Understanding Ion Rejection Mechanism of Freeze Desalination by Molecular Dynamics Simulation

Mahbuba Jannat
Utah State University

UNDERSTANDING ION REJECTION MECHANISM OF FREEZE DESALINATION BY MOLECULAR DYNAMICS SIMULATION

by

Mahbuba Jannat

A thesis paper submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Mechanical Engineering

Approved:

Hailei, Wang, Ph.D.	Nicholas Roberts, Ph.D.
Major professor	Committee member
Yanqing Su, Ph.D.	D. Richard Cutler, Ph.D.
Committee member	Vice Provost of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2024

ABSTRACT

Understanding Ion Rejection Mechanism of Freeze Desalination by Molecular Dynamics Simulation

by

Mahbuba Jannat, Master of Science

Utah State University, 2024

Major Professor: Dr. Hailei Wang
Department: Mechanical and Aerospace Engineering

Despite its promise for producing both drinking and agricultural water efficiently, freeze desalination as a method to purify water by forming ice crystals to segregate salt ions remains mostly untapped. While some works have been done, its underlying mechanisms of ion rejection remain unclear. This research aims to improve understanding of ion rejection mechanisms in freeze desalination using molecular dynamics simulations. Specifically, the study attempts to elucidate the process by evaluating various parameters under different ensembles and temperature conditions. To find the ion rejection rate, the tetrahedrality parameter is calculated to find the liquid-solid interface at each timestep. Additionally, the radial distribution function was assessed, and free energy was calculated using umbrella sampling. Results showed that ion rejection rate is exceeding 90%, with the largest ice growth in the simulation at 235K with NPxT used as the ensemble. The free energies for Na and Cl ions were found to be 8.6 & 8.81 kcal/mol, indicating the Cl ions have higher free energy barrier to escape from the ice structure. As a result, the simulations have shown the Na ions are more effectively rejected into the brine solution. Furthermore, the radial distribution values were found to be 7.5 for Na ion and 3.9 for Cl ion.

(101 pages)

PUBLIC ABSTRACT

Understanding Ion Rejection Mechanism of Freeze Desalination by Molecular Dynamics Simulation

Mahbuba Jannat

This study explores a method called freeze desalination, which uses the natural process of ice formation to remove salt from water at lower than freezing temperature of water, which is 235K (Freezing temperature of this water model is 249K), making it safe for potable water. Unlike traditional methods, freeze desalination can be more efficient and environmentally friendly, but how it rejects salt at the molecular level is not very clear to understand. Using molecular dynamics simulation, this research aims to uncover the details of this process. We focused on understanding how water molecules interact with salt ions during freezing. Our findings showed that over 90% of salt ions can be successfully removed from the water under optimal conditions, with certain temperatures and settings leading to the best results. Specifically, we found that sodium ions are removed more effectively than chloride ions. This difference is due to how sodium and chloride ions interact with the surrounding water molecules, which was measure by calculation of free energies. These insights are crucial for improving the technique and making it a viable option for purifying water in different settings. Overall, this work provides a clearer picture of how freeze desalination works at a microscopic level, offering guidance on how to optimize this process for better performance. This could have significant implications for water purification technology, potentially leading to more widespread use of freeze desalination in the future.

ACKNOWLEDGMENTS

I am immensely thankful to my major professor, Dr. Hailei Wang, whose guidance, wisdom, and unwavering support have been a guiding light throughout my academic journey. His mentorship has not only shaped my work but has also profoundly influenced my personal and professional growth. I would also like to express my deepest gratitude to Adam Rasmussen from ETRI lab, whose invaluable assistance and expertise in simulations significantly contributed to the success of my work. His dedication and insights were instrumental in navigating the complexities of this project. All LAMMPS simulations were completed by using the resources of Center for High Performing Computers (CHPC) from University of Utah. I am grateful to have access to such incredible facility.

This journey would not have been the same without these remarkable professors, Dr. Yanqin Su and Dr. Nick Roberts. Their collective contributions have left an indelible mark on my work. Finally, I would like to thank my parents, Rawshan Ara & Reaz Uddin and my husband, Abdullah Al Mamun to be always there for me. I could not do this without their constant support.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Page

NOMENCLATURE

Acronyms

Cl^-	=	Chloride Ion
FD	=	Freeze Desalination
LAMMPS	=	Large-scale Atomic/Molecular Massively Parallel Simulator
LJ	=	Lennard-Jones
MD	=	Molecular Dynamics
NaCl	=	Sodium Chloride
Na^+	=	Sodium Ion
NPT	=	Isothermal-Isobaric Ensemble
NVT	=	Canonical Ensemble
PMEMD	=	Particle Mesh Ewald Molecular Dynamics
TIP4P/2005	=	Modified version of TIP4P model for general use

CHAPTER 1

INTRODUCTION

1.1 Motivation of Research

With the worldwide population and industrial activity on the rise, water scarcity is a growing concern. Issues such as climate change and inadequate management strategies in the past few decades have resulted in 4 billion individuals facing water scarcity for at least one month each year [1]. Moreover, roughly 700 million people continually struggle with water scarcity [2]. Recent analysis of groundwater systems highlights an alarming increase in depletion rates in several regions, including South-East Asia, the Middle East, and Central and North America [3]. Desalination is considered a comparatively modern solution, which could pave the way for more efficient desalination technologies to meet global water needs [2]. Essentially, desalination is the process of removing salt and other impurities from seawater or brackish water to produce fresh water suitable for human consumption and irrigation. While the concept is not new, having been practiced in some form for centuries, advancements in technology have made it increasingly efficient and sustainable. There are two primary methods employed in modern desalination: reverse osmosis and thermal. Reverse osmosis (RO), where water is forced through semi-permeable membranes to separate it from salts, and thermal distillation, which involves heating seawater to create vapor and then condensing it to produce fresh water. As global populations continue to grow and climate change exacerbates drought conditions in many regions, the demand for fresh water is expected to surge. Desalination plants, especially in arid and coastal areas, are playing a crucial role in bridging the gap between demand and supply. While the process is energy-intensive and has environmental considerations, ongoing research and innovation are paving the way for more eco-friendly and cost-effective solutions in the realm of desalination [4]. As research is going on desalination, there is another approach called freeze desalination (FD). FD removes salts from saline water, producing purified ice and a concentrated brine byproduct [4,6,7]. Despite advances in FD systems, a harmony on the rates of salt or brine rejection is yet to be achieved, and the molecular mechanisms that drive FD remain unclear [6,7]. To investigate ion rejection and molecular mechanisms that drives freeze desalination, the ion rejection phenomenon is to be investigated. The focus on ion rejection is about pushing the boundaries of what's known in freeze desalination, aiming to improve the technology both in terms of its fundamental understanding and its practical application.

Ion rejection is a natural phenomenon occurring during the freezing of water-based solutions. Unlike methods that rely on heat or pressure to separate salts from water, freeze desalination leverages the natural tendency of ice crystals to form in a purer state, pushing away dissolved solutes. As the water solidifies, the ions are systematically rejected or squeezed out from the forming ice structure. The result is freshwater in the form of ice, which can be melted for use, and a concentrated brine left behind. While the process has its set of challenges and efficiencies to address, the fundamental science of ion rejection in freeze desalination offers a unique and environment friendly advantage to address water scarcity issues. Foreign particles are thrown out from the forming ice crystal lattice and remain in the residual solution, thus affecting the concentration of ions in both the ice and remaining solution [8-12]. This phenomenon is also seen in the upper atmosphere and potentially influences thundercloud electrification and the Workman-Reynolds Freezing Potential [13-16]. While there has been extensive research on water and its unique characteristics, less focus has been given to the mechanisms and kinetics of ion rejection [27–33]. This is where molecular dynamics simulations can play a vital role. They provide a detailed view of complex processes at the molecular level [22, 23, 27], which can contribute significantly to the understanding and refinement of freeze desalination [22, 23, 27, 28]. Through this research, innovative desalination technologies can be developed leading to more efficient, affordable, and environmentally friendly water purification solutions [1, 2]. This knowledge could also facilitate advancements in various industries such as agriculture, water and energy production, where water purification is a critical element.

Numerous research efforts have been devoted to understanding the elements that influence the rate of ion rejection via experimental studies [9–14]. Nonetheless, majority of previous MD studies have primarily focused on how ions influence the kinetics of ice nucleation, with minimal exploration of the microscopic mechanism of ion rejection. This area requires more comprehensive studies. This research aims to deepen current understanding of the ion rejection mechanism in freeze desalination through molecular dynamics simulations. By exploring the influence of various parameters like temperature on ion rejection rate, this study hopes to optimize the freeze desalination process. Ultimately, these findings seek to not only enhance the efficiency of this sustainable desalination method but also inspire further innovations in water purification to address global water scarcity.

1.2 Molecular Dynamics Simulation

This part of the chapter introduces about molecular dynamics simulations where it aims to present molecular view of ice growth while facilitating freeze desalination of salt water. This chapter also looks at ion rejection rate and its mechanism. So, for freeze desalination, a clear view of microscopic details of molecular interactions is required in order to get knowledge of the experimental expected results of the system. Molecular dynamics (MD) simulation represents a potent method to accomplish this goal. MD simulation is a computational technique that examines the temporal advancement of a system of particles, like atoms, molecules, and charges, each characterized by specific properties. The movement of particles is determined by solving Newton's classical equations of motion. The detailed molecular information from MD simulations, including positions and energies of particles, can be analyzed using statistical methods to derive thermodynamic and dynamic properties of the system, such as temperature and free energy. The molecular details of ice formation which can shed light on the ion rejection mechanisms while forming ice are very challenging to obtain by experimental measurements. MD simulation, with its ability to capture detailed trajectories of individual particles, can serve as a powerful tool for gaining insights into the microscopic aspects of freeze desalination. The primary limitations of MD simulations are the time and size scales [55,56], which are restricted to $\sim 10^5$ molecules at a time scale of nano-/microseconds. Freeze desalination is a long process in MD simulation compared to real time, even to get a very small amount of ice. To observe the spontaneous formation of ice growth in the timescale accessible to the simulation, lower temperature fairly below the freezing point is advised [57]. Moreover, system size must be over 5000 molecules to avoid finite size effect and stochastic effects [47]. Despite these challenges, the advancement of computational technologies has led to the increasing use of molecular dynamics (MD) simulations to explore key aspects of crystallization and its fundamental mechanisms.

In molecular simulations, specifically those involving sodium chloride (NaCl), the choice of potentials is crucial, as they determine how ions and molecules interact. It will also have a big impact on how long it takes to run a single simulation. Different potentials cater to varied simulation requirements based on the level of accuracy and computational efficiency needed. Ion models and water models have been listed in Table 1.1.

Table 1. 1 Using MD simulations, water-ion model combinations are utilized to study ion rejection. Tm, assuming pure water, is the melting temperature related to the given water mode [49]

Group	Year	Water Model	Ion Model	T_m (K)	Ref.
Vrbka	2005	SPC/E	SPC/E	215	[33]
Vrbka	2007	SPC/E	Smith-Dang	215	[32]
Conde	2017	TIP4P/2005	Madrid	250	[30]
Tsironi	2020	TIP4P	OPLS-AA	232	[29]
Luo	2021	TIP4P/2005	Madrid-2019	249	[27]
Conde	2021	TIP4P/2005	Madrid-2019	249	[20]

The Madrid Model is preferred for its balance between accuracy and computational demand, making it suitable for simulations where a realistic representation of ionic behavior in aqueous environments is essential. Madrid model captures the nuances of ion-water and ion-ion interactions effectively, vital for studying processes like salt dissolution or ion transport in water. [30,35] The Nonpolarized rigid Monomar and Smith-Dang are other variants, each fine-tuned for specific scenarios such as ion pairing or interactions with specific types of water models, like the SPC/E water model [32]. These potentials allow for tailored approaches to simulating NaCl in various environments, essential for accurately modeling processes like ice growth in FD studies, where the interaction between salt ions, water, and ice plays a critical role [27, 49].

In few studies on the freezing behavior of saltwater solutions, the direct coexistence method has emerged as a critical technique in molecular dynamics simulations. This approach typically involves placing an ice slab in contact with a saline solution slab [34]. By observing the interface between these two phases, the growth or melting of ice can be monitored at various temperatures to accurately determine the melting point or freezing point [34]. Below mentioned Figure 1.1 explains how melting temperature has been taken using direct coexistence.

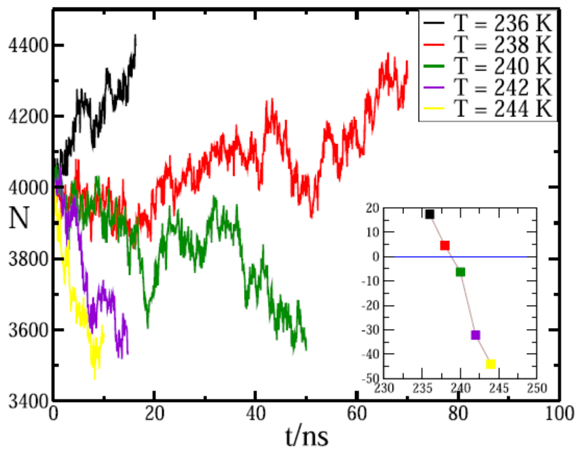


Fig. 1. 1 Number of molecules belonging to the ice slab as a function of time at different temperatures. Inset: slope of a linear fit to the curves in the main panel as a function of temperature. The melting temperature is taken as that for which the interpolated slope is zero (239 K) [34].

They are determining number of molecules in the ice phase, then taking a slope of it to determine melting temperature where slope is zero. They found that the melting temperature is around 239K because slope was zero at 240K. In addition, they also found that if slope is positive then ice is growing and if slope is negative ice is melting. While estimating melting temperature, salt concentration is changing but not more than 1% [34]. This method has been effectively applied in several studies, which observed ice growth and salt ion doping in simulations with seawater concentrations [24]. Similarly, the 2022 study using the Madrid-2019 model employed direct coexistence to ascertain the freezing temperatures of both pure water and electrolyte solutions by noting whether the ice phase grows or melts in contact with the solution [24]. Additionally, the 2018 research on the freezing-point depression of NaCl in water used this technique to study energy stabilization in a water/NaCl system at the point of phase transition [30]. The system was simulated for 200 ns at three different temperatures between the freezing point of water and the sodium chloride eutectic point (at one atmospheric pressure). At the solid-liquid interface, water changes from liquid to ice, ice plates start to expand, and the rate of ice front growth gradually slows down. The simulated temperature has a direct impact on the final growing ice volume. The volume of forming ice increases as the simulated temperature decreases away from the melting point [30]. These investigations collectively underscore the significance of the direct coexistence method in enhancing our understanding of the complex interplay between salt, water, and ice at the molecular level, particularly in the context of freezing point depression and ice formation processes [24, 30].

After setting up seed ice with salt water and setting the system size in MD simulation, ice growth can be observed at lower temperature. To understand it further, first one should look into melting temperature of ice varies with water. Vrbka and Jungwirth conducted simulations using four different types of boxes to observe how the melting temperature of ice varies when it is in contact with water [32,33]. They first performed these simulations for pure water using the SPC/E water model and then repeated the simulations for saltwater to see how the presence of salt affects the melting temperature. They stated that these melting point values can be influenced by system size and potential cut off distance. Now if partially freezing is happening in lower temperature, system can be kinetically trapped in metastable glass like state. Then again, if any system has small portion of unit cells in liquid phase, it will be easier to overcome the kinetic barriers during freezing. They found lower melting temperature when cut off distance was increased a bit. If system is melting in lower temperature, interfaces become kinetically stable, that's why, full system does not change. These changes found for neat water freezing [33]. Vrbka and Jungwirth found three double peaks in the density profile in the initial structure after the production phase, where it should have four according to them. These three fully developed double peaks that they found, shows a specific degree of pre-melting of the interface. These double peaks in the density profiles shows consequent immediate pre-melting of the interface. Which causes freezing shows down [38].

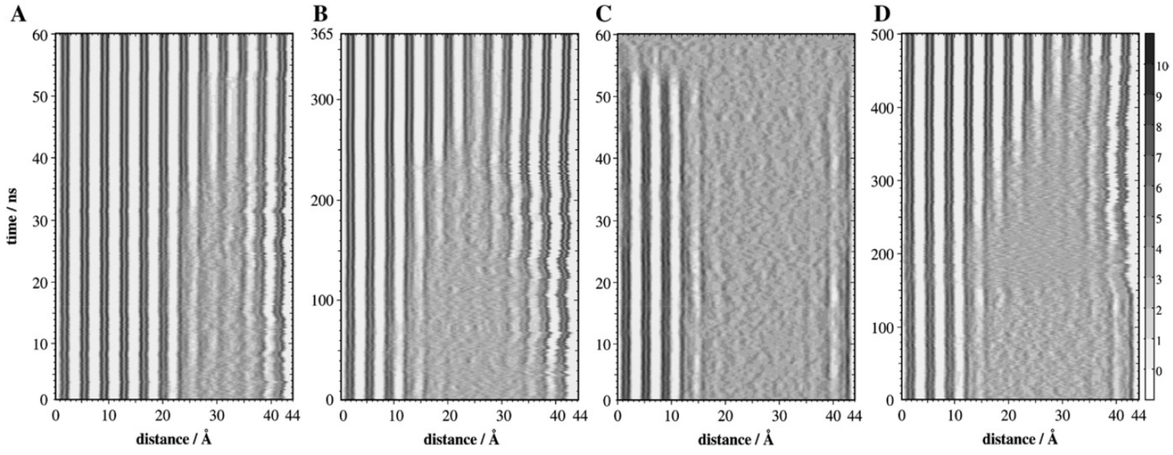


Fig. 1. 2 Time evolution of the density profiles for neat water simulations. A) box360 at −5° B) box180 at −15° C) box180 at −5° D) box180 at −10° (combined constant volume/constant pressure simulation). Note the partial melting after the switch to constant pressure around 150 ns [32].

Vrbka and Jungwirth established a procedure for neat water freezing. This protocol was helpful to find information on brine rejection and density profiles [32]. At first, they have equilibrated system with constant volume. They took cubic ice and SPC/E water model. Temperature was kept within 15K of 215K (melting temperature of SPC/E). Vrbka et al. wanted to demonstrate the kinetic antifreeze effect of the added salt at the molecular level through brine rejection. They found that if salt concentration is low, ions do not get trapped, see the figure (c) below where NaCl is 2 in pairs. But for figure (d) where NaCl is 4 in pairs, ions get trapped. They also found increase in freezing time when salt concentration is increased. This freezing time increase is direct demonstration of kinetic anti-freeze effect of the added salt [32].

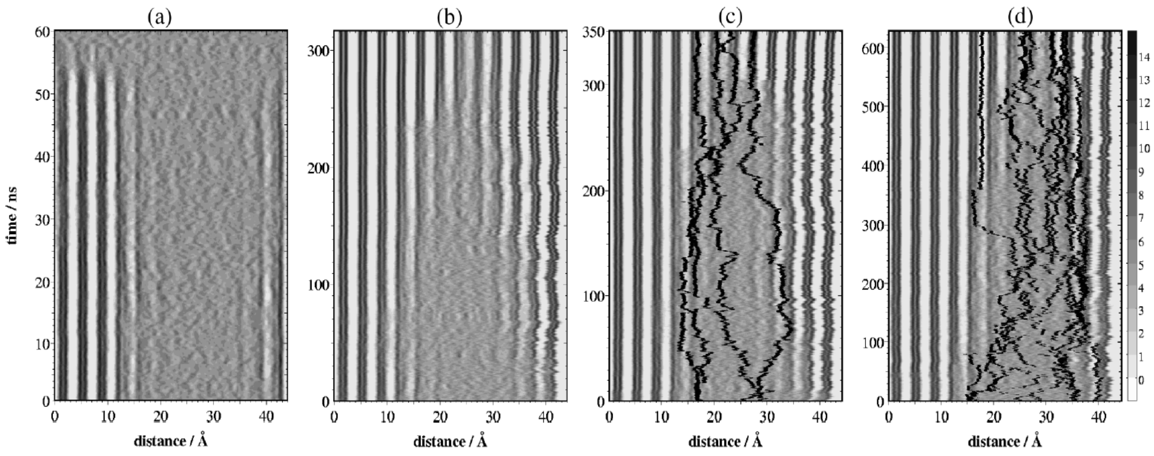


Fig. 1. 3 Time evolution of the density profiles for salt water freezing simulations. Trajectories of Na+ and Cl− ions are displayed as black lines. Temperature was held at −10° in all cases. (a) box180 with two NaCl ion pairs, (b) box180 with four ion pairs, (c) box360 with two ion pairs, and (d) box360 with five ion pairs [32].

In couple of literature, ice growth was visualized via simulations and visualizations tools (VMD, Ovitto etc.). Where ice growth can be observed in different temperature and in different salinity. Couple these figures have been added down here.

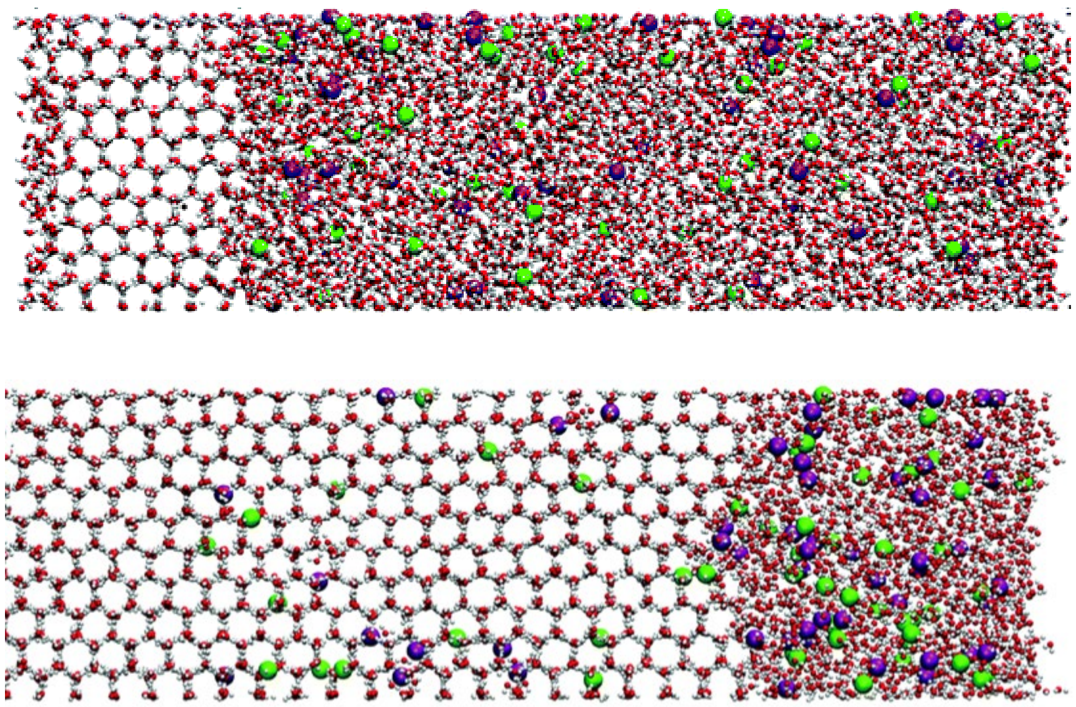


Fig. 1. 4 Molecular dynamics simulation system ice growth over time, 500ns. [27].

In Figure 1.4, ion has been trapped after 500ns. To further investigate this, interface of ice growth needs to be calculated. In literature, there were two ways to calculate interface. One way to go is calculating the average bond order parameter [27], another way is to calculate tetrahedrality parameter [30].

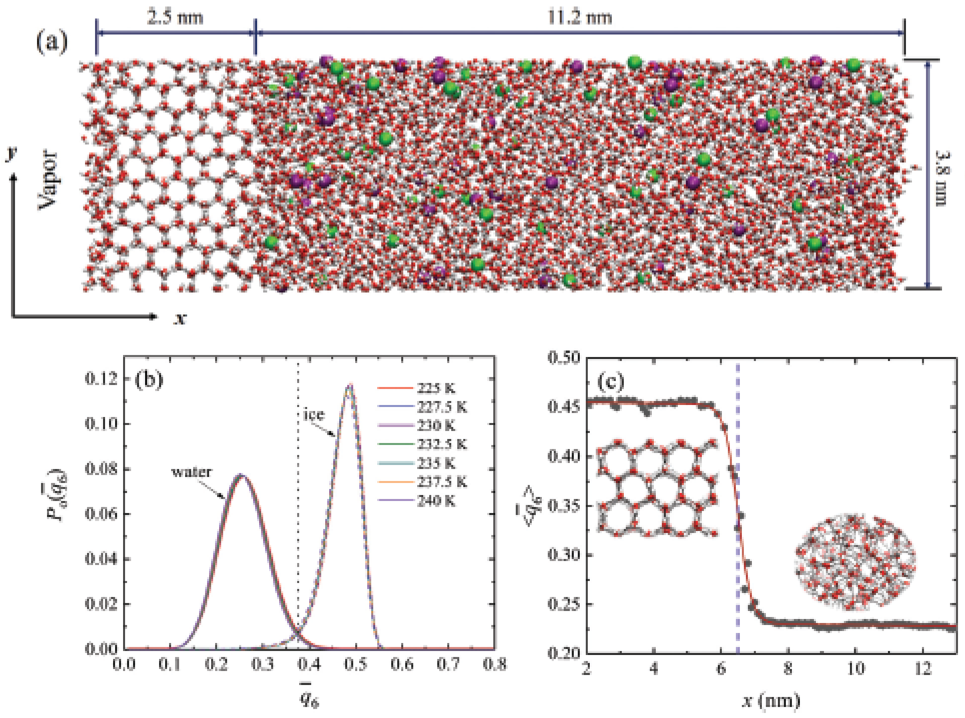


Fig. 1. 5 (a) Molecular dynamics simulation system, (b) Probability distribution of average bond order parameter vs average bond order parameter at different temperature, (c) Values of average bond parameter in x direction. So, from figure, average bond order parameter greater than 0.38 are ice [27].

To further investigate ion rejection mechanism, one has to identify ice growth interface. As discussed earlier, one of the interface identification methods is to calculate tetrahedrality parameter. After calculating tetrahedrality parameter, interface location can be found by it. So, the tetrahedrality parameter was calculated by Luo et al [50]. It was calculated to see the effect of ions on the local ice structures which allows to calculate the orientational order of the water molecules in the simulation as a function of the distance to the closest ion, to assess the impact of ions on the local ice structure [50].

$$q = 1 - \frac{9}{2n(n-1)} \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} (\cos\theta_{jik} + \frac{1}{3}) \qquad (2)$$

$$q_{avg}(y) = \sum_{i=1}^{N_y} \frac{q_i}{N_y} \tag{3}$$

$$q_{avg}(y) = q_{ice} + \frac{q_{ice} - q_{liq}}{2} [\tanh\left(\frac{y - I_{SL}}{w}\right) - \tanh\left(\frac{y - I_{LS}}{w}\right)] \tag{4}$$

In these equation, jik is the angle given by the lines joining the oxygen atom of a given molecule and those of its nearest neighbors j and k (less than or equal 4) [50].

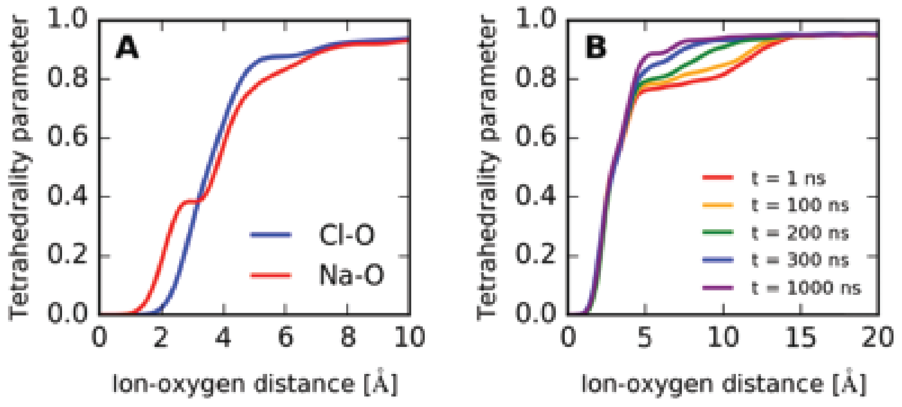


Fig. 1. 6 (A) Tetrahedrality parameter as a function of distance between the water's oxygen and the closest ion, for Cl–O and Na–O. (B) The tetrahedrality parameter for different simulation times, indicated in the legend [29].

Here ion rejection rate was determined by this R equation. [27]

$$R = \left(1 - \frac{C_{ice}}{C_{initial}}\right) * 100\% \tag{1}$$

In Luo et al.'s paper, the retention or rejection of NaCl ions depends in part on the events that take place at the ice-water interface, therefore the researchers continued by analyzing the free energy profiles that were discovered using PMF. An ion is less likely to escape at higher energy barrier values because it lacks the necessary kinetic energy. Ions are then trapped in ice as a result. Soria et al also discussed about free energy calculation to see where interfacial free energy is higher, whether in solution or in pure water. They measured interfacial free energy with classical nucleation theory. Finally, they found that ice solution free energy is higher in solution than pure water [27,40].

For calculating salinity, authors also calculated molar enthalpy of water. Molar enthalpy of water is directly dependent on Avogadro number, system's enthalpy, and number of water molecules. Salinity was kept at 1.85m for salt solution. As enthalpy difference is very small for the system, they had to run long simulations to find the difference (800ns at 190K to 100ns at 300K). For calculating enthalpy difference, they had taken two system of solution where salinity was 1.836m and 1.863m. There are two major benefits of thermodynamic integration, they could avoid finite size effect and stochasticity associated with the finite size of the system. Soria et al took spherical ice I_h as a sample for direct coexistence with brine. Ice seeds with any ion were embedded into the solution [34]. A crucial observation made was in the realm of mass percent concentration of ions within the ice. As the temperature climbs, there is a discernible decline in the ion concentration in the ice. In conditions where the temperature is 237.5 K or more (with a degree of supercooling not exceeding 14.5 K), the ion concentration dwindles to around 0.5% m/m, deemed apt for agricultural irrigation, and drops further to a mere 0.1% m/m, making it ideal for drinking purposes. [27] The rate of Na is 1.21 times higher than Cl- and there was no general decrease in the rate for Na+ as temperature rose [27]. Both Na and Cl had diffusion coefficients about 10 x less than bulk [27]. By examining the ice's growth rate, they evaluated the effect of temperature and diffusion for this as well. Ice grows at different rates depending on temperature, and if it grows too quickly, it will trap ions. They discovered that ions might escape if the ratio of ice growth rate to diffusion coefficient was less than 1. There is less ion retention or trapping in the ice at higher temperatures since it has been observed that ion diffusion is accelerated at these temperatures. And because the ice growth rate is decreased. Ice growth rate had a nonlinear relationship. The development of $NaCl.2H_2O$ hydrates (hydrohalites) during cooling at T = 233 K, which coexist with ice I_h [20]. Luo et al also looked at the final mass percent (concentration) of the solution and discovered a decreasing relationship between temperature and mass percent. As a result, they were able to simulate water that was pure enough to be utilized for drinking or farming at higher temperatures (about 238 K). [27]

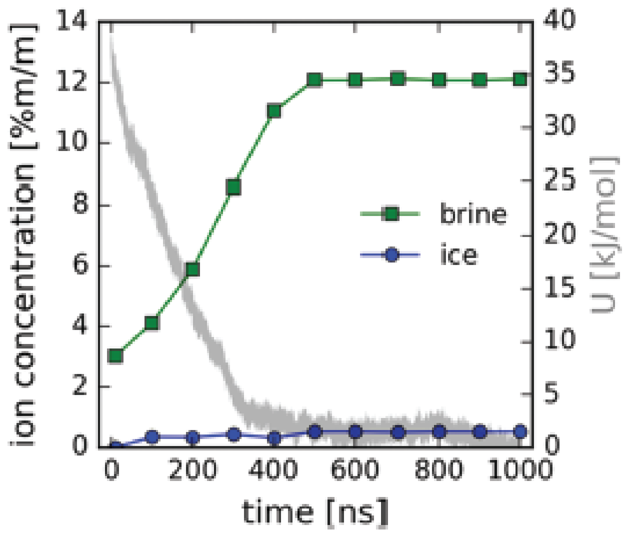


Fig. 1. 7 Ion concentration (left y-axis) in ice (blue circles) and brine (green squares) and the potential energy in the MD simulation (gray line – right y-axis) as a function of time [29].

In the context of chemistry and physics, reactions often require a certain amount of energy to proceed. This required energy is typically termed the "activation energy". It's like a hill a reaction must climb before it can proceed to completion. This metaphorical "hill" represents an energy barrier. Luo et al (2021) identified that activation energy was overcome by increasing the system's temperature, as higher temperatures gave the molecules more kinetic energy [27]. Additionally, first energy barrier refers to the initial free energy barriers that ions face at the ice-water interface. This barrier is a result of the free energy variation at the interface, primarily caused by the hydration energy differences between ions in water and ions in ice. It was noticed that the first energy barriers (ΔG) tend to be the highest while the other energy barriers (ΔG) are often smaller. Additionally, Na^+ has a lower energy barrier (ΔG) than Cl^-, which makes it simpler for Na^+ to break through the energy barrier. [27] The temperature plays a pivotal role in the freezing process of NaCl aqueous solutions, particularly in determining the rate at which ions are rejected or trapped. In the study, it was observed that ions encounter energy barriers when they attempt to move away from the freezing front. This is due to the free energy profiles that have multiple local minima, which are influenced by the layering of water molecules near the ice surface. Specifically, the first free energy barriers for Na^+ and Cl^- were identified to be 0.70 kcal mol^{-1} and 0.92 kcal mol^{-1}, respectively. The chance of an ion overcoming these barriers is directly proportional to $\exp(-\Delta G^*/k_BT)$. Given the determined values of ΔGI^*, it can be inferred that Na+ ions have a relatively lower likelihood of being trapped in the ice in comparison to Cl− ions. [27] As for the ion rejection rate (RR) in relation to temperature, it was noted that the RR ascends with increasing temperature for both ions. Intriguingly, at any given temperature, the RR for Na+ is approximately 1.21 times that of Cl−. This correlation is consistent with the theoretical predictions which suggest $P \sim \exp(-\Delta G^*/k_BT)$. Diving into the specifics of ion diffusion coefficients, both Na+ and Cl− display diminished diffusion coefficients at the ice-water interface as temperatures decrease. To put this into perspective, at 235 K, the diffusion coefficient D_x stands at 7.4×10^{-11} $m^2 s^{-1}$ for Na^+ and 9.1×10^{-11} $m^2 s^{-1}$ for Cl^-. It's noteworthy that these values are roughly an order of magnitude less than what is seen in bulk solutions. [27]. Temperature was 235 K for this bulk solution. Delving into the dynamics of ice growth, the study highlighted that as the temperature elevates, there is a zenith in the ice growth rate before it starts to decline. This phenomenon has significant implications; when the ice is forming rapidly, ions might not get sufficient time to escape, thus resulting in a subdued ion rejection rate at cooler temperatures. However, a shift in behavior is observed at higher temperatures, where the augmented diffusion of ions promotes their migration into the liquid phase, ensuring a heightened rejection rate [27].

The average potential energy between an ion and the water molecules in the ion's hydration shell is known as hydration energy [27]. The hydration energy of Na+ and Cl- is stronger with water than with ice, and the differences are higher than the average thermal energy, indicating that the NaCl ions prefer to stay in solution as the ice expands. It underscores the preference of Na^+ and Cl^- ions to remain in the liquid phase over being incorporated into ice due to the energetics of their interactions with water molecules, which are significantly influenced by thermal energy. Through a thorough analysis of the radial distribution functions for Na+ and Cl-, it became evident that the variations in their hydration structures between solid and liquid phases significantly influenced their behavior. Specifically, the RDF served as a quantitative tool to measure the hydration energy, shedding light on why these ions exhibited distinct preferences in different phases. If the Na-ice interaction is stronger than the Na-water interaction, the Na would not leave the ice. Luo et al. claim that the Na-Water interaction is stronger than Na-Ice. They concluded that this indicates that the hydration energy difference for Na+ is therefore greater than the hydration energy difference for Cl- and that the ion rejection rate of Na+ is higher than that of Cl^-. The concept of hydration energy difference plays a pivotal role in understanding the behavior of ions in freezing processes. Essentially, this difference measures an ion's attraction to water molecules in comparison to ice molecules. An ion with a pronounced hydration energy difference shows a marked preference for the liquid water phase over the solid ice phase. Delving into the specifics of the study by Luo et al., it was discerned that Na+ possesses a more pronounced hydration energy difference when compared to Cl-. This means Na+ has a considerably stronger affinity for water, indicating a more robust interaction with water molecules relative to its interaction with ice, especially when compared to that of Cl-. This distinction becomes evident during the freezing process; the stronger water affinity of Na+ ensures it's more likely to be expelled from the ice structure, causing it to predominantly remain in the liquid phase. In contrast, the hydration energy difference for Cl- is less pronounced, making it less predisposed to stay in the liquid phase when compared to Na+. Consequently, the ion rejection rate of Na+ surpasses that of Cl-. In essence, a larger hydration energy difference equates to a higher likelihood of an ion being rejected during ice formation, and in this scenario, sodium’s rejection rate exceeds that of chloride due to its more pronounced hydration energy difference. To support this, they included simulation snapshots that showed more Cl- ions trapped in the ice structure than Na^+ ions. [27]

The RDF, or Radial Distribution Function, is a tool often used in molecular dynamics (MD) simulations to analyze the spatial distribution of particles (The spatial distribution of particles refers to how particles are arranged or spread out in space). In the context of the study by Luo et al., the RDF was employed to examine how ions and water molecules are spatially distributed relative to each other. By calculating the RDF, they aimed to understand the behavior of ions near the freezing front and how this behavior contributes to the phenomenon of ion rejection during the freezing of NaCl solutions. The information from the RDF helped in determining the energy barriers ions face when moving away from the freezing front, and thus, it played a crucial role in their analysis of ion rejection mechanisms. Discovering a particle (a molecule or an ion) at a specific distance from a reference particle is described by the radial distribution function (RDF). It is employed to gather data regarding the interactions of two particles. In Tsironi et al.'s work, radial distribution function, g(r) was computed to look at the development and regional structure of the ice encased NaCl dihydrate crystals. In figure we can see that changes observed as a function of simulation time where those changes happened due to freezing.

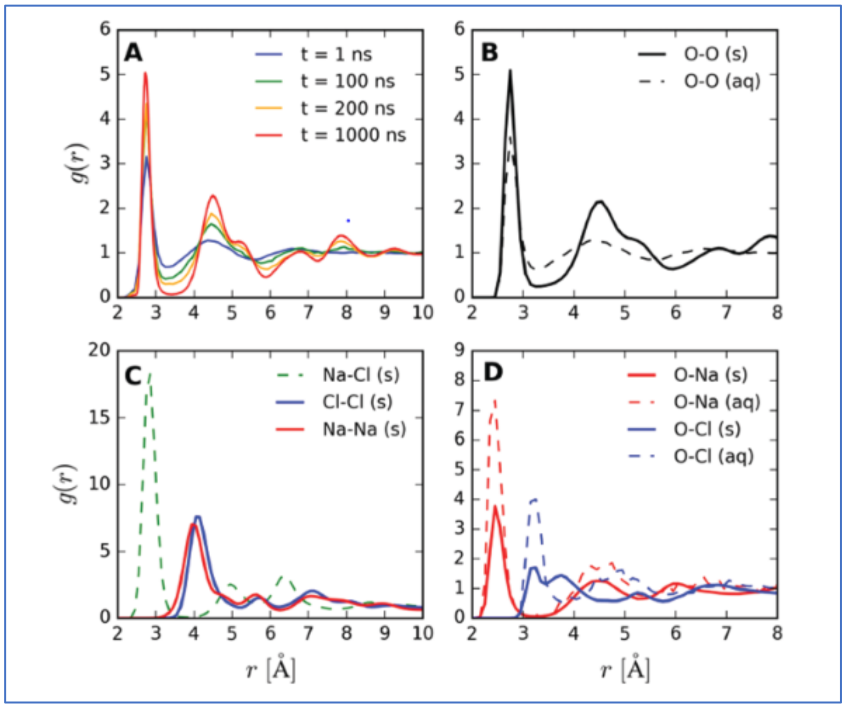


Fig. 1. 8 This figure captured four results, (A) The time evolution of the total g(r), (B) oxygen-oxygen bond for ice and water both, (C) Cl–Cl, Na–Na as well as the cross correlations between Na–Cl, (D) O–Cl, O–Na for both ice and water components. The self-correlations of various atoms, such as O-O, Cl-Cl, and Na-Na, are distinguished from the corresponding cross-correlations, such as Na-Cl, O-Cl, and O-Na. The crystal's O-O radial distribution function g(r) (solid line) resembles that of ice Ih with the addition of disorder brought on by the interaction between water and brine [29].

This hints at the possibility that the ions in the studied system may be adopting an arrangement akin to the FCC lattice. On the other hand, the cross-correlation between Na+ and Cl- ions presented a distinct peak at 2.8 Å. This indicates a tendency for sodium and chloride ions to position themselves roughly 2.8 Å apart within the solution. Taking these observations together, a compelling picture emerges. The patterns unveiled by the RDFs bear a striking resemblance to the structure of brine crystals. Typically found in high-concentration salt solutions like NaCl, brine, when on the cusp of solidification, manifests in a unique crystalline structure [29]. The data suggests that the ions in the studied solution seem to be mirroring this crystalline structure. This uncanny resemblance insinuates that, given the prevailing conditions, the system might either be teetering on the brink of crystallization or displaying short-range ordering patterns evocative of crystallized brine. [27,34,35]

To understand structural information inside ice lattice and salt water, Radial Distribution Function (RDF), often symbolized as g(r), is calculated. RDF is a measure used in the field of molecular physics and statistical mechanics specially to describe how the density of a system varies as a function of distance from a reference particle. It provides insight into the structural organization of particles (atoms, molecules, colloids, etc.) within a given system, typically in liquids or solids but also applicable to gases at high densities. Mathematically, the RDF is defined such that g(r)dr represents the probability of finding a particle within a spherical shell of radius r and thickness dr away from a reference particle, normalized by the average number density of particles ρ in the system. Essentially, it's a ratio of local density to the overall density of the system [48, 51]. Here is list of values and their meaning in RDF [51].

- $g(r)$=0: No particles are found at distance r from a reference particle, indicating a prohibited or highly unlikely region due to repulsive forces or physical constraints.

- $g(r)= 1$: The density of particles at distance r is the same as the average density of the system, suggesting no structural correlation between particles at this distance.
- $g(r)> 1$: A higher likelihood of finding a particle at distance r, indicating regions of attraction or structural ordering.
- $g(r)< 1$: A lower likelihood of finding a particle at distance r, suggesting repulsion, or excluded volume effects.

1.3 Objectives

The main goal of this research is to advance the understanding of the ion rejection mechanism in freeze desalination using molecular dynamics simulations, ultimately aiming to improve the efficiency of the process, contribute to the development of more sustainable desalination technologies, and address global water scarcity challenges.

- Utilize molecular dynamics simulations as an effective tool to study the complex interactions and dynamics of ions and water molecules during the freezing process, providing insights that may not be easily accessible through experimental methods alone.
- Gain a comprehensive understanding of the ion rejection mechanism in freeze desalination at the molecular level, which is a crucial aspect of optimizing the process and improving its efficiency.
- Evaluate the impact of various parameters, including temperature, pressure, and solute concentration, on the efficiency of ion rejection in freeze desalination, which can guide the optimization of the process.

- Contribute to the body of scientific knowledge on freeze desalination and its ion rejection mechanisms, inspiring further research, and innovation in the field of water purification and desalination technologies.

CHAPTER 2

METHODOLOGY

2.1 Potentials

We are using TIP4P/2005 model and Madrid 2019 model, where both made with Lennard Jones Potential and Electrostatic Potential. Each ion also employs a particular modified LJ potential with an additional electrostatic component. [35]

$$u(r_{ij}) = \frac{1}{4\pi\epsilon_0}\frac{q_i q_j}{r_{ij}} + 4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] \tag{6}$$

The additional parameters are defined as the Lennard-Jones potential, where qi is for atom i and 0 is the vacuum permittivity.

The Lennard-Jones (LJ) potential is a widely used mathematical model to describe the interaction between a pair of neutral atoms or molecules. It is particularly useful in simulations of molecular dynamics because it captures both attractive and repulsive forces that occur due to van der Waals interactions. The general form of the LJ potential is given by: [45]

$$V(r) = 4\epsilon[(\frac{\sigma}{r})^{12} -(\frac{\sigma}{r})^{6}] \tag{7}$$

Where:

- $V(r)$ is the potential energy as a function of the distance r between two particles.
- ϵ is the depth of the potential well, representing the strength of the attraction.
- σ is the distance at which the potential energy is zero; this parameter also represents the effective diameter of the particles.
- The $(\frac{\sigma}{r})^{12}$ term describes the repulsion between the particles, which becomes significant when particles are very close to each other.
- The $(\frac{\sigma}{r})^{6}$ term accounts for the attraction between the particles, which dominates at intermediate distances

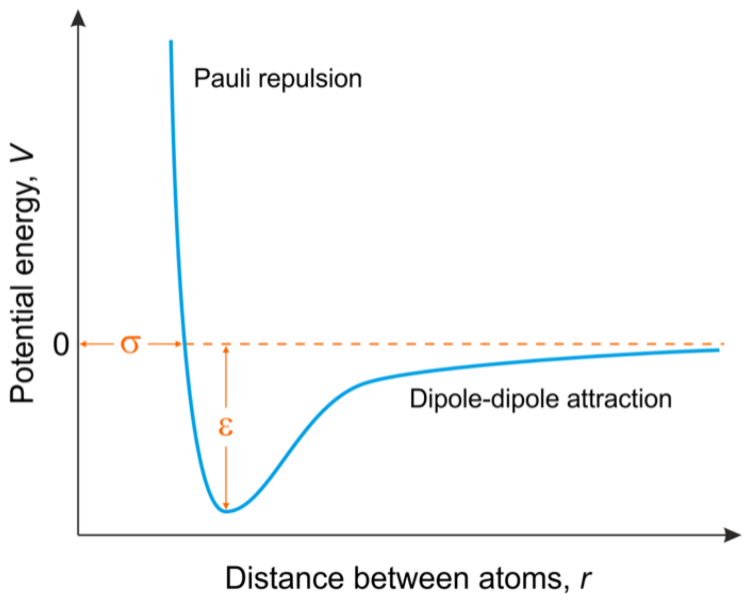


Fig. 2. 1 Graph of the LJ potential function: Intermolecular potential energy V(r) as a function of the distance of a pair of particles. [45].$r=r_{min}=2^{1/6}\sigma$.

The plot above illustrates the Lennard-Jones potential as a function of the distance between two particles. Here's a breakdown of the key features:

- Repulsive Region: When the distance r between particles is less than about $2^{1/6}\sigma$, the potential energy rapidly increases. This sharp rise represents the repulsive force that prevents the particles from getting too close to each other, corresponding to the r^{-12} term in the equation.
- Attractive Region: As the distance increases beyond $2^{1/6}\sigma$, the potential energy dips into negative values, indicating an attractive force between the particles. This is the region where the r^{-6} term dominates, modeling the van der Waals attraction.
- Minimum Energy Point: The minimum of the curve occurs at $r=2^{1/6}\sigma$, which is the most energetically favorable distance between the particles under this model. At this point, the attractive and repulsive forces are balanced in such a way that the system's potential energy is minimized.

The electrostatic potential, denoted as φ, quantifies the potential energy per unit charge at a given point in an electric field, which is created by stationary electric charges. It is a scalar quantity and integral in fields like electrochemistry, capacitor physics, and molecular modeling. The potential φ at a point in space due to a point charge Q can be expressed using the formula: [46]

$$\phi(r) = \frac{Q}{4\pi\epsilon 0\boldsymbol{r}} \tag{8}$$

Here, **r** is the position vector relative to the charge's location, r is the magnitude of **r** (the distance to the charge), ϵ_0 is the permittivity of free space, and Q is the charge. This equation essentially states that the potential at a point is directly proportional to the charge and inversely proportional to the distance from the charge. In molecular dynamics, the electrostatic potential is crucial for calculating the forces between ions in a system, such as in a saltwater solution. Here, the Coulombic interactions described by this potential dictate how ions like Na^+ and Cl^- influence each other, crucial for predicting behaviors such as crystal formation or dissolution. This potential also explains why charged particles move towards regions of lower potential in an electric field.

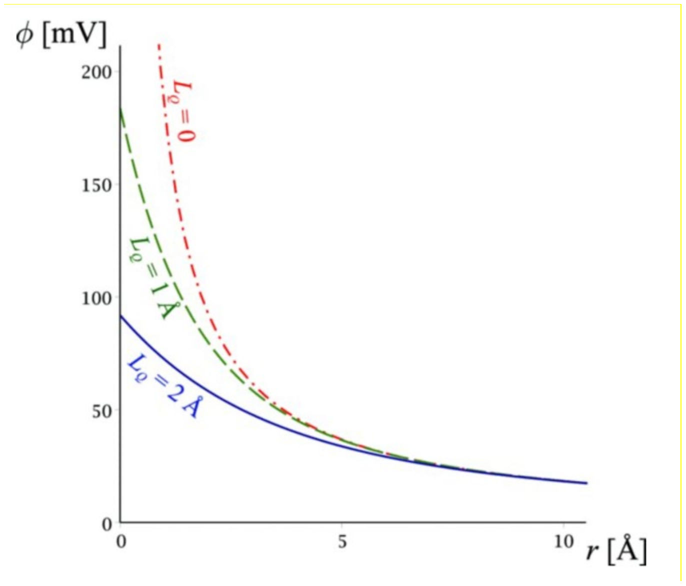


Fig. 2. 2 Electrostatic potential φ of a point charge in a quadrupolar medium vs. the distance r from the point charge e in water, Eq. 8. [46]

The plot above shows the electrostatic potential φ as a function of distance r from a point charge. The potential φ decreases as $\frac{1}{r}$ which is evident from the curve falling off as distance increases. Near the charge, the potential is high, indicating a strong influence of the charge at close distances. As its moving further away, the potential decreases, reflecting the diminishing influence of the charge over greater distances. This relationship is fundamental in understanding how charged particles interact with each other and influence surrounding fields. [46]

2.2 Simulation Setup

Molecular Dynamics (MD) simulations provide a powerful computational approach to model the motions and interactions of atoms and molecules over time, based on principles of classical Newtonian physics. Within the context of this project, MD simulations offer a microscopic lens, enabling the visualization and analysis of complex processes that occur when water molecules freeze, and ions are either incorporated into the forming ice structure or rejected into the remaining liquid. Specifically, as water molecules arrange themselves into an ice lattice, MD simulations can vividly illustrate how ions such as Na+ and Cl- navigate this changing environment. By analyzing these simulations, we can gather insights into the ion rejection mechanisms, discerning patterns in molecular movement, interaction energies, and spatial distributions that would not be readily apparent or feasible to measure through traditional experimental methods alone. Thus, MD simulations not only complement empirical studies but also reveal the nuanced, molecular-scale intricacies of ion rejection during the freeze desalination process, offering potential avenues to refine and optimize the process.

The methodology for conducting molecular dynamics simulations in the context of freeze desalination involves the use of sophisticated software tools such as GROMACS, LAMMPS, and PMEMD. The preparatory phase of the simulation mandates the definition of the system's thermodynamic conditions. This entails choosing a particular thermodynamic ensemble, for example NPT, NPxT, NVT etc., which sets the parameters for the system. With the system boundaries defined, the simulation's next stage focuses on establishing specific thermodynamic values, such as temperature (T) and pressure (P). This stage encompasses detailing both the extensive and intensive properties of the system: N= Number of atoms, Px= Pressure in x direction, T= Temperature & V= Volume. Here, we are using NPxT and NVT as ensembles. Next step is setting boundaries, we are using periodic boundaries. The next crucial step is setting equations of motions. The equations of motion are generally considered to follow one set of definitions. Thus, the Leapfrog algorithm or the Velocity-Verlet algorithm are not different “equations of motion” but are different numerical approximations of the equations of motion as defined by Newton’s Laws of Motion. Various equations of motion are available for this purpose, including the Velocity-Verlet, Leapfrog algorithms, Numerical propagation, and Verlet integration.

This stage is pivotal because molecular dynamics simulations require stable environments for accurate predictions. To achieve this stability and control, various algorithms and theorems are utilized. For instance:

- The Buss thermostat, Velocity rescale thermostat, and Velocity scaling thermostat are advanced methodologies aimed at adjusting and stabilizing the temperature by moderating particle velocities. The Nose-Hoover Theorem offers a deterministic approach to maintaining the system's temperature.
- When it comes to pressure regulation, Berendsen barostat and Parrinello-Rahman Barostat are commonly employed. (Add which ones are you using)

Each of these methods has its unique characteristics and applications, ensuring that the system remains within the desired thermodynamic parameters throughout the simulation. Following this, the simulation box must be configured, with most papers opting for a rectangular geometry. The dimensions of the simulation box should be defined. Ours is being 5.40 nm x 4.65 nm x 17.59 nm.

To maintain the rigidity of water molecules, algorithms like SHAKE, SETTLE, and LINCS can be employed.

Molecular interactions, especially those involving water molecules, are commonly defined using specific potentials. One of the primary potentials for water interactions is the Lennard-Jones potential, typically implemented with a cutoff of 1.2 nm [35]. Other important interactions include Coulombic (often handled by methods like Particle Mesh Ewald) and Van der Waals [36]. To address long-range electrostatic interactions, specialized algorithms such as Particle-Particle Particle Mesh, Particle Mesh Ewald summation, Particle Mesh Ewald method, and Smooth Particle Mesh Ewald procedure are employed [37]. Hydrogen bonds in water are pivotal for its unique properties. Each water molecule can form up to four hydrogen bonds, referring to its two hydrogen atoms and two lone pairs on the oxygen atom. The hydrogen bond is a type of dipole-dipole interaction, where the hydrogen atom of one water molecule is attracted to the oxygen atom of another. This bond is partly electrostatic, partly LJ potential arising from the Coulombic force between the positively charged hydrogen and the negatively charged oxygen. However, hydrogen bonding is stronger than typical dipole-dipole interactions due to the small size of hydrogen and the high electronegativity of oxygen, forcing closer approach of molecules.

Here, we are using TIP4P/2005 water model for water and ice, and Madrid Model for salt. The simulation consists of 18104 atoms and ran for 200ns for four cases, and 500ns for one cases. Here, two different ensembles were used to observe ion rejection rate and salinity change in brine solution, those are NVT (constant number of particles, volume, and temperature) and NPxT (constant number of particles, pressure in x direction, and temperature). Temperature was set at 235K and 240K. These were the key parameters for LAMMPS simulation.

2.3 Tetrahedrality Parameter

The tetrahedrality parameter, often termed q or q_t, is a measure that helps discern the local tetrahedral structure around an atom or molecule, particularly in the context of water and its hydrogen-bonding nature. In the MD simulations, this parameter is instrumental in characterizing the arrangement of water molecules as they transition between liquid and solid phases. The q parameter for a molecule, particularly a water molecule in our context, can be computed using the following approach:

Select a Central Molecule:

- For every water molecule taken as the central molecule, identify its four nearest neighbors, typically other water molecules. These neighbors are usually determined based on the oxygen-oxygen distance, given the nature of hydrogen bonding in water.

Compute the Distance Vector:

- For each pair of these four nearest neighbors, compute the distance vector between their oxygen atoms. Let's consider these vectors as r_{ij} where i and j are two neighboring molecules among the four closest ones.

Calculate the Angle Between Vectors:

- For every combination of these vectors (6 combinations for 4 neighbors), compute the angle θ between them. The cosine of this angle is given by eq. 9 [50],

$$\cos(\theta_{ij}) = \frac{r_i \cdot r_j}{|r_i| \times |r_j|} \tag{9}$$

Tetrahedrality Parameter Calculation:

- The tetrahedrality parameter q for the central molecule is then given by eq. 10 [50],

$$q=1-\frac{3}{8}\sum_{i<j}(\cos(\theta_{ij})+13)^2 \tag{10}$$

Where the summation runs over the 6 combinations of the 4 nearest neighbors.

The value of q will be close to 1 for a perfect tetrahedral arrangement (as seen in ice) and will be significantly lower in the liquid phase where the arrangement is more distorted. Here, q_{ice} ≈0.95 and q_{liq}≈0.77 are the local order parameters for the bulk ice and liquid, respectively [50]. By calculating the tetrahedrality parameter, the phase transition of water molecules can be calculated from a liquid-like to an ice-like environment during the freeze desalination process. Areas with high q values would correspond to regions where water is freezing or already in the ice phase, while areas with lower q values would represent still liquid or transitioning regions. In addition, this parameter can aid in identifying the ice-water interface and can provide information into how solute ions (like Na^+ and Cl^-) are behaving near this interface—whether these are being incorporated into the ice matrix or rejected into the remaining liquid phase. For practical calculations, most molecular dynamics software packages or associated analysis tools provide functionalities or scripts to compute tetrahedrality parameters, given the relevance of this measure in studying water and aqueous systems.

2.4 Interface Calculation

Identifying the ice-water interface in a freeze desalination process using the tetrahedrality parameter is an important task. Once the tetrahedrality parameter is calculated for each water molecule in the system, these values can be used to delineate regions of ice from regions of liquid water, thereby identifying the interface. Defining a threshold value for the tetrahedrality parameter, q threshold, that distinguishes between liquid-like and ice-like environments. Here,

q_{ice}≈0.95 and q_{liq}≈0.77 are the local order parameters for the bulk ice and liquid, respectively like mentioned earlier. This implies that a value close to 0.95 would indicate a structure very close to perfect tetrahedral geometry (as found in ice), whereas a value around 0.77 would be more representative of the less ordered liquid water structure. Water molecules with q values higher than this threshold will be considered as being in an ice-like environment, while those below it will be in a liquid-like state. This threshold can be set based on literature values or a separate calibration simulation where the states of water are well-defined. [50]

2.5 Ion Rejection Rate

Calculating the ion rejection rate in the context of a freeze desalination process using molecular dynamics (or similar simulations) involves tracking the behavior of ions (like Na^+ and Cl) with respect to the growing ice interface. Here's how to proceed:

Temporal Analysis:

- For a dynamic understanding, calculate the rejection rate at various time points or time intervals throughout the simulation. This will provide insights into how the rejection rate might change as the ice front grows and as ion concentrations in the remaining liquid phase potentially increase.

Comparative Analysis:

- To provide context for findings, compare the rejection rate calculated with any available literature values or with rates from other simulations or experimental setups. This can give insights into the efficiency and accuracy of this freeze desalination process.

As temperature increases, the viscosity of water decreases, which can affect the flow rate and the behavior of ions within the system. Salinity, on the other hand, indicates the concentration of salt ions in the water. Higher salinity means more ions need to be rejected to produce freshwater. These two factors are interconnected, and their interplay can have nuanced effects on ion rejection rates. Hence, understanding the comparative rejection rates across different temperature and salinity gradients is crucial for optimizing desalination and other water treatment processes.

The ratio of C_ice (the ratio of the number of ions trapped in the solid phase) and C_brine (the number of ions in the liquid phase) can be combined to determine the ratio R [27].

$$R = \frac{C\_ice}{C\_brine} \tag{12}$$

The ion rejection rate can be calculated using the following formula: [27]

$$\text{Ion Rejection Rate, } R_R = (1 - R) * 100 \tag{13}$$

This formula represents the percentage of ions that are effectively rejected from the ice phase and remain in the brine phase during the freeze desalination process. We can investigate various control parameters to understand their impact on the ion rejection rate. This will help to optimize the freeze desalination process for improved efficiency [27].

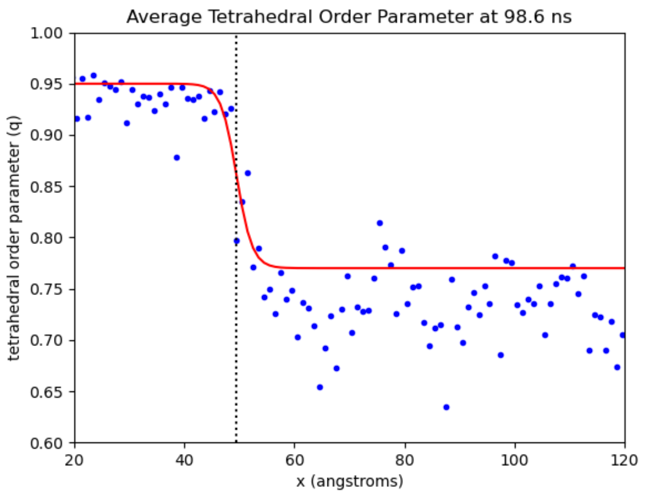


Fig. 2. 3 Values of q in y direction and simulation box bin distances in the x direction. This plot was generated by post processing of frozen data at 53ns with NPxT ensemble at 240K.

By analyzing these parameters, especially in relation to temperature and salinity, we can gain insights into the optimal conditions (for ensemble, salinity, and temperature) for maximizing ion rejection rates. This is crucial for designing more efficient freeze desalination systems, ensuring better water quality and resource management. The structural analysis provided by q aids in understanding the microscopic mechanisms driving these macroscopic outcomes.

2.6 Free Energy

Calculating the free energy or energy barrier at a growing ice interface in a freeze desalination process requires determining the energy difference between the ion being in the bulk liquid phase and the ion being at or near the ice interface.

Here's the approach:

Umbrella Sampling:

- This is a commonly used technique to calculate free energy profiles using MD simulations.
- Begin by placing the ion of interest (Na^+ or Cl^-) at various distances from the ice interface, then apply a harmonic biasing potential to "restrain" the ion at that position during the simulation.
- This is repeated for various positions spanning from deep within the liquid phase to deep within the ice phase, effectively "sampling" the ion's behavior across the interface.

Run Simulations:

- For each restrained position, run an MD simulation to gather data. by using Umbrella Sampling and MD simulations, the free energy profile of an ion at an ice interface can be calculated. This involves running simulations for the ion at various positions, collecting positional data, and then using methods like WHAM to construct a PMF, which gives a detailed view of the energy landscape the ion experiences near the interface. The biasing potential ensures that the ion explores its local environment but remains close to the chosen position. [48]

Weighted Histogram Analysis Method (WHAM):

- To calculate the PMF, the data from all simulations (different positions) are combined using a technique called the WHAM. This method uses the collected energy and force data to construct a probability distribution of the ion's position. Use the WHAM technique to combine the data from all simulations into a single free energy profile (or PMF) as a function of distance from the interface. [48]
- The result will show the relative free energy of the ion at various positions.

Interpretation:

- Examine the PMF:
    - Regions with higher free energy represent unfavorable positions for the ion. free energy peaks within the liquid phase in a PMF suggest regions where ions are more stable. For FD, this could imply more effective ion rejection if the energy associated with ion entrapment in the ice is higher than the energy associated with dissolution in the liquid phase, making it energetically unfavorable for them to be incorporated into the forming ice. However, the overall impact on the desalination process would depend on a detailed analysis of these energy barriers in relation to the ice-liquid interface dynamics. If this region coincides (direct coexistence of ice and brine) with the ice interface, it suggests that the ion is rejected by the growing ice.
    - Regions with lower free energy (or deep valleys) can suggest positions where the ion is stabilized. If such valleys exist in the liquid phase but not in the ice phase, it supports the notion of ion rejection in freeze desalination.

Repeat for Both Ions:

- The process should be repeated for both Na^+ and Cl^- to determine the energy barriers or free energies associated with each ion type.

Comparative Analysis:

- Once PMFs have been calculated for both ions, the relative stabilities and energy barriers can be compared.

By calculating the free energy profiles, valuable insights can be gained into the fundamental interactions that dictate ion behavior at the ice interface, providing a deeper understanding of the freeze desalination process at the molecular level.

2.7 Radial Distribution Function (RDF)

The RDF can be used to study the local structure and organization of particles in a system, such as liquids, gases, or solids. It can reveal the presence of short-range and long-range order in a system, as well as provide insights into the nature of interactions between particles, like attractive or repulsive forces. By using the RDF in MD simulations, valuable insights into the local structure and organization of water molecules and ions during the freeze desalination process can be gained. The results will yield valuable insights into the ion rejection mechanism, enabling us to optimize the process for enhanced efficiency and sustainability, especially when applied to molecular dynamics (MD) simulations of systems like freeze desalination. Because: [44,48]

Understanding Local Structure: The RDF is a key tool in understanding the arrangement of particles, like water molecules and ions, in a system. It is like a map how likely you are to find a particle at a certain distance from another particle, compared to just random chance. In the context of freeze desalination, this becomes especially useful because as water freezes, it forms an ice-liquid interface. The RDF can show us how the water molecules and ions are arranged near this boundary. When the RDF shows a sharp peak at a specific distance, it means that particles are more likely to be separated by that distance – they prefer to be this far apart. On the flip side, if there's a dip or trough in the RDF at a certain distance, it indicates that it's less likely for particles to be separated by that distance. This information is crucial in freeze desalination studies because it helps us understand the local structure of water and ions near the freezing interface, which is vital for optimizing the process and making it more efficient. The Radial Distribution Function (RDF) is pivotal in optimizing freeze desalination by revealing the molecular arrangement of water and ions near the freezing interface. By analyzing RDF peaks and troughs, we can fine-tune freezing conditions to enhance ice purity and exclude impurities more effectively. RDF plots the density probability of particles at different distances from a reference particle, revealing where particles are more likely to be positioned relative to each other. Sharp peaks in the RDF indicate preferred distances where water molecules align to form ice crystals, suggesting optimal freezing conditions that promote the formation of pure ice and the exclusion of impurities. Troughs, on the other hand, show less likely distances for particle arrangements, helping identify where ions or impurities are pushed out as the ice lattice forms. By analyzing these peaks and troughs, the freezing process can be fine-tuned—adjusting the freezing rate and temperature—to maximize ice purity and exclude impurities effectively. This analysis aids in setting precise control parameters to improve the efficiency and scalability of freeze desalination, making it possible to achieve higher purity and yield in ice production for water desalination. Overall, RDF serves as a vital tool in refining freeze desalination techniques, enhancing process efficiency, and aiding in the scale-up from laboratory to industrial applications. This insight enables the adjustment of cooling rates, temperature gradients to maximize salt removal efficiency. Furthermore, analysis of the RDF informs the design of additives that modify water structure, improving the selective exclusion of impurities. By analyzing these peaks and troughs in the RDF, we can gain insights into how the microscopic arrangements of particles influence the larger process of separating salt from water through freezing. Analyzing the Radial Distribution Function (RDF) allows for precise adjustments in MD simulation. Peaks indicate preferred particle distances, facilitating the formation of purer ice by aligning water molecules optimally. Troughs highlight less likely distances, helping to exclude impurities. This understanding guides the adjustment of cooling rates and temperature gradients, maximizing salt removal. Additionally, RDF analysis informs the development of additives that further optimize water structure for selective impurity exclusion, enhancing overall desalination efficiency. [44,48]

Ion Exclusion from the Ice Front: As water freezes, it forms a crystalline structure that typically excludes impurities, including ions. By examining the RDF of ions with respect to water molecules, we can observe how ions are pushed away from growing ice interfaces. If, for example, the RDF between water molecules and ions shows a depletion zone (a decrease in probability) at distances corresponding to the ice-liquid interface, this suggests that ions are being rejected from the solidifying ice front. As water freezes into ice, its crystalline structure naturally excludes impurities such as ions. The RDF between water molecules and ions can reveal a depletion zone at distances typical of the ice-liquid interface, indicating a reduced probability of ion presence where ice forms. This depletion suggests that ions are actively pushed away from the ice front, preventing their incorporation into the ice lattice, and thereby enhancing the purity of the frozen water. As water freezes into ice, its molecules form a hexagonal crystalline structure that excludes impurities, including ions. This process, known as ion exclusion, occurs because the regular ice lattice cannot incorporate ions without disrupting its hydrogen-bonded network. This exclusion is crucial for understanding natural phenomena, like the concentration of salts in unfrozen water bodies, and applications such as freeze desalination, highlighting the intricate dance between water's structure and impurity distribution during freezing. [44,48]

Nature of Interactions: The positions and heights of peaks in the RDF can indicate the strength and nature of interactions between particles. Strong attractive forces between particles, for instance, can lead to pronounced peaks at specific distances in the RDF, corresponding to preferred binding or interaction distances. In the context of FD, if there are specific interactions between ions and water molecules that either promote or hinder ion rejection, these could be revealed in the RDF. It reveal a depletion zone at distances typical of the ice-liquid interface, indicating a reduced probability of ion presence where ice forms. This depletion suggests that ions are actively pushed away from the ice front, preventing their incorporation into the ice lattice, and thereby enhancing the purity of the frozen water. Peaks at certain distances indicate, suggesting specific binding configurations between ions and water. Such insights can identify interactions that either promote or hinder ion rejection during freezing. By analyzing the position and height of these peaks, scientists can deduce how ions influence the water structure, potentially improving freeze desalination efficiency by identifying optimal conditions for maximizing ion exclusion from the forming ice, thus enhancing water purity [44, 48].

To calculate RDF $g_{ab}(r)$, we used eq. 11, [51]

$$\mathrm{g_{ab}(r)} = (\mathrm{N_a N_b})^{-1} \sum_{i=1}^{N_a} \sum_{j=1}^{N_b} \langle \delta(|\boldsymbol{r}_i - \boldsymbol{r}_j| - r) \rangle \tag{11}$$

In a homogeneous system, the radial distribution function (RDF) is adjusted so that its value reaches 1 at large distance between particles. The RDF measures the average number of type 'b' particles located within a certain distance 'r' from a type 'a' particle, and presents this count as a density value [51]. In summary, by using the RDF in conjunction with MD simulations for freeze desalination processes, we can gain better understanding of how ions are rejected at the molecular level.

We have few simulations to achieve above mentioned goals. Here is the list of it.

Table 2. 1 These are our simulation summary for LAMMPS.

Lammps Simulation	Time (days)	Core hours (nodes)
NVT_235K_TIP4P/2005_0.6M_200ns	5	46,080 (6)
NVT_245K_TIP4P/2005_0.6M_200ns	6	36,864 (4)
NPxT_235K_TIP4P/2005_0.6M_200ns	6	36,864 (4)
NPxT_240K_TIP4P/2005_0.6M_200ns	5	36,864 (4)
NVT_235K_TIP4P/ICE_0.6M_200ns	6	36,864 (4)
NVT_235K_TIP4P/2005_0.6M_500ns	3	30,720 (4)
Umbrella Sampling	5	46,080 (6)
RDF_calculation	1	10,000 (4)
NVT_240K_TIP4P/2005_0.6M_200ns	5	36,864 (4)

CHAPTER 3

RESULTS AND DISCUSSIONS

3.1 Ice Growth and Ion Trap Comparison with Different Ensembles (NVT and NPxT)

The phenomena of ice growth and ion entrapment offer insights into presenting unique challenges and opportunities for exploration. Particularly, the investigation of these processes under different thermodynamic ensembles, namely NVT (constant number of particles, volume, and temperature) and NPxT (constant number of particles, pressure in x direction and temperature), reveals details about system equilibriums, structural formations, and energy distributions. This thesis delves into the comparative study of ice growth and ion trap mechanisms within the NVT and NPxT ensembles. We have run the simulation for 500ns to allow for adequate freezing with NVT ensemble at 235K in Figure 14.

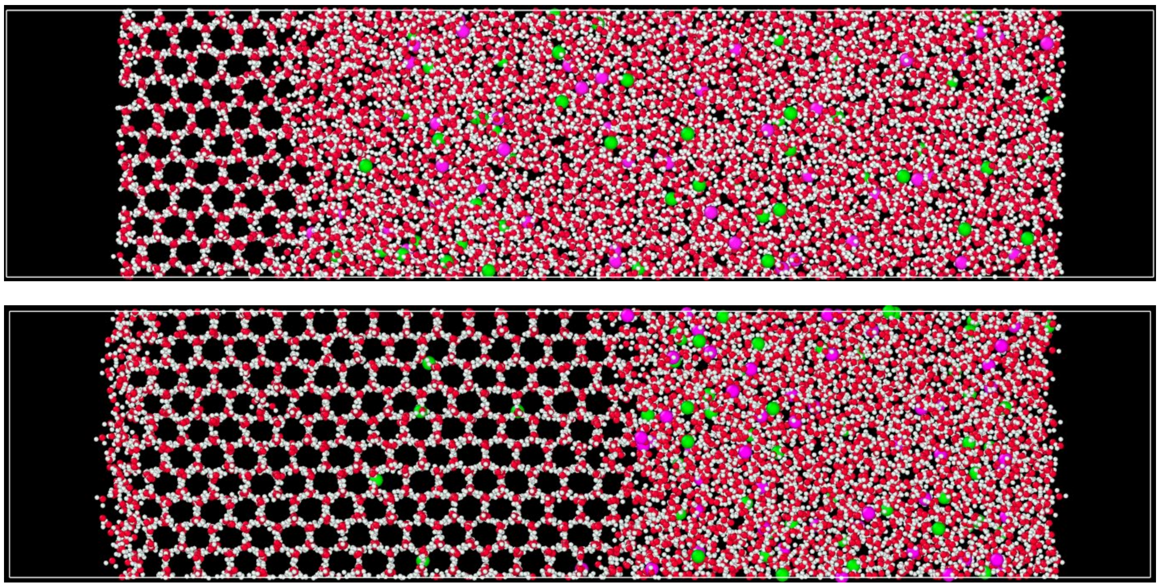
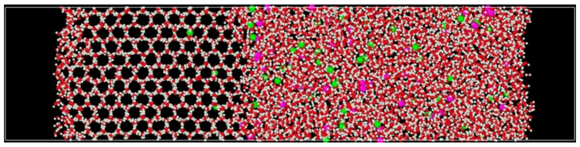
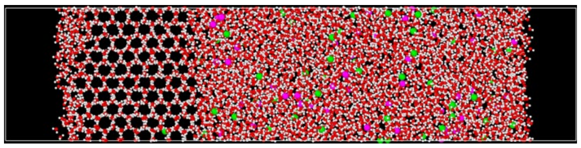
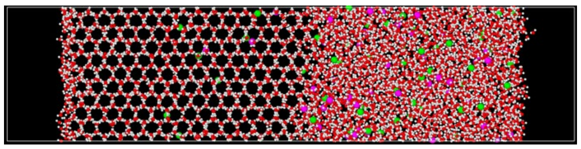


Fig. 3. 1 Molecular Dynamic Simulation of growing ice front in NaCl solution using Lammps simulation. On the left is the hexagonal ice commonly found in natural conditions in contact with a solution of 0.6 M NaCl. Red – Oxygen, White – Hydrogen, Purple – Sodium, Green - Chloride.

When looking closely each trapped ion, it is clear how ions are surrounded by water atoms. Chloride is found to be surrounded by hydrogen atoms. In the other hand, sodium was found to be surrounded by oxygen atoms due to opposite charge. There was visible ice growth for 100ns simulation. After 100ns, ice growth was reduced substantially but ion entrapment was very less within 200ns. We ran the same simulation with NPxT ensemble to compare ice growth and ion entrapment.


(a) NVT_235K_TIP4P/2005


(b) NVT_240K_TIP4P/2005


(c) NPxT_235K_TIP4P/2005

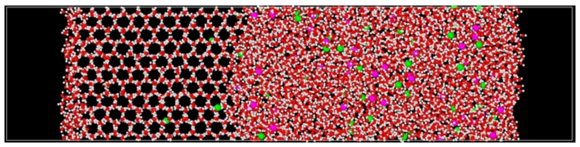

(d) NPxT_240K_TIP4P/2005

Fig. 3. 2 (a) NVT_235K_TIP4P/2005, (b) NVT_240K_TIP4P/2005, (c) NPxT_235K_TIP4P/2005, (d) NPxT_240K_TIP4P/2005. Molecular Dynamic Simulation of growing ice front in NaCl solution using Lammps simulation. On the left is the hexagonal ice commonly found in natural conditions in contact with a solution of 0.6 M NaCl. All simulation consists of 18104 atoms.

As we know from literature that melting temperature of TIP4P/2005 is 249K [20,27]. So, the temperature, set at 235K and 240K, is below the freezing point of water, creating a convenient environment to ice growth. The fixed volume in the NVT and fixed pressure in NPxT ensemble plays a crucial role in freezing. By setting the temperature below the freezing point but not too low (235K and 240K here), the simulation achieves an imbalance where the kinetic energy of the molecules is low enough to allow them to come together and form stable ice crystal structure, but not so low that the molecular motion is overly restricted. The fixed volume of the NVT ensemble further influences this balance by limiting the space in which the molecules can move, thereby affecting the density and arrangement of water molecules and their interactions, which is critical for the growth of ice crystals in the solution. The use of the TIP4P/2005 model for water further impacts this process, affecting how water molecules interact with each other and with the NaCl molecules in the solution. Thus, the substantial ice growth observed in these simulations can be attributed to the interplay between the reduced kinetic energy at 235K. In MD simulations described here, salinity, ion models and water models was same all four cases, but ensembles and temperatures were different. If we compare temperatures with same ensemble in Fig. 15(a) & 15(b) and Fig. 15(c) & Fig. 15(d), lower freezing temperature gave better freezing. This agrees with our expectation, as we have seen in literature [27, 47, 48, 49]. If we further lower the temperature, it will give more freezing evidently up to 220K from literature. If we compare ensembles for same temperature in Fig. 15(a) & 15(c) and Fig. 15(b) & 15(d), the most notable ice growth was for NPxT over NVT. Here is a table to show ice growth along x axis, expressed in interface shift.

Table 3. 1 Interface start and end in x axis of simulation box for four cases.

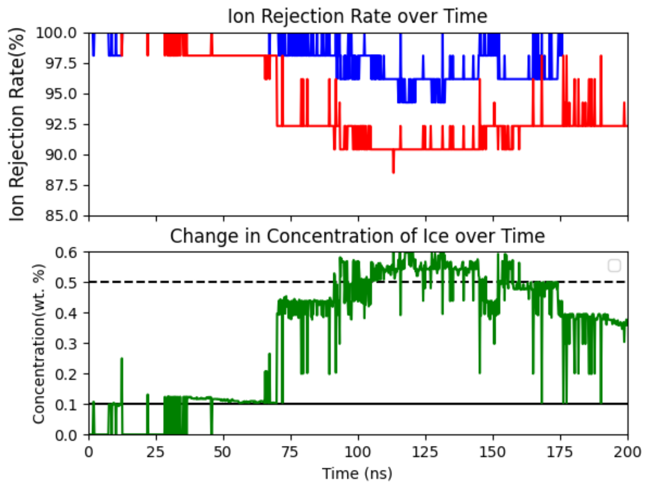
Cases	Interface start	Interface end
(a) NVT_235K_TIP4P/2005	26.97	51.56
(b) NVT_240K_TIP4P/2005	26.97	47.07
(c) NPxT_235K_TIP4P/2005	26.97	62.50
(d) NPxT_240K_TIP4P/2005	26.97	46.59

From Table 3 and Figure 15, we can say case (c) provided most notable ice growth where temperature was low enough to freeze and ensemble provided constant pressure to ensure freezing.

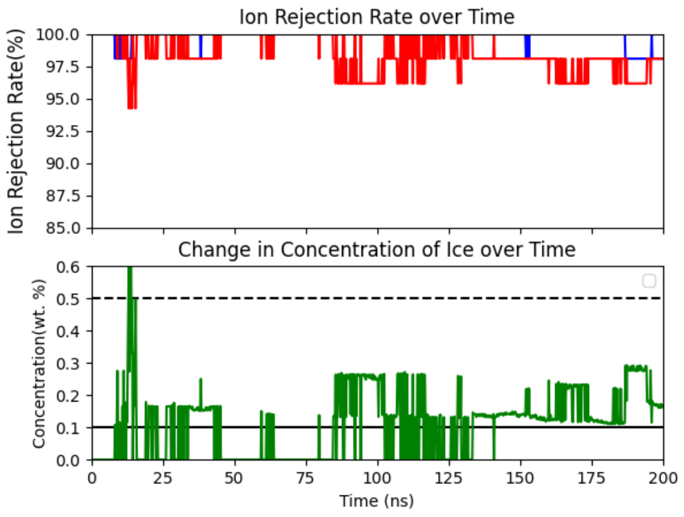
The differences in ice growth and ion trapping between the two ensembles are indicative of how thermophysical properties can be influenced by the choice of ensemble. The NPxT ensemble, by allowing volume changes, might more accurately reflect natural processes where pressure and temperature are constant, but volume can change. In this investigation, we explore the kinetic properties of water molecules and ions under varying thermal and physical conditions at different temperatures and under distinct constraints of fixed and variable volume. The findings of Luo and colleagues highlight that the rejection of ions is significantly influenced by temperature variations, which affect both ion diffusion and the speed of ice formation [27, 48]. This interaction directly impacts the efficiency with which ions are excluded. Notably, as the temperature approaches the melting point (249 K, according to the TIP4P/2005 model), the rate at which ions are rejected reaches its maximum.

3.2 Ion Rejection Rate Comparison in Different Temperature and Ensembles

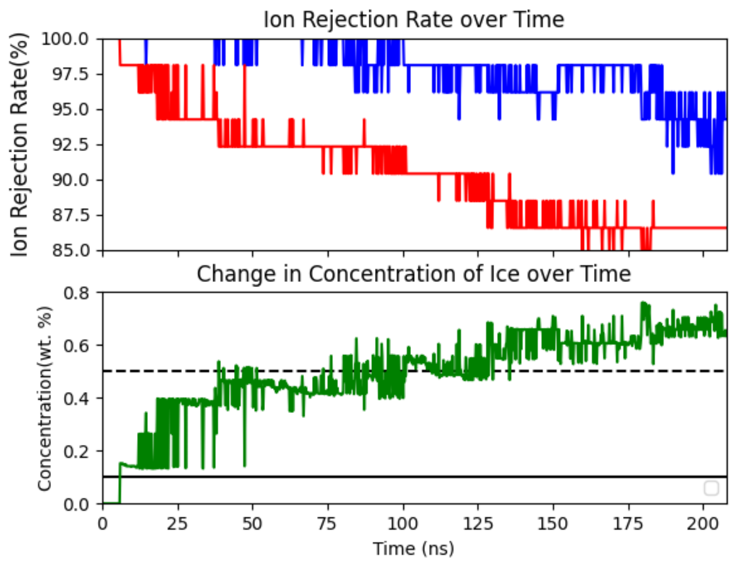
In the molecular dynamics simulations under consideration, ion rejection rates were compared across varying temperatures using the TIP4P/2005 water model within NVT and NPxT ensembles, the latter allowing for pressure variation in one direction. It was observed that the most effective ion rejection occurred at a lower temperature of 235K with the NPxT ensemble, suggesting enhanced efficiency under these specific conditions in Fig. 16 and Tab. 4 below. The correlation with ice growth indicates a potential interest in the interplay between ice formation and ion rejection. It is found R_R (Ion Rejection Rate) for Na^+ is around 1.01-1.04 times higher than that for Cl^- at both for temperatures 235K & 240K and for both NVT & NPxT. This probability ratio, P_{Na^+}/P_{Cl^-}. is lower than what we found in literature [42], which is 1.25. But this ration we have found still aligns with fact that R_R for Na+ is higher that Cl-. There could couple of reasons why we have found lower ratio. We have used different temperature 240K and different ensemble NPxT for two of the cases. But NVT ensemble and 235K were identical to literature.



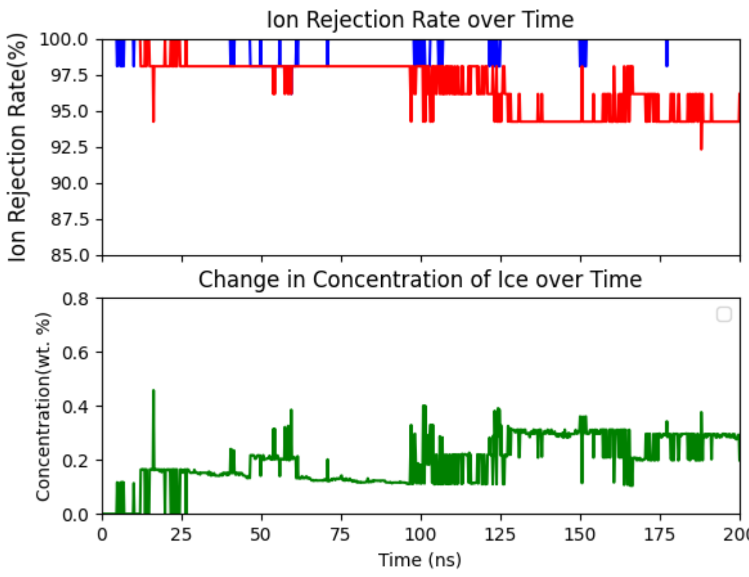
(a) NVT_235K_TIP4P/2005



(b) NVT_240K_TIP4P/2005



(c) NPxT_235K_TIP4P/2005



(d) NPxT_240K_TIP4P/2005

Fig. 3. 3 (a) NVT_235K_TIP4P/2005, (b) NVT_240K_TIP4P/2005, (c) NPxT_235K_TIP4P/2005, (d) NPxT_240K_TIP4P/2005. Values of q in y direction and simulation box bin in angstroms distances in the x direction.

Figure 3.3 gives significant visualization that ions are getting rejected as a result salinity goes higher in brine water with rejected ions. Here is listed results in Table 3.2.

Table 3. 2 Ion Rejection Rate in four distinguished cases with salinity of brine water with rejected ion as freezing progresses.

Cases	Ion Rejection Rate at 200ns (%)		Salinity of ice at 200ns (m/m %)	Salinity of brine water at 200 ns (m/m %)	Probability Ratio
	Na+	Cl-			
(a) NVT_235K_TIP4P/2005	97	92	0.4	4.2	1.05
(b) NVT_240K_TIP4P/2005	99	97	0.2	4.2	1.02
(c) NPxT_235K_TIP4P/2005	92	86	0.7	4.6	1.07
(d) NPxT_240K_TIP4P/2005	99	94	0.25	4.0	1.05

As shown Table 3.2, higher salinity is found by using NPxT ensemble. This can be attributed to the increased system volume (constant system pressure) associated with the NPxT ensemble. As volume increases in NPxT (more ice growth), ion gets more rejected. While comparing the temperatures, higher temperature provides better ion rejection because higher temperature, ions obtain more kinetic energy to overcome the free energy barrier.

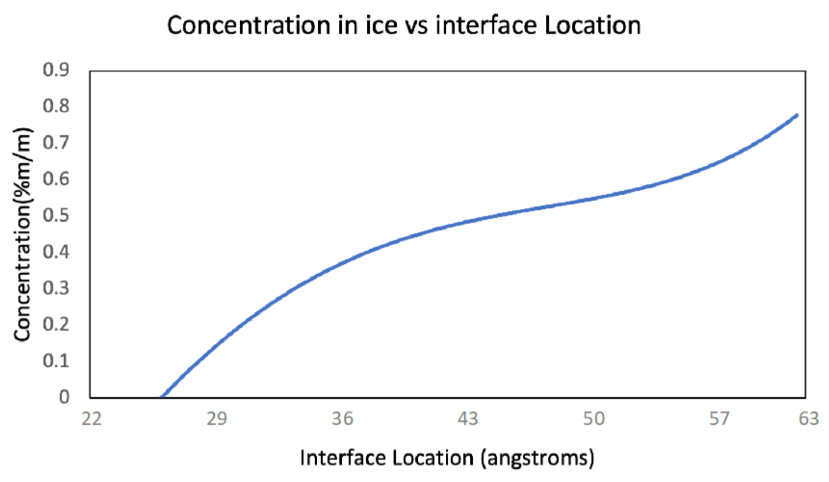


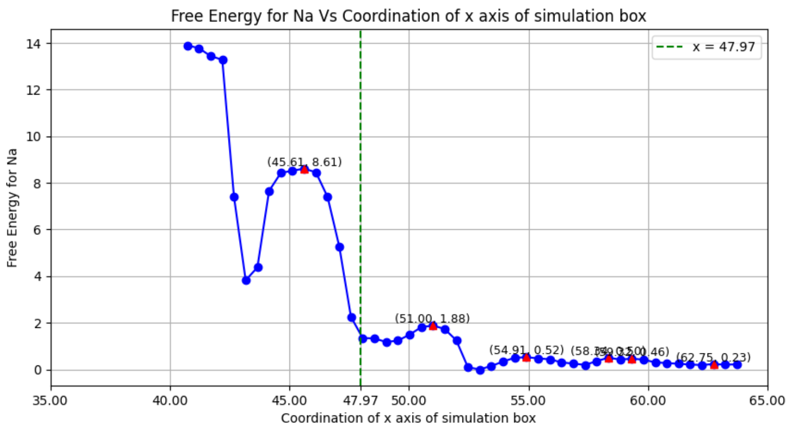
Fig. 3. 4 Ion concentration in ice over time vs interface location over time for NPxT ensemble at 235K.

As shown Figure 3.4, ion concentration in the growing ice section fluctuates as ions close to the ice-water interface are captured or rejected. To address this "noise" and obtain a clearer representation of the underlying trend, a moving average technique was applied. Specifically, a moving average over a 10 ns window was employed to smooth out the ion concentration in the ice section. The choice of a 10 ns window for the moving average was based on the need to balance between smoothing the noise and preserving the essential features of the concentration curve. This approach ensures that the key trends and variations in concentration are still visible. The result after applying the 10 ns moving average for case 3 (NPxT_235K) is plotted in Figure 3.4. Instead of using time as the x-axis, the interface location is used as it is directly related to time. As shown, the ion concentration in ice is gradually increasing indicating the ion rejection rate decreases slightly as the ice section grows larger.

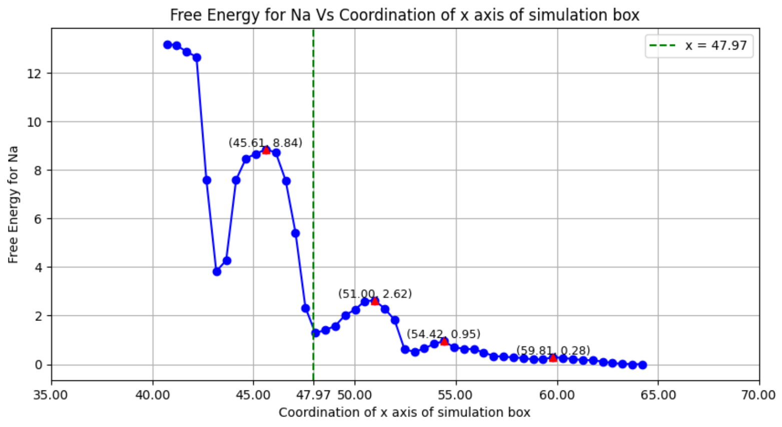
3.3 Umbrella Sampling and Free Energy Calculation

Whether a Na^+ or Cl^- is trapped or rejected by the ice is largely determined by its dynamics at the ice-water interface, which can be probed by examining the free energy profile of ions in the vicinity of the ice front. The free energy ΔG for Na^+ and Cl^- in the x direction, i.e., the ice growth direction, is shown in Figures 16a and 16b. It is seen that the free energy profiles exhibit multiple local minima. The fluctuation of the free energy is caused by the layering of water molecules near the ice surface (Fig. 17). The free energy variation generates energy barriers ΔG*, which are the free energy difference between a local minimum and the adjacent maximum, as indicated in Figure 14. If ΔG* is high, ions at a local free energy minimum (lowest point of 1st peak) may not have sufficient kinetic energy to escape the attraction of water molecules in the ice structure and eventually can be captured by the ice as the ice grows. On the other hand, the thermal fluctuation of ions causes them to diffuse around the ice-water interface. The probability P of an ion overcoming an energy barrier is proportional to $\exp(-\Delta G^* \; k\, T\,)$. In Figure 16, it is seen that the first free energy barriers for Na+ and Cl- are the largest, which are 8.6 kcal/mol and 8.8 kcal/mol, respectively. The second and third free energy barriers ΔG* and ΔG* are relatively small. In addition, it is found that ΔGI* for Na^+ is lower than that for Cl^-. Combined with the effect of temperature and the free energy landscape at the ice-water interface, are key factors determining the rate and mechanism of ion rejection during freezing.

- The peaks are energy barrier to escape ice and go into liquid side
- Cl's energy barrier is greater than Na, which is consistent with our literature review.
- From these plots, we can say that it is much more likely to escape ice for Na than Cl.



(a)



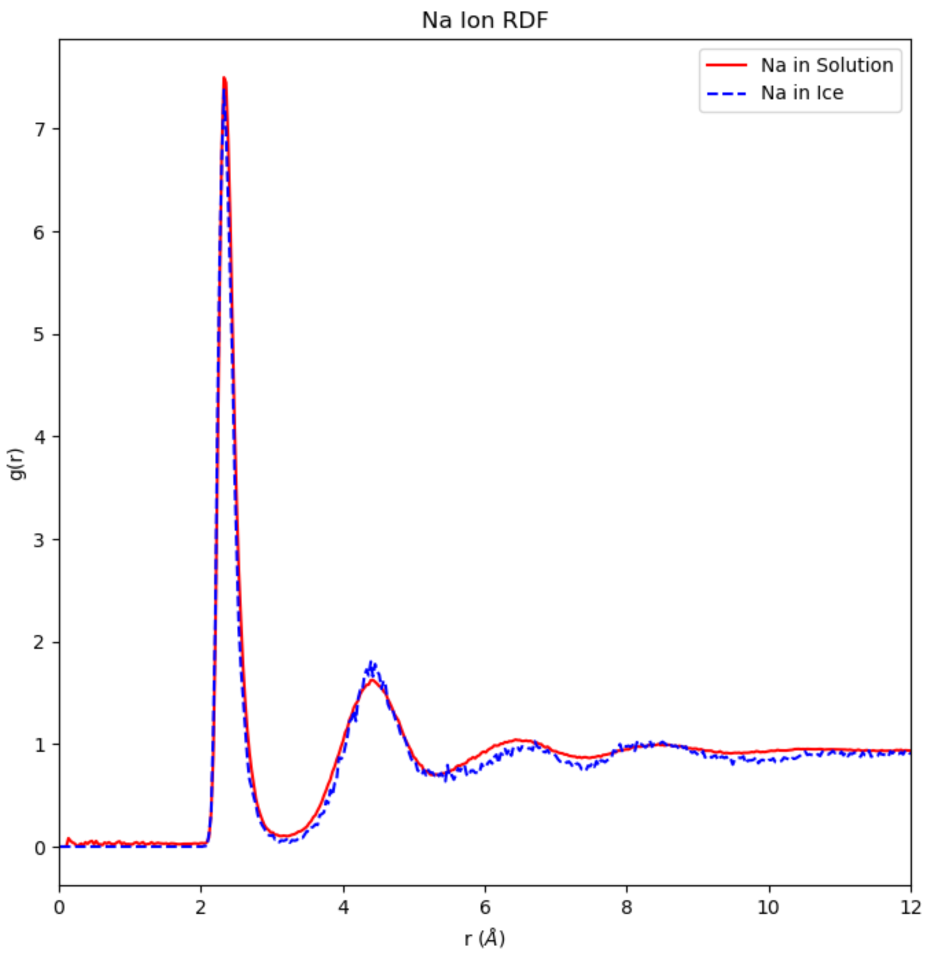
(b)

Fig. 3. 5 The free energy change of sodium and chloride as they progress from the ice front must progress past the 2 energy barriers. The interface location is x = 47.97 indicated by perpendicular dashed line on x axis.

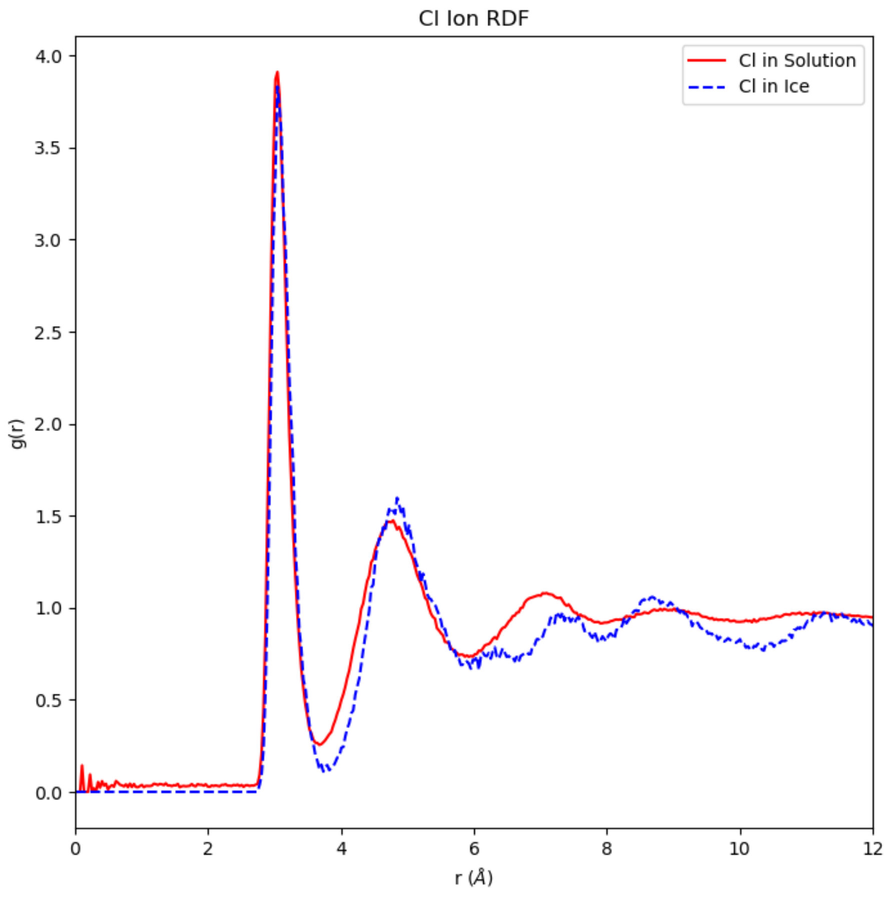
Figure 3.5 (a) and (b) are instrumental in elucidating the molecular-level dynamics of ion rejection in freezing NaCl solutions. They illustrate the free energy landscape that ions encounter at the ice-water interface during the freezing process. The peaks depicted in the graphs represent energy barriers that Na^+ and Cl^- ions must overcome to transition from the ice to the liquid side. According to the study, the energy barrier for Cl^- is greater than that for Na^+, which aligns with findings from the literature review [27, 48, 49]. This difference in energy barriers is indicative of the likelihood of ion escape from ice, suggesting that Na^+ ions have a higher propensity to enter the liquid phase compared to Cl^- ions. Additionally, the size of the Na^+ ion plays a significant role in its interaction with water molecules. Due to its smaller size relative to Cl^-, Na^+ tends to attract water molecules more effectively, leading to the formation of hydration shells around the ion [20, 30].

3.4 Radial Distribution Function (RDF) Calculation

At the molecular scale, the dynamics of ions are governed by the molecular forces acting on the ions by the surrounding molecules, which depend on the hydration energy per ion, which is defined as the average potential energy between an ion and the water molecules in the hydration shell of the ion. The hydration energy doesn't include the other energetic interactions or entropy. The variation of the hydration energy in water and ice is caused by the change of ionic hydration structure. Figures 17 depict the radial distribution functions g(r) of water and ice around Na+ and Cl-. It is seen that the first peak in g(r) for Na+ in water is 7.5 than that in ice, whereas the g(r) curves of Cl- in water is 3.9. The integration of the g(r) curve gives the average number of water/ice molecules N_h inside ionic hydration shells. The Na^+-water interaction is stronger than the Na^+-ice interaction. The preferential rejection of sodium – or inclusion of chloride – has been repeatedly shown in molecular dynamics studies by the absence of sodium or presence of chlorides trapped in the ice [18,42,43,45–48]. Vrbka and Jungwirth first noted the preferential nature of ion rejection when they only found chloride ions trapped at higher concentrations while at lower concentrations both sodium and chloride ions were rejected [16]. It is due to the differences in potential energy barriers. Figure 18(a) & (b) are showing the difference in the RDF of chlorine and sodium in solid versus aqueous phases. The RDF quantifies the organizational nature of atoms/molecules around a specific atom. For ions in solution, the RDF is directly correlated with the amount of water molecules surrounding the given ion. The energy barriers that chloride must overcome are larger than for sodium ions [42].



(a)



(b)

Fig. 3. 6 RDF shown in salt water and in ice for (a) Na^+, (b) Cl^-.

RDF data is important to understand how close other molecules is to each ions both in ice lattice and in solution. From the values we have found in here, it is more likely that Cl^- is surrounded by more molecules compare to Na^+. From this, we cannot draw conclusion yet whether Cl^- is more likely to get trapped or Na^+. For that we need to calculate coordination number and hydration energy first. It is important to note that RDF values are first step to calculate coordination number and hydration energies.

CHAPTER 4

CONCLUSION AND FUTURE WORK SECTION

In this thesis, MD simulations have been used to investigate the microscopic kinetics of ice growth and the molecular mechanism of ion rejection from freezing salt solutions has been explored. Results showed that ion rejection rate is exceeding 90%, with the largest ice growth in the simulation at 235K with NPxT used as the ensemble. As the temperature increases, the ion rejection rate increases. Moreover, the rejection rate of Na+ is higher than that of Cl^- (not exceeding melting points). Furthermore, the molecular interactions between ions and water/ice form energy barriers at the ice-water interface, which may cause possible entrapment of ions in the ice. On the other hand, the inherent kinetic motion of ions allows them to overcome the energy barriers and leads to ion rejection. The free energies for Na and Cl ions were found to be 8.6 & 8.81 kcal/mol, indicating the Cl ions have higher free energy barrier to escape from the ice structure. As a result, the simulations have shown the Na ions are more effectively rejected into the brine solution. Furthermore, the radial distribution values were found to be 7.5 for Na ion and 3.9 for Cl ion. So, Na^+ is surrounded by more molecules than Cl^-. This information is crucial to determine which ions will get trapped during freezing.

4.1 Perspectives for Future Work

The coordination number plays an essential role in this process as it determines the number of nearest neighbors of ions, thereby influencing how ions interact with surrounding water molecules and ice structures. Additionally, the hydration energy of ions, which reflects the stability of ions when solvated, changes significantly as water transitions to ice, affecting ion mobility and stability. Moreover, utilizing brackish water as a medium could further understanding of ion rejection, as its unique ionic composition offers a varied context for examining how multiple ion types at different concentrations impact ice nucleation and growth dynamics.

The TIP4P/ICE water model, which accurately represents water's properties at low temperatures, is particularly useful for these studies. By simulating the effects of external electric fields using this model, researchers can gain deeper insights into how field strength and frequency modify ion mobility and ice growth kinetics. Such investigations are crucial for developing more precise models of ion dynamics at the ice-water interface and could lead to innovative techniques for controlling ion concentrations in environments where water is in the process of freezing. These approaches promise to enhance theoretical understanding of freeze desalination.

REFERENCES

[1] M. Salehi, Global water shortage and potable water safety; Today's concern and tomorrow's crisis, Environ. Int. 158 (2022) 106936. https://doi.org/10.1016/j.envint.2021.106936

[2] M. El Haj Assad, M. Nooman AlMallahi, M.A. Abdelsalam, M. AlShabi, W.N. AlMallahi, Desalination Technologies: Overview, in: 2022 Adv. Sci. Eng. Technol. Int. Conf. ASET, 2022: pp. 1–4. https://doi.org/10.1109/ASET53988.2022.9734991.

[3] B. Zarei, E. Parizi, S. Mossa Hosseini, and B. Ataie-Ashtiani, A multifaceted quantitative index for sustainability assessment of groundwater management: application for aquifers around Iran, (n.d.).

[4] I. Janajreh, H. Zhang, K. El Kadi, N. Ghaffour, Freeze desalination: Current research development and future prospects, Water Res. 229 (2023) 119389. https://doi.org/10.1016/j.watres.2022.119389.

[5] L. Erlbeck, M. Rädle, R. Nessel, F. Illner, W. Müller, K. Rudolph, T. Kunz, F.-J. Methner, Investigation of the depletion of ions through freeze desalination, Desalination. 407 (2017) 93–102. https://doi.org/10.1016/j.desal.2016.12.009.

[6] A. Najim, A review of advances in freeze desalination and future prospects, Npj Clean Water. 5 (2022) 1–15. https://doi.org/10.1038/s41545-022-00158-1.

[7] L. Macias-Bu, M. Guerra-Valle, G. Petzold, P. Orellana-Palma, Technical and Environmental Opportunities for Freeze Desalination, Sep. Purif. Rev. 0 (2022) 1–10. https://doi.org/10.1080/15422119.2022.2098504.

[8] K. Aagaard, E.C. Carmack, The role of sea ice and other fresh water in the Arctic circulation, J. Geophys. Res. Oceans. 94 (1989) 14485–14498. https://doi.org/10.1029/JC094iC10p14485.

[9] A.K. Peterson, Observations of brine plumes below melting Arctic sea ice, Ocean Sci. 14 (2018) 127–138. https://doi.org/10.5194/os-14-127-2018.

[10] R. Ferrari, M.F. Jansen, J.F. Adkins, A. Burke, A.L. Stewart, A.F. Thompson, Antarctic sea ice control on ocean circulation in present and glacial climates, Proc. Natl. Acad. Sci. 111 (2014) 8753–8758. https://doi.org/10.1073/pnas.1323922111.

[11] A.Y. Shcherbina, L.D. Talley, D.L. Rudnick, Direct Observations of North Pacific Ventilation: Brine Rejection in the Okhotsk Sea, Science. 302 (2003) 1952–1955. https://doi.org/10.1126/science.1088692.

[12] R.A. Eastwood, R.W. Macdonald, J.K. Ehn, J. Heath, L. Arragutainaq, P.G. Myers, D.G. Barber, Z.A. Kuzyk, Role of River Runoff and Sea Ice Brine Rejection in Controlling Stratification Throughout Winter in Southeast Hudson Bay, Estuaries Coasts. 43 (2020) 756–786. https://doi.org/10.1007/s12237-020-00698-0.

[13] A. Bogdan, Ice Clouds: Atmospheric Ice Nucleation Concept versus the Physical Chemistry of Freezing Atmospheric Drops, J. Phys. Chem. A. 122 (2018) 7777–7781. https://doi.org/10.1021/acs.jpca.8b07926.

[15] E.J. Smith, A.D.J. Haymet, Ion Solubility in Ice: Calculation of Potentially Favorable Positions of Cl− and Na+ Ions in the SPC/E Model of Ice 1 h*, Mol. Simul. 30 (2004) 827–830. https://doi.org/10.1080/08927020410001709325.

[16] Y. Yashima, Y. Okada, M. Harada, T. Okada, Structures of ions accommodated in salty ice Ih crystals, Phys. Chem. Chem. Phys. 23 (2021) 17945–17952. https://doi.org/10.1039/D1CP01624E.

[17] M.R. Frank, C.E. Runge, H.P. Scott, S.J. Maglio, J. Olson, V.B. Prakapenka, G. Shen, Experimental study of the NaCl–H2O system up to 28GPa: Implications for ice-rich planetary bodies, Phys. Earth Planet. Inter. 155 (2006) 152–162.

https://doi.org/10.1016/j.pepi.2005.12.001.

[18] B. Journaux, I. Daniel, R. Caracas, G. Montagnac, H. Cardon, Influence of NaCl on ice VI and ice VII melting curves up to 6GPa, implications for large icy moons, Icarus. 226 (2013) 355–363. https://doi.org/10.1016/j.icarus.2013.05.039.

[19] J.-A. Hernandez, R. Caracas, S. Labrosse, Stability of high-temperature salty ice suggests electrolyte permeability in water-rich exoplanet icy mantles, Nat. Commun. 13 (2022) 3303. https://doi.org/10.1038/s41467-022-30796-5.

[20] M.M. Conde, M. Rovere, P. Gallo, Spontaneous NaCl-doped ices Ih, Ic, III, V and VI. Understanding the mechanism of ion inclusion and its dependence on the crystalline structure of ice, Phys. Chem. Chem. Phys. 23 (2021) 22897–22911. https://doi.org/10.1039/D1CP02638K.

[21] Water: A Tale of Two Liquids | Chemical Reviews, (n.d.). https://pubs.acs.org/doi/full/10.1021/acs.chemrev.5b00750 (accessed August 29, 2022).

[22] B. Guillot, A reappraisal of what we have learnt during three decades of computer simulations on water, J. Mol. Liq. 101 (2002) 219–260. https://doi.org/10.1016/S0167-7322(02)00094-6.

[24] C.P. Lamas, C. Vega, E.G. Noya, Freezing point depression of salt aqueous solutions using the Madrid-2019 model, J. Chem. Phys. 156 (2022) 134503. https://doi.org/10.1063/5.0085051.

[25] V. Bianco, M.M. Conde, C.P. Lamas, E.G. Noya, E. Sanz, Phase diagram of the NaCl–water system from computer simulations, J. Chem. Phys. 156 (2022) 064505. https://doi.org/10.1063/5.0083371.

[26] M.M. Conde, C. Vega, G.A. Tribello, B. Slater, The phase diagram of water at negative pressures: Virtual ices, J. Chem. Phys. 131 (2009) 034510. https://doi.org/10.1063/1.3182727.

[27] Luo, S., Jin, Y., Tao, R., Li, H., Li, C., Wang, J., & Li, Z. (2021). Molecular understanding of ion rejection in the freezing of aqueous solutions. Physical Chemistry Chemical Physics, 23(23), 13292–13299. https://doi.org/10.1039/D1CP01733K.

[28] S. Mahmood Fatemi Sh., M. Foroutan, Study on formation of unstable clathrate-like water molecules at freezing/melting temperatures of water and salty water, Fluid Phase Equilibria. 384 (2014) 73–81. https://doi.org/10.1016/j.fluid.2014.10.007.

[29] Tsironi, I., Schlesinger, D., Späh, A., Eriksson, L., Segad, M., & Perakis, F. (2020). Brine rejection and hydrate formation upon freezing of NaCl aqueous solutions. Physical Chemistry Chemical Physics, 22(14), 7625–7632. https://doi.org/10.1039/C9CP05436G https://doi.org/10.1039/C9CP05436G.

[30] Conde, M. M., Rovere, M., & Gallo, P. (2018). Molecular dynamics simulations of freezing-point depression of TIP4P/2005 water in solution with NaCl. *Journal of Molecular Liquids*, *261*, 513–519. https://doi.org/10.1016/j.molliq.2018.03.126.

[32] Vrbka, L., & Jungwirth, P. (2007). Molecular dynamics simulations of freezing of water and salt solutions. *Journal of Molecular Liquids*, *134*(1), 64–70. https://doi.org/10.1016/j.molliq.2006.12.011.

[33] Vrbka, L., & Jungwirth, P. (2005). Brine Rejection from Freezing Salt Solutions: A Molecular Dynamics Study. *Physical Review Letters*, *95*(14), 148501. https://doi.org/10.1103/PhysRevLett.95.148501.

[34] Soria, G. D., Espinosa, J. R., Ramirez, J., Valeriani, C., Vega, C., & Sanz, E. (2018). A simulation study of homogeneous ice nucleation in supercooled salty water. *The Journal of Chemical Physics*, *148*(22), 222811. https://doi.org/10.1063/1.5008889.

[35] Blazquez, S., Conde, M. M., Abascal, J. L. F., & Vega, C. (2022). The Madrid-2019 force field for electrolytes in water using TIP4P/2005 and scaled charges: Extension to the ions F−, Br−, I−, Rb+, and Cs+. *The Journal of Chemical Physics*, *156*(4), 044505. https://doi.org/10.1063/5.0077716.

[36] J.L.F. Abascal, C. Vega, A general purpose model for the condensed phases of water: TIP4P/2005, J. Chem. Phys. 123 (2005) 234505. https://doi.org/10.1063/1.2121687.

[37] C. Vega, J.L.F. Abascal, E. Sanz, L.G. MacDowell, C. McBride, Can simple models describe the phase diagram of water?, J. Phys. Condens. Matter. 17 (2005) S3283. https://doi.org/10.1088/0953-8984/17/45/013.

[38] C. Vega, J.L.F. Abascal, Simulating water with rigid non-polarizable models: a general perspective, Phys. Chem. Chem. Phys. 13 (2011) 19663–19688. https://doi.org/10.1039/C1CP22168J.

[39] S.P. Kadaoluwa Pathirannahalage, N. Meftahi, A. Elbourne, A.C.G. Weiss, C.F. McConville, A. Padua, D.A. Winkler, M. Costa Gomes, T.L. Greaves, T.C. Le, Q.A. Besford, A.J. Christofferson, Systematic Comparison of the Structural and Dynamic Properties of Commonly Used Water Models for Molecular Dynamics Simulations, J. Chem. Inf. Model. 61 (2021) 4521–4536. https://doi.org/10.1021/acs.jcim.1c00794.

[40] A.L. Benavides, J.L. Aragones, C. Vega, Consensus on the solubility of NaCl in water from computer simulations using the chemical potential route, J. Chem. Phys. 144 (2016) 124504. https://doi.org/10.1063/1.4943780.

[41] T. Yagasaki, M. Matsumoto, H. Tanaka, Lennard-Jones Parameters Determined to Reproduce the Solubility of NaCl and KCl in SPC/E, TIP3P, and TIP4P/2005 Water, J. Chem. Theory Comput. 16 (2020) 2460–2473. https://doi.org/10.1021/acs.jctc.9b00941.

[42] A.L. Benavides, M.A. Portillo, V.C. Chamorro, J.R. Espinosa, J.L.F. Abascal, C. Vega, A potential model for sodium chloride solutions based on the TIP4P/2005 water model, J. Chem. Phys. 147 (2017) 104501. https://doi.org/10.1063/1.5001190.

[43] S. Yue, A.Z. Panagiotopoulos, Dynamic properties of aqueous electrolyte solutions from non-polarisable, polarisable, and scaled-charge models, Mol. Phys. 117 (2019) 3538–3549. https://doi.org/10.1080/00268976.2019.1645901.

[44] Sneha, P., & George Priya Doss, C. (2016). Chapter Seven - Molecular Dynamics: New Frontier in Personalized Medicine. In R. Donev (Ed.), *Advances in Protein Chemistry and Structural Biology* (Vol. 102, pp. 181–224). Academic Press. https://doi.org/10.1016/bs.apcsb.2015.09.004.

[45] Generalic, Eni. "Lennard-Jones potential." *Croatian-English Chemistry Dictionary & Glossary*. 29 June 2022. KTF-Split. 13 Apr. 2024. <https://glossary.periodni.com>.

[46] Slavchov, R., & Ivanov, T. (2014). Quadrupole terms in the Maxwell equations: Born energy, partial molar volume, and entropy of ions. *The Journal of Chemical Physics*, *140*, 074503. https://doi.org/10.1063/1.4865878.

[47] Liu, M., Shi, Q., & Sun, Z. (2023). Molecular dynamics simulation of ammonium ion removal by freezing concentration. *Nano Express*, *3*(4), 045005. https://doi.org/10.1088/2632-959X/acad1a.

[48] *Thesis MECH 2021 Luo—991012980218103412.pdf*. (n.d.). Retrieved April 15, 2024, from https://lbezone.hkust.edu.hk/pdfviewer/web/viewer.php?file=aHR0cHM6Ly9sYmV6b25lLmhrdXN0LmVkdS5oay9vYmovMS9vLzk5MTAxMjk4MDIxODEwMzQxMi85OTEwMTI5ODAyMTgxMDM0MTIucGRm#page=1

[49] Rasmussen, A., Jannat, M., & Wang, H. (2024). Fundamentals of freeze desalination: Critical review of ion inclusion and rejection studies from molecular dynamics perspective. *Desalination*, *573*, 117216. https://doi.org/10.1016/j.desal.2023.117216.

[50] Luo, S., Li, C., Li, F., Wang, J., & Li, Z. (2019). Ice Crystallization in Shear Flows. *The Journal of Physical Chemistry C*, *123*(34), 21042–21049. https://doi.org/10.1021/acs.jpcc.9b06225.

[51] *4.7.3. Radial Distribution Functions—MDAnalysis.analysis.rdf—MDAnalysis 1.1.0 documentation*. (n.d.). Retrieved April 15, 2024, from https://docs.mdanalysis.org/1.1.0/documentation_pages/analysis/rdf.html.

[52] Luo, S., Li, C., Li, F., Wang, J., & Li, Z. (2019). Ice Crystallization in Shear Flows. The Journal of Physical Chemistry C, 123(34), 21042–21049. http://dx.doi.org/10.1021/acs.jpcc.9b06225.

[53] Espinosa, J. R., Sanz, E., Valeriani, C., & Vega, C. (2013). On fluid-solid direct coexistence simulations: The pseudo-hard sphere model. *The Journal of Chemical Physics*, *139*(14), 144502. https://doi.org/10.1063/1.4823499.

[54] Frenkel, D. (2012). Simulations: The dark side (arXiv:1211.4440). arXiv. http://arxiv.org/abs/1211.4440.

[55] P. Nielaba, M. Mareschal, and G. Ciccotti, Bridging the Time Scales. (n.d.). https://link.springer.com/book/10.1007/3-540-45837-9.

[56] Abrams, C., & Bussi, G. (2014). Enhanced Sampling in Molecular Dynamics Using Metadynamics, Replica-Exchange, and Temperature-Acceleration. *Entropy*, *16*(1), Article 1. https://doi.org/10.3390/e16010163.

[57] G. C. Sosso, J. Chen. (2016). Crystal Nucleation in Liquids: Open Questions and Future Challenges in Molecular Dynamics Simulations. *Chemical Reviews*. (n.d.). https://pubs.acs.org/doi/10.1021/acs.chemrev.5b00744.

[58] Frenkel, D., Smit, B., & Smit, B. (2001). *Understanding Molecular Simulation: From Algorithms to Applications*. Elsevier Science & Technology. http://ebookcentral.proquest.com/lib/usu/detail.action?docID=307221.

[59] Rapaport, D. C. (2004). *The Art of Molecular Dynamics Simulation* (2nd ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511816581.

[60] Meller, J. (2001). Molecular Dynamics. In *Encyclopedia of Life Sciences*. John Wiley & Sons, Ltd. https://doi.org/10.1038/npg.els.0003048.

[61] Ghasemi, M., Shafiei, A., & Foroozesh, J. (2022). A systematic and critical review of application of molecular dynamics simulation in low salinity water injection. *Advances in Colloid and Interface Science*, *300*, 102594. https://doi.org/10.1016/j.cis.2021.102594.

[62] Zhang, S., Zhang, C., Wu, S., Zhou, X., He, Z., & Wang, J. (2021). Ion-Specific Effects on the Growth of Single Ice Crystals. *The Journal of Physical Chemistry Letters*, *12*(36), 8726–8731. https://doi.org/10.1021/acs.jpclett.1c02601.

APPENDICES

APPENDIX A

NVT_235K_0.6M_TIP4P/2005

```
#'TIP4P/2005'
#Total 11520 molecules 3840,20layers,576 atoms,192 molecules per
layer.
# 2ice 6liq 4ice 6liq 2ice=20layer
units               real
dimension           3
boundary      p p p
atom_style          full
comm_modify     vel yes

#parallel

read_data           ice01.data
#read_restart   iceE002_9.eq

#  ' interactions 1O 2H 3Na 4Cl'
pair_style     lj/cut/tip4p/long 1 2 1 1 0.1546 12.0 10.0
pair_coeff        1 1   0.1852 3.1589
pair_coeff        2 2   0.0 0.0
pair_coeff        3 3   0.3519   2.21737
pair_coeff        4 4   0.01839 4.69906
pair_coeff        1 2   0.0 0.0
pair_coeff        1 3   0.18962 2.60838
pair_coeff        1 4   0.01481 4.23867
pair_coeff        2 3 0.0 0.0
pair_coeff        2 4 0.0 0.0
pair_coeff        3 4 0.3439 3.00512


bond_style         harmonic
bond_coeff      1 0.0 0.9572
angle_style        harmonic
angle_coeff     1 0.0 104.52

kspace_style   pppm/tip4p 1.0e-5
#kspace_modify      order 7


group            water  type 1 2
group            kation   type 3
group               anion type 4
group            hydrogen  type 2
group            oxygen   type 1
group               liquid id 3601:18000
group               liqoxy intersect oxygen liquid
group               ion type 3 4
```

```
neighbor       2.0 bin
neigh_modify     delay 1

timestep      2


compute         Tliq water temp
compute       Tion ion temp
compute         Poperatom water pe/atom
compute         Pope water reduce sum c_Poperatom
compute       Tpar all temp/partial 1 0 1



fix                SHAKE water shake 0.0001 200 0 b 1 a 1

#fix               NPT all npt temp 235.0 235.0 200 z 1.0 1.0
2000.0
#fix_modify        NPT temp Tpar

#fix            zeromom all momentum 1 linear 0 0 1

fix             zeromom all momentum 100 linear 1 1 1

fix                NVT all nvt temp 235.0 235.0 200
fix_modify         NVT temp Tpar

#fix            fxfld all efield 0.0 0.02 0.0



thermo_style     custom step temp press pe ke c_Tliq c_Tion lx ly
lz
thermo              100000
thermo_modify        flush yes

fix           Cp water ave/time 100000 1 100000 c_Pope file
potential.dat


# ====================================================
# dump atom positions for visualization, e.g. using VMD
# ====================================================
dump             all all custom 100000 all.lammpstrj id type x y
z
dump_modify   all flush yes

dump          dumpoxygen oxygen xyz 20000 dump.oxygen.xyz
dump_modify   dumpoxygen flush yes
dump          dumphydrogen hydrogen xyz 20000 dump.hydrogen.xyz
dump_modify   dumphydrogen flush yes
```

```
dump            dumpkation kation xyz 20000 dump.kation.xyz
dump_modify     dumpkation flush yes
dump            dumpanion anion xyz 20000 dump.anion.xyz
dump_modify     dumpanion flush yes

# =====================================================
# dump atom velocities
# =====================================================

#dump    dumpoxygenvelocity liqoxy custom 10000 velocityO.dat x vy
#dump_modify dumpoxygenvelocity flush yes sort id

#dump    dumpionvelocity ion custom 100 velocityIon.dat x vy
#dump_modify dumpionvelocity flush yes sort id

compute oxyvel oxygen chunk/atom     bin/1d x lower 2.0 units box
discard no

fix vel oxygen ave/chunk 1 10000000 10000000 oxyvel vy norm
sample file velO.dat

compute  VelperNa kation property/atom vy

compute  VelperCl anion property/atom vy

compute  VelNa kation reduce sum c_VelperNa

compute  VelCl anion reduce sum c_VelperCl

fix Velocity ion ave/time 1 250000000 250000000 c_VelNa c_VelCl
file Velion.dat


#Enter loop
variable a loop 20
label loop

#Runtime 100ns
run 5000000

write_data output/system_gb_test_$a.data
write_restart restart/system_gb_test_restart_$a.data

#write_restart            iceE002_2.eq
#write_data                     iceE002_2.data
```

APPENDIX B

ION REJECTION RATE CALCULATION

(A)MD_data_analysis and store

```python
"""
Imports
"""
#Imports from Python libraries/packages
import os
import time

#Import from custom modules and classes
from util.database_handling import pickleStorage
from util.Ion_Atom import Atom
from util.Timestep import Timestep
from util.util import timeReq

"""
Receive User Input for the saving and processing of the MD simulation Data
"""
#Recieve user input for the datafile name:
print("MD Simulation Data File Name: ")
fileName = input()

print("Directory for Pickled Object Data: ")
directory = input()

print("Prefix for Pickled Object Files: ")
filePrefix = input()

#Generate the directory where the pickled files will be saved.
if not os.path.isdir(directory):
    command = "mkdir " + directory
    os.system(command)

print("Store file names in database (1 = yes): ")
doDatabaseStorage = input() == "1"
databaseName = ""
if doDatabaseStorage:
    print("Enter Database Name: ")
    databaseName = "databases/" + input() + "_db.sqlite"
```

```
"""
MD Simulation Data file Processing
"""
#Track the time required for the program
start = time.time()
print("Beginning File Processing")

#initialize parameters
timesteps = {}
timestep_values = []
atom_ids = []

#Open the data file
dataFile = open(fileName)

#Pull in the data from the file. Store them as Atom objects
line = dataFile.readline().removesuffix("\n")
count = 0
while line != "":
    if "TIMESTEP" in line:
        #Set up values for new timestep
        #if count >= 20:
        #    break
        #count += 1
        timestep = int(dataFile.readline().removesuffix("\n"))
        timesteps[timestep] = Timestep(timestep)
        timestep_values.append(timestep)

        #if count % 25 == 0:
        #    print("Timestep = " + str(timestep))

    elif "NUMBER" in line:
        num_atoms = int(dataFile.readline().removesuffix("\n"))

    elif "BOX BOUNDS" in line:
        #Box Bounds
        xlo, xhi = dataFile.readline().removesuffix("\n").split(" ")
        ylo, yhi = dataFile.readline().removesuffix("\n").split(" ")
        zlo, zhi = dataFile.readline().removesuffix("\n").split(" ")

        #Float Values
        xLo = float(xlo)
        xHi = float(xhi)
        yLo = float(ylo)
        yHi = float(yhi)
        zLo = float(zlo)
        zHi = float(zhi)
        #Add Dimensions to the timestep
        #if timestep % 200000 == 0:
        timesteps[timestep].addDimensions((xHi, xLo, yHi, yLo, zHi, zLo))

        #Skip next line
        dataFile.readline()
    else:
        #Generate new atoms and add them to the system
        atom_id, atom_type, x, y, z = line.split(" ")
        atom_ids.append(int(atom_id))
        timesteps[timestep].addAtom(Atom(int(atom_id), float(x), float(y), float(z), atom_type))

    #Move to the next line
    line = dataFile.readline().removesuffix("\n")

#Close the file
dataFile.close()

#Pull the time required for reading the data
fileReadTime = time.time()

"""
Call the Method to store the timestep data in storage and in the database
"""
print("Storing Timestep Data in Pickled Files and Database...")
for timestep in timestep_values:
    pickleStorage(timestep, timesteps[timestep], filePrefix, directory, databaseName, doDatabaseStorage)

#Pull the time required for neighbor addition and tetrahedrality Calculations
print("...Pickling == COMPLETE")
pickleTime = time.time()


"""
Output Total Times
"""
endTime = time.time()
print("File Processing Time:")
timeReq(start, fileReadTime)
print("Pickling and Database Storage time: ")
timeReq(fileReadTime, pickleTime)
print("Total Time:")
timeReq(start, pickleTime)
```

(B)Tetrahedrality Calculation

```
from mpi4py import MPI
import numpy as np
import math
import util.database_handling as db_handle
import util.neighbor_handling as nb_handle
import util.util as utilities
import sys
import sqlite3
import time

if len(sys.argv) < 3:
    print("Usage: Input number of timesteps being analyzed and timestep database path.")
    sys.exit(1)
numSteps = int(sys.argv[1])
database = sys.argv[2]

comm = MPI.COMM_WORLD
size = comm.Get_size()
rank = comm.Get_rank()

numPerProc = math.ceil(numSteps / size)

split_timesteps = None
if rank == 0:
    timesteps = db_handle.pullTimestepList(database)
    split_timesteps = utilities.splitList(timesteps, size, numPerProc)
    print(split_timesteps)

recvbuf = np.empty(numPerProc, dtype="l")
comm.Scatter(split_timesteps, recvbuf, root=0)

print("I'm ", rank, " and I recieved: ", recvbuf)

#Connect to the database
try:
    query = "file:" + database + "?mode=rw"
    conn = sqlite3.connect(query, uri=True)
except:
    print("That database doesn't exist.\nCheck your pathname and retry.")

print("Neighbor Addition and Tetra Calculations beginning on processer ", rank)
start = time.time()
for i in recvbuf:
    if i == 1:
        continue
    start_2 = time.time()
    timestep, file_path = db_handle.pullTimestep(i, conn)
    dimensions = timestep.getDimensions()
    delta_x = dimensions[1] - dimensions[0]
    delta_y = dimensions[3] - dimensions[2]
    delta_z = dimensions[5] - dimensions[4]
    nb_handle.constructNeighborhood(timestep, "1", 4, delta_x, delta_y, delta_z)
    db_handle.pickleObject(timestep, file_path)
    print("\tCompleted Neighbor Addition and Tetrahedrality Calculations for timestep: " + str(i))
    end_2 = time.time()
    print("\tTime required: ",end='')
    utilities.timeReq(start_2, end_2)
end = time.time()
print("Analysis Complete on rank: ", rank,". Time required:",end="")
utilities.timeReq(start, end)
print()
```

(C) Interface Calculation

```
"""
This file pulls data from the pickled timesteps and determines the interface location and plots it across the
    MD Simulation data with the curve_fit data as well.
"""

#Imports
import sqlite3
import matplotlib.pyplot as plt
import util.tetrahedral_calculations as tetrahedral_calculations
import util.database_handling as db_handle
import os
import time
import sys

#User enters input if none is given in command line
if len(sys.argv) < 2:
    #Connect to the database given by the user
    cont = False
    conn = None
    while not cont:
        print("Enter the Database File Path: ",end="")
        database = input()
        try:
            query = "file:" + database + "?mode=rw"
            conn = sqlite3.connect(query, uri=True)
        except:
```

```
        print("That database doesn't exist.\nCheck your pathname and retry.")
    else:
        print("That is an existing database. Connection Established.")
        cont = True
cur = conn.cursor()
poss_timesteps = [timestep[0] for timestep in cur.execute("SELECT timestep FROM timesteps")]
#timesteps = [str(i * 10000000) for i in range(0, 103)]
print("Here are the possible timesteps to analyze: ")
count = 0
output = ""
while count < len(poss_timesteps):
    count2 = 0
    while count2 < 10 and count < len(poss_timesteps):
        output += str(poss_timesteps[count]) + "\t"
        count += 1
        count2 += 1
    output += "\n"
print(output)

#More User Input
print("What timesteps to analyze ('all' for all timesteps, comma-separated list otherwise): ")
toAnalyze = input()
print("Enter Folder to save output in: ",end='')
folderName = input()
print("Add final interface calculations to a database (1 = yes): ",end='')
doDatabase = input() == "1"
if doDatabase:
    print("Enter the name of the database: ",end='')
    database2 = "databases/" + input() + "db.sqlite"
    if "databases/" not in database2:
        database2 = "databases/" + database2
    if "_db.sqlite" not in database2:
        database2 += "_db.sqlite"
print("Save to file for Excel (1 = yes): ",end='')
doExcel = input() == "1"
if doExcel:
    print("Enter the name of the file to save the data to for Excel: ",end='')
    fileName = folderName + input()
    if ".csv" not in fileName:
        fileName += ".csv"
print("Plot Calculations (1 = yes): ",end='')
doPlotting = input() == "1"


#Generate the directory where the pickled files will be saved.
if not os.path.isdir(folderName):
    command = "mkdir " + folderName
    os.system(command)
```

```python
    folderName += "/"

    #Get the timesteps to analyze
    if toAnalyze == "all":
        timesteps = poss_timesteps
    else:
        timesteps = toAnalyze.split(",")
else:
    try:
        conn = None
        try:
            query = "file:" + sys.argv[1] + "?mode=rw"
            conn = sqlite3.connect(query, uri=True)
        except:
            print("That database doesn't exist.\nCheck your pathname and retry.")
            exit()
        else:
            print("That is an existing database. Connection Established.")

        #Generate the directory where the pickled files will be saved.
        folderName = sys.argv[2]
        if not os.path.isdir(folderName):
            command = "mkdir " + folderName
            os.system(command)
        folderName += "/"
        if len(sys.argv) < 5:
            #Pull Timesteps to analyze
            if sys.argv[3] == "all":
                cur = conn.cursor()
                timesteps = [timestep[0] for timestep in cur.execute("SELECT timestep FROM timesteps")]
            else:
                if len(sys.argv) < 5:
                    timesteps = sys.argv[3].split(",")
                else:
                    timesteps = sys.argv[3:]
            doDatabase = False
            doExcel = False
            doPlotting = False
        else:
            if len(sys.argv) < 7:
                raise Exception("Incorrect Command Line Entries")
            if not (sys.argv[3].lower() == "true" or sys.argv[3].lower() == "false"):
                raise Exception("Incorrect Command Line Entries")
            if not (sys.argv[4].lower() == "true" or sys.argv[4].lower() == "false"):
                raise Exception("Incorrect Command Line Entries")
            if not (sys.argv[5].lower() == "true" or sys.argv[5].lower() == "false"):
                raise Exception("Incorrect Command Line Entries")
        doDatabase = sys.argv[3].lower() == "true"
        doExcel = sys.argv[4].lower() == "true"
        doPlotting = sys.argv[5].lower() == "true"
        if (doDatabase and doExcel) and len(sys.argv) < 9:
            raise Exception("Incorrect Command Line Entries")
        if (doDatabase or doExcel) and len(sys.argv) < 8:
            raise Exception("Incorrect Command Line Entries")
        if doDatabase and doExcel:
            database2 = sys.argv[6]
            fileName = sys.argv[7]
            if "databases/" not in database2:
                database2 = "databases/" + database2
            if "_db.sqlite" not in database2:
                database2 += "_db.sqlite"
            if ".csv" not in fileName:
                fileName += ".csv"
            if sys.argv[8] == "all":
                cur = conn.cursor()
                timesteps = [timestep[0] for timestep in cur.execute("SELECT timestep FROM timesteps")]
            else:
                timesteps = sys.argv[8].split(",")
        if doDatabase and not doExcel:
            database2 = sys.argv[6]
            if "databases/" not in database2:
                database2 = "databases/" + database2
            if "_db.sqlite" not in database2:
                database2 += "_db.sqlite"
            if sys.argv[7] == "all":
                cur = conn.cursor()
                timesteps = [timestep[0] for timestep in cur.execute("SELECT timestep FROM timesteps")]
            else:
                timesteps = sys.argv[7].split(",")
        if not doDatabase and doExcel:
            fileName = sys.argv[6]
            if ".csv" not in fileName:
                fileName += ".csv"
            if sys.argv[7] == "all":
                cur = conn.cursor()
                timesteps = [timestep[0] for timestep in cur.execute("SELECT timestep FROM timesteps")]
            else:
                timesteps = sys.argv[7].split(",")
        if not doDatabase and not doExcel:
            if sys.argv[6] == "all":
                cur = conn.cursor()
                timesteps = [timestep[0] for timestep in cur.execute("SELECT timestep FROM timesteps")]
            else:
                timesteps = sys.argv[6].split(",")
    except Exception as exc:
        print(exc)
        print("Incorrect Command line entries.")
        print("Must have at least the following: ")
        print("\ttimestep database path")
        print("\toutput folder path")
        print("\ttimesteps to analyze (all or comma-seperated values)")
        print("Additional Arguements include: ")
        print("*Note, timesteps to analyze must always be last argument")
        print("**Note, if any additional arguments are include, all save inquiries must be input.")
        print("\tSave to a database (true or false)")
        print("\tSave to csv for Excel file (true or false) ")
        print("\tPlot Calculations (true or false)")
        print("\tDatabase file name (if saving)")
        print("\tCSV file name (if saving)")
        print("Examples:")
        print("python interface_calculations.py databases/test_db.sqlite output_folder true true false
test.db_sqlite test.csv all")
        print("python interface_calculations.py databases/test_db.sqlite output_folder false true false test.csv
0,500000000")
        exit()
    else:
        for i in range(len(timesteps)):
            try:
                timesteps[i] = int(timesteps[i])
            except:
                print("You entered an invalid timestep.")
                conn.close()
                exit()



#Pull the Timestep Objects from their pickled location
data_timesteps = []
for timestep in timesteps:
    data_timesteps.append(db_handle.pullTimestep(timestep, conn)[0])
conn.close()

#Calculate the averages of each timestep
print("Calculating Averages...")
averages = []
for i in range(len(data_timesteps)):
    start = time.time()
    averages.append(tetrahedral_calculations.average_tetra(data_timesteps[i], 1))
    end = time.time()
    print("Averages Calculated for timestep in " + str(round(end - start, 3)) + " seconds: " + str(timesteps[i]))

#Calculate interfacial parameters
interface_params = []
```

```
fitData = []
for i in range(len(averages)):
    time_tuple = (int(timesteps[i]))
    popt, perr, fit_data = tetrahedral_calculations.locate_interface(*averages[i])
    test_tuple = (timesteps[i], *popt)
    interface_params.append(test_tuple)
    fitData.append(fit_data)

if doDatabase:
    try:
        conn = sqlite3.connect(database2)
        cur = conn.cursor()
        query = "CREATE TABLE data " + """
            (timestep INTEGER, interface_location REAL, interface_width REAL, c REAL, PRIMARY KEY
(timestep))"""
        cur.execute(query)
        conn.commit()
        script = "INSERT INTO data " + "(timestep, interface_location, interface_width, c) VALUES (?,?,?,?)"
        cur.executemany(script, interface_params)
        conn.commit()
        conn.close()
    except Exception as exc:
        print("Database Addition Failed. Continuing with Program.")
        print(exc)
    else:
        print("Database Generation Successful!\n")

if doExcel:
    if ".csv" not in fileName:
        fileName += ".csv"
    try:
        newFile = open(fileName, "w")
        newFile.write("timesteps (ns),Interface Location (angstroms),Interfacial Width (angstroms),c\n")
        for params in interface_params:
            outString = str(params[0]) + "," + str(params[1]) + "," + str(params[2]) + "," + str(params[3]) + "\n"
            newFile.write(outString)
        newFile.close()
    except Exception as exc:
        print("File Generation Failed. Continuing with Program")
        print(exc)
    else:
        print("File Generation Successful!\n")

if doPlotting:
    fig, ax = plt.subplots()
    print("Beginning Plotting...")
    for i in range(len(averages)):
        title = "Average Tetrahedral Order Parameter at " + str(int(timesteps[i]) / 10**6) + " ns"
        pngName = folderName + "tetraplot_" + str(timesteps[i]) + ".png"
        tetrahedral_calculations.plotTetra(ax, title, averages[i][0], averages[i][1], interface_params[i][1],
fitData[i], pngName)
    print("...Plotting Complete.")
```

(D) Ion Rejection Rate Calculation and Plot

```
#Imports
import sqlite3
import sys
import matplotlib.pyplot as plt
from util.database_handling import pullTimestep
from util.get_ion_param import getMolarMass
import util.ion_rejection_functions as ion_rejection_functions

"""
    *Other Ideas:
    ? Store the rejection rates in a database
    ? Have a method for calculating of averages?
    ? Have a method to plot against temperature? - This is most likely going to happen.
"""
"""
Main Running Parameters
"""
#*User Input
#print("Calculate Parameters based on averages (1 = yes): ",end='')
#doAverages = input() == "1"


if len(sys.argv) < 2:
    """
    !This section recieves input for the interface location sqlite3 database
    """
    #Initialize flag and the connection varaiable.
    complete = False
    conn_interface = None
    while not complete:
        print("Enter the Database File Path for the interface location data (-1 to quit): ",end="")
        database = "databases/interfacial_0710_params_db.sqlite"#input()
        if database == "-1":
            quit
        try:
            query = "file:" + database + "?mode=rw"
            conn_interface = sqlite3.connect(query, uri=True)
        except:
            print("That database doesn't exist.\nCheck your pathname and retry.")
        else:
```

```
        print("That is an existing database. Connection Established.")
        complete = True

#Generate the Cursor connected to the database
cur_interface = conn_interface.cursor()

"""
!This section receives input for the timestep storage sqlite3 database
"""
#Initialize flag and the connection varaiable.
complete = False
conn_timesteps = None
while not complete:
    print("Enter the Database File Path for the timestep data (-1 to quit): ",end="")
    database_2 = "databases/0710_test_db.sqlite"#input()
    if database_2 == "-1":
        quit()
    try:
        query = "file:" + database_2 + "?mode=rw"
        conn_timesteps = sqlite3.connect(query, uri=True)
    except:
        print("ERROR: That database doesn't exist.\nCheck your pathname and retry.")
    else:
        print("That is an existing database. Connection Established.")
        complete = True

#Generate the Cursor connected to the database
cur_timesteps = conn_timesteps.cursor()

"""
!This section recieves input for the molar mass sqlite3 database
"""
#Initialize flag and the connection varaiable.
complete = False
conn_ion_params = None
while not complete:
    print("Enter the Database File Path for the molar mass data (-1 to quit): ",end="")
    database_3 = "databases/ion_params_db.sqlite"#input()
    if database_3 == "-1":
        quit()
    try:
        query = "file:" + database_3 + "?mode=rw"
        conn_ion_params = sqlite3.connect(query, uri=True)
        conn_ion_params.close()
    except:
        print("That database doesn't exist.\nCheck your pathname and retry.")
    else:
        print("That is an existing database. Thank you.")
```

```
            complete = True

    #Determine the ions to be analyzed:
    print("Enter the ions to be analyzed as a commma-separated list (no comma if only 1 ion): ")
    print("\t\tExample: Na,Cl,Ca")
    ions = input()
    if "," in ions:
        ions = ions.removesuffix("\n").split(",")
    else:
        ions = [ions.removesuffix("\n")]
    print("Enter the atom_type id for each ion entered: ")
    atom_types = []
    for ion in ions:
        print("\t for ion " + ion + ": ",end="")
        atom_types.append(input())

    #Continuing User Input - Name of plot and should it be saved/where
    print("Enter the name of the plot (-1 to quit): ",end='')
    plotName = "test_output/trial_1.png"#input()
    if plotName == "-1":
        quit()
    print("Save the plot (1 = yes): ",end='')
    doPlotSave = input() == "1"

    #Continuing User Input - Name of the csv file for data storage
    print("Enter the file name for a csv file (none to not save.):",end="")
    csvFileName = input()
    doCSV = csvFileName.lower() != "none"
    if ".csv" not in csvFileName:
        csvFileName += ".csv"

else:
    if len(sys.argv) < 7 or len(sys.argv) > 8:
        print("ERROR: Invalid CMD Line Argumennts")
        print("CMD line arguments are:")
        print("\tInterface Database")
        print("\tData Storage Database")
        print("\tAtom Information Database")
        print("\tIons to analyze - as a comma-seperated list")
        print("\tAtom types for the list - as a comma-seperated list")
        print("\tCSV file name for storage (if none, no data will be saved to a csv file)")
        print("\t(Optional) Plot File Names")
        print("\tExamples:")
        print("\t\tpython ion_rejection_calc.py interfaces_db.sqlite object_data_db.sqlite atom_info_db.sqlite
Na,Cl,Ca 3,4,5 plot_plot.png")
        print("\t\tpython ion_rejection_calc.py interfaces_db.sqlite object_data_db.sqlite atom_info_db.sqlite
Na,Cl,Ca 3,4,5")
        exit()
    database = sys.argv[1]
    database_2 = sys.argv[2]
    database_3 = sys.argv[3]
    ions = sys.argv[4].split(",")
    atom_types = sys.argv[5].split(",")
    csvFileName = sys.argv[6]
    doCSV = csvFileName.lower() != "none"
    if ".csv" not in csvFileName:
        csvFileName += ".csv"


    #!This section recieves input for the interface location sqlite3 database
    try:
        query = "file:" + database + "?mode=rw"
        conn_interface = sqlite3.connect(query, uri=True)
    except:
        print("That database doesn't exist.\nCheck your pathname and retry.")
        exit()
    else:
        print("That is an existing database. Connection Established.")

    #Generate the Cursor connected to the database
    cur_interface = conn_interface.cursor()

    #!This section receives input for the timestep storage sqlite3 database
    try:
        query = "file:" + database_2 + "?mode=rw"
        conn_timesteps = sqlite3.connect(query, uri=True)
    except:
        print("ERROR: That database doesn't exist.\nCheck your pathname and retry.")
        exit()
    else:
        print("That is an existing database. Connection Established.")

    #Generate the Cursor connected to the database
    cur_timesteps = conn_timesteps.cursor()

    #Continuing User Input - Name of plot and should it be saved/where
    if len(sys.argv) == 8:
        plotName = sys.argv[7]
        doPlotSave = True
    else:
        plotName = "placeholder.png"
        doPlotSave = False

#*Get Timestep List:
print("These are the timesteps in the data chosen: ")
```

```
timesteps = [values[0] for values in cur_timesteps.execute("SELECT timestep from timesteps")]
ion_rejection_functions.printDataTables(timesteps, columns=10)

#*Loop through all the timesteps:
interface_locations = [] #& List of interface locations as a tuple with the location, interface width, and c value
ion_rejection_vals = [[] for i in range(len(atom_types))] #& 2D matrix of the ion rejection values per ion type.
ions_in_liq = [[] for i in range(len(timesteps))] #& 2D matrix storing the number of ions/ion_type/timestep
concentration_vals = [] #& The concentrations calculated using the total quantity of ions.

#*Get the molar masses of each ion and store in a list
molar_masses = [] #& List of the molar masses of each ion in the solution
for ion in ions:
    molar_masses.append(getMolarMass(database_3, ion))

#*Get the molar mass of water
molar_mass_water = getMolarMass(database_3, "Ow") + 2 * getMolarMass(database_3, "H")

for i in range(len(timesteps)):
    #*Pull the timestep
    timestep_obj = pullTimestep(timestep=timesteps[i], conn=conn_timesteps)[0]

    #*Pull the interface location information
    interface_data = ion_rejection_functions.pullInterfaceLocation(timestep=timesteps[i],
database=conn_interface, table="data")
    interface_locations.append(interface_data)

    #*Determine the domains to count the atoms
    left_domain = (timestep_obj.getDimensions()[1], interface_data[1])
    right_domain = (interface_data[1], timestep_obj.getDimensions()[0])

    #*Count ions/atoms of interest in each section
    water_count = ion_rejection_functions.count_atoms(timestep=timestep_obj, atom_type="1",
domain=right_domain)
    for j in range(len(atom_types)):
        atoms_left = ion_rejection_functions.count_atoms(timestep=timestep_obj, atom_type=atom_types[j],
domain=left_domain)
        atoms_right = ion_rejection_functions.count_atoms(timestep=timestep_obj, atom_type=atom_types[j],
domain=right_domain)

        #*Determine ion rejection for each atom type
        ion_rejection = ion_rejection_functions.calc_ion_rejection(ion_trapped=atoms_left, ion_total=(atoms_left
+ atoms_right))

        #Add to lists/totals#
        ion_rejection_vals[j].append(ion_rejection)
        ions_in_liq[i].append(atoms_right)

    #*Calculate Concentration of Salts:
```

```
    concentration_vals.append(ion_rejection_functions.calc_ppt_concentration(ions_in_liq[i], water_count,
molar_masses, molar_mass_water))

#*Plot Changes in Concentrations
x_values = [(x * 2) / 1000000 for x in timesteps]

fig, (ax1, ax2)= plt.subplots(2, sharex=True)
ax1.set_xlim(0, x_values[len(x_values) - 1])
ax1.set_ylim(60, 100)
ax1.set_ylabel("Ion Rejection Rate(%)", fontsize=12)
ax1.plot(x_values, ion_rejection_vals[0], "b-", label=ions[0])
ax1.plot(x_values, ion_rejection_vals[1], "r-", label=ions[1])
ax1.set_title("Rate of Ion Rejection over Time")
ax1.legend(loc="lower left")
ax2.set_ylim(3.4, 5.5)
ax2.set_ylabel("Concentration(m/m %)")
ax2.set_title("Change in Concentration over Time")
ax2.plot(x_values, concentration_vals, "g-", label="NaCl Salt")
ax2.set_xlabel("Time (ns)")
plt.savefig(plotName)
conn_interface.close()
conn_timesteps.close()

if doCSV:
    csvFile = open(csvFileName, "w")
    header = "Timestep(ns),Concentration(%m/m),"
    for i in ions:
        header += str(i) + ","
    csvFile.write(header+"\n")
    for i in range(len(x_values)):
        line = str(x_values[i]) + ","
        line += str(concentration_vals[i]) + ","
        for j in ion_rejection_vals:
            line += str(j[i]) + ","
        line += "\n"
        csvFile.write(line)
    csvFile.close()
```

APPENDIX C

UMBRELLA SAMPLING CALCULATION

```
#========================
# Initialization
#========================
units            real
dimension   3
atom_style  full
bond_style  harmonic
angle_style      harmonic
pair_style  lj/cut/tip4p/long 1 2 1 1 0.1546 12.0 10.0
kspace_style     pppm/tip4p 1.0e-5
boundary   p p p

#Various necessary variables
variable interface equal 47.97  #Change this to the interface
variable particle_mass equal 22.99  #Change based on the ion being
used.
variable y_high equal 39.1867
variable y_low equal 0.07815
variable z_high equal 36.874
variable z_low equal 0.12754
variable particle_x_pos equal ${interface}
variable particle_y_pos equal 0.5*(${y_high}+${y_low})
variable particle_z_pos equal 0.5*(${z_high}+${z_low})
variable k equal 1000/100
variable k_ice equal 2500/100

#read in system data and add together
read_data system_gb_test_20.data extra/atom/types 1  #Change the
data file to the one you are using for Ion Rejection Simulation

#Add the umbrella sampling particle
create_atoms 5 single ${particle_x_pos} ${particle_y_pos}
${particle_z_pos}

# Water Pair coefficients
pair_coeff 1 1 0.1852 3.1589
pair_coeff 2 2 0.0      0.0
pair_coeff 1 2 0.0      0.0

# Ion Pair Coefficients
pair_coeff 3 3 0.35190153 2.21737
pair_coeff 4 4 0.01838504 4.69906
pair_coeff 1 3 0.18962 2.60838
pair_coeff 1 4 0.01481 4.23867
pair_coeff 2 3 0.0 0.0
pair_coeff 2 4 0.0 0.0
pair_coeff 3 4 0.3439039 3.00512

#US Extra Atom Coefficients - Change per ion (this is for sodium
currently)
pair_coeff 1 5 0.18962 2.60838
pair_coeff 2 5 0.0 0.0
pair_coeff 3 5 0.3439039 3.00512
pair_coeff 4 5 0.35190153 2.21737
pair_coeff 5 5 0.35190153 2.21737

#Bond and Angle Coefficients
bond_coeff 1 0.0 0.9572
angle_coeff 1 0.0 104.52

#Neighbor Handling
neighbor 2.0 bin
neigh_modify delay 1

#Mass for the 5th particle
mass 5 ${particle_mass}

#Groups
region ice_reg block EDGE ${interface} EDGE EDGE EDGE EDGE
group oxygen           type 1
group hydrogen     type 2
group sodium           type 3
group chloride         type 4
group water         type 1 2
group salts         type 3 4
group ice           region ice_reg
group ice_oxygens   intersect oxygen ice
group topull        type 5

#==============================================
#     Minimization Running
#==============================================
#Change bonds to have high values to imitate the SHAKE algorithm
#Bond Coefficient for Harmonic Bond
bond_coeff 1 100000.0 0.9572

#Angle Coefficient for Harmonic Angle style
angle_coeff 1 10000.0 104.52

#Thermodynamic Handling
thermo_style custom step temp etotal press
thermo 10

#Spring fix to force minimization to be focused on solely ions
fix tether water spring/self 1000

#Minimization Parameters
min_style cg
minimize 1e-5 1e-5 5000 10000
```

```
unfix tether

#==============================================
#   Umbrella Sampling Parameters
#==============================================
#Reset the timestep
reset_timestep 0

#Bond and Angle Coefficients
bond_coeff 1 0.0 0.9572
angle_coeff 1 0.0 104.52

#Shake Algorithm
fix SHAKE all shake 0.0001 200 0 b 1 a 1

#NPT running
fix NPT all npt temp 260.0 260.0 100 y 1.0 1.0 2000.0 z 1.0 1.0
2000.0
compute Tpar all temp/partial 1 1 1
fix_modify NPT temp Tpar

#Force Fix - to keep the ice from melting.
fix springforce ice_oxygens spring/self ${k_ice}

#Momentum Fix - keeps the particles centered in the box
fix zeroMom all momentum 100 linear 1 1 1
fix recenter all recenter INIT INIT INIT

#Timestep (same for equilibration and production run)
timestep 2

#Output for visualizaiton and tetrahedrality order parameter
calculations.
dump all all custom 500000 vis_us.lammpstrj id type x y z

thermo 500000

#Umbrella Sampling Loop
variable b loop 48 #Change for a differnt number of bins
label loop2

#Variables
variable xdes equal (${interface}+7)+(${b}/2-15) #Change this to
change the range of x-dist sampled
variable xave equal xcm(topull,x)

#Bias Potential for Umbrella Sampling
fix bias topull spring tether ${k} ${xdes} 0 0 0

#Equilibration Running
run 1000000

#Data Collection Fix
```

```
fix myat1 all ave/time 10 10 100 v_xave v_xdes file
position.${b}.dat

#Data Collection Run Command
run 1000000

#Reset
unfix myat1
next b
jump SELF loop2
```

APPENDIX D

RADIAL DISTRIBUTION FUNCTION CALCULATION

```python
import MDAnalysis as mda
from MDAnalysis.analysis import rdf
import matplotlib.pyplot as plt
import util.ion_rejection_functions as ion_rej_fn
import sqlite3
import time
from util.util import timeReq
import sys
import numpy as np

#Handle command line input
if len(sys.argv) < 2:

    #Input required for system if not used in command line or if command line options are not going to be
used.
    sysFile = "input/rdf_adf_test.data"
    trajectFile = "input/vis_test.data"
    interfaceDatabase = "databases/md_test_interfaces_db.sqlite"
    timestepDatabase = "databases/md_test_db.sqlite"
    ions = ["Na", "Cl"]
    doAllFrames = False
    output_folder = "output"
    output_prefix = "new_test_2"
    dummyFile = "input/new_test_sw_input.data"

elif len(sys.argv) < 9:
    print("ERROR: Missing Command Line Arguments. Format and Example below:")
    print("FORMAT: python rdf_calculations.py <system_file> <dump_file> <ice/water_interface_file>
<timestep_database> <ions> <do_all_frames> <output_folder> <output_prefix>")
    print("EXAMPLE: python rdf_calculations.py input/rdf_adf_test.data input/vis_test_mod.data
databases/md_test_interfaces_db.sqlite databases/md_test_db.sqlite Na,Cl True output test_2_7")
else:
    sysFile = sys.argv[1]
    trajectFile = sys.argv[2]
    interfaceDatabase = sys.argv[3]
    timestepDatabase = sys.argv[4]
    ions = sys.argv[5].split(",")
    doAllFrames = sys.argv[6] == "True"
    output_folder = sys.argv[7]
    output_prefix = sys.argv[8]
    dummyFile = "dummyFile.data"

#Load and Modify System
print("Shrinkwrapping System")
time0 = time.time()
system = mda.Universe(sysFile, [(trajectFile, 'LAMMPSDUMP')], atomstyle="id resid type charge x y z")

system_bounds = []

for i in system.trajectory:

    #Shrinkwrap the system for RDF Calculations
    max_x = 0
    min_x = 1000
    max_y = 0
    min_y = 1000
    max_z = 0
    min_z = 1000
    all_atoms = system.select_atoms("type 1 or type 2 or type 3 or type 4")

    for atom in all_atoms:
        x_pos, y_pos, z_pos = atom.position
        max_x = x_pos if x_pos > max_x else max_x
        min_x = x_pos if x_pos < min_x else min_x
        max_y = y_pos if y_pos > max_y else max_y
        min_y = y_pos if y_pos < min_y else min_y
        max_z = z_pos if z_pos > max_z else max_z
        min_z = z_pos if z_pos < min_z else min_z
    dataset = (max_x, min_x, max_y, min_y, max_z, min_z)
    #print(dataset)
    system_bounds.append(dataset)

#Regenerate files
dump_file = open(trajectFile)
new_dump_file = open(dummyFile, "w")

count = -1
line = dump_file.readline()
while line != "":
    if "TIMESTEP" in line:
        count += 0
        new_dump_file.write(line)
    elif "BOX BOUNDS" in line:
        new_dump_file.write(line)
        new_dump_file.write("" + str(system_bounds[count][1]) + " " + str(system_bounds[count][0]) + "\n")
        new_dump_file.write("" + str(system_bounds[count][3]) + " " + str(system_bounds[count][2]) + "\n")
        new_dump_file.write("" + str(system_bounds[count][5]) + " " + str(system_bounds[count][4]) + "\n")
        dump_file.readline()
        dump_file.readline()
        dump_file.readline()
    else:
        new_dump_file.write(line)
    line = dump_file.readline()

#Close files
dump_file.close()
new_dump_file.close()
time01 = time.time()
print("System Shrinkwrap completed in ",end="")
timeReq(time0, time01)

#Generate System
print("Loading Shrinkwrapped System")
time1 = time.time()
system = mda.Universe(sysFile, [(dummyFile, 'LAMMPSDUMP')], atomstyle="id resid type charge x y z")
time2 = time.time()
print("System loaded in:")
timeReq(time1, time2)
#

#Get the length of the trajectory
numTimesteps = len(system.trajectory)

#Pull the timestep interface data
try:
    query = "file:" + timestepDatabase + "?mode=rw"
    conn_timesteps = sqlite3.connect(query, uri=True)
except:
    print("ERROR: That database doesn't exist.\nCheck your pathname and retry.")
    exit()
else:
    print("That is an existing database. Connection Established.")

#Generate the cursor
cur_timesteps = conn_timesteps.cursor()

#Get the timesteps
timesteps = [values[0] for values in cur_timesteps.execute("SELECT timestep from timesteps")]

#Pull Interface Database
try:
    query = "file:" + interfaceDatabase + "?mode=rw"
    conn_interface = sqlite3.connect(query, uri=True)
except:
    print("That database doesn't exist.\nCheck your pathname and retry.")
    exit()
else:
    print("That is an existing database. Connection Established.")


#Get the interface location information using the timesteps
interfaceLocations = []
for i in range(len(timesteps)):
    interfaceLocations.append(float(ion_rej_fn.pullInterfaceLocation(timestep=timesteps[i],
database=conn_interface, table="data")[1]))

#Close sqlite connections
conn_interface.close()
conn_timesteps.close()

#array for the results to be added together
rdf_results_ice = [[] for i in ions]
rdf_results_water = [[] for i in ions]

#Loop through the timesteps (using the trajectory) and compute the rdf for each timestep for a set number of
bins
count = 0
print("Beginning RDF Calculations")
time5 = time.time()
while count < 5:
    time3 = time.time()
    #Set the frame of the trajectory:
    system.trajectory[count]

    #Get the atom groups
    interface = str(interfaceLocations[count])
    selPhrase = "prop x < " + interface + " and type 1"
    oxygens = system.select_atoms("type 1")
    ionsInWater = []
    ionsInIce = []
    bins = []
    typeCount = 3
    #Loop through the ions, analyzing each set's RDF and storing in a list.
    for i in range(len(ions)):
        selPhrase = "prop x > " + interface + " and type " + str(typeCount)
        ionsInWater.append(system.select_atoms(selPhrase))
        selPhrase = "prop x < " + interface + " and type " + str(typeCount)
        ionsInIce.append(system.select_atoms(selPhrase))
        iceRDFCalculated = False
        waterRDFCalculated = False
        if len(ionsInIce[i]) > 0:
            rdfIon = rdf.InterRDF(ionsInIce[i], oxygens, nbins=500, norm="rdf", range=(0,15))
            rdfIon.run()
            rdf_results_ice[i].append(rdfIon.results.rdf)
            bins = rdfIon.results.bins
            iceRDFCalculated = True
        if len(ionsInWater[i]) > 0:
            rdfIon = rdf.InterRDF(ionsInWater[i], oxygens, nbins=500, norm="rdf", range=(0,15))
            rdfIon.run()
            rdf_results_water[i].append(rdfIon.results.rdf)
            bins = rdfIon.results.bins
            waterRDFCalculated = True
        if doAllFrames:
            fig, ax1 = plt.subplots()
            fig.set_figwidth(8)
            fig.set_figheight(8)
            if waterRDFCalculated:
                label = ions[i] + " in Solution"
                ax1.plot(bins, rdf_results_water[i][len(rdf_results_water[i])-1], "r-", label=label)
            if iceRDFCalculated:
                label = ions[i] + " in Ice"
                ax1.plot(bins, rdf_results_ice[i][len(rdf_results_water[i])-1], "b--", label=label)
            label = "r (" + r'$\AA$' + ")"
            ax1.set_xlabel(label)
            ax1.set_ylabel("g(r)")
            title = ions[i] + " Ion RDF"
            ax1.set_xlim(0,12.0)
            ax1.legend(loc="upper right")
            figName = output_folder + "/" + output_prefix + "_" + ions[i] + "_" + str(count) + ".png"
            fig.savefig(figName)
            plt.close(fig)

    #Increment
    count += 1

    #Calculate Times and print
    time4 = time.time()
    print("Frame " + str(count) + " completed in:")
    timeReq(time3, time4)

#End of While loop
time6 = time.time()
print("RDF Calculations completed in:")
timeReq(time5, time6)

for i in range(len(ions)):
    averaged_rdf_ice = [0 for i in range(len(bins))]
    averaged_rdf_water = [0 for i in range(len(bins))]
    count = 1
    for j in range(len(rdf_results_ice[i])):
        for k in range(len(bins)):
            averaged_rdf_ice[k] += rdf_results_ice[i][j][k]
            averaged_rdf_water[k] += rdf_results_water[i][j][k]
        count += 1
    for l in range(len(averaged_rdf_ice)):
        averaged_rdf_ice[l] = averaged_rdf_ice[l] / count
        averaged_rdf_water[l] = averaged_rdf_water[l] / count

    fig, ax1 = plt.subplots()
    fig.set_figwidth(8)
    fig.set_figheight(8)
    label1 = ions[i] + " in Solution"
    label2 = ions[i] + " in Ice"
    ax1.plot(bins, averaged_rdf_water, "r-", label=label1)
    ax1.plot(bins, averaged_rdf_ice, "b--", label=label2)
    label = "r (" + r'$\AA$' + ")"
    ax1.set_xlabel(label)
    ax1.set_ylabel("g(r)")
    title = ions[i] + " Ion RDF"
    ax1.set_title(title)
    ax1.set_xlim(0, 12.0)
    ax1.legend(loc="upper right")
    plt.close(fig)

    #Save Fig
    figName = output_folder + "/" + output_prefix + "_" + ions[i] + "_total.png"
    fig.savefig(figName)

#Save the RDF data for use in coordination number calculations.
ice_np = np.array(rdf_results_ice)
water_np = np.array(rdf_results_water)
bins_np = np.array(bins)
np.save("input/ice_rdf_data.npy", ice_np)
np.save("input/water_rdf_data.npy", water_np)
np.save("input/bins.npy", bins_np)
```