval(line),float)
                    molecules=molecules.tolist()
                    ids=[]
                    for mol in molecules:
                        point=[mol[4],mol[5],mol[6]]
                        id=int(mol.pop(0))
                        points.append(point)
                        ids.append(id)
                    if len(points)>3:
                        try:
                            calculate_volume(points)
                        except:
                            f5.write(str(len(points))+': '+line)
                        else:
                            volume=round(calculate_volume(points),4)
                            density= round(len(ids)/volume,4)
                            # i indicates the frame
```

```
                                info=[len(ids),ids,volume,density,i]
                                f2.write(str(info)+'\n')
                                if len(ids)==4:
                                    f3.write(str(info)+'\n')
                                if len(ids)==5:
                                    f4.write(str(info)+'\n')

'''
analyze('finalThree/method3.2/angle',
'finalThree/method3.2/convex_hull_all.txt',
'finalThree/method3.2/convex_hull_c_4.txt',
'finalThree/method3.2/convex_hull_c_5.txt',
'finalThree/method3.2/convex_hull_errors.txt')
'''
```

N kmeans.py

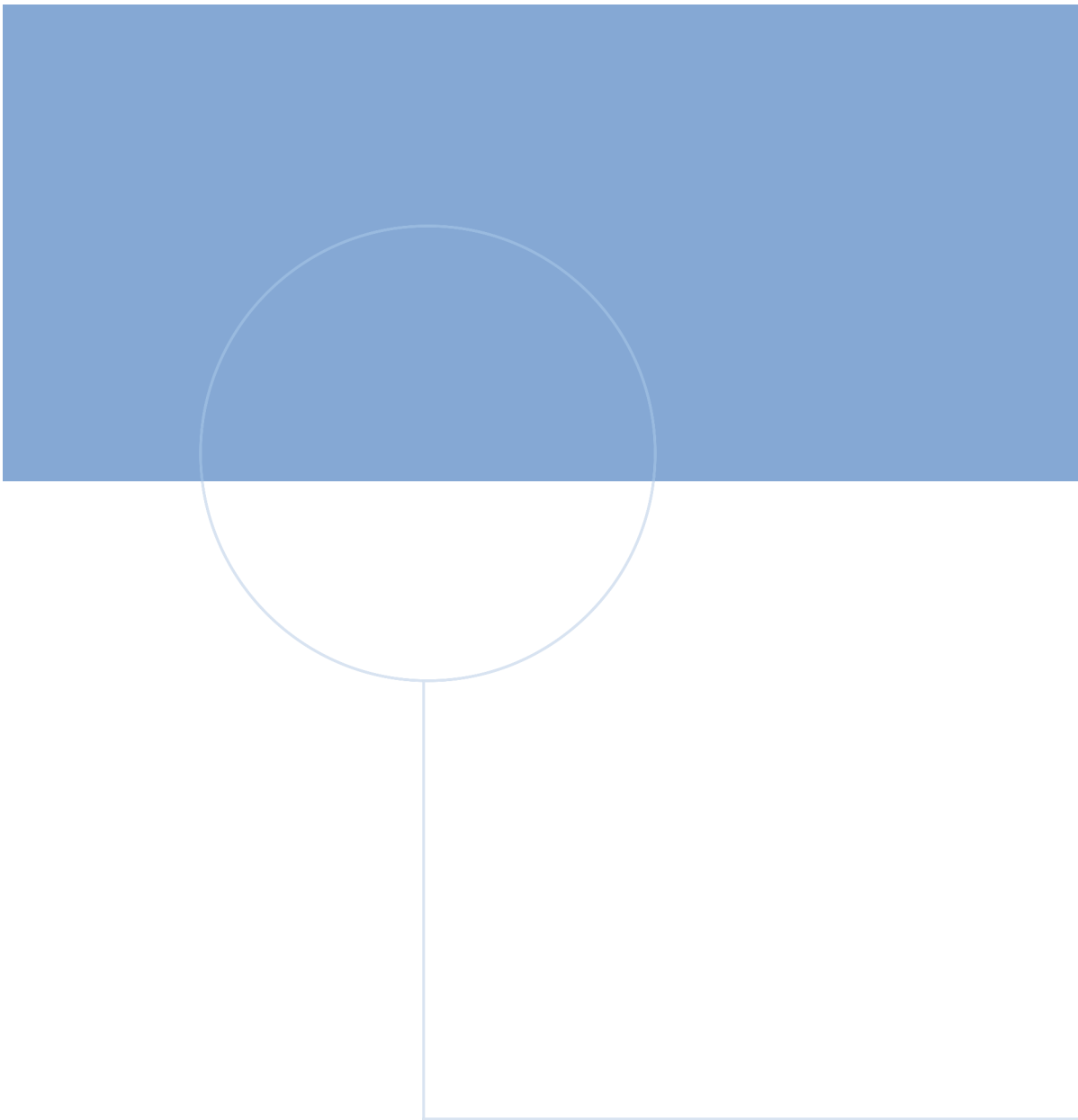
```
#imports
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler

#This function performs the k means clustering.
#The variable readfile is the file that will be analyzed by the function.
#The variable convex_hull is a boolean that is True if a convex_hull file is
# inputed as a readfile.
def kmeans(readfile, convex_hull):
    with open(readfile,'r') as f:
        data=[]
        volumes=[]
        for line in f:
            info = eval(line)
            values=[info[2],info[3]]
            volumes.append(info[2])
            data.append(values)

    # Separate the x and y coordinates into separate lists
    x = [point[0] for point in data]
    y = [point[1] for point in data]

    # Create a scatter plot
    plt.scatter(x, y)

    # Add labels and title
    if convex_hull:
        plt.xlabel('Volume')
        plt.ylabel('Density')
    else:
        plt.xlabel('Average distance')
        plt.ylabel('Total distance')
    plt.title('Scatter Plot')

    # Display the plot
    plt.show()

     # Create an instance of StandardScaler
    scaler = StandardScaler()

    # Fit and transform the data

    # Convert the points to a NumPy array
    if convex_hull!=True:
        normalized_data = scaler.fit_transform(data)
        X = np.array(normalized_data)
    else:
        X = np.array(data)
    X2 = np.array(data)

    # Define the range of cluster numbers to evaluate
    min_clusters = 2
    max_clusters = 9
```

```
# Perform silhouette analysis for different cluster numbers
best_score = -1
optimal_clusters = -1

#For the silhouette analysis
for k in range(min_clusters, max_clusters+1):
    # Perform k-means clustering
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(X)
    labels = kmeans.labels_

    # Compute the silhouette score
    score = silhouette_score(X, labels)

    # Update the best score and optimal number of clusters
    if score > best_score:
        best_score = score
        optimal_clusters = k

# Define the number of clusters
k = optimal_clusters

# Perform k-means clustering
kmeans = KMeans(n_clusters=k)
kmeans.fit(X)

# Get the cluster labels and cluster centers
labels = kmeans.labels_
centers = kmeans.cluster_centers_

# Count the number of points in each cluster
counts = np.bincount(labels)

# Plot the points with different colors for each cluster
plt.scatter(X2[:, 0], X2[:, 1], c=labels)

# Plot the cluster centers
if convex_hull!=True:
    #Scale back the centers to fit the original data (non normalized data)
    centers=scaler.inverse_transform(centers)
plt.scatter(centers[:, 0], centers[:, 1], c='red', marker='x')

# Add labels and title
if convex_hull:
    plt.xlabel('Volume')
    plt.ylabel('Density')
else:
    plt.xlabel('Average distance')
    plt.ylabel('Total distance')
plt.title('Scatter Plot')

# Display the number of points in each cluster
for i, count in enumerate(counts):
    plt.text(centers[i, 0], centers[i, 1], f'Count: {count}'
    , color='black', ha='center')

# Display the plot
```

```
    plt.show()

#method1
#kmeans('finalThree/method1/convex_hull_all.txt',True)
#kmeans('finalThree/method1/convex_hull_c_4.txt',True)
#kmeans('finalThree/method1/convex_hull_c_5.txt',True)
#kmeans('finalThree/method1/intermediates_method1_n_1_c_3.txt',False)
#kmeans('finalThree/method1/intermediates_method1_n_1_c_4.txt',False)
#kmeans('finalThree/method1/intermediates_method1_n_1_c_5.txt',False)

#Method2
#kmeans('finalThree/method2/avg_dist_c3.txt',False)
#kmeans('finalThree/method2/avg_dist_c4.txt',False)
#kmeans('finalThree/method2/avg_dist_c5.txt',False)



#Method3
#kmeans('finalThree/method3.2/intermediates_method_3_avg_dist_and_angle_c_3.txt'
,False)
#kmeans('finalThree/method3.2/intermediates_method_3_avg_dist_and_angle_c_4.txt'
,False)
#kmeans('finalThree/method3.2/intermediates_method_3_avg_dist_and_angle_c_5.txt'
,False)
#kmeans('finalThree/method3.2/convex_hull_all.txt',True)
#kmeans('finalThree/method3.2/convex_hull_c_4.txt',True)
#kmeans('finalThree/method3.2/convex_hull_c_5.txt',True)
```

O life_cycle_simulation.py

```
#imports
import numpy as np

#This writes a trajectory file that simulate the "life" of a #
intermediate structure
#The variable cluster is the intermediate structure that will be simulated
#The variable writefile is a string that contains the information on where the
# trajectory file will be saved and the name of the new trajectory file
def simulate(cluster, writefile):
    atoms=cluster[1]
    with open(writefile,'w') as f:
        for i in range(1,102):
            atoms_frame_info=[]
            lines_to_write=[]
            readfile='finalThree/growth/frame_'+str(i)+'.txt'
            with open(readfile,'r') as f2:
                lines=f2.readlines()
                first=lines[:3]
                second=lines[4:9]
                data=lines[9:]
                for atom in atoms:
                    for line in data:
                        info=line.split()
                        if int(atom)==int(info[0]):
                            atoms_frame_info.append([int(info[0]),
                            float(info[2]),float(info[3]),float(info[4])])
                for atom2 in atoms_frame_info:
                    x=atom2[1]
                    y=atom2[2]
                    z=atom2[3]
                    for line2 in data:
                        info=line2.split()
                        x2=float(info[2])
                        y2=float(info[3])
                        z2=float(info[4])
                        distance=np.sqrt((x2-x)**2+(y2-y)**2+(z2-z)**2)
                        if distance<=3.5:
                            lines_to_write.append(line2)
            for l in first:
                f.write(l)
            f.write(str(len(lines_to_write))+'\n')
            for l in second:
                f.write(l)
            for l in lines_to_write:
                f.write(l)
            print(i)

simulate([4, [20290, 14882, 13457, 15448], 1.939, 3.68, 37]
, 'finalThree/method3.2/testCycleIce4.txt')
```



NTNU

Norwegian University of
Science and Technology