CLASSICAL MOLECULAR DYNAMICS SIMULATIONS OF POLYMERIC IONIC LIQUIDS USING MACHINE LEARNING NEURAL NETWORK POTENTIAL

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Shengjie Tang

August 2023

© 2023 Shengjie Tang
ALL RIGHTS RESERVED

ABSTRACT

Imidazolium-based ionic liquids (ILs) are widely studied, for their enormous potential in electrolytes and battery innovations. Polymeric ionic liquids (PILs), which combine the strengths of ILs and the thermal-mechanical stability of polymer matrix, are yet to be researched thoroughly. Molecular Dynamics (MD), being one of the most popular computational techniques in materials science, has been broadly utilized to analyze the properties of materials on an atomistic scale. Different force fields (FF), or potential functions, represent the interatomic interactions of the molecules. Recently, by taking advantage of machine learning (ML) methods, neural network (NN) based potential functions have been developed. In this study, we use MD simulations to investigate the bulk and ion-transport properties of two types of PILs: Poly(1-butyl-3-methylimidazolium) bis(trifluoromethylsulfonyl)imide and Poly(1-butyl-3-methylimidazolium) Chloride (PolyBMIM_TFSI and PolyBMIM_Cl). Meanwhile, we benchmark ANI-2x NN potential against two empirical FF: All-atomic Optimized Potential for Liquids Systems (OPLS-AA) and Polarizable FF with Drude Oscillator model to provide some insights on how well ML potential works on charged macromolecules systems.

BIOGRAPHICAL SKETCH

Tang was born in Chongqing, southwestern China in 1998. He obtained his dual Bachelor's degrees from the Dalian University of Technology and the University of Leicester in June 2021. He moved to Cornell University in Ithaca for a Master's study in August 2021. He is a member of Professor Jingjie Yeo's J^2 lab, where he is studying the capability of machine learning interatomic potential for molecular dynamics simulations of macromolecules.

To my late grandpa.

ACKNOWLEDGEMENTS

I would like to thank my parents for their unconditional support for me to pursue my study in the States. They are industrious and hardworking. They would have cheap meals and wear inexpensive clothes to save some more money for my educational and living expenses. Especially, I would appreciate the help from my advisor Professor Jingjie Yeo and my committee member Professor Meredith Silberstein. They are very smart and warmhearted, offering me much knowledge and experience in polymer materials and simulation techniques. Without their guidance and suggestions, I could not conduct my research in a relatively smooth manner. My lab mates, Hanfeng, Haolin, Kaiyang, Haoyuan, Tianjiao, Ruye, and Liming, also provided advice for my research work. So I would like to thank them as well. I feel very lucky and grateful to be surrounded by these fantastic people who made me realize that I am never alone on the path to scientific research success.

TABLE OF CONTENTS

Biographical Sketch . . . iii
Dedication . . . iv
Acknowledgements . . . v
Table of Contents . . . vi
List of Tables . . . viii
List of Figures . . . ix

1 Introduction **1**
1.1 Polymeric ionic liquids . . . 1
1.2 Molecular dynamics simulation . . . 3
1.3 Machine learning-assisted force fields . . . 5
1.4 Problem statement and research question . . . 8

2 Methodology **10**
2.1 Simulation Targets . . . 10
2.2 Simulation details . . . 12
2.2.1 Force fields details . . . 12
2.2.2 Initial configurations . . . 17
2.2.3 Energy minimization and equilibration . . . 26
2.2.4 Simulation software . . . 27

3 Results and Discussions **33**
3.1 Energy of the systems . . . 33
3.2 Density . . . 35
3.3 Radius of gyration . . . 36
3.4 Radial distribution functions . . . 38
3.5 Mean-square displacement and self-diffusion coefficients . . . 41
3.6 Run-time comparisons . . . 45

4 Conclusion and Outlooks **48**
4.1 Conclusion . . . 48
4.2 Outlooks . . . 49

A OPLS-AA force field parameters [61, 29] **52**
A.1 Lenard-Jones interaction and atomic polarizability . . . 52
A.2 Partial charges . . . 53
A.3 Bond parameters . . . 54
A.4 Angle parameters . . . 55
A.5 Dihedral and improper dihedral parameters . . . 57
A.6 Input scripts . . . 59
A.6.1 OPLS-AA input script . . . 59
A.6.2 Polarizable force field input script . . . 73
A.6.3 ANI-2x input script . . . 121

Bibliography **124**

LIST OF TABLES

3.1 Time-averaged density (g/cm^3) of each simulations. Errors are in percentages, in which the plus and minus signs represent overestimate and underestimate, respectively. Both the reference data and experimental data are based on ILs, instead of PILs. 35

3.2 Run-time comparisons between different computation modes. For LAMMPS simulations of four systems, G2 cluster and Stampede2 were used, whereas for the ANI-2x simulations, the personal laptop was used. For the simulations on the G2 cluster, 160 cores were used. And for the simulations on Stampede2, 128 cores were used for smaller systems, and 256 cores were employed for most complex systems. 46

A.1 Force field interactions for L-J interactions and polarizability. . . 52

A.2 Force field parameters for Coulomb interactions. 53

A.3 Force field parameters for bond interactions. 54

A.4 Force field parameters for angle interactions. 55

A.5 Force field parameters for angle interactions. (continued) 56

A.6 Force field parameters for dihedral interactions. 57

A.7 Force field parameters for dihedral interactions. (continued) . . . 58

A.8 Force field parameters for improper dihedrals. 58

LIST OF FIGURES

1.1 Different types of PILs, reproduced from reference [38]. In the middle of the graph, there is a positively charged imidazolium ring that contains two nitrogen atoms. Especially we are interested in the polycations in this research. 2

1.2 A schematic graph of ANI NNP, reproduced with permission from reference [45]. In the left graph, $\vec{q}$ is the molecular coordinates and $\vec{G}_i^X$ is the atomic environment vector. These two variables are fed into the hidden layers represented by l_k to produce the atomic energy E_i^X. Noting that the $\vec{G}_i^X$ is for the *ith* atom in the molecule with the atomic number X. The right graph shows how the total energy of a water molecule can be obtained. 7

2.1 Atom type nomenclature for polycation BMIM+, anions Cl- and TFSI-. The middle two structures are the monomer and polymer chain with ten monomers. (note that if the CM1 is at the beginning of the polymer chain, then its atom type becomes CM2, and so as HM1 (HM2)). 11

2.2 The initial structures of two polymeric ionic liquids, PolyBMIM_Cl and PolyBMIM_TFSI. The upper row represents two initial structures for PolyBMIM_Cl, and the lower row shows two initial configurations of PolyBMIM_TFSI. (a) and (c) have only one polymer chain, with 10 corresponding anions; (b) and (d) has four chains and 40 moving anions in their simulations cells. . 24

2.3 The initial structures of two polymeric ionic liquids with Drude particles harmonically bonded to the polarizable atoms. The upper row represents two initial structures for PolyBMIM_Cl, and the lower row shows two initial configurations of PolyBMIM_TFSI. The Drude particles are highlighted in tangerine color. 25

3.1 The total energy during the last nanosecond of the production runs for four simulated systems using different force fields. The energies calculated from ANI-2x potential in ASE were converted into unit kcal/mol for comparisons. 34

3.2 The Rg of the polycation chain in the PolyBMIM_Cl_1 and PolyBMIM_TFSI_1 systems. The trajectory of the last 1 ns was collected for analysis. ANI-2x underestimated Rg for both structures. . . . 37

3.3 Radial distribution functions (RDFs) between polycation PolyBMIM and Cl/TFSI anions in the PolyBMIM_Cl_1 (a) and PolyBMIM_TFSI_1 (b) systems. Comparisons are done among the three force fields. 39

3.4 Radial distribution functions (RDFs) between polycation PolyBMIM and Cl/TFSI anions in the PolyBMIM_Cl_4 (a) and PolyBMIM_TFSI_4 (b) systems. Comparisons are done between two empirical force fields. 41
3.5 Mean square displacement of Cl- and TFSI- anions in PolyBMIM_Cl_1 and PolyBMIM_TFSI_1 systems. Data is collected from the last 1 ns of the production of each trajectory. (a), (c) is the actual plot of the calculated MSD of anions, and (b), (d) are the linear fits of MSDs. 43

CHAPTER 1

INTRODUCTION

1.1 Polymeric ionic liquids

Polymeric ionic liquids (PILs) represent a broad category of battery electrolytes consisting of either the cation or anion of traditional ionic liquids (ILs) as polymeric repeating units [37]. Besides, they are of interest as next-generation ion transport materials because of their potential to combine the thermodynamic and mechanical stability of polymers with the ion transport facility of ionic liquids [35]. Most of the PILs are ionically bonded organic compounds at room temperature. However, there is a limited amount of research data on the PILs since they are expensive to study experimentally.

Structurally speaking, PILs are mainly categorized into polycation-based PILs and polyanion-based PILs, according to which charged particles are polymerized. In addition, there are other categories of PILs in terms of configurations. Fig. 1.1 demonstrates various types of PILs, which are reproduced from reference [38]. Typical counteranions of PILs are hydrophobic anions such as tetrafluoroborate (BF_4-), hexafluorophosphate (PF_6-), while the typical counteranions of PILs are organic cations such as dialkyl imidazolium, alkylpyridinium [35]. In this work, two imidazolium-based PILs containing polycations, poly(1-butyl-3-methylimidazolium) bis(trifluoromethylsulfonyl)imide (pBMIM-TFSI) and poly(1-butyl-3-methylimidazolium) chloride (pBMIM-Cl), are investigated.

In the past study of polymeric ionic liquids, various methods have been im-

plemented. Zhang *et al.* has done an extensive research study on the polyelectrolytes using molecular dynamics simulations, investigating influences and effects of transference numbers [60], morphology [58], and polarizability [61] on the ion transport within the polymer system. Sangoro *et al.* [43] carried out experiments on the polymeric ionic liquids, suggesting a "decoupling" between ionic conductivity and glass transition/segmental dynamics in PILs. The combination of experimental insights and computational validations enables the study of specific properties of certain materials to be more comprehensive.

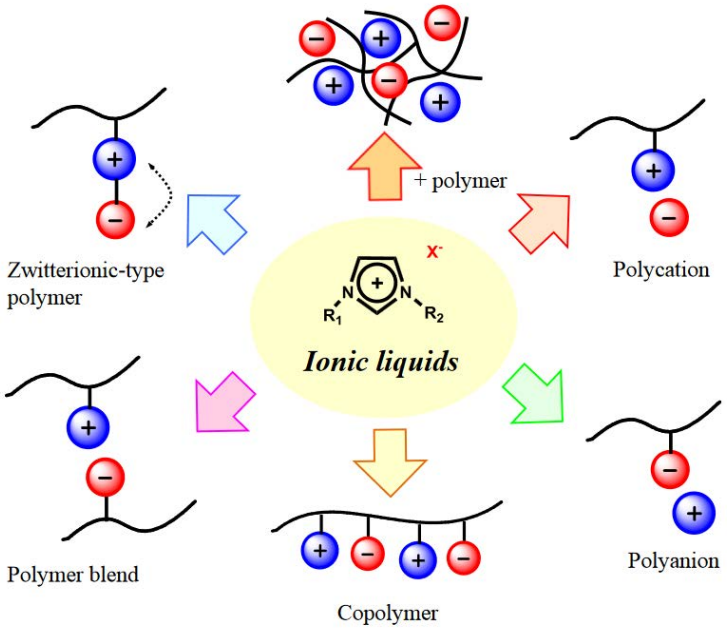

Figure 1.1: Different types of PILs, reproduced from reference [38]. In the middle of the graph, there is a positively charged imidazolium ring that contains two nitrogen atoms. Especially we are interested in the polycations in this research.

1.2 Molecular dynamics simulation

In this work, pure molecular dynamics simulations of the polymer materials were performed, so the content of this work is more or less MD-biased. Admittedly, the increase in computational power and accessibility of massively parallel architecture combined with the maturation of advanced modeling techniques and force fields allowed atomistic molecular dynamics simulations to transform into an essential tool for providing molecular-scale insight into the structure-property relationships and virtual design of novel materials. [3]

In atomistic simulations of different molecules, interactions between atoms should be described as accurately as possible to approach real situations. These interactions are represented as interatomic potential, or force field, and should be specified for simulations before files are executed. For example, in the MD simulator LAMMPS [50], the potential is determined in the *pair_style* section to set up the simulation. And there is a diverse range of force fields that are specifically parameterized for certain simulation tasks. Originally, Lennard Jones proposed a simplified mathematical model that can approximately describe the interactions between two neutral atoms or molecules, which is called L-J potential, or 12-6 potential. The L-J potential is especially accurate when describing the interactions between noble gas molecules [31].

$$V(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \tag{1.1}$$

The L-J potential treats the distance between two bodies as the only variable, and there are two constants, ϵ, and σ. ϵ stands for the depth of the energy well, and σ is the distance at which the potential energy between two bodies is

zero. Usually, the determination of the two parameters is done by fitting known experimental data. By taking the derivative of the potential energy, the force between the two particles could be obtained.

$$\mathbf{F}(r) = -\nabla V(r) = -\frac{d}{dr}V(r)\hat{\mathbf{r}} = 4\epsilon\left(12\frac{\sigma^{12}}{r^{13}} - 6\frac{\sigma^{6}}{r^{7}}\right)\hat{\mathbf{r}} \tag{1.2}$$

However, with the development of microscale studies on the structure and properties of various materials, the L-J potential is not able to capture a high level of energy accuracy for different materials under various circumstances. Based on the traditional L-J potential model, more potentials, or force fields, have been developed for atomistic simulations of specific properties of certain materials. Bond order potentials such as Tersoff potential [49] were developed to depict the atomic correlations in Carbon-Silicon systems such as SiC. van Duin [51] developed Reactive Force Field (ReaxFF) for addressing the processes of dissociation and formation of bonds, hence providing a way to study the chemical reaction process of hydrocarbons up to 1000s atoms using MD simulations. Jorgensen and coworkers [24] developed all-atomic optimized potential for organic liquid systems (OPLS-AA). The nonbonded and torsional parameters were obtained from *ab initio* calculations. The OPLS-AA force field has been validated to accurately model the bond, angle, dihedral, improper torsions, and nonbonded interactions in the polymer systems [61, 59]. Later on, Wang *et al.* proposed a general AMBER force field, which is also called GAFF, which uses various atom types to account for almost all the chemical space for drug-like molecules that are composed of H, C, N, O, S, P, F, Cl, Br, and I. Their bond angle, length, and torsional information were extracted from empirical calculations [54]. When it comes to the specific problem of ionic liquids with charged

molecules, Polarizable force fields (PFs) play an important role in accurately capturing the effects induced by electronic polarizations. In the non-PFs, each of the atoms is assigned with a fixed partial charge, but in the real situation, the partial charges on the atoms are continuously changing along with the electrostatic interactions. The PFs consider the charge fluctuations. There are mainly three types of PFs, which are Fluctuating Charge Model (FCM), the Drude Oscillator Model (DOM), and Induced Point Dipoles [3]. Goloviznina *et al.* [17] proposed a transferable PF for ILs, which is called CL&Pol, based on a fixed-charge force field CL&P that has been extensively used for MD simulations ILs [6].

1.3 Machine learning-assisted force fields

Parameterizations of the force field from quantum chemical calculations data is indeed a time-consuming task. Besides, there is always a trade-off between accuracy and computational cost in terms of time and memory storage in computers. Researchers have been thinking about using the power of machine learning for materials designs and atomistic simulations. Again, taking the force field as an example, there has been plenty of studies that use machine learning approaches to develop interatomic potentials that can be trained on maximally informative quantum chemistry data through active learning and adversarial uncertainty attacks [1]. Bartók *et al.* [2] employed Gaussian approximation for automatically generating finite range interatomic potential models from quantum-mechanically calculated atomic forces and energies. Wang *et al.* [55] used the active learning technique to create a neural network potential trained on Density Functional Theory (DFT) calculation data for simulating the solvate ionic liquids. Willman and coworkers [57] used relevant quantum molecular dynam-

ics database to develop a spectral neighbor analysis machine learning potential, enabling atomic-scale insights at the experimental time and length scales. Recently, Chen and Ong [7] proposed a universal graph neural network potential based on three-body interactions (M3GNet), which is trained on a massive structural relaxation database generated by Materials Project, that is proven to be accurate when simulating lithium-containing systems.

In this work, another type of neural network potential developed by Smith *et al.* [45], Accurate NeurAl networK engINe for Molecular Energies (ANI potential), is investigated to explore its capability in simulating macromolecules. Knowing the construction of such black-box neural network potential can be crucial because this is a kind of guidance to improve their abilities further. The ANI potential was originally based on the symmetry functions proposed by Behler and Parrinello [4] in their paper about a generalized neural network representation of a high-dimensional energy surface. The symmetry functions are used for representing the chemical environment information of atoms and are composed of radial part and angular parts. While in the formation of ANI potential, the angular symmetry function was heavily modified for probing the local angular environment of complex chemical systems [45]. The atomic environment vectors (AEVs) are obtained from the calculations of the symmetry function and then fed into the neural network architecture. What is also been input into the neural network are the atomic positions. Furthermore, to ensure a fair degree of accuracy in the simulations, the dataset for the training of the network is extremely important. The ANI potential is trained on a subset of GDB-11 [14] database with up to 8 heavy atoms to predict total energies for organic molecules containing four atom types: H, C, N, and O. Fig. 1.2 demonstrates the structure of the ANI neural network potential.

The ANI potential has experienced a period of evolution from ANI-1 obtained from normal sampling method [45], ANI-1ccx potential with improved computational speed and accuracy that reaches couple cluster methods [46], ANI-1x potential [47] which utilized active learning data selection processes and trained much larger data set compared to ANI-1ccx. The most recent potential model from the ANI potentials family is the ANI-2x [9], which is an advanced version of the ANI-1x model and has extended the accessibility of ANI potentials to drug-like small molecules that contain three more elements S, Cl, and F.

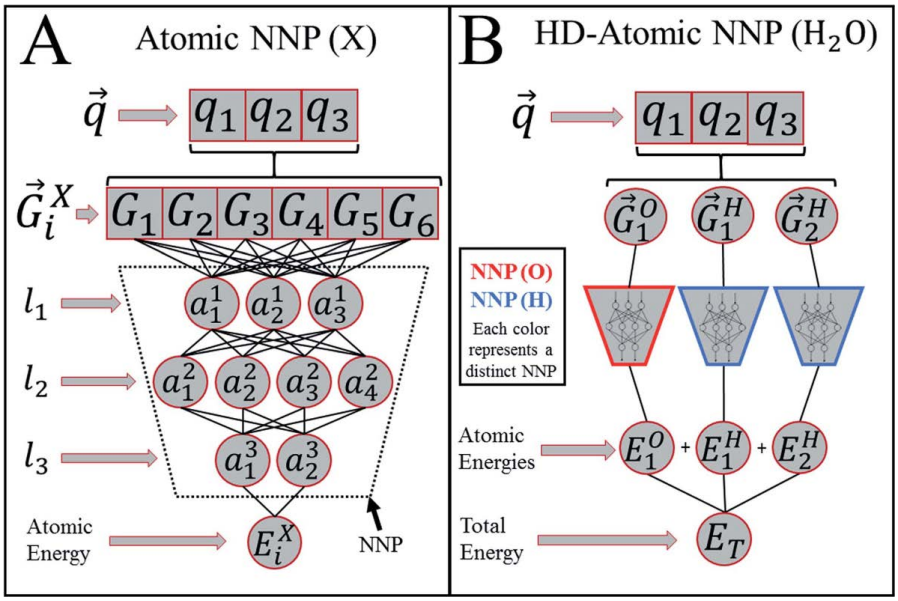

Figure 1.2: A schematic graph of ANI NNP, reproduced with permission from reference [45]. In the left graph, $\vec{q}$ is the molecular coordinates and $\vec{G}_i^X$ is the atomic environment vector. These two variables are fed into the hidden layers represented by l_k to produce the atomic energy E_i^X. Noting that the $\vec{G}_i^X$ is for the *ith* atom in the molecule with the atomic number X. The right graph shows how the total energy of a water molecule can be obtained.

The presence of ions in PILs results in strong local electric fields polarizing

molecules and large ions [3], so PFs are implemented in the MD simulations. While PFs can provide accurate ion transport properties in ILs according to several studies [33], they are proven to be computationally expensive even for non-polymeric ILs. In this work, for results comparison purposes, the Drude oscillator model of the PFs, ANI neural network potential model, and OPLS-AA force fields are used for simulations of the polyelectrolyte material. Aspects including the density of the simulation box (bulk material), the total energy of the structure, radial distribution functions (RDFs), and radii of gyration (RG) will be investigated in detail and will be discussed in the later sections.

1.4 Problem statement and research question

Due to the limited knowledge about the properties of PILs at an atomic scale, there have been quantities of MD simulations [61, 60, 59, 58] and even combined MD-DFT investigation on the ionic conductivity properties of the materials [44]. However, MD simulations on PILs require parameters that can precisely describe atomic and bonding interactions when evolving the atoms along with Newton's equations. Furthermore, all the constants in the bonded and nonbonded interaction potentials between all the atom types in the system, the initial configuration one input into the simulation, and the selected simulation protocol all have a significant impact on the accuracy of this atomistic models [15]. In the previously mentioned research work, none of them have deployed machine learning force field models for MD simulations on the PILs materials, hence the accuracy of such novel potential models in simulating polymer systems is worth investigating. The simulation results obtained from ANI potential model will be discussed and compared with conventionally used force

fields to provide some guidance on how to improve the machine learning force field for practical uses in the modeling of polymeric systems.

CHAPTER 2

METHODOLOGY

2.1 Simulation Targets

Imidazolium is the most popular cation, which has been included in polymer backbones obtained from vinyl, styrenic, (meth)acrylic and (meth)acrylamide, ethylene glycol, vinyl ether, and norbornene monomers [35]. They possess various applications in different industries. In this simulation work, two types of PILs are chosen for research. The first type of the PILs is poly(1-butyl-3-methylimidazolium) bis(trifluoromethylsulfonyl)imide (PolyBMIM-TFSI), and the second one is poly(1-butyl-3-methylimidazolium) chloride (PolyBMIM-Cl). Fig 2.1 demonstrates the structure of the polymeric ionic liquids. The PILs consist of monomers of 1-butyl-3-methylimidazolium and the main carbon backbones are connected. There are two nitrogen atoms on the imidazolium ring, with one of them positively charged, connecting the butyl alkyl radical. The formal charge of the positively charged nitrogen atom is +1, but the partial charges of the monomer and, especially for the two nitrogen atoms, are not the same as the formal charges. This should be noted since the partial charges of the system are essential when conducting simulations and hence should be carefully and accurately specified before the simulations, or the results might be non-physical. The mobile anions are the chloride anion and the bis(trifluoromethylsulfonyl)imide. Recall that the ANI-2x neural network potential was trained on a database containing C, H, O, N, S, F, and Cl drug-like small molecules; choosing the corresponding PILs possessing these atom types is necessary.

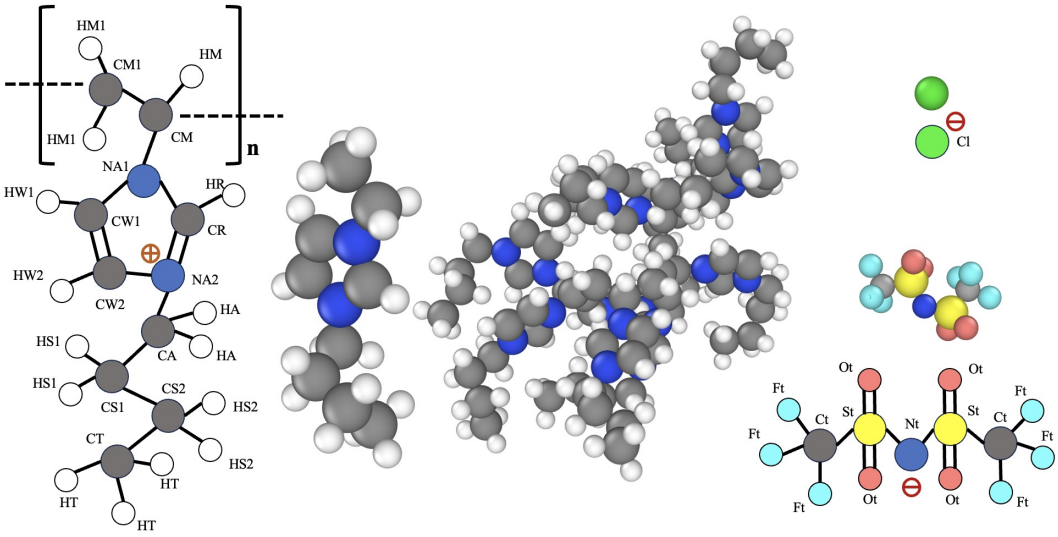

Figure 2.1: Atom type nomenclature for polycation BMIM+, anions Cl- and TFSI-. The middle two structures are the monomer and polymer chain with ten monomers. (note that if the CM1 is at the beginning of the polymer chain, then its atom type becomes CM2, and so as HM1 (HM2)).

In this study, there are ten monomers on each polymer chain. In addition, two configurations are simulated: 1. one chain with ten monomers and 2. four chains with 40 anions. Detailed initial configurations are described in the following section.

2.2 Simulation details

2.2.1 Force fields details

OPLS-AA force field

Because of the tunable properties of OPLS-AA FF parameters, it is frequently used to model ILs and PILs. Nonetheless, the parameters of the OPLS-AA force field can be tricky to obtain. Sometimes they are acquired by empirical data from carrying out a tremendous amount of experiments. Luckily, this force field has been under development for decades, and there are parameter generators like LigParGen [11, 10, 25] that can assign specific OPLS-type interaction to each atom in a simulation system. These two packages take SMILES string or pdb (protein data bank) as input and output corresponding parameter files for simulations. However, the tool has limitations on the size of the system in terms of number of atoms. One effective way is to create a trimer system as an input and get the information for the head, middle, and tail monomers and replicate the parameters of the middle monomer to create a complete polymer chain. During the assignment of the OPLS parameters, it is essential to know which atom type the atoms belong to. The generated OPLS-AA parameters files cannot be directly used in the LAMMPS because of the different life formats. Open Babel [39] is a format converter that can transform GROMACS format files, such as itp files, into LAMMPS data files. In the non-polarizable OPLS-AA force field formalism [24], the total energies for each ionic liquid system are evaluated as a sum of individual energies for the harmonic bond stretching and angle bending terms, a cosine series for torsional energetics, and Coulomb and

12-6 Lennard-Jones terms for the non-bonded interactions [12]. The interactions can be expressed in the equations as follows.

$$
\begin{aligned}
E_{bonds} &= \sum_i k_{\mathrm{b},i}(r_i - r_{\mathrm{o},i})^2 \\
E_{angles} &= \sum_i k_{\theta,i}\,(\theta_i - \theta_{0,i})^2 \\
E_{torsion} &= \sum_i \left[\frac{1}{2}V_{1,i}(1+\cos\phi) + \frac{1}{2}V_{2,i}(1-\cos 2\phi)\right. \\
&\qquad \left. + \frac{1}{2}V_{3,i}(1+\cos 3\phi) + \frac{1}{2}V_{4,i}(1-\cos 4\phi)\right] \\
E_{nonbond} &= \sum_i \sum_{j>i} \left\{ \frac{q_i q_j e^2}{r_{ij}} + 4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right]\right\}
\end{aligned}
\tag{2.1}
$$

In the equations, the bonds, angles, and torsional and non-bonded interactions are defined. The parameters are the force constants k, the r_0 and θ_o equilibrium bond and angle values, Fourier coefficients V, partial atomic charges q, and Lennard-Jones radii and well-depths, σ and ϵ [12]. Correspondingly, the bond interaction is ***harmonic***, and so does the angle interaction. The torsional interaction, which is equivalent to the dihedral interaction, is set specifically as ***opls*** in the simulation. In terms of nonbonded interaction (pairwise interaction), the style is specified as ***lj/cut/coul/long***. For the anions in this study, there are no improper styles, but for the polycation chains, there is a setting for the improper style, which is ***cvff***.

PF with Drude oscillator model

The Drude oscillator model introduces an auxiliary particle linked to each polarizable atom through a harmonic spring to represent induced electronic polar-

ization. Drude dipoles are used in the CHARMM [5] and NAMD [41] molecular dynamics algorithms to express explicit polarization, and Drude polarization is a feature of the CHARMM force field for biomolecules. The PF simulation with the Drude oscillator model can be enabled in LAMMPS by activating the USER-DRUDE package. It can also be used in OpenMM [13] using their Plugin. One important thing that should be paid special attention to is that how much mass to assign to the Drude Particle (DP) and how much charge to assign to the Drude Particle from the Drude Core (DC). Due to new degrees of freedom, real/reduced coordinate transforms, and the extra computations for the Thole screening of the DP interactions, the polarizable simulations are more time-consuming than their non-polarizable equivalents [8]. The aim of the Drude model is to place a virtual particle to mimic the dipole effect but the initial parameters of the Drude Oscillator model are actually from non-PF, in this case, the OPLS-AA. The Nosé-Hoover thermostat and the Langevin thermostat can be utilized to control the temperature in the simulation systems. The relationship between the polarizability of the polarizable atoms and their charges is described by the following equation.

$$k = \frac{q^2}{\alpha} \tag{2.2}$$

where the k is the stiffness constant for the spring that connects the DC and DP, q is the charge of the DP, and the α is the polarizability of the polarizable atom. Lamoureux and Roux [30] suggest adopting a global half stiffness, $k = 4184\,\mathrm{kJ/mol/\AA^2}$, for all types of DC-DP bond. The USER-DRUDE package provides users with a Python script that can automatically assign the DP to the polarizable atoms by identifying the comments after the atom types in the Mass

section in the LAMMPS data file. In this case, the polarizable atoms are all the atoms except for the hydrogen atoms. The obtained data file has DP interactions embedded, but there is no pairwise interaction defined. To include the pair interactions, the ***pair_coeff*** is set separately in the input file to account for pair interactions of atoms. In the input ***pair_style*** is set to a mixed one: ***hybrid/overlay lj/cut/coul/long 12.0 thole 2.600 12.0***.

ANI Neural Network force field

ANI is one of the newest neural network potential models that has not been added to LAMMPS as a ready-to-use force field because the parameters cannot be simply defined as Lennard-Jones style. In this work, only the latest ANI model, namely the ANI-2x, is invoked for simulations. As described in the previous section, the ANI model was trained on the GDB-11 database, including small organic molecules. But for the trained neural network applicable for simulating S, Cl, and F containing molecules, more databases were used to form the training set of ANI-2x. A list of chemically possible organic compounds comprising the heavy elements C, N, O, and F can be found in the GDB-11 data set. For all compounds possessing up to eight non-hydrogen atoms, the authors performed a combinatorial substitution of the chemical symbols O with S and F with Cl using this database. Moreover, molecules with S, F, and Cl elements were selected from the ChEMBL [16] database. To produce training data for the sampling of non-bonded interactions, the s66x847 [27] benchmark was utilized.

To enable the ANI family force field, several packages like torchani and torch are imported at the beginning of the Python script, making the ANI force field model use autograd function in Pytorch [40] for faster calculations of forces and

energy in the systems. Additionally, it is possible to specify whether it is the CPU or GPU we would like the ANI to run on. Using the ASE to invoke the ANI force field follows similar steps as conventional MD simulations. First initial configuration is read, then velocities are assigned, and the following energy minimization and equilibrations before the actual production run.

2.2.2 Initial configurations

Every MD simulation needs topology files as the input. For LAMMPS, the topology and parts of the force field definitions are stored in the LAMMPS data files. To obtain the data files, one can start by generating pdb files of a single polymer chain and the anions, which can be done by using Enhanced Monte Carlo (EMC) [23].

Enhanced Monte Carlo software is a computational tool that uses the Monte Carlo algorithm to generate molecular structures for multiple types of simulations, especially MD simulations. This software's main job is to simulate how molecules behave in different chemical environments, like solutions or biological systems. The Monte Carlo algorithm is a stochastic method that randomly generates conformations of a molecule, evaluating the energy of each conformation using a scoring function. The scoring function typically includes terms for bond stretching, angle bending, torsional rotation, and non-bonded interactions. EMC is extremely useful for generating polymer structures. There are several steps for generating the polycations and anions structures.

The esh file, which is aimed at describing the chemical and environmental properties of the polymer chains, is the starting point. In this file, several ITEM paragraphs are defined. Each paragraph starts with ITEM, plus the name of that paragraph, and ends with ITEM END.

OPTIONS paragraph determines the overall chemical and environmental conditions for a molecule. For example, as a chemical option, density defines the chemical density of the system as an initial guess. On the other hand, the ntotal determines the number of atoms in the system as an approximation. The exact

number of atoms will be adjusted according to the Simplified molecular-input line-entry system (SMILES) [56] string defined in the later paragraph section.

```
ITEM OPTIONS

replace     true
mass        true
field       pcff
number      false
ntotal      265
density     1.0
field_debug reduced
charge      true

ITEM END
```

In the OPTIONS paragraph, replace means to replace or overwrite any existing files within the directory, while field_debug reduced makes sure that there will be an output log file after executing build.emc, describing details of the run jobs. Any errors or possible warnings will be displayed in that file. The charge determines that the system is considered with charges.

In GROUPS paragraph, bmim_head, bmim_middle, and bmim_tail define the names of the three groups, with the SMILES string following them. For the polymer chain, there are three types of monomers, comprising two ends and the middle. Note that the asterisk sign * indicates the connectivity between the monomers. The first pair 1, bmim_middle:1 means, that the first occurring * of C(n1c[n+](CCCC)cc1)C * connects to the first occurring * of group

bmim_middle. The whole polymer chain ends with a methyl group, which is defined in bmim_tail group where hydrogens are omitted.

```
ITEM GROUPS

bmim_head  C(n1c[n+](CCCC)cc1)C*,1, bmim_middle:1
bmim_middle *C(n1c[n+](CCCC)cc1)C*,1, bmim_middle:2
bmim_tail  *C,1, bmim_middle:2

ITEM END
```

The parentheses are used to represent the aromatic ring structure of the imidazolium group, positive sign determines the formal positive charges on the Nitrogen atoms. All the atoms with charges should be enclosed by square brackets [56].

Finally, the CLUSTERS and POLYMERS sections define the overall configuration.

```
ITEM CLUSTERS
poly  random,1
ITEM END

ITEM POLYMERS
poly
1    bmim_head,1, bmim_middle,9, bmim_tail,1
ITEM END
```

In CLUSTERS section, the name of the system is poly and random means that the monomers are randomly distributed on the chain. In POLYMERS paragraph, the numbers mean that there are 1 head monomer, 9 middle monomers, and 1 tail terminator on each polymer chain.

The above procedure is for generating a structure for the polycations in my target molecule. The pdb structure files of Cl and TFSI anions can be prepared using similar steps. After obtaining the pdb structures data from EMC, post-processing should be done to construct the actual data files for LAMMPS simulations since the force field used in EMC is Polymer Consistent Force Field (PCFF), which is completely different from OPLS-AA. Information on partial charges, atom types, bonds, angles, dihedrals, and improper dihedrals should be rigorously edited. For a simpler manipulation of the structures, the polymer chain and two distinct anions are defined separately in protein databank (PDB) format. Then, Packmol [32] and Topo Tools [52] in Visual Molecular Dynamics (VMD) [22] are used for generating proper initial data files with OPLS-AA parameters.

Assuming PDB files for PolyBMIM polycations, TFSI, and Chloride anions are obtained from EMC, inp files can be written for merging the cations and anions. In the inp file, simulation box size, number of polymer chains and corresponding anions can be manually defined. Inp files from Packmol read PDB files and also generate PDB files. For instance, if a small single cell is to be constructed, an inp file can be scripted in the following format.

```
tolerance 2.0
filetype pdb
output PolyBMIM_Cl_1.pdb
```

```
structure PolyBMIM.pdb
  number 1
  inside box 0. 0. 0. 30. 30. 30.
end structure

structure Cl.pdb
  number 10
  inside box 0. 0. 0. 30. 30. 30.
end structure
```

Packmol is very useful when many combinations of molecules are desired for calculations since the number of the specific molecules can be defined and they will be randomly distributed within the box. PDB file does not specify the bond connections or bond angles with details. only the positions of atoms and residue types are defined. Hence, re-assigning atom types and partial charges are needed. This is done via Topo Tools plugin in VMD. Topo Tools scripts can be written in a tcl file for being called by the VMD TK console. The following piece of tcl script describes how atoms are selected, and charges are assigned to specific atom types.

```
# structure topology definition
mol new PolyBMIM_Cl_1.pdb autobonds no waitfor all
set sel [atomselect top all]
$sel set radius 1.537
mol bondsrecalc top
```

```
topo retypebonds
topo guessangles
topo guessdihedrals
topo guessimpropers
mol reanalyze top

set selCA [atomselect top {name CA}]
set selCM [atomselect top {name CM}]
set selCM1 [atomselect top {name CM1}]
.... # select other atom types

$selCA set charge -0.071916
$selCM set charge -0.191281
$selCM1 set charge -0.286559
.... # assign charges to other atom types

topo writelammpsdata PolyBMIM_Cl_1.data full
.... # write lammps data file
```

Although the bond connections can be defined through the above script, the detailed interaction force field parameters are not set. Looking through the data file generated by tcl script, the interaction types are automatically commented at the end of each line.

```
# Pair Coeffs
# 1 CA
...
```

```
# Bond Coeffs
# 1 CA-CS1
...

# Angle Coeffs
# 1 CA-CS1-CS2
...

# Dihedral Coeffs
# 1 CA-CS1-CS2-CT
...

# Improper Coeffs
# 1 CR-HR-NA1-NA2
...
```

Accurately defining the interaction parameters ensures accurate and physical simulation processes. All the OPLS-AA parameters are adapted from work from Zhang et al. [61] and Kubisiak et al. [29]. Specific parameters will be provided in the supporting information in the appendix section. However, the Atoms section is fully and accurately defined. In addition, the bonds, angles, dihedrals, and impropers are also clearly specified. In these sections, the connections of each atom are defined, and serial numbers and types of those connections are assigned. The initial structures that are written in the data files can be visualized and are shown in Figure 2.2. The polycations and actions are randomly distributed in the simulation cell, and the center of mass (COM) is placed at the center of the cells.

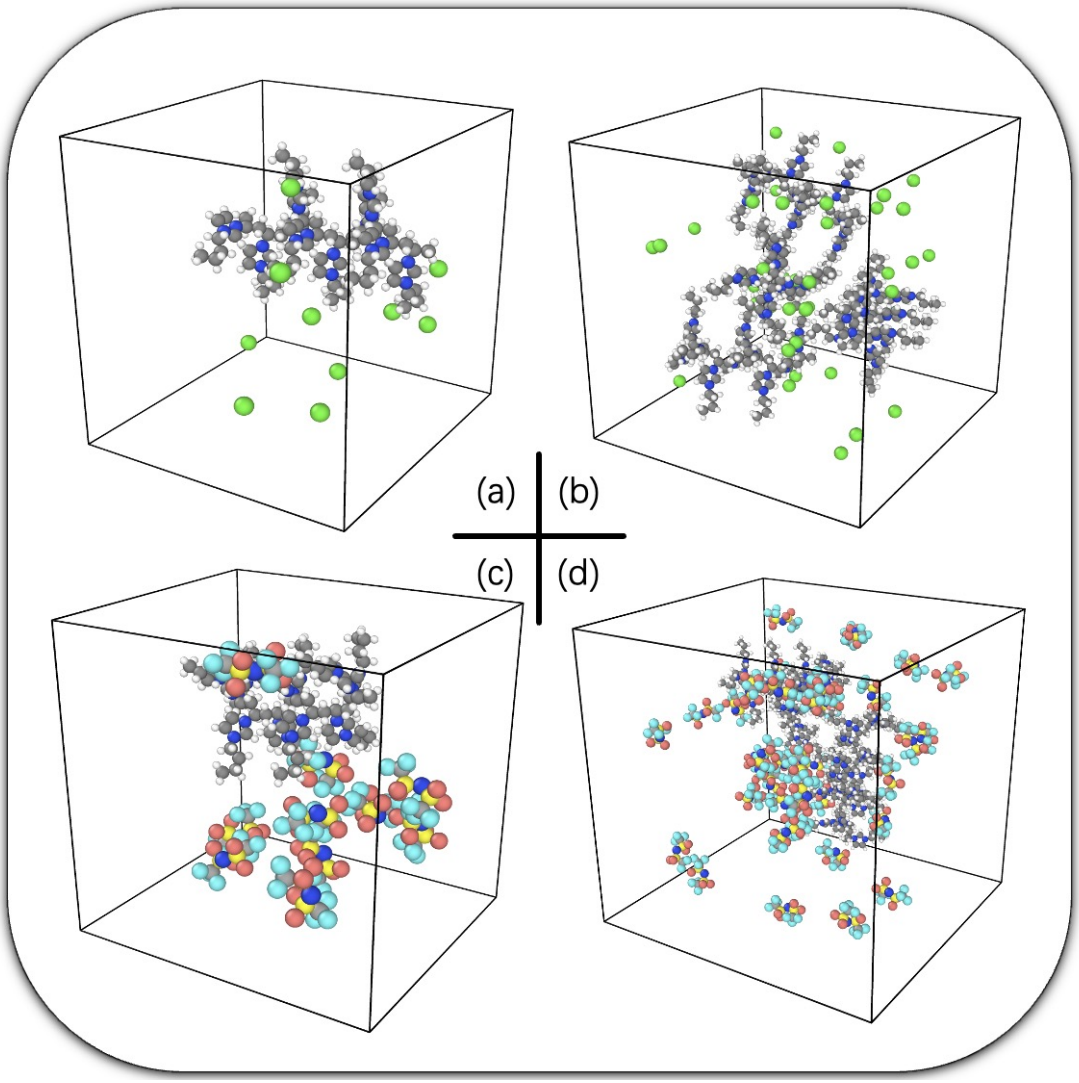

Figure 2.2: The initial structures of two polymeric ionic liquids, PolyBMIM_Cl and PolyBMIM_TFSI. The upper row represents two initial structures for PolyBMIM_Cl, and the lower row shows two initial configurations of PolyBMIM_TFSI. (a) and (c) have only one polymer chain, with 10 corresponding anions; (b) and (d) has four chains and 40 moving anions in their simulations cells.

The parameters of PF with the Drude Oscillator are based on the OPLS-AA parameters, with some modifications on the data files and input scripts. Taking the fully defined data files from OPLS-AA simulations, it is convenient to use

a 'polarizer' Python script for adding Drude particles to the original systems. Attaching Drude particles to the polarizable atoms does not change the total charge and total mass. However, this leads to an increase in the atom types.

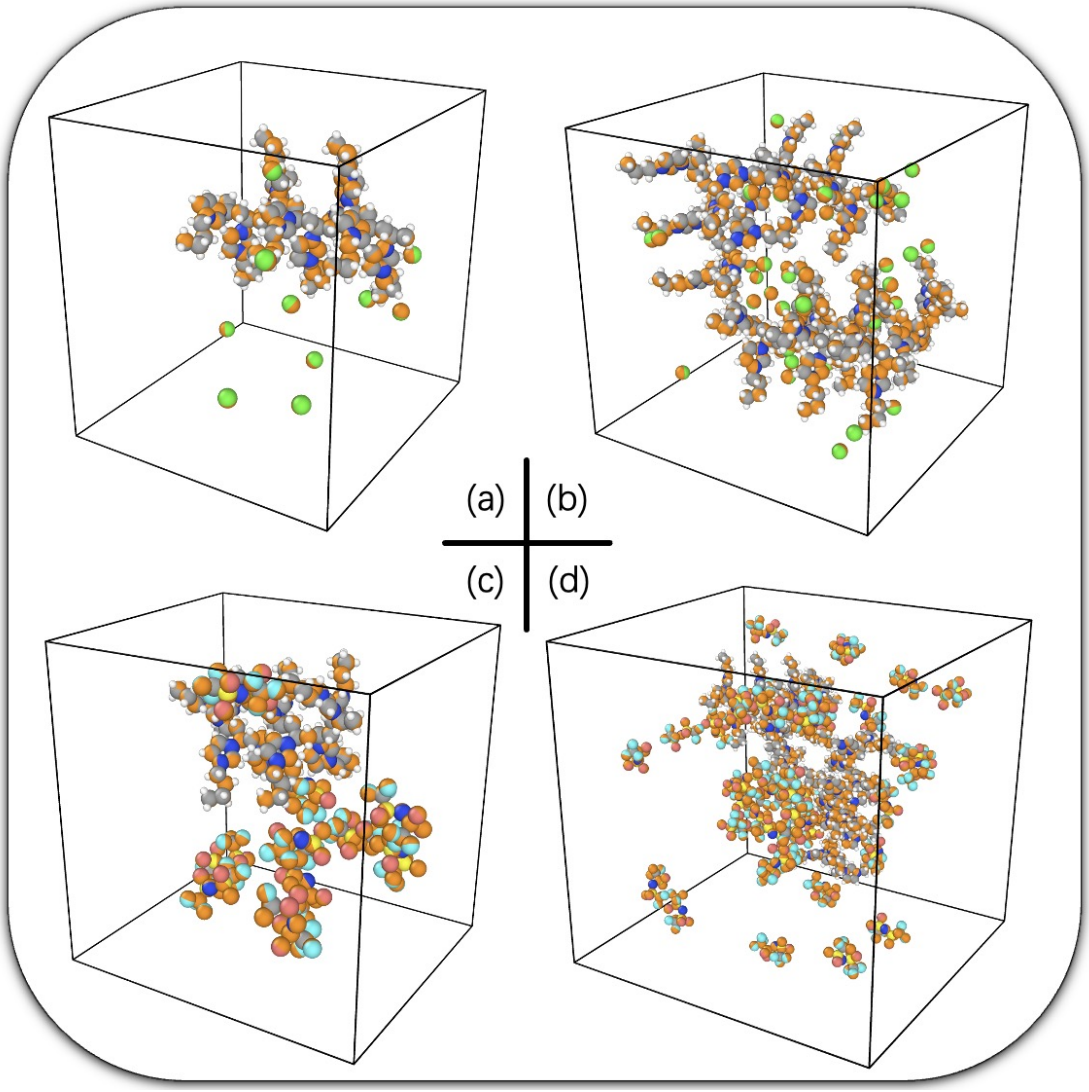

Figure 2.3: The initial structures of two polymeric ionic liquids with Drude particles harmonically bonded to the polarizable atoms. The upper row represents two initial structures for PolyBMIM_Cl, and the lower row shows two initial configurations of PolyBMIM_TFSI. The Drude particles are highlighted in tangerine color.

2.2.3 Energy minimization and equilibration

The step of energy minimization entails optimizing the initial atomic positions in order to achieve a stable configuration with minimal potential energy. LAMMPS provides a variety of minimization algorithms, such as steepest descent (sd) and conjugate gradient (cg), which can be selected based on the simulation's particular requirements. The process of energy minimization helps to eliminate unrealistic atomic overlaps or geometries, thereby preventing the simulation from producing excessively high forces or non-physical behavior. Using the 'minimize' command in LAMMPS input programs, minimization can be performed with parameters such as energy tolerance, force tolerance, and a maximum number of iterations.

The system is equilibrated to the intended thermodynamic state (temperature and pressure) after energy minimization. This method requires running short periods of MD simulations with temperature and pressure controlled by thermostats and barostats. To equilibrate the system, LAMMPS supports a variety of thermostat algorithms, such as Berendsen and Langevin and barostat algorithms, such as Parrinello-Rahman and Nose-Hoover. During the equilibration steps, ensembles are specified. In all the simulations in this work, the canonical ensemble (NVT, constant number of particles, volume, and temperature) and the isothermal-isobaric ensemble (NPT, constant number of particles, pressure, and temperature) are both utilized for stabler systems. Inspired by the procedure from Zhang and co-workers [61], a single loop of multistep pre-equilibration consists of (1) 0.1 NVT simulation at 1000 K, (2) 0.1 NPT simulation at 600 K and 100 bar, (3) 0.1 NPT simulation at 600 K and 1 bar. This loop is repeated several times to decompress the system to experimental densities.

2.2.4 Simulation software

The atomic MD simulations in this research are done through two platforms, the first one is a Python-based platform named Atomic Simulation Environment (ASE) [19]. In ASE, the powerful syntax of Python combined with the NumPy array library makes it possible to perform very complex simulation tasks such as density functional theory (DFT) calculations of electronic structures and *ab initio* MD (AIMD). For MD simulation specifically, ASE can import different geometry optimization algorithms and force fields for simulations.

On the other hand, the classical MD simulator, Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) [50], is also used for the MD simulations of PILs. In LAMMPS, it is more convenient to specify the OPLS-AA parameters for the specific simulation system. One should note that the use of a force field has a direct influence on the simulation accuracy. Besides, the parameters of the OPLS-AA force field are not predefined and hard to be implemented in ASE. In the LAMMPS input file, the OPLS-AA parameters can be easily called to initialize the simulations.

Apart from the difficulty of using the OPLS-AA force field in the ASE, there is another reason for the separate use of two simulators. Until now, the newest ASE neural network ANI-2X can only be employed in the ASE platform, concerning its unique interpretations (black box neural network) of pairwise interactions that lack actual physical descriptions. There is a plugin for OpenMM that allows the ANI Neural Network Potential to be used to compute forces and energies, but only for the older ones like ANI-1 and ANI-1x. Similarly, there are well-studied machine learning pair potentials such as DeePMD-kit [53] that can be employed in both LAMMPS and ASE but are out of the scope of this research

project.

In ASE, there are multiple functions that can be used for reading the initial structures, such as readproteindatabank, read_xyz functions. Alternatively, it is convenient to construct a lattice structure by creating arrays and imposing periodic boundary conditions on them. In this study, read_lammps_data function is employed for reading the configuration that is also used for LAMMPS simulations. Yet, ASE is not able to automatically identify the atom types in the system, even though the atom types are specified according to their relative atomic masses in the Masses section in the LAMMPS data file. Z_of_type argument in the read_lammps_data function is used for ASE to identify the atom types in a system. This argument assigns atomic numbers to specific atom types according to the periodic table.

Prior to the MD simulations, random velocities are applied to the atoms, which is done by MaxwellBoltzmannDistribution function, which has two main arguments, atoms and temperature in Kelvin, distributing momenta to all atoms in the system by Maxwell-Boltzmann distribution. In addition, the 'calculator' in ASE refers to the potential function that is utilized for simulations, which is the ANI-2x model in this case, that has to be explicitly determined. To run the actual dynamics, it is required to all the integrator, thermostat, and barostat functions. ASE provides several integrators for running MD simulations. Here, only the NPT integrator for the production run is called since the initial configuration used for ANI-2x potential had already reached a reasonably stable state from LAMMPS simulations. Besides, it is useful to call the MDLogger function to record the data during the simulation and the Trajectory function to output the trajectory for post-processing of the simulations and visualization.

In the NPT integrator, the simulation box is limited to moving along three principal directions, excluding the shear movement, which is realized by setting an identity tensor to the mask argument.

```
from ase.io.trajectory import Trajectory
from ase.md.npt import NPT
from ase.io.lammpsdata import read_lammps_data
from ase.io.lammpsdata import write_lammps_data
from ase.md.velocitydistribution import
   MaxwellBoltzmannDistribution
from ase.md import MDLogger
from ase import units
import torch
import torchani

atoms = read_lammps_data('data.PolyBMIM_Cl_1.npt2',
   style='full', units='real', Z_of_type={1: 6, 2: 6, 3: 6, 4:
   6, 5: 6, 6: 6, 7: 6, 8: 6, 9: 6, 10: 6, 11: 1, 12: 1, 13:
   1, 14: 1, 15: 1, 16: 1, 17: 1, 18: 1, 19: 1, 20: 1, 21: 7,
   22: 7, 23: 17})

device = torch.device('cpu')
calculator=torchani.models.ANI2x().to(device).ase()
atoms.set_calculator(calculator)
MaxwellBoltzmannDistribution(atoms, temperature_K=T)

T = 300
timestep = 1 * units.fs
```

```
pressure = 1 * units.bar
interval = 100
tpro = 1000000

print("Beginning NPT production run...")

dyn3 = NPT(atoms, timestep=1 * units.fs, externalstress=(-1 *
   units.bar, -1 * units.bar, -1 * units.bar, 0, 0, 0),
   ttime=2 * units.fs, pfactor=2 * units.fs,
   trajectory='PolyBMIM_Cl_1_ANI.traj', temperature_K=T,
   logfile=None, loginterval=100, mask=([1, 0, 0], [0, 1, 0],
   [0, 0, 1]), append_trajectory=True)

logger_production = MDLogger(dyn3, atoms,
   'PolyBMIM_Cl_1_ANI.log', header=True, stress=True, mode='a')
traj = Trajectory('PolyBMIM_Cl_1_ANI.traj', 'a', atoms)
dyn3.attach(traj.write, interval=interval)
dyn3.attach(logger_production, interval=interval)
dyn3.run(tpro)
```

The code above demonstrates a complete process of MD simulation using ANI-2x potential, from setting up the system to starting running. Similarly, one can define a function to print out desired properties of the system during the MD simulation, such as the density of the PILs and the volume of the simulation box.

```
def printenergy(a=atoms, file_name="PolyBMIM_Cl_1_ANI_my.log"):
   global call_count
```

```
call_count += 1 # increment the call count

"""Function to print the potential, kinetic and total
   energy."""

epot = a.get_potential_energy() * units.kcal / units.mol
ekin = a.get_kinetic_energy() / len(atoms) * units.kcal /
   units.mol
ekin_all = a.get_kinetic_energy() * units.kcal / units.mol
volume = atoms.get_volume()
mass = atoms.get_masses().sum()
density = mass/volume * 1.66053906660e-24 / (1e-8)**3

system_info = 'Simulation Info: Epot = %.3f kcal/mol Ekin =
   %.3f kcal/mol (T=%3.0f K) ' \
          'Etot = %.3f kcal/mol Volume = %.3f Angstrom^3
             Density = %.3f g/cm^3'\
          % (epot, ekin_all, ekin / (units.kcal /
             units.mol) / (1.5 * units.kB), epot + ekin,
             volume, density)

print(system_info)

with open(file_name, "a") as f: # "a" means append mode
   f.write(f'{call_count * interval}: {system_info}\n') #
      write call_count and energy_info to the file
```

When the properties are printed throughout the simulations, one has to pay

attention to the unit conversion. According to the ASE documentation, the energy is in *eV*, length in , time in *fs*, mass in atomic mass unit *amu* [19]. This is the reason that units attribute is used for this 'printenergy' function for convenient unit conversion.

Before the actual simulations, it is worth conducting small simulations. The term 'small' here refers to sample systems with shorter time scales and fewer atoms. By doing this, researchers are able to estimate the total simulation time on the materials. Besides, some of the errors will pop out during the small test trials, which can be fixed and eliminated during the actual runs. This is quite important since, for much larger atomic systems, large amounts of computational resources such as memory and electricity might be wasted if meaningless simulations are run. Time estimation is also essential because, for most simulations running on supercomputers, time and computational cores are allocated prior to the simulations. It is not desired that computational resources are excessive for the simulations, but at the same time, they are expected to finish as soon as possible for analyses.

Some of the approaches for studying the MD simulations include analyzing the trajectory files and processing the thermodynamics output data. The visualizations of the simulations are carried out in Open Visualization Tool (OVITO) [48] and VMD. For the simulations with ANI-2x potential, the trajectory of the simulations is in traj file format, which is a file format that can only be opened by ASE. In this case, the traj files are converted into xyz and pdb coordinates files for analyses. Additionally, MDAnalysis [36] is taken for investigating the variations of some of the key properties of the PILs systems during production runs.

CHAPTER 3

RESULTS AND DISCUSSIONS

3.1 Energy of the systems

There are some differences between the three force fields when calculating the energy. As mentioned before, when using the OPLS-AA empirical force field, the potential energy of the system is composed of energy components that are contributed by angles, bonds, dihedrals, improper dihedrals, and electrostatic interactions, i.e., the Coulomb's force. While the kinetic energy is calculated from the velocities of each atom.

In addition, when performing the simulations using a polarizable force field, the energy is not only calculated from the above parts, but the contributions from the Drude Oscillators are also taken into account. When simulating small systems like PolyBMIM_Cl_1 and PolyBMIM_TFSI_1, the energy differences between the empirical and polarizable force fields are not obvious, which means that they can comparably describe the systems. When these two force fields are used for larger systems, PolyBMIM_Cl_4 and PolyBMIM_TFSI_4, the energies are approximately 4 times that of smaller systems, which is consistent with the increase of simulation sizes.

However, the ANI-2x potential cannot predict the energy of the system well. This is because of the nature of the ANI potential family. They are designed for simulations of small drug-like molecules, and the DFT-calculated data of polymers and molecules containing over 100 atoms are not included in their training set. ANI-2x potential lacks the ability to describe the chemical environment of

complex polymer structures, so it is normal to find discrepancies in the energies when compared with well-developed empirical force fields.

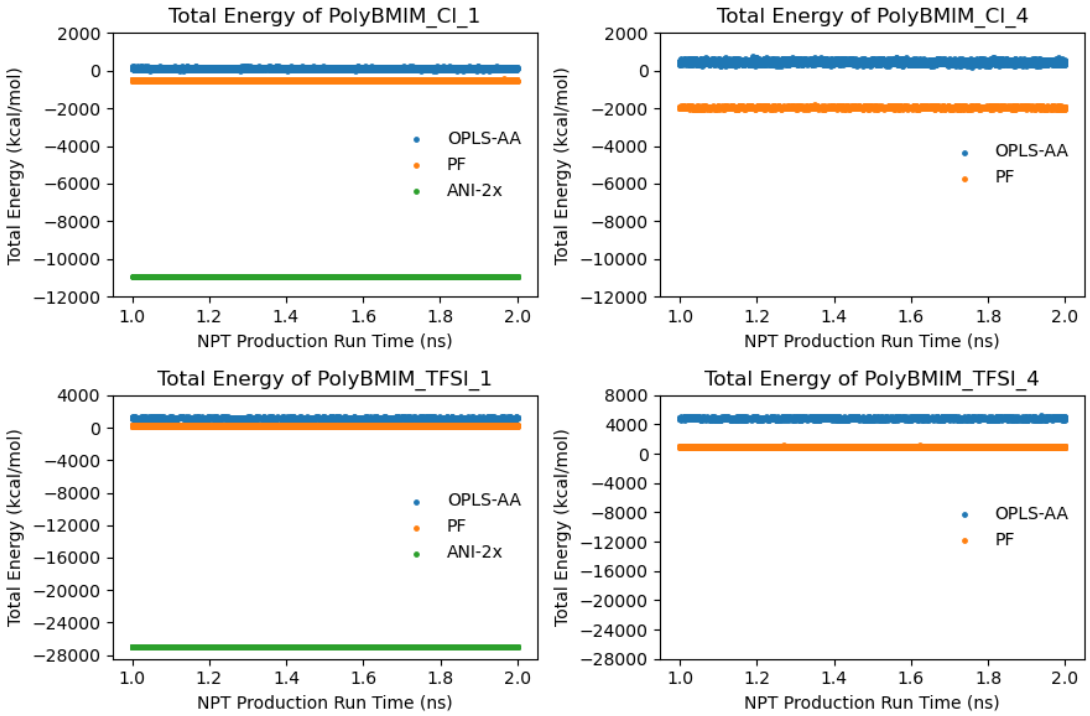

Figure 3.1: The total energy during the last nanosecond of the production runs for four simulated systems using different force fields. The energies calculated from ANI-2x potential in ASE were converted into unit kcal/mol for comparisons.

Figure 3.1 demonstrated the total energy variations during the production runs. It is obvious to find that all the total energies during their respective production runs are fairly stable, indicating that when collecting the data, the structures have been all equilibrated.

3.2 Density

The densities of each simulated system are tabulated in Table 3.1. After each density values column, there are two error columns that correspond to the errors with respect to Acevedo and coworkers' simulation data [12] and weighted average from previous experiments [21, 34]. It can be seen from the plot that for the OPLS-AA force field, the density results are often underestimated. Only for the bigger system of PolyBMIM_Cl_4 is the value overestimated. On the other hand, when predicting the density of the PolyBMIM_Cl_1 system, the PF with Drude oscillators resulted in a larger error compared to OPLS-AA. However, the densities of the three other systems possess smaller errors when using PF. Due to the run-time limitation, the ANI-2x potential was only used for MD simulations of PolyBMIM_Cl_1 and PolyBMIM_TFSI_1. Both simulations yield comparable results with that of empirical force field counterparts, and especially when comparing the OPLS-AA and ANI-2x potential, the latter one performed better.

	OPLS-AA	% Error	% Error	PF	% Error	% Error	ANI-2x	% Error	% Error	Ref.	Exptl.
PolyBMIM_Cl_1	1.040	-3.673	-3.673	1.132	4.807	4.807	1.037	–4.015	-4.015	1.080	1.080
PolyBMIM_Cl_4	1.127	4.382	4.382	1.032	-4.483	-4.483	N/A	N/A	N/A	1.080	1.080
PolyBMIM_TFSI_1	1.330	-14.098	-7.399	1.544	-0.261	7.518	1.422	-8.121	-0.955	1.548	1.436
PolyBMIM_TFSI_4	1.373	-11.325	-4.408	1.532	-1.008	6.712	N/A	N/A	N/A	1.548	1.436

Table 3.1: Time-averaged density (g/cm^3) of each simulations. Errors are in percentages, in which the plus and minus signs represent overestimate and underestimate, respectively. Both the reference data and experimental data are based on ILs, instead of PILs.

Although the parameters that form the OPLS-AA force field are optimized particularly for the MD simulations on ILs, their parameters are fixed and do not

account for the effects from the atomic environments, while ANI-2x is based on quantum chemical data and takes radial and angular chemical environments of each atom in the system into account. Hence, to some extent, the ANI-2x outperforms OPLS-AA in density calculations for small molecule systems. Nonetheless, even if the predictions of the densities by the three force fields have different degrees of deviation, which are not that big, all simulated systems should be properly and sufficiently equilibrated before the production runs. The fact that OPLS-AA is worse at simulating PolyBMIM_TFSI systems in this research is due partially to the insufficient equilibration. And since PolyBMIM_TFSI is more complex than PolyBMIM_Cl, so less equilibration might not affect PolyBMIM_Cl that much.

3.3 Radius of gyration

When analyzing the dimensional distribution of polymer chains, the radius of gyration (Rg) is a crucial descriptor. This physical quantity measures the structural compactness of polymers or simply the 'size' of the polymers. In LAMMPS, the computation of Rg is done by compute gyration command, which needs to be specified with which group of atoms are calculated. Similarly, in MDAnalysis, there is a function called radius_of_gyration() that helps estimate the Rg of selected atoms from a trajectory. In Equation 3.1, M is the total mass of the polymer chain, and r_{cm} is the center-of-mass position of that chain, and the sum is over all atoms in the polymer chain. R_g^2 is a six-component vector, which corresponds to one global scalar R_g.

$$R_g^2 = \frac{1}{M} \sum_i m_i \left(r_i - r_{\text{cm}}\right)^2 \tag{3.1}$$

It can be seen from the Fig 3.2 that, when simulating the system PolyBMIM_Cl_1, the Rg values for the three force fields are comparable and consistent with one another, with ANI-2x potential underestimated the Rg in the first half of the production run. The Rg values fluctuate during the last 1 ns in the production run without obvious increases or decreases, which indicates a stable status of the polymer chain. On the other hand, even though the polymer chain in the PolyBMIM_TFSI_1 system has reached a steady state, the Rg values from the three force fields deviate from each other. OPLS-AA has the highest measurements of Rg, followed by PF and then ANI-2x.

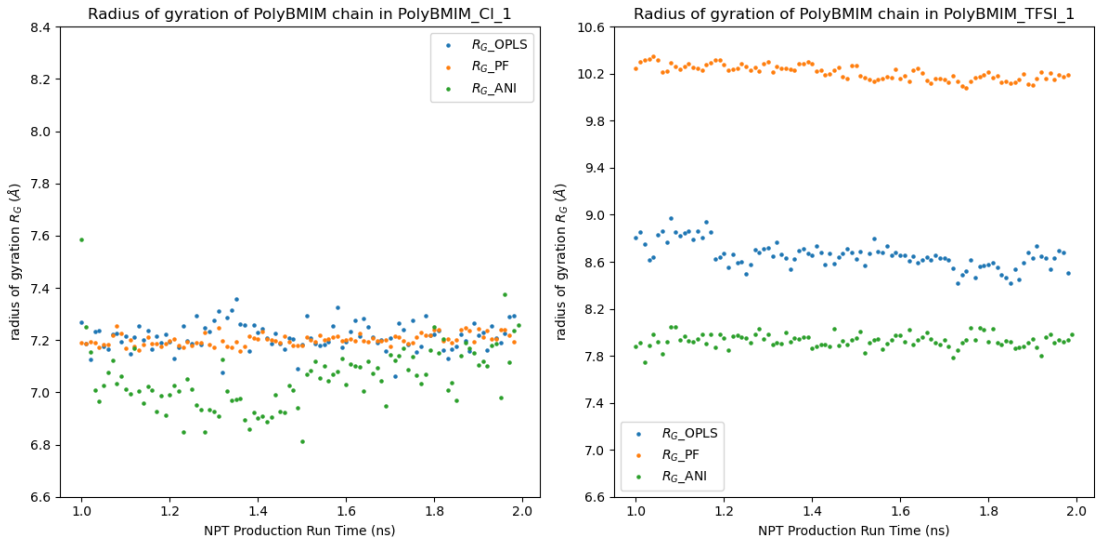

Figure 3.2: The Rg of the polycation chain in the PolyBMIM_Cl_1 and PolyBMIM_TFSI_1 systems. The trajectory of the last 1 ns was collected for analysis. ANI-2x underestimated Rg for both structures.

The reason why the PolyBMIM chain in PF simulation has a slightly smaller Rg compared to OPLS-AA might result from the constraints of C-H bonds and different fixed temperatures for Drude cores and Drude particles, leading to a decrease in the kinetic energy and a stiffer structure. Again, ANI-2x underestimated the Rg, not only in the first half but throughout the whole production run. This further enhanced the conclusion that ANI-2x is not performing well when used for larger system MD simulation since the TFSI anion is structurally more complex, and there are more atoms in the PolyBMIM_TFSI_1 than in PolyBMIM_Cl_1. Since there are already discrepancies between ANI-2x and empirical force fields for a system that contains 275 atoms (PolyBMIM_Cl_1), from this study, it can be predicted that ANI-2x would be better for a PILs system containing less than 200 atoms for more accuracy. However, simulations on small systems may not yield statistically meaningful results when the bulk properties are to be studied.

3.4 Radial distribution functions

The concept of radial distribution function (RDF) comes from statistical mechanics. Being a structural interpreter, it provides a statistical description of the local packing and particle density of the system by describing the average distribution of particles around a central reference particle [18]. In order to account for the locations of two different types of atom groups i and j, the radial pair-distribution function can be written as [61]:

$$g_{ij}(r) = \frac{V}{N_i N_j} \left\langle \sum_{i=1}^{N_i} \sum_{j=1}^{N_j} \frac{\delta\left(r_{ij} - r\right)}{4\pi r^2} \right\rangle \tag{3.2}$$

In which N_i and N_j are the numbers of atoms of polycation PolyBMIM+ and anions Cl- or TFSI-, V is the time-averaged volume of the simulation box, and δ is the Dirac delta function. By conventional normalization of the RDF, g_{ij} tends to reach 1 at the large separation between the two atom groups, which means that the system is homogenous.

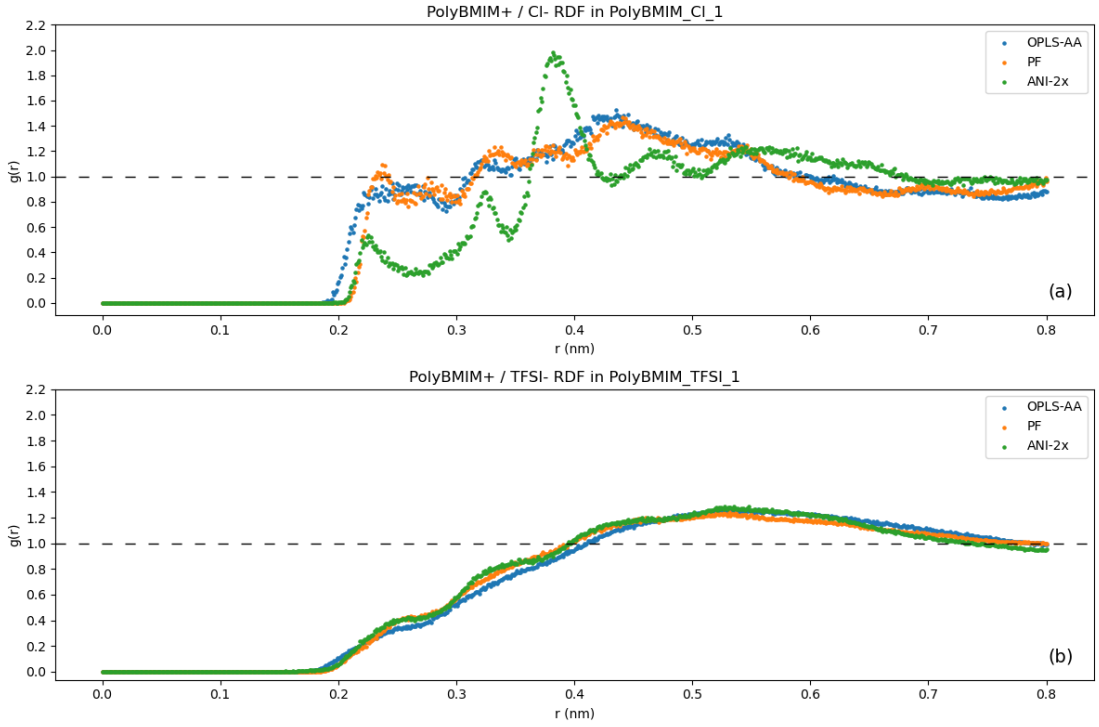

Figure 3.3: Radial distribution functions (RDFs) between polycation PolyBMIM and Cl/TFSI anions in the PolyBMIM_Cl_1 (a) and PolyBMIM_TFSI_1 (b) systems. Comparisons are done among the three force fields.

Figure 3.3 shows the RDFs calculated from the two small systems and three different force fields. For both PolyBMIM_Cl_1 and PolyBMIM_TFSI_1 systems, the differences between the two empirical force fields are relatively small. Many parts of the orange dots and blue dots overlap with each other. The first solvation shells for OPLS-AA and PF occur at around 0.43 nm. This indicates that at

this distance, there is a higher probability of finding anions with the polycation as the reference, and also elucidates that the ion pairs are more stable at this distance. However, the RDF calculated from ANI–2x tells another story. Although before its first solvation shell appears, RDF of ANI-2x is fluctuates as empirical ones do, its peak value is much higher than OPLS-AA and PF. Also, the first solvation shell occurs at a shorter distance, at approximately 0.38 nm.

Discrepancies are expected. ANI-2x was trained on the GDB data set, which contains small drug-like molecules. Therefore, in terms of structures (angles, bonds, dihedrals), ML potential cannot describe the dynamics of polycations and counterions accurately. In addition, ANI-2x does not take the partial charges into account, while OPLS-AA and PF do, so when updating the positions of atoms within the system, it is predicted that differences will arise.

Nonetheless, the RDFs calculated from PolyBMIM_TFSI_1 are amazingly self-consistent. All three RDF curves stack on one another. The curves start to grow steadily when the distance is around 0.18 nm and reach a peak at 0.52 nm, where the first solvation shells appear altogether. From this study, it can be concluded that ANI-2x ML potential is as good as empirical counterparts when predicting the structural aspect of the PolyBMIM and TFSI anion-containing system, and only this particular system. But further simulations should be done on larger time scales and size scales to exclude the possibility that this is just a coincidence.

Fig 3.4 demonstrates the RDFs for two larger systems. It can be observed from the plots that RDFs obtained from OPLS-AA and PF are comparable. Both Cl and TFSI anions start to emerge when they are approximately 0.2 nm from the polycation, which is consistent with the simulations of smaller systems.

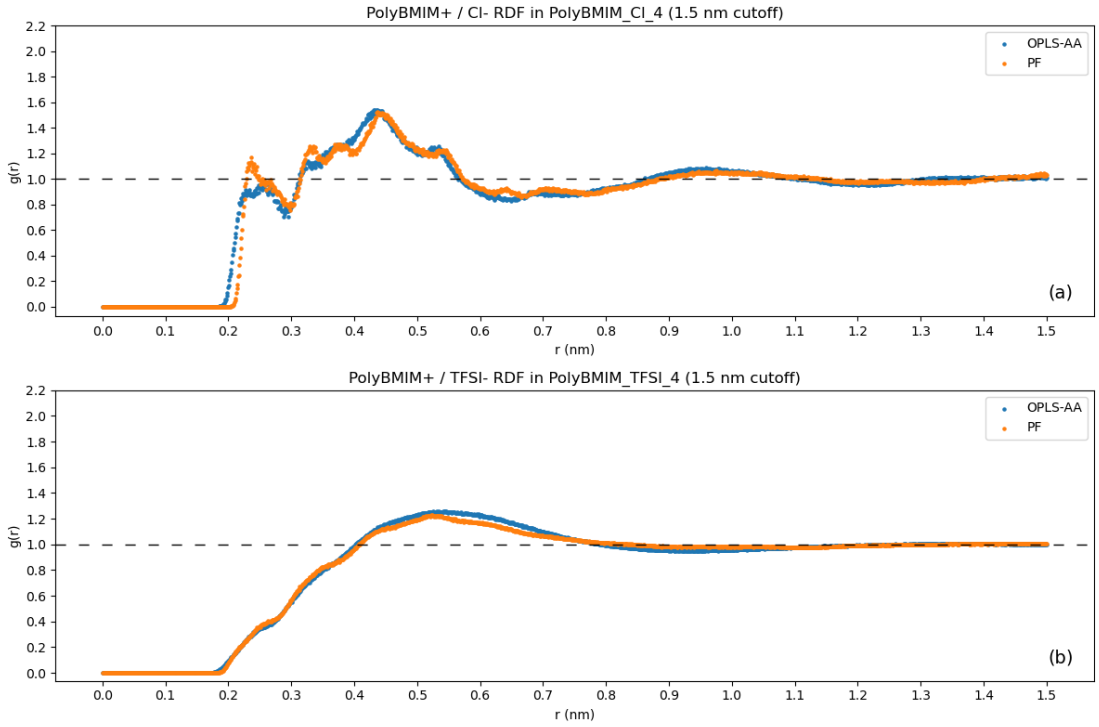

Figure 3.4: Radial distribution functions (RDFs) between polycation PolyBMIM and Cl/TFSI anions in the PolyBMIM_Cl_4 (a) and PolyBMIM_TFSI_4 (b) systems. Comparisons are done between two empirical force fields.

3.5 Mean-square displacement and self-diffusion coefficients

The diffusivity of anions can be quantified as the mean square displacement (MSD). MSD of anions can be calculated by LAMMPS and by processing trajectory files from ANI-2x simulations in MDAnalysis. Given the trajectory files, one can calculate how much a certain group of atoms move from their original position. (position at the first frame of the trajectory) The quantity of MSD is averaged over all atoms. The equation for the calculation of MSD of anion type i at time t is as follows:

$$MSD(t) = \frac{1}{N_{\text{anions}}} \sum_{i=1}^{N_{\text{anions}}} (r_i(t) - r_i(0))^2 \tag{3.3}$$

According to the Einstein relation, diffusivity of type i anion can be calculated [26]:

$$D_{\text{i}} = \lim_{t \to \infty} \frac{1}{6t} MSD(t) \tag{3.4}$$

This means that when time becomes infinity, the value of MSD and ion diffusivity D is in a linear relation. Starting from the first frame of the last 1 ns of the production run, the time-averaged MSD values are shown in Figure 3.5. Taking a trajectory of 10000 frames, the MSD is averaged every 200 frames, and x, y, and z directions are all taken into account.

For PolyBMIM_Cl_1, the MSD calculated from OPLS-AA and PF are reasonably comparable. There are slight increases in the MSD of the Cl anions after 1.8 ns. While the MSD from ANI-2x potential is much different from the empirical force field results, forming a much steeper increase with an obvious deduction after 1.8 ns. The fitted values of $D_{OPLS-AA-Cl}$ and D_{PF-Cl} are 6.683×10^{-13} m^2/s and 1.442×10^{-12} m^2/s, which are factors of 60.3 and 27.9 smaller than $D_{ANI-2x-Cl}$ (4.027×10^{-11} m^2/s), respectively.

Again, the empirical force fields predicted the MSD for TFSI anions similarly. Interestingly, the MSD of TFSI anions experiences a rise after 1.8 ns in the production run, just like the Cl anions do. However, ANI-2x potential performed drastically differently when simulating PolyBMIM_TFSI_1. The MSD of TFSI anions undergoes a slow, stable increase throughout the simulation. The fitted values of $D_{OPLS-AA-TFSI}$ and $D_{PF-TFSI}$ are 3.4×10^{-11} m^2/s and 4.079×10^{-11} m^2/s,

which are factors of 9 and 10.8 greater than $D_{ANI-2x-TFSI}$ (3.786×10^{-12} m^2/s), respectively.

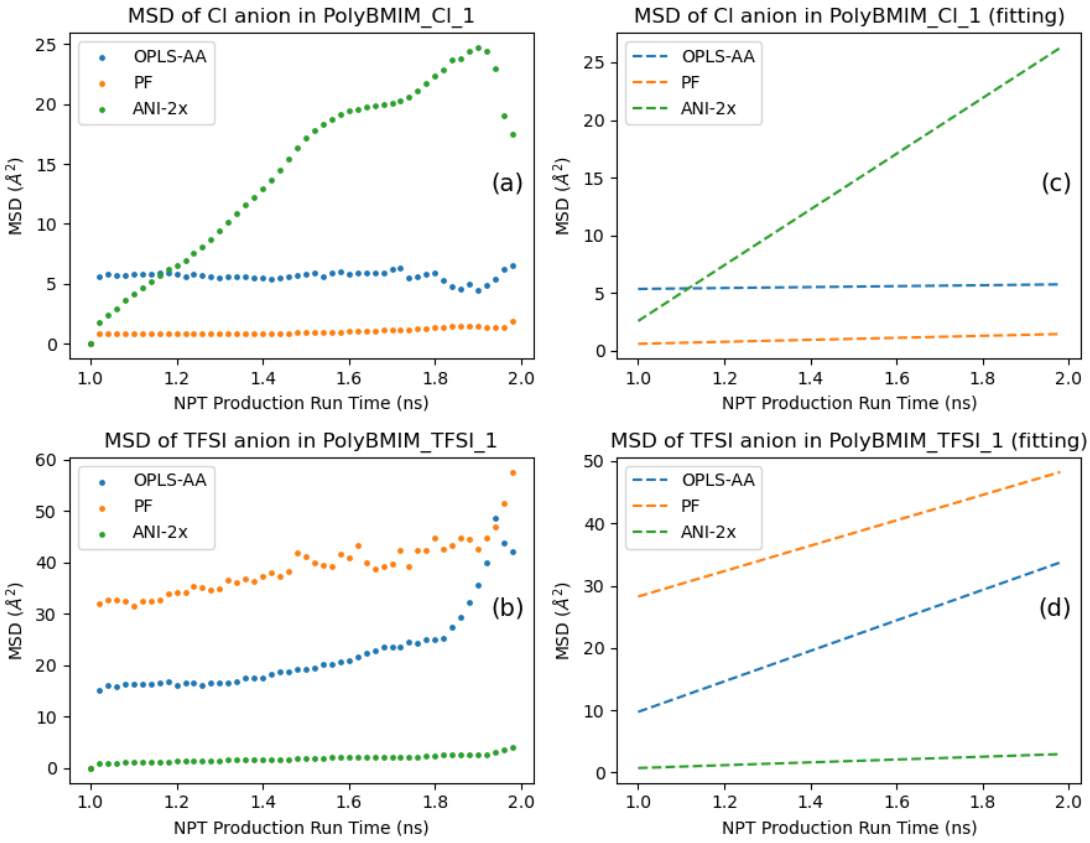

Figure 3.5: Mean square displacement of Cl- and TFSI- anions in PolyBMIM_Cl_1 and PolyBMIM_TFSI_1 systems. Data is collected from the last 1 ns of the production of each trajectory. (a), (c) is the actual plot of the calculated MSD of anions, and (b), (d) are the linear fits of MSDs.

For the BMIM_Cl ionic liquid, the simulated diffusion coefficients for cation and anion are comparable and equal to 6.8×10^{-12} m^2/s and 6.7×10^{-12} m^2/s, respectively [28]. This value is larger than both $D_{OPLS-AA-Cl}$ and D_{PF-Cl}, but much smaller than $D_{ANI-2x-Cl}$. The reason for this might be that in the study, the anions are hard to reach diffusive regimes, remaining in the sub-diffusive or even

ballistic regimes with extreme slow motion. Another reason might be that the literature value was calculated based on BMIM_Cl, rather than PolyBMIM_Cl. The polymer chains in the PILs system might impede the diffusion of mobile anions, yielding a much slower diffusion coefficient.

On the other hand, when accounting for the diffusion of TFSI anions, the experimental reference value is 2×10^{-11} m^2/s, according to Rollet and co-workers' study [42]. The experimental value and the values from two empirical force fields are within the same order of magnitude. The reason that $D_{OPLS-AA-TFSI}$ and $D_{PF-TFSI}$ are slightly higher might be the simulation temperature. The simulation temperature is around 600 K, which is much greater than the room temperature at which the experiments were conducted. Similarly, the literature was examining the diffusion coefficient of TFSI in the BMIM_TFSI without any polymeric matrices.

The overestimated $D_{ANI-2x-Cl}$ and underestimated $D_{ANI-2x-TFSI}$ elucidates the inaccuracy of ANI-2x potential for demonstrating the diffusion of anions in the PILs systems. However, longer production runs (up to hundreds of ns) should be carried out to make sure that the anions are within the regime of interest, the diffusive regime. When the simulation time is long enough, the whole system might reach an equilibrium state, where the MSD of the anion exhibits a steady increase without fluctuations. Zhang et al. [61] state that the production run should contain more than 200 ns second for collecting more data for analysis. In their work of analyzing the PF_6 anions in the PILs system, the production run is about 220 ns, and the last 200 ns results were used for calculating related properties.

3.6 Run-time comparisons

Due to the natural, inherent advantages of parallel computation of LAMMPS, it passes information about the simulation system through a message-passing interface (MPI) to the computation units. MPI can call multiple processors for the computation of the whole system by distributing computation tasks to them. For example, if 8 processors are called during a computation, 1/8 of the forces and equations of motions (EOM) are calculated and solved by one processor. If two processors deal with two adjacent parts of the simulation box, these two processors have to communicate and exchange information about the atoms at their boundary.

However, although ANI-2x potential can be used via a graphics processing unit (GPU) for the computation of forces and energy, the GPU-enabled ANI-2x potential is not available for MD simulation in ASE, which means it is only working on the central processing unit (CPU). This is primarily a problem of ASE since the integrator, thermostat, barostat, etc., are written in plain Python on the CPU. When running the MD simulations, ASE takes approximately 10 ms for the ANI model to evaluate forces. if a GPU is used, the MD code will be bottle-necked by ASE calls, resulting in an invalid usage of the GPU. In this case, only one core on the CPU can be utilized for the calculation, which massively decreases computational efficiency.

Apart from the differences between the hardware for computations, the structure of the computed systems will affect the calculation time as well. PF brings more degrees of freedom by introducing extra Drude particles, which increases the computation expenses, including the additional mass, angle, and

bond parameters for describing the Drude core-Drude particle interactions.

Thanks to Prof. Yeo and the G2 cluster provided by the College of Computing and Information Science at Cornell University for allocating large amounts of computational resources, parallel computing for the MD simulations with over a hundred cores can be possible. Prof. Benedek also shared computational resources for running MD simulations on the TACC Stampede2 supercomputer. The figure below demonstrates the run-time comparisons between different runs conducted on different computational platforms.

OPLS-AA simulations		PolyBMIM_Cl_1	PolyBMIM_TFSI_1	PolyBMIM_Cl_4	PolyBMIM_TFSI_4
Parrallel computation	G2 cluster / 160 cores	336 mins			326 mins
	Stampede2 / 128 cores		111 mins	167 mins	
Drude PF simulations		PolyBMIM_Cl_1	PolyBMIM_TFSI_1	PolyBMIM_Cl_4	PolyBMIM_TFSI_4
Parrallel computation	G2 cluster / 160 cores		961 mins		
	Stampede2 / 128 cores	467 mins		547 mins (256 cores)	671 mins (256 cores)
ANI-2x simulations		PolyBMIM_Cl_1	PolyBMIM_TFSI_1	PolyBMIM_Cl_4	PolyBMIM_TFSI_4
CPU computation	AMD Ryzen 9 5900HX / 8 cores	12000 mins	20000 mins		
	Google Colab Intel Xeon / 1 core	2640 mins (estimated)	4800 mins (estimated)		

Table 3.2: Run-time comparisons between different computation modes. For LAMMPS simulations of four systems, G2 cluster and Stampede2 were used, whereas for the ANI-2x simulations, the personal laptop was used. For the simulations on the G2 cluster, 160 cores were used. And for the simulations on Stampede2, 128 cores were used for smaller systems, and 256 cores were employed for most complex systems.

For the simulation of PolyBMIM_Cl_1 on the G2 cluster using the OPLS-AA force field, the simulation time took relatively longer. This is because, during the equilibration stage, multiple runs were used to avoid the circumstances that atoms are moving too fast and the particle-particle particle-mesh algorithm can-

not solve long-range electron interactions. However, each 'run' command used in the LAMMPS input script requires a re-initialization of the calculations, resulting in a longer simulation time.

Although the supercell simulation box is 4 times the single cell, which means the atoms are also 4 times the single cell, the computation run-time is not increasing linearly with the system size. And the number of interactions between the atoms is magnified exponentially. This is directly affecting the computation time, which is most obvious in the largest system, PF simulations with more than 2700 atoms, including the Drude particles.

The ASE can support ANI-2x potential for energy and forces calculations during energy minimization of a simple cell. However, AN-2x cannot be directly used for parallel computation either GPU-accelerated calculations in MD simulations. This brings a severely increased computation time, and a single core is doomed to yield slower calculations and evolution of Newton's equations. For the production run of an equilibrated PolyBMIM_Cl_1, the total runtime approaches 12000 minutes, the equivalent of 36 times the computation time from the G2 cluster on the same system.

To compare the computation performance with a local laptop and cloud computing Google Colab, 2 nanosecond production was tested both on Ryzen CPU and Colab, whose CPU model is Intel(R) Xeon(R) CPU @ 2.30GHz. It can be seen that with good resource distribution and memory management, Colab outperforms Ryzen and is approximately 3.4 times faster. The reason for not performing ASE ANI-2x based MD on G2 cluster or supercomputer Stampede2 is that it will take up too much memory, and the submitted job will be automatically terminated without warning.

CHAPTER 4

CONCLUSION AND OUTLOOKS

4.1 Conclusion

This study mainly investigated the properties of two types of polymeric ionic liquids (PILs), namely the PolyBMIM_Cl and PolyBMIM_TFSI systems. Simulations on single cells and 4-times larger supercells were conducted using three kinds of force field models: OPLS-AA, Polarizable force field with Drude Oscillators, and machine learning deep neural network potential ANI-2x. G2 cluster and supercomputer TACC Stampede2 were accessed for the LAMMPS MD simulations using empirical force fields for all systems, whereas the laptop was used for ANI-2x simulations on PolyBMIM_Cl_1 and PolyBMIM_TFSI_1. After energy minimization and equilibrations, the last 1 ns trajectory from the production runs is taken for analysis.

Compared with empirical force fields, ANI-2x is not predicting the energy of the systems well, which might be resulted of its inaccurate long-range interactions in the polymer structure. However, the densities information of the ANI-2x simulations is consistent with OPLS-AA and PF simulations. In addition, ANI-2x underestimated the radii of gyration of polymer chains both in PolyBMIM_Cl_1 and PolyBMIM_TFSI_1. Another comparison criterion is the radial distribution functions (RDFs) between the polycation and the mobile anions. For the PolyBMIM_Cl_1 system, the calculated distance at which the first solvation shell appears from ANI-2x is not comparable with the other two empirical force fields. However, for the PolyBMIM_TFSI_1, the RDF from three potentials are incredibly similar. The diffusion coefficient of anions calculated from

three force fields vary. OPLS-AA and PF have comparable results, while ANI-2x overestimated the self-diffusion coefficient of Cl- anion and underestimated the self-diffusion coefficient of TFSI- anion.

4.2 Outlooks

Looking through the whole project, ANI-2x is performing well in predicting the densities of S, F, N, C, and O-containing molecules. When analyzing the energies and structural perspectives of the PILs, ANI-2x seems to be weak and inaccurate. Although it has been proven to be accurate when calculating the RDFs of PolyBMIM_TFSI_1 system and Rg of PolyBMIM in PolyBMIM_Cl_1, most of the properties from ANI-2x are not comparable with OPLS-AA and PF. This is probably because of the inherent disadvantages of ANI-2x for simulating charged macromolecules.

Although the ANI neural network model can be easily implemented, the atomic interactions in the polymer are not embedded, i.e., the model may not recognize polymer chains when the initial structures are read and sequentially output meaningless data. The aim of ANI potential was for drug-like molecule discovery, so all the data sets that have been used for training ANI potential are small-molecules based, and polymeric configurational and conformational information was not taken into consideration. In addition, the ANI potential cannot be directly used in traditional MD simulators, instead, it is invoked in a Python-based simulation package. Moreover, it is impossible to intuitively tell what atomic interactions are defined from the architecture of the machine learning potential, so there is a lack of physical representations, therefore, it

is reasonably hard to capture thermodynamic properties from the simulations results. To further enhance the conclusions from this preliminary study and improve the capability of ANI-2x for simulating polymer systems, efforts can be put into several aspects:

- In this study, although PolyBMIM_TFSI_4 contains more than 2700 atoms, it is still a relatively small system, not to mention other even smaller simulation cells. In Zhang et al. [61] study, there are 32 monomers on each chain, and 8 chains in a single simulation cell, forming a roughly 9000-atoms system. System size is closely associated with simulation quality. If the system is too small, nonphysical and unrealistic data might be calculated. Furthermore, limited simulation time is another factor affecting the simulation results. As mentioned earlier, simulations up to hundreds of ns might be ideal for researching ion diffusion in ionic liquids systems. Therefore, conducting simulations with larger cell sizes and longer time span is essential.
- Concerning the same system, the ML force field is already much faster than DFT calculations. However, when compared with the empirical force files with parallel computation acceleration, it is much slower. The source code from ASE only enables pure CPU calculations when running MD, while ANI-2x can only be implemented in ASE. If a 200-ns simulation is run with ANI-2x potential in Python on a single CPU, the run time might be 3 years. Fortunately, the ANI team is working on combining ANI potential with LAMMPS, facilitating the realization of parallel computation using such ML potential. With the LAMMPS plugin, running for a longer time becomes possible.
- Based on the ANI-2x model, train our ML neural network potential based

on databases containing polymeric structures. ANI-2x was trained on DFT data from small organic drug-like molecules, so it is expected that they are not able to perform well with polymeric systems. Ramprasad et al. proposed a DFT-calculation-based polymer database that contains optimized structures of 1073 polymers [20]. Accessing the data is easy, but the main difficulty is parsing the structural data and feeding them into a neural network for training. Bonds, angles, dihedrals, and other configurational descriptors should be prepared accordingly.

In the longer term, the investigations of PILs should focus on larger systems and longer simulation time, implementation of ANI-2x potential in LAMMPS; and training our own ML potential based on ANI neural network architecture. Indeed, apart from ANI potential, there are other types of ML potentials such as Gaussian Approximation Potential [2] and Graph Neural Network Potential [7] for the benchmark. The booming of ML potential is promising for studying the complexity of materials comprehensively.

APPENDIX A

OPLS-AA FORCE FIELD PARAMETERS [61, 29]

A.1 Lenard-Jones interaction and atomic polarizability

type i	$\epsilon_{ii}(kcal/mol)$	$\sigma_{ii}(Angstrom)$	$\alpha(Angstrom^3)$
CA	0.066	3.5	1.016
CM	0.066	3.5	1.016
CM1	0.066	3.5	1.016
CM2	0.066	3.5	1.016
CR	0.07	3.55	1.122
CS1	0.066	3.5	1.016
CS2	0.066	3.5	1.016
CT	0.066	3.5	1.016
CW1	0.07	3.55	1.122
CW2	0.07	3.55	1.122
HA	0.03	1.92	0.323
HM	0.03	1.92	0.323
HM1	0.03	2.5	0.323
HM2	0.03	2.5	0.323
HR	0.03	1.72	0.323
HS1	0.03	2.5	0.323
HS2	0.03	2.5	0.323
HT	0.03	2.5	0.323
HW1	0.03	1.72	0.323
HW2	0.03	1.72	0.323
NA1	0.17	3.25	1.208
NA2	0.17	3.25	1.208
ct	0.066	3.5	1.050
ft	0.053	2.95	0.600
nt	0.17	3.25	1.450
ot	0.21	2.96	1.360
st	0.25	3.55	0.500
Cl	0.148	3.77	3.610

Table A.1: Force field interactions for L-J interactions and polarizability.

A.2 Partial charges

type i	$Mass(a.m.u)$	$q_i(e)$
CA	12.011	-0.071916
CM	12.011	-0.191281
CM1	12.011	-0.286559
CM2	12.011	-0.1432795
CR	12.011	-0.08164
CS1	12.011	-0.112314
CS2	12.011	0.058799
CT	12.011	-0.168031
CW1	12.011	-0.203382
CW2	12.011	-0.165764
HA	1.008	0.103288
HM	1.008	0.200553
HM1	1.008	0.198717
HM2	1.008	0.066239
HR	1.008	0.215218
HS1	1.008	0.06337
HS2	1.008	0.032434
HT	1.008	0.056042
HW1	1.008	0.23804
HW2	1.008	0.244756
NA1	14.007	0.276722
NA2	14.007	0.083055
ct	12.011	0.35
ft	18.997999	-0.16
nt	14.007	-0.66
ot	15.999	-0.53
st	32.060001	1.02
Cl	35.452999	-1

Table A.2: Force field parameters for Coulomb interactions.

A.3 Bond parameters

type i - type j	$k_B(kcal/mol/\text{Å}^2)$	b_0(Å)
CA-CS1	280.159	1.534
CA-HA	357.723	1.092
CA-NA2	287.313	1.478
CM-CM1	276.101	1.532
CM-CM2	276.101	1.532
CM-HM	363.651	1.083
CM-NA1	274.189	1.486
CM1-HM1	348.875	1.095
CM2-HM2	348.875	1.095
CR-HR	401.204	1.074
CR-NA1	464.417	1.385
CR-NA2	552.388	1.335
CS1-HS1	347.494	1.091
CS1-CS2	281.515	1.536
CS2-CT	287.616	1.532
CS2-HS2	349.1	1.095
CT-HT	351.78	1.093
CW1-CW2	531.524	1.352
CW1-HW1	396.395	1.081
CW1-NA1	410.333	1.386
CW2-HW2	395.457	1.079
CW2-NA2	408.436	1.384
ct-ft	441.92	1.323
ct-st	233.03	1.818
nt-st	374.88	1.57
ot-st	637.07	1.437

Table A.3: Force field parameters for bond interactions.

A.4 Angle parameters

type i - type j - type k	$k_\theta(kcal/mol/rad^2)$	$\theta_0(degrees)$
CA-CS1-CS2	80.939	112.443
CA-CS1-HS1	67.208	109.146
CA-NA2-CR	84.594	126.178
CA-NA2-CW2	88.387	126.334
CM-CM1-CM	93.377	116.755
CM-CM1-HM1	65.863	109.352
CM-CM2-HM2	65.863	109.352
CM-NA1-CR	90.685	126.384
CM-NA1-CW1	97.374	125.886
CM1-CM-CM1	86.605	113.354
CM1-CM-CM2	86.605	113.354
CM1-CM-HM	62.36	109.018
CM1-CM-NA1	100.742	109.948
CM2-CM-HM	62.36	109.018
CM2-CM-NA1	100.742	109.948
CR-NA1-CW1	266.725	107.813
CR-NA2-CW2	97.245	108.165
CS1-CA-HA	63.034	111.966
CS1-CA-NA2	78.829	112.124
CS1-CS2-CT	80.4	114.212
CS1-CS2-HS2	59.461	107.908
CS2-CS1-HS1	55.219	108.941
CS2-CT-HT	63.976	111.352
CT-CS2-HS2	58.884	109.477
CW1-CW2-HW2	64.721	130.027
CW1-CW2-NA2	268.303	107.291
CW2-CW1-HW1	58.627	130.162
CW2-CW1-NA1	255.171	106.709
HA-CA-HA	63.013	107.853
HA-CA-NA2	71.809	106.307
HM-CM-NA1	81.775	104.182

Table A.4: Force field parameters for angle interactions.

type i - type j - type k	$k_\theta(kcal/mol/rad^2)$	$\theta_0(degrees)$
HM1-CM1-HM1	54.233	108.068
HM2-CM2-HM2	54.233	108.068
HR-CR-NA1	64.617	125.669
HR-CR-NA2	66.082	125.132
HS1-CS1-HS1	56.563	107.458
HS2-CS2-HS2	82.095	106.005
HT-CT-HT	68.335	108.358
HW1-CW1-NA1	68.306	122.79
HW2-CW2-NA2	61.818	122.991
NA1-CR-NA2	252.744	109.164
ct-st-nt	91.3	103.5
ct-st-ot	103.97	102.6
ft-ct-ft	93.33	107.1
ft-ct-st	82.93	111.7
nt-st-ot	94.29	113.6
ot-st-ot	115.8	118.5
st-nt-st	80.19	125.6

Table A.5: Force field parameters for angle interactions. (continued)

A.5 Dihedral and improper dihedral parameters

type *i* - type *j* - type *k* - type *l*	K1	K2	K3	K4
CA-CS1-CS2-CT	1.3	-0.05	0.2	0
CA-CS1-CS2-HS2	0	0	0.366	0
CM1-CM-CM1-CM	-0.775803	0.313249	4.69238	0.31007
CM1-CM-CM1-HM1	0	0	2.67777	0
CM1-CM-CM2-HM2	0	0	2.67777	0
CM1-CM-NA1-CR	2.45324	-0.559275	-0.813571	1.52831
CM1-CM-NA1-CW1	0	0	2.37904	0
CM2-CM-CM1-CM	-0.775803	0.313249	4.69238	0.31007
CM2-CM-CM1-HM1	0	0	2.67777	0
CM2-CM-NA1-CR	2.45324	-0.559275	-0.813571	1.52831
CM2-CM-NA1-CW1	0	0	2.37904	0
CS1-CA-NA2-CR	-1.659	-0.555	-0.375	0
CS1-CA-NA2-CW2s	-1.91	-1.5	0.29	0
CS1-CS2-CT-HT	0	0	0.366	0
CW1-CW2-NA2-CA	0	3	0	0
CW1-CW2-NA2-CR	0	3	0	0
CW2-CW1-NA1-CM	0	3	0	0
CW2-CW1-NA1-CR	0	3	0	0
HA-CA-CS1-CS2	0	0	0.366	0
HA-CA-CS1-HS1	0	0	0.318	0
HA-CA-NA2-CR	0	0	0	0
HA-CA-NA2-CW2	-1.4	-2.65	0.175	0
HM-CM-CM1-CM	0	0	2.67777	0
HM-CM-CM1-HM1	0	0	2.78174	0
HM-CM-CM2-HM2	0	0	2.78174	0
HM-CM-NA1-CR	0	0	0	0
HM-CM-NA1-CW1	0	0	0.124	0
HR-CR-NA1-CM	0	4.651	0	0
HR-CR-NA1-CW1	0	4.651	0	0
HR-CR-NA2-CA	0	4.651	0	0

Table A.6: Force field parameters for dihedral interactions.

type *i* - type *j* - type *k* - type *l*	K1	K2	K3	K4
HR-CR-NA2-CW2	0	4.651	0	0
HS1-CS1-CS2-CT	0	0	0.366	0
HS1-CS1-CS2-HS2	0	0	0.318	0
HS2-CS2-CT-HT	0	0	0.318	0
HW1-CW1-CW2-HW2	0	10.75	0	0
HW1-CW1-CW2-NA2	0	10.75	0	0
HW1-CW1-NA1-CM	0	3	0	0
HW1-CW1-NA1-CR	0	3	0	0
HW2-CW2-NA2-CA	0	3	0	0
HW2-CW2-NA2-CR	0	3	0	0
NA1-CM-CM1-CM	0	0	4.44212	0
NA1-CM-CM1-HM1	-0.839253	-0.310509	4.68038	0.846371
NA1-CM-CM2-HM2	-0.839253	-0.310509	4.68038	0.846371
NA1-CR-NA2-CA	0	4.651	0	0
NA1-CR-NA2-CW2	0	4.651	0	0
NA1-CW1-CW2-HW2	0	10.75	0	0
NA1-CW1-CW2-NA2	0	10.75	0	0
NA2-CA-CS1-CS2	-1.788	0.756	-0.288	0
NA2-CA-CS1-HS1	0	0	0	0
NA2-CR-NA1-CM	0	4.651	0	0
NA2-CR-NA1-CW1	0	4.651	0	0
Ft-Ct-St-Nt	0	0	0.316	0
Ft-Ct-St-Ot	0	0	0.347	0
St-Nt-St-Ct	7.832	-2.49	-0.764	0
St-Nt-St-Ot	0	0	-0.004	0

Table A.7: Force field parameters for dihedral interactions. (continued)

type *i* - type *j* - type *k* - type *l*	$K(kcal/mol)$	*d*	*n*
CR-HR-NA1-NA2	1	-1	2
CW1-CW2-HW1-NA1	1.1	-1	2
CW2-CW1-HW2-NA2	1.1	-1	2
NA2-CA-CR-CW2	1.1	-1	2

Table A.8: Force field parameters for improper dihedrals.

A.6 Input scripts

Input scripts of PolyBMIM_TFSI_1 system are provided in this section, including setting up of the configuration, energy minimization, equilibrations, and production run. For ANI-2x potential intup, for saving some time, equilibrated structure was taken from OPLS-AA simualtion, so only production run was performed in ASE using ANI-2x.

A.6.1 OPLS-AA input script

```
# PolyBMIM_TFSI simulation for a single small cell: 1 chain – 10 monomers – 10 TFSI anions

units real
boundary p p p
atom_style full

# ------------- Variables definitions -------------- #

variable TEMP equal 300.0
variable TEMP_D equal 1.0
variable PRESS equal 1.0
variable dt equal 1.0
variable dthero equal 100
variable dtdump equal 100
variable tnvt equal 100000.0
variable tnpt equal 100000.0
variable tpro equal 1000000.0
variable Tdamp equal ${dt}*100
```

```
variable  Tdamp_D equal ${dt}*20
variable  Pdamp  equal ${dt}*1000
variable  Nevery equal 1000
variable  Nrepeat equal 10
variable  Nfreq    equal 10000
variable  cutoff    equal 12.0

# equilibration  variables

variable  TEMP_nvt      equal 1000.0
variable  TEMP_npt      equal 600.0
variable  PRESS_npt_1   equal 100.0
variable  PRESS_npt_2   equal 1.0
variable  tnvt              equal 100000.0
variable  tnpt              equal 10000.0

# Interaction  potential  definition

 pair_style     lj /cut/coul/long ${cutoff} ${cutoff}
bond_style        harmonic
angle_style       harmonic
dihedral_style    opls
improper_style   cvff
pair_modify     mix geometric tail  yes

# OPLS considers 1-4 interactions with 50%
special_bonds     lj /coul 0.0  0.0  0.5

neighbor          2.0  bin
neigh_modify    delay 0  every 1  check yes
neigh_modify    one 10000
```

```
kspace_style      pppm 1.0e-4

# Read data from initially generated data file (created by VMD, Topo Tools)

read_data         PolyBMIM_TFSI_1.data

velocity all create ${TEMP} 12345 rot yes dist gaussian

group ATOMS type 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
    27
group PolyBMIM type 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
group TFSI        type 23 24 25 26 27

# OPLS force field parameters defined in data file

# Energy Minimization

min_style         cg
minimize          1e-45 1e-45 10000 10000

write_data        PolyBMIM_TFSI_1_minimized.data

# -------------- Equilibration Stage 1 (NVT dynamics at 1000 K for 0.1 ns)
    --------------- #

print "loop 1 starts ..... "

fix               NVT_ATOMS all nvt temp ${TEMP_nvt} ${TEMP_nvt} ${Tdamp}

thermo_style      custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
    vol fmax
thermo            ${dthero}
```

```
timestep ${dt}

run         ${tnvt}    # 0.1  ns

unfix           NVT_ATOMS

# ------------- Equilibration Stage 2 (NPT dynamics at 600 K 100 bar for 0.1 ns)
    -------------- #

fix         NPT_ATOMS all npt temp ${TEMP_npt} ${TEMP_npt} ${Tdamp} iso
    ${PRESS_npt_1} ${PRESS_npt_1} ${Pdamp}

thermo_style    custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
    vol fmax
thermo          ${dthero}
timestep ${dt}

run 5000
run 5000
run 5000
run 5000
run 5000
run 5000
run 5000
run 5000
run 5000
run 5000
run 5000
run 5000
run 5000
run 5000
run 5000
run 5000
run 5000
```

```
run 5000
run 5000
run 5000
run 5000
run 5000 # 0.1 ns

unfix NPT_ATOMS

# ---------- Equilibration Stage 3 (NPT dynamics at 600 K 1 bar for 0.1 ns)
    ---------- #

fix NPT_ATOMS all npt temp ${TEMP_npt} ${TEMP_npt} ${Tdamp} iso
    ${PRESS_npt_2} ${PRESS_npt_2} ${Pdamp}

thermo_style custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
    vol fmax
thermo ${dthero}
timestep ${dt}


run ${tnpt}
run ${tnpt}
run ${tnpt}
run ${tnpt}
run ${tnpt}
run ${tnpt}
run ${tnpt}
run ${tnpt}
run ${tnpt}
run ${tnpt} # 0.1 ns

unfix NPT_ATOMS
```

```
# ------------- Equilibration Stage 1 (NVT dynamics at 1000 K for 0.1 ns)
    ------------- #

print "loop 2 starts ..... "

fix             NVT_ATOMS all nvt temp ${TEMP_nvt} ${TEMP_nvt} ${Tdamp}

thermo_style    custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
    vol fmax
thermo          ${dthero}
timestep ${dt}

run     ${tnvt}    # 0.1 ns

unfix       NVT_ATOMS

# ------------- Equilibration Stage 2 (NPT dynamics at 600 K 100 bar for 0.1 ns)
    ------------- #

fix     NPT_ATOMS all npt temp ${TEMP_npt} ${TEMP_npt} ${Tdamp} iso
    ${PRESS_npt_1} ${PRESS_npt_1} ${Pdamp}

thermo_style    custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
    vol fmax
thermo          ${dthero}
timestep ${dt}

run     ${tnpt}
run     ${tnpt}
run     ${tnpt}
run     ${tnpt}
```

```
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}    # 0.1 ns

unfix       NPT_ATOMS

# ---------------- Equilibration Stage 3 (NPT dynamics at 600 K 1 bar for 0.1 ns)
      ---------------- #

 fix        NPT_ATOMS all npt temp ${TEMP_npt} ${TEMP_npt} ${Tdamp} iso
      ${PRESS_npt_2} ${PRESS_npt_2} ${Pdamp}

thermo_style     custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
      vol fmax
thermo           ${dthero}
timestep ${dt}



run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}    # 0.1 ns
```

```
unfix      NPT_ATOMS

# ------------- Equilibration Stage 1 (NVT dynamics at 1000 K for 0.1 ns)
      ------------- #

print "loop 3 starts ..... "

fix                NVT_ATOMS all nvt temp ${TEMP_nvt} ${TEMP_nvt} ${Tdamp}

thermo_style    custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
      vol fmax
thermo          ${dthero}
timestep ${dt}

run        ${tnvt}    # 0.1 ns

unfix          NVT_ATOMS

# ------------- Equilibration Stage 2 (NPT dynamics at 600 K 100 bar for 0.1 ns)
      ------------- #

fix        NPT_ATOMS all npt temp ${TEMP_npt} ${TEMP_npt} ${Tdamp} iso
      ${PRESS_npt_1} ${PRESS_npt_1} ${Pdamp}

thermo_style    custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
      vol fmax
thermo          ${dthero}
timestep ${dt}

run        ${tnpt}
run        ${tnpt}
run        ${tnpt}
```

```
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}    # 0.1 ns

unfix       NPT_ATOMS

# ------------- Equilibration Stage 3 (NPT dynamics at 600 K 1 bar for 0.1 ns)
    ------------- #

fix         NPT_ATOMS all npt temp ${TEMP_npt} ${TEMP_npt} ${Tdamp} iso
    ${PRESS_npt_2} ${PRESS_npt_2} ${Pdamp}

thermo_style    custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
    vol fmax
thermo          ${dthero}
timestep ${dt}


run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}    # 0.1 ns
```

```
unfix      NPT_ATOMS

# ------------- Equilibration Stage 1 (NVT dynamics at 1000 K for 0.1 ns)
      ------------- #

print "loop 4 starts ..... "

fix                NVT_ATOMS all nvt temp ${TEMP_nvt} ${TEMP_nvt} ${Tdamp}

thermo_style     custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
      vol fmax
thermo           ${dthero}
timestep ${dt}

run        ${tnvt}    # 0.1 ns

unfix          NVT_ATOMS

# ------------- Equilibration Stage 2 (NPT dynamics at 600 K 100 bar for 0.1 ns)
      ------------- #

fix        NPT_ATOMS all npt temp ${TEMP_npt} ${TEMP_npt} ${Tdamp} iso
      ${PRESS_npt_1} ${PRESS_npt_1} ${Pdamp}

thermo_style     custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
      vol fmax
thermo           ${dthero}
timestep ${dt}

run        ${tnpt}
run        ${tnpt}
```

```
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}    # 0.1 ns

unfix       NPT_ATOMS

# ------------- Equilibration Stage 3 (NPT dynamics at 600 K 1 bar for 0.1 ns)
    -------------- #

fix         NPT_ATOMS all npt temp ${TEMP_npt} ${TEMP_npt} ${Tdamp} iso
    ${PRESS_npt_2} ${PRESS_npt_2} ${Pdamp}

thermo_style    custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
    vol fmax
thermo          ${dthero}
timestep ${dt}


run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
```

```
run         ${tnpt}    # 0.1 ns

unfix       NPT_ATOMS



write_restart       restart .PolyBMIM_Cl_1_equilibrated.npt2
write_data          data.PolyBMIM_Cl_1_equilibrated.npt2

print "production run starts .... "

# ------------- NPT dynamics -------------- #

variable    TEMP_pro equal 600
variable    PRESS_pro equal 1.01325


fix         PRO_ATOMS all npt temp ${TEMP_pro} ${TEMP_pro} ${Tdamp} iso ${PRESS_pro}
      ${PRESS_pro} ${Pdamp}

thermo_style      custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
      vol fmax
thermo            ${dthero}
timestep ${dt}

run         ${tpro}    # 1 ns


# ------------- record the data for the last 1 ns -------------- #

dump              1 ATOMS custom ${dtdump} PolyBMIM_TFSI_1.xyz id type mol x y z vx vy
      vz
```

```
dump     2 ATOMS custom ${dtdump} PolyBMIM_TFSI_1.dump id type mol x y z vx vy vz
dump     3 ATOMS custom ${dtdump} PolyBMIM_TFSI_1.pdb id type mol x y z vx vy vz

# ------------- Calculate RDFs / Rg ------------- #

compute  TFSI_TFSI all rdf  500 23*27 23*27 cutoff  ${cutoff}
compute  PolyBMIM_PolyBMIM all rdf 500 1*22 1*22 cutoff ${cutoff}
compute  TFSI_PolyBMIM all rdf 500 23*27 1*22 cutoff ${cutoff}
compute  PolyBMIM_TFSI all rdf 500 1*22 23*27 cutoff ${cutoff}

compute  PolyBMIM_Rg PolyBMIM gyration
compute      TFSI_msd_1 TFSI msd com yes
compute      TFSI_msd_2 TFSI msd com yes

fix          TFSI_TFSI all  ave/time ${Nevery} ${Nrepeat} ${Nfreq} c_TFSI_TFSI[*] file
    TFSI_TFSI.rdf mode vector
fix          PolyBMIM_PolyBMIM all ave/time ${Nevery} ${Nrepeat} ${Nfreq}
    c_PolyBMIM_PolyBMIM[*] file PolyBMIM_PolyBMIM.rdf mode vector
fix          TFSI_PolyBMIM all ave/time ${Nevery} ${Nrepeat} ${Nfreq} c_TFSI_PolyBMIM[*]
    file TFSI_PolyBMIM.rdf mode vector
fix          PolyBMIM_TFSI all ave/time ${Nevery} ${Nrepeat} ${Nfreq} c_PolyBMIM_TFSI[*]
    file PolyBMIM_TFSI.rdf mode vector

fix          PolyBMIM_Rg_v PolyBMIM ave/time ${Nevery} ${Nrepeat} ${Nfreq}
    c_PolyBMIM_Rg file data.PolyBMIM_Rg_v mode vector
fix          PolyBMIM_Rg_s PolyBMIM ave/time ${Nevery} ${Nrepeat} ${Nfreq}
    c_PolyBMIM_Rg[*] file data.PolyBMIM_Rg_s mode scalar
fix          PolyBMIM_Rg_s_1 PolyBMIM ave/time ${Nevery} ${Nrepeat} ${Nfreq}
    c_PolyBMIM_Rg file data.PolyBMIM_Rg_s_1 mode scalar

fix          TFSI_msd_1 TFSI ave/time ${Nevery} ${Nrepeat} ${Nfreq} c_TFSI_msd_1 file
    data.TFSI_msd_v mode vector
```

```
fix         TFSI_msd_2 TFSI ave/time ${Nevery} ${Nrepeat} ${Nfreq} c_TFSI_msd_2[*] file
    data.TFSI_msd_s mode scalar

run         ${tpro}    # 1 ns

unfix               PRO_ATOMS

write_restart       restart .PolyBMIM_TFSI_1.final
write_data          data.PolyBMIM_TFSI_1.final

print "All done!"
```

A.6.2 Polarizable force field input script

```
# PolyBMIM_TFSI simulation for a single small cell: 1 chain - 10 monomers - 10 TFSI anions

units real
boundary p p p
atom_style full
timestep 1.0

# Interaction potential definition
# we use hybrid pair_style to account for Drude cores and Drude particles

pair_style hybrid/overlay lj/cut/coul/long 12.0 thole 2.600 12.0
bond_style harmonic
angle_style harmonic
dihedral_style opls
improper_style cvff
pair_modify tail yes

kspace_style pppm 1.0e-5

# OPLS considers 1-4 interactions with 50%
special_bonds lj /coul 0.0 0.0 0.5

# Read data from initially generated data file (created by VMD Topo Tools,
    LAMMPS-Drude package)

read_data PolyBMIM_TFSI_1-p.data extra/special/per/atom 6

group ATOMS type 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
    27 # DCs and non-polarizable atoms
```

```
group CORES      type 1 2 3 4 5 6 7 8 9 10 21 22 23 24 25 26 27
                                        # DCs
group DRUDES     type 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
                                    # DPs

group TFSI       type 23 24 25 26 27
                                                        # TFSI anions
group PolyBMIM type 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
                        # polycations
group TFSI_and_dp type 23 24 25 26 27 40 41 42 43 44
                                                # polycations and their Drude particles

neighbor         2.0 bin
neigh_modify     delay 0 every 1 check yes
neigh_modify     one 10000

# ------------ Define pair interactions ------------ #

pair_coeff 1 1 lj/cut/coul/long 0.066 3.500
pair_coeff 1 2 lj/cut/coul/long 0.066 3.500
pair_coeff 1 3 lj/cut/coul/long 0.066 3.500
pair_coeff 1 4 lj/cut/coul/long 0.066 3.500
pair_coeff 1 5 lj/cut/coul/long 0.068 3.525
pair_coeff 1 6 lj/cut/coul/long 0.066 3.500
pair_coeff 1 7 lj/cut/coul/long 0.066 3.500
pair_coeff 1 8 lj/cut/coul/long 0.066 3.500
pair_coeff 1 9 lj/cut/coul/long 0.068 3.525
pair_coeff 1 10 lj/cut/coul/long 0.068 3.525
pair_coeff 1 11 lj/cut/coul/long 0.044 2.710
pair_coeff 1 12 lj/cut/coul/long 0.044 2.710
pair_coeff 1 13 lj/cut/coul/long 0.044 3.000
pair_coeff 1 14 lj/cut/coul/long 0.044 3.000
```

pair_coeff 1 15 lj/cut/coul/long 0.044 2.610

pair_coeff 1 16 lj/cut/coul/long 0.044 3.000

pair_coeff 1 17 lj/cut/coul/long 0.044 3.000

pair_coeff 1 18 lj/cut/coul/long 0.044 3.000

pair_coeff 1 19 lj/cut/coul/long 0.044 2.610

pair_coeff 1 20 lj/cut/coul/long 0.044 2.610

pair_coeff 1 21 lj/cut/coul/long 0.106 3.375

pair_coeff 1 22 lj/cut/coul/long 0.106 3.375

pair_coeff 1 23 lj/cut/coul/long 0.066 3.500

pair_coeff 1 24 lj/cut/coul/long 0.059 3.225

pair_coeff 1 25 lj/cut/coul/long 0.106 3.375

pair_coeff 1 26 lj/cut/coul/long 0.118 3.230

pair_coeff 1 27 lj/cut/coul/long 0.128 3.525

pair_coeff 2 2 lj/cut/coul/long 0.066 3.500

pair_coeff 2 3 lj/cut/coul/long 0.066 3.500

pair_coeff 2 4 lj/cut/coul/long 0.066 3.500

pair_coeff 2 5 lj/cut/coul/long 0.068 3.525

pair_coeff 2 6 lj/cut/coul/long 0.066 3.500

pair_coeff 2 7 lj/cut/coul/long 0.066 3.500

pair_coeff 2 8 lj/cut/coul/long 0.066 3.500

pair_coeff 2 9 lj/cut/coul/long 0.068 3.525

pair_coeff 2 10 lj/cut/coul/long 0.068 3.525

pair_coeff 2 11 lj/cut/coul/long 0.044 2.710

pair_coeff 2 12 lj/cut/coul/long 0.044 2.710

pair_coeff 2 13 lj/cut/coul/long 0.044 3.000

pair_coeff 2 14 lj/cut/coul/long 0.044 3.000

pair_coeff 2 15 lj/cut/coul/long 0.044 2.610

pair_coeff 2 16 lj/cut/coul/long 0.044 3.000

pair_coeff 2 17 lj/cut/coul/long 0.044 3.000

pair_coeff 2 18 lj/cut/coul/long 0.044 3.000

pair_coeff 2 19 lj/cut/coul/long 0.044 2.610

pair_coeff 2 20 lj/cut/coul/long 0.044 2.610

pair_coeff 2 21 lj /cut/coul/long 0.106 3.375

pair_coeff 2 22 lj /cut/coul/long 0.106 3.375

pair_coeff 2 23 lj /cut/coul/long 0.066 3.500

pair_coeff 2 24 lj /cut/coul/long 0.059 3.225

pair_coeff 2 25 lj /cut/coul/long 0.106 3.375

pair_coeff 2 26 lj /cut/coul/long 0.118 3.230

pair_coeff 2 27 lj /cut/coul/long 0.128 3.525

pair_coeff 3 3 lj /cut/coul/long 0.066 3.500

pair_coeff 3 4 lj /cut/coul/long 0.066 3.500

pair_coeff 3 5 lj /cut/coul/long 0.068 3.525

pair_coeff 3 6 lj /cut/coul/long 0.066 3.500

pair_coeff 3 7 lj /cut/coul/long 0.066 3.500

pair_coeff 3 8 lj /cut/coul/long 0.066 3.500

pair_coeff 3 9 lj /cut/coul/long 0.068 3.525

pair_coeff 3 10 lj /cut/coul/long 0.068 3.525

pair_coeff 3 11 lj /cut/coul/long 0.044 2.710

pair_coeff 3 12 lj /cut/coul/long 0.044 2.710

pair_coeff 3 13 lj /cut/coul/long 0.044 3.000

pair_coeff 3 14 lj /cut/coul/long 0.044 3.000

pair_coeff 3 15 lj /cut/coul/long 0.044 2.610

pair_coeff 3 16 lj /cut/coul/long 0.044 3.000

pair_coeff 3 17 lj /cut/coul/long 0.044 3.000

pair_coeff 3 18 lj /cut/coul/long 0.044 3.000

pair_coeff 3 19 lj /cut/coul/long 0.044 2.610

pair_coeff 3 20 lj /cut/coul/long 0.044 2.610

pair_coeff 3 21 lj /cut/coul/long 0.106 3.375

pair_coeff 3 22 lj /cut/coul/long 0.106 3.375

pair_coeff 3 23 lj /cut/coul/long 0.066 3.500

pair_coeff 3 24 lj /cut/coul/long 0.059 3.225

pair_coeff 3 25 lj /cut/coul/long 0.106 3.375

pair_coeff 3 26 lj /cut/coul/long 0.118 3.230

pair_coeff 3 27 lj /cut/coul/long 0.128 3.525

pair_coeff 4 4 lj/cut/coul/long 0.066 3.500

pair_coeff 4 5 lj/cut/coul/long 0.068 3.525

pair_coeff 4 6 lj/cut/coul/long 0.066 3.500

pair_coeff 4 7 lj/cut/coul/long 0.066 3.500

pair_coeff 4 8 lj/cut/coul/long 0.066 3.500

pair_coeff 4 9 lj/cut/coul/long 0.068 3.525

pair_coeff 4 10 lj/cut/coul/long 0.068 3.525

pair_coeff 4 11 lj/cut/coul/long 0.044 2.710

pair_coeff 4 12 lj/cut/coul/long 0.044 2.710

pair_coeff 4 13 lj/cut/coul/long 0.044 3.000

pair_coeff 4 14 lj/cut/coul/long 0.044 3.000

pair_coeff 4 15 lj/cut/coul/long 0.044 2.610

pair_coeff 4 16 lj/cut/coul/long 0.044 3.000

pair_coeff 4 17 lj/cut/coul/long 0.044 3.000

pair_coeff 4 18 lj/cut/coul/long 0.044 3.000

pair_coeff 4 19 lj/cut/coul/long 0.044 2.610

pair_coeff 4 20 lj/cut/coul/long 0.044 2.610

pair_coeff 4 21 lj/cut/coul/long 0.106 3.375

pair_coeff 4 22 lj/cut/coul/long 0.106 3.375

pair_coeff 4 23 lj/cut/coul/long 0.066 3.500

pair_coeff 4 24 lj/cut/coul/long 0.059 3.225

pair_coeff 4 25 lj/cut/coul/long 0.106 3.375

pair_coeff 4 26 lj/cut/coul/long 0.118 3.230

pair_coeff 4 27 lj/cut/coul/long 0.128 3.525

pair_coeff 5 5 lj/cut/coul/long 0.070 3.550

pair_coeff 5 6 lj/cut/coul/long 0.068 3.525

pair_coeff 5 7 lj/cut/coul/long 0.068 3.525

pair_coeff 5 8 lj/cut/coul/long 0.068 3.525

pair_coeff 5 9 lj/cut/coul/long 0.070 3.550

pair_coeff 5 10 lj/cut/coul/long 0.070 3.550

pair_coeff 5 11 lj/cut/coul/long 0.046 2.735

pair_coeff 5 12 lj/cut/coul/long 0.046 2.735

```
pair_coeff  5 13  lj /cut/coul/long 0.046 3.025
pair_coeff  5 14  lj /cut/coul/long 0.046 3.025
pair_coeff  5 15  lj /cut/coul/long 0.046 2.635
pair_coeff  5 16  lj /cut/coul/long 0.046 3.025
pair_coeff  5 17  lj /cut/coul/long 0.046 3.025
pair_coeff  5 18  lj /cut/coul/long 0.046 3.025
pair_coeff  5 19  lj /cut/coul/long 0.046 2.635
pair_coeff  5 20  lj /cut/coul/long 0.046 2.635
pair_coeff  5 21  lj /cut/coul/long 0.109 3.400
pair_coeff  5 22  lj /cut/coul/long 0.109 3.400
pair_coeff  5 23  lj /cut/coul/long 0.068 3.525
pair_coeff  5 24  lj /cut/coul/long 0.061 3.250
pair_coeff  5 25  lj /cut/coul/long 0.109 3.400
pair_coeff  5 26  lj /cut/coul/long 0.121 3.255
pair_coeff  5 27  lj /cut/coul/long 0.132 3.550
pair_coeff  6 6  lj /cut/coul/long 0.066 3.500
pair_coeff  6 7  lj /cut/coul/long 0.066 3.500
pair_coeff  6 8  lj /cut/coul/long 0.066 3.500
pair_coeff  6 9  lj /cut/coul/long 0.068 3.525
pair_coeff  6 10  lj /cut/coul/long 0.068 3.525
pair_coeff  6 11  lj /cut/coul/long 0.044 2.710
pair_coeff  6 12  lj /cut/coul/long 0.044 2.710
pair_coeff  6 13  lj /cut/coul/long 0.044 3.000
pair_coeff  6 14  lj /cut/coul/long 0.044 3.000
pair_coeff  6 15  lj /cut/coul/long 0.044 2.610
pair_coeff  6 16  lj /cut/coul/long 0.044 3.000
pair_coeff  6 17  lj /cut/coul/long 0.044 3.000
pair_coeff  6 18  lj /cut/coul/long 0.044 3.000
pair_coeff  6 19  lj /cut/coul/long 0.044 2.610
pair_coeff  6 20  lj /cut/coul/long 0.044 2.610
pair_coeff  6 21  lj /cut/coul/long 0.106 3.375
pair_coeff  6 22  lj /cut/coul/long 0.106 3.375
```

pair_coeff 6 23 lj/cut/coul/long 0.066 3.500

pair_coeff 6 24 lj/cut/coul/long 0.059 3.225

pair_coeff 6 25 lj/cut/coul/long 0.106 3.375

pair_coeff 6 26 lj/cut/coul/long 0.118 3.230

pair_coeff 6 27 lj/cut/coul/long 0.128 3.525

pair_coeff 7 7 lj/cut/coul/long 0.066 3.500

pair_coeff 7 8 lj/cut/coul/long 0.066 3.500

pair_coeff 7 9 lj/cut/coul/long 0.068 3.525

pair_coeff 7 10 lj/cut/coul/long 0.068 3.525

pair_coeff 7 11 lj/cut/coul/long 0.044 2.710

pair_coeff 7 12 lj/cut/coul/long 0.044 2.710

pair_coeff 7 13 lj/cut/coul/long 0.044 3.000

pair_coeff 7 14 lj/cut/coul/long 0.044 3.000

pair_coeff 7 15 lj/cut/coul/long 0.044 2.610

pair_coeff 7 16 lj/cut/coul/long 0.044 3.000

pair_coeff 7 17 lj/cut/coul/long 0.044 3.000

pair_coeff 7 18 lj/cut/coul/long 0.044 3.000

pair_coeff 7 19 lj/cut/coul/long 0.044 2.610

pair_coeff 7 20 lj/cut/coul/long 0.044 2.610

pair_coeff 7 21 lj/cut/coul/long 0.106 3.375

pair_coeff 7 22 lj/cut/coul/long 0.106 3.375

pair_coeff 7 23 lj/cut/coul/long 0.066 3.500

pair_coeff 7 24 lj/cut/coul/long 0.059 3.225

pair_coeff 7 25 lj/cut/coul/long 0.106 3.375

pair_coeff 7 26 lj/cut/coul/long 0.118 3.230

pair_coeff 7 27 lj/cut/coul/long 0.128 3.525

pair_coeff 8 8 lj/cut/coul/long 0.066 3.500

pair_coeff 8 9 lj/cut/coul/long 0.068 3.525

pair_coeff 8 10 lj/cut/coul/long 0.068 3.525

pair_coeff 8 11 lj/cut/coul/long 0.044 2.710

pair_coeff 8 12 lj/cut/coul/long 0.044 2.710

pair_coeff 8 13 lj/cut/coul/long 0.044 3.000

pair_coeff 8 14 lj/cut/coul/long 0.044 3.000
pair_coeff 8 15 lj/cut/coul/long 0.044 2.610
pair_coeff 8 16 lj/cut/coul/long 0.044 3.000
pair_coeff 8 17 lj/cut/coul/long 0.044 3.000
pair_coeff 8 18 lj/cut/coul/long 0.044 3.000
pair_coeff 8 19 lj/cut/coul/long 0.044 2.610
pair_coeff 8 20 lj/cut/coul/long 0.044 2.610
pair_coeff 8 21 lj/cut/coul/long 0.106 3.375
pair_coeff 8 22 lj/cut/coul/long 0.106 3.375
pair_coeff 8 23 lj/cut/coul/long 0.066 3.500
pair_coeff 8 24 lj/cut/coul/long 0.059 3.225
pair_coeff 8 25 lj/cut/coul/long 0.106 3.375
pair_coeff 8 26 lj/cut/coul/long 0.118 3.230
pair_coeff 8 27 lj/cut/coul/long 0.128 3.525
pair_coeff 9 9 lj/cut/coul/long 0.070 3.550
pair_coeff 9 10 lj/cut/coul/long 0.070 3.550
pair_coeff 9 11 lj/cut/coul/long 0.046 2.735
pair_coeff 9 12 lj/cut/coul/long 0.046 2.735
pair_coeff 9 13 lj/cut/coul/long 0.046 3.025
pair_coeff 9 14 lj/cut/coul/long 0.046 3.025
pair_coeff 9 15 lj/cut/coul/long 0.046 2.635
pair_coeff 9 16 lj/cut/coul/long 0.046 3.025
pair_coeff 9 17 lj/cut/coul/long 0.046 3.025
pair_coeff 9 18 lj/cut/coul/long 0.046 3.025
pair_coeff 9 19 lj/cut/coul/long 0.046 2.635
pair_coeff 9 20 lj/cut/coul/long 0.046 2.635
pair_coeff 9 21 lj/cut/coul/long 0.109 3.400
pair_coeff 9 22 lj/cut/coul/long 0.109 3.400
pair_coeff 9 23 lj/cut/coul/long 0.068 3.525
pair_coeff 9 24 lj/cut/coul/long 0.061 3.250
pair_coeff 9 25 lj/cut/coul/long 0.109 3.400
pair_coeff 9 26 lj/cut/coul/long 0.121 3.255

pair_coeff 9 27 lj /cut/coul/long 0.132 3.550

pair_coeff 10 10 lj /cut/coul/long 0.070 3.550

pair_coeff 10 11 lj /cut/coul/long 0.046 2.735

pair_coeff 10 12 lj /cut/coul/long 0.046 2.735

pair_coeff 10 13 lj /cut/coul/long 0.046 3.025

pair_coeff 10 14 lj /cut/coul/long 0.046 3.025

pair_coeff 10 15 lj /cut/coul/long 0.046 2.635

pair_coeff 10 16 lj /cut/coul/long 0.046 3.025

pair_coeff 10 17 lj /cut/coul/long 0.046 3.025

pair_coeff 10 18 lj /cut/coul/long 0.046 3.025

pair_coeff 10 19 lj /cut/coul/long 0.046 2.635

pair_coeff 10 20 lj /cut/coul/long 0.046 2.635

pair_coeff 10 21 lj /cut/coul/long 0.109 3.400

pair_coeff 10 22 lj /cut/coul/long 0.109 3.400

pair_coeff 10 23 lj /cut/coul/long 0.068 3.525

pair_coeff 10 24 lj /cut/coul/long 0.061 3.250

pair_coeff 10 25 lj /cut/coul/long 0.109 3.400

pair_coeff 10 26 lj /cut/coul/long 0.121 3.255

pair_coeff 10 27 lj /cut/coul/long 0.132 3.550

pair_coeff 11 11 lj /cut/coul/long 0.030 1.920

pair_coeff 11 12 lj /cut/coul/long 0.030 1.920

pair_coeff 11 13 lj /cut/coul/long 0.030 2.210

pair_coeff 11 14 lj /cut/coul/long 0.030 2.210

pair_coeff 11 15 lj /cut/coul/long 0.030 1.820

pair_coeff 11 16 lj /cut/coul/long 0.030 2.210

pair_coeff 11 17 lj /cut/coul/long 0.030 2.210

pair_coeff 11 18 lj /cut/coul/long 0.030 2.210

pair_coeff 11 19 lj /cut/coul/long 0.030 1.820

pair_coeff 11 20 lj /cut/coul/long 0.030 1.820

pair_coeff 11 21 lj /cut/coul/long 0.071 2.585

pair_coeff 11 22 lj /cut/coul/long 0.071 2.585

pair_coeff 11 23 lj /cut/coul/long 0.044 2.710

pair_coeff 11 24 lj/cut/coul/long 0.040 2.435
pair_coeff 11 25 lj/cut/coul/long 0.071 2.585
pair_coeff 11 26 lj/cut/coul/long 0.079 2.440
pair_coeff 11 27 lj/cut/coul/long 0.087 2.735
pair_coeff 12 12 lj/cut/coul/long 0.030 1.920
pair_coeff 12 13 lj/cut/coul/long 0.030 2.210
pair_coeff 12 14 lj/cut/coul/long 0.030 2.210
pair_coeff 12 15 lj/cut/coul/long 0.030 1.820
pair_coeff 12 16 lj/cut/coul/long 0.030 2.210
pair_coeff 12 17 lj/cut/coul/long 0.030 2.210
pair_coeff 12 18 lj/cut/coul/long 0.030 2.210
pair_coeff 12 19 lj/cut/coul/long 0.030 1.820
pair_coeff 12 20 lj/cut/coul/long 0.030 1.820
pair_coeff 12 21 lj/cut/coul/long 0.071 2.585
pair_coeff 12 22 lj/cut/coul/long 0.071 2.585
pair_coeff 12 23 lj/cut/coul/long 0.044 2.710
pair_coeff 12 24 lj/cut/coul/long 0.040 2.435
pair_coeff 12 25 lj/cut/coul/long 0.071 2.585
pair_coeff 12 26 lj/cut/coul/long 0.079 2.440
pair_coeff 12 27 lj/cut/coul/long 0.087 2.735
pair_coeff 13 13 lj/cut/coul/long 0.030 2.500
pair_coeff 13 14 lj/cut/coul/long 0.030 2.500
pair_coeff 13 15 lj/cut/coul/long 0.030 2.110
pair_coeff 13 16 lj/cut/coul/long 0.030 2.500
pair_coeff 13 17 lj/cut/coul/long 0.030 2.500
pair_coeff 13 18 lj/cut/coul/long 0.030 2.500
pair_coeff 13 19 lj/cut/coul/long 0.030 2.110
pair_coeff 13 20 lj/cut/coul/long 0.030 2.110
pair_coeff 13 21 lj/cut/coul/long 0.071 2.875
pair_coeff 13 22 lj/cut/coul/long 0.071 2.875
pair_coeff 13 23 lj/cut/coul/long 0.044 3.000
pair_coeff 13 24 lj/cut/coul/long 0.040 2.725

pair_coeff 13 25 lj/cut/coul/long 0.071 2.875
pair_coeff 13 26 lj/cut/coul/long 0.079 2.730
pair_coeff 13 27 lj/cut/coul/long 0.087 3.025
pair_coeff 14 14 lj/cut/coul/long 0.030 2.500
pair_coeff 14 15 lj/cut/coul/long 0.030 2.110
pair_coeff 14 16 lj/cut/coul/long 0.030 2.500
pair_coeff 14 17 lj/cut/coul/long 0.030 2.500
pair_coeff 14 18 lj/cut/coul/long 0.030 2.500
pair_coeff 14 19 lj/cut/coul/long 0.030 2.110
pair_coeff 14 20 lj/cut/coul/long 0.030 2.110
pair_coeff 14 21 lj/cut/coul/long 0.071 2.875
pair_coeff 14 22 lj/cut/coul/long 0.071 2.875
pair_coeff 14 23 lj/cut/coul/long 0.044 3.000
pair_coeff 14 24 lj/cut/coul/long 0.040 2.725
pair_coeff 14 25 lj/cut/coul/long 0.071 2.875
pair_coeff 14 26 lj/cut/coul/long 0.079 2.730
pair_coeff 14 27 lj/cut/coul/long 0.087 3.025
pair_coeff 15 15 lj/cut/coul/long 0.030 1.720
pair_coeff 15 16 lj/cut/coul/long 0.030 2.110
pair_coeff 15 17 lj/cut/coul/long 0.030 2.110
pair_coeff 15 18 lj/cut/coul/long 0.030 2.110
pair_coeff 15 19 lj/cut/coul/long 0.030 1.720
pair_coeff 15 20 lj/cut/coul/long 0.030 1.720
pair_coeff 15 21 lj/cut/coul/long 0.071 2.485
pair_coeff 15 22 lj/cut/coul/long 0.071 2.485
pair_coeff 15 23 lj/cut/coul/long 0.044 2.610
pair_coeff 15 24 lj/cut/coul/long 0.040 2.335
pair_coeff 15 25 lj/cut/coul/long 0.071 2.485
pair_coeff 15 26 lj/cut/coul/long 0.079 2.340
pair_coeff 15 27 lj/cut/coul/long 0.087 2.635
pair_coeff 16 16 lj/cut/coul/long 0.030 2.500
pair_coeff 16 17 lj/cut/coul/long 0.030 2.500

pair_coeff 16 18 lj/cut/coul/long 0.030 2.500
pair_coeff 16 19 lj/cut/coul/long 0.030 2.110
pair_coeff 16 20 lj/cut/coul/long 0.030 2.110
pair_coeff 16 21 lj/cut/coul/long 0.071 2.875
pair_coeff 16 22 lj/cut/coul/long 0.071 2.875
pair_coeff 16 23 lj/cut/coul/long 0.044 3.000
pair_coeff 16 24 lj/cut/coul/long 0.040 2.725
pair_coeff 16 25 lj/cut/coul/long 0.071 2.875
pair_coeff 16 26 lj/cut/coul/long 0.079 2.730
pair_coeff 16 27 lj/cut/coul/long 0.087 3.025
pair_coeff 17 17 lj/cut/coul/long 0.030 2.500
pair_coeff 17 18 lj/cut/coul/long 0.030 2.500
pair_coeff 17 19 lj/cut/coul/long 0.030 2.110
pair_coeff 17 20 lj/cut/coul/long 0.030 2.110
pair_coeff 17 21 lj/cut/coul/long 0.071 2.875
pair_coeff 17 22 lj/cut/coul/long 0.071 2.875
pair_coeff 17 23 lj/cut/coul/long 0.044 3.000
pair_coeff 17 24 lj/cut/coul/long 0.040 2.725
pair_coeff 17 25 lj/cut/coul/long 0.071 2.875
pair_coeff 17 26 lj/cut/coul/long 0.079 2.730
pair_coeff 17 27 lj/cut/coul/long 0.087 3.025
pair_coeff 18 18 lj/cut/coul/long 0.030 2.500
pair_coeff 18 19 lj/cut/coul/long 0.030 2.110
pair_coeff 18 20 lj/cut/coul/long 0.030 2.110
pair_coeff 18 21 lj/cut/coul/long 0.071 2.875
pair_coeff 18 22 lj/cut/coul/long 0.071 2.875
pair_coeff 18 23 lj/cut/coul/long 0.044 3.000
pair_coeff 18 24 lj/cut/coul/long 0.040 2.725
pair_coeff 18 25 lj/cut/coul/long 0.071 2.875
pair_coeff 18 26 lj/cut/coul/long 0.079 2.730
pair_coeff 18 27 lj/cut/coul/long 0.087 3.025
pair_coeff 19 19 lj/cut/coul/long 0.030 1.720

pair_coeff 19 20 lj/cut/coul/long 0.030 1.720
pair_coeff 19 21 lj/cut/coul/long 0.071 2.485
pair_coeff 19 22 lj/cut/coul/long 0.071 2.485
pair_coeff 19 23 lj/cut/coul/long 0.044 2.610
pair_coeff 19 24 lj/cut/coul/long 0.040 2.335
pair_coeff 19 25 lj/cut/coul/long 0.071 2.485
pair_coeff 19 26 lj/cut/coul/long 0.079 2.340
pair_coeff 19 27 lj/cut/coul/long 0.087 2.635
pair_coeff 20 20 lj/cut/coul/long 0.030 1.720
pair_coeff 20 21 lj/cut/coul/long 0.071 2.485
pair_coeff 20 22 lj/cut/coul/long 0.071 2.485
pair_coeff 20 23 lj/cut/coul/long 0.044 2.610
pair_coeff 20 24 lj/cut/coul/long 0.040 2.335
pair_coeff 20 25 lj/cut/coul/long 0.071 2.485
pair_coeff 20 26 lj/cut/coul/long 0.079 2.340
pair_coeff 20 27 lj/cut/coul/long 0.087 2.635
pair_coeff 21 21 lj/cut/coul/long 0.170 3.250
pair_coeff 21 22 lj/cut/coul/long 0.170 3.250
pair_coeff 21 23 lj/cut/coul/long 0.106 3.375
pair_coeff 21 24 lj/cut/coul/long 0.095 3.100
pair_coeff 21 25 lj/cut/coul/long 0.170 3.250
pair_coeff 21 26 lj/cut/coul/long 0.189 3.105
pair_coeff 21 27 lj/cut/coul/long 0.206 3.400
pair_coeff 22 22 lj/cut/coul/long 0.170 3.250
pair_coeff 22 23 lj/cut/coul/long 0.106 3.375
pair_coeff 22 24 lj/cut/coul/long 0.095 3.100
pair_coeff 22 25 lj/cut/coul/long 0.170 3.250
pair_coeff 22 26 lj/cut/coul/long 0.189 3.105
pair_coeff 22 27 lj/cut/coul/long 0.206 3.400
pair_coeff 23 23 lj/cut/coul/long 0.066 3.500
pair_coeff 23 24 lj/cut/coul/long 0.059 3.225
pair_coeff 23 25 lj/cut/coul/long 0.106 3.375

```
pair_coeff  23 26  lj /cut/coul/long 0.118 3.230
pair_coeff  23 27  lj /cut/coul/long 0.128 3.525
pair_coeff  24 24  lj /cut/coul/long 0.053 2.950
pair_coeff  24 25  lj /cut/coul/long 0.095 3.100
pair_coeff  24 26  lj /cut/coul/long 0.105 2.955
pair_coeff  24 27  lj /cut/coul/long 0.115 3.250
pair_coeff  25 25  lj /cut/coul/long 0.170 3.250
pair_coeff  25 26  lj /cut/coul/long 0.189 3.105
pair_coeff  25 27  lj /cut/coul/long 0.206 3.400
pair_coeff  26 26  lj /cut/coul/long 0.210 2.960
pair_coeff  26 27  lj /cut/coul/long 0.229 3.255
pair_coeff  27 27  lj /cut/coul/long 0.250 3.550

pair_coeff  * 28*44  lj /cut/coul/long 0.000 0.000 # No LJ for  Drudes

# Thole damping if more than 1 Drude per molecule
pair_coeff      1     1 thole    1.016
pair_coeff      1     2 thole    1.016
pair_coeff      1     3 thole    1.016
pair_coeff      1     4 thole    1.016
pair_coeff      1     5 thole    1.068
pair_coeff      1     6 thole    1.016
pair_coeff      1     7 thole    1.016
pair_coeff      1     8 thole    1.016
pair_coeff      1     9 thole    1.068
pair_coeff      1    10 thole    1.068
pair_coeff      1    21 thole    1.108
pair_coeff      1    22 thole    1.108
pair_coeff      1    23 thole    1.033
pair_coeff      1    24 thole    0.781
pair_coeff      1    25 thole    1.214
pair_coeff      1    26 thole    1.175
```

```
pair_coeff     1    27 thole    0.713
pair_coeff     1    28 thole    1.016
pair_coeff     1    29 thole    1.016
pair_coeff     1    30 thole    1.016
pair_coeff     1    31 thole    1.016
pair_coeff     1    32 thole    1.068
pair_coeff     1    33 thole    1.016
pair_coeff     1    34 thole    1.016
pair_coeff     1    35 thole    1.016
pair_coeff     1    36 thole    1.068
pair_coeff     1    37 thole    1.068
pair_coeff     1    38 thole    1.108
pair_coeff     1    39 thole    1.108
pair_coeff     1    40 thole    1.033
pair_coeff     1    41 thole    0.781
pair_coeff     1    42 thole    1.214
pair_coeff     1    43 thole    1.175
pair_coeff     1    44 thole    0.713
pair_coeff     2     2 thole    1.016
pair_coeff     2     3 thole    1.016
pair_coeff     2     4 thole    1.016
pair_coeff     2     5 thole    1.068
pair_coeff     2     6 thole    1.016
pair_coeff     2     7 thole    1.016
pair_coeff     2     8 thole    1.016
pair_coeff     2     9 thole    1.068
pair_coeff     2    10 thole    1.068
pair_coeff     2    21 thole    1.108
pair_coeff     2    22 thole    1.108
pair_coeff     2    23 thole    1.033
pair_coeff     2    24 thole    0.781
pair_coeff     2    25 thole    1.214
```

```
pair_coeff      2    26 thole    1.175
pair_coeff      2    27 thole    0.713
pair_coeff      2    28 thole    1.016
pair_coeff      2    29 thole    1.016
pair_coeff      2    30 thole    1.016
pair_coeff      2    31 thole    1.016
pair_coeff      2    32 thole    1.068
pair_coeff      2    33 thole    1.016
pair_coeff      2    34 thole    1.016
pair_coeff      2    35 thole    1.016
pair_coeff      2    36 thole    1.068
pair_coeff      2    37 thole    1.068
pair_coeff      2    38 thole    1.108
pair_coeff      2    39 thole    1.108
pair_coeff      2    40 thole    1.033
pair_coeff      2    41 thole    0.781
pair_coeff      2    42 thole    1.214
pair_coeff      2    43 thole    1.175
pair_coeff      2    44 thole    0.713
pair_coeff      3     3 thole    1.016
pair_coeff      3     4 thole    1.016
pair_coeff      3     5 thole    1.068
pair_coeff      3     6 thole    1.016
pair_coeff      3     7 thole    1.016
pair_coeff      3     8 thole    1.016
pair_coeff      3     9 thole    1.068
pair_coeff      3    10 thole    1.068
pair_coeff      3    21 thole    1.108
pair_coeff      3    22 thole    1.108
pair_coeff      3    23 thole    1.033
pair_coeff      3    24 thole    0.781
pair_coeff      3    25 thole    1.214
```

```
pair_coeff     3    26 thole    1.175
pair_coeff     3    27 thole    0.713
pair_coeff     3    28 thole    1.016
pair_coeff     3    29 thole    1.016
pair_coeff     3    30 thole    1.016
pair_coeff     3    31 thole    1.016
pair_coeff     3    32 thole    1.068
pair_coeff     3    33 thole    1.016
pair_coeff     3    34 thole    1.016
pair_coeff     3    35 thole    1.016
pair_coeff     3    36 thole    1.068
pair_coeff     3    37 thole    1.068
pair_coeff     3    38 thole    1.108
pair_coeff     3    39 thole    1.108
pair_coeff     3    40 thole    1.033
pair_coeff     3    41 thole    0.781
pair_coeff     3    42 thole    1.214
pair_coeff     3    43 thole    1.175
pair_coeff     3    44 thole    0.713
pair_coeff     4     4 thole    1.016
pair_coeff     4     5 thole    1.068
pair_coeff     4     6 thole    1.016
pair_coeff     4     7 thole    1.016
pair_coeff     4     8 thole    1.016
pair_coeff     4     9 thole    1.068
pair_coeff     4    10 thole    1.068
pair_coeff     4    21 thole    1.108
pair_coeff     4    22 thole    1.108
pair_coeff     4    23 thole    1.033
pair_coeff     4    24 thole    0.781
pair_coeff     4    25 thole    1.214
pair_coeff     4    26 thole    1.175
```

```
pair_coeff     4   27 thole    0.713
pair_coeff     4   28 thole    1.016
pair_coeff     4   29 thole    1.016
pair_coeff     4   30 thole    1.016
pair_coeff     4   31 thole    1.016
pair_coeff     4   32 thole    1.068
pair_coeff     4   33 thole    1.016
pair_coeff     4   34 thole    1.016
pair_coeff     4   35 thole    1.016
pair_coeff     4   36 thole    1.068
pair_coeff     4   37 thole    1.068
pair_coeff     4   38 thole    1.108
pair_coeff     4   39 thole    1.108
pair_coeff     4   40 thole    1.033
pair_coeff     4   41 thole    0.781
pair_coeff     4   42 thole    1.214
pair_coeff     4   43 thole    1.175
pair_coeff     4   44 thole    0.713
pair_coeff     5    5 thole    1.122
pair_coeff     5    6 thole    1.068
pair_coeff     5    7 thole    1.068
pair_coeff     5    8 thole    1.068
pair_coeff     5    9 thole    1.122
pair_coeff     5   10 thole    1.122
pair_coeff     5   21 thole    1.164
pair_coeff     5   22 thole    1.164
pair_coeff     5   23 thole    1.085
pair_coeff     5   24 thole    0.820
pair_coeff     5   25 thole    1.275
pair_coeff     5   26 thole    1.235
pair_coeff     5   27 thole    0.749
pair_coeff     5   28 thole    1.068
```

```
pair_coeff     5   29 thole    1.068
pair_coeff     5   30 thole    1.068
pair_coeff     5   31 thole    1.068
pair_coeff     5   32 thole    1.122
pair_coeff     5   33 thole    1.068
pair_coeff     5   34 thole    1.068
pair_coeff     5   35 thole    1.068
pair_coeff     5   36 thole    1.122
pair_coeff     5   37 thole    1.122
pair_coeff     5   38 thole    1.164
pair_coeff     5   39 thole    1.164
pair_coeff     5   40 thole    1.085
pair_coeff     5   41 thole    0.820
pair_coeff     5   42 thole    1.275
pair_coeff     5   43 thole    1.235
pair_coeff     5   44 thole    0.749
pair_coeff     6    6 thole    1.016
pair_coeff     6    7 thole    1.016
pair_coeff     6    8 thole    1.016
pair_coeff     6    9 thole    1.068
pair_coeff     6   10 thole    1.068
pair_coeff     6   21 thole    1.108
pair_coeff     6   22 thole    1.108
pair_coeff     6   23 thole    1.033
pair_coeff     6   24 thole    0.781
pair_coeff     6   25 thole    1.214
pair_coeff     6   26 thole    1.175
pair_coeff     6   27 thole    0.713
pair_coeff     6   28 thole    1.016
pair_coeff     6   29 thole    1.016
pair_coeff     6   30 thole    1.016
pair_coeff     6   31 thole    1.016
```

```
pair_coeff     6    32 thole    1.068
pair_coeff     6    33 thole    1.016
pair_coeff     6    34 thole    1.016
pair_coeff     6    35 thole    1.016
pair_coeff     6    36 thole    1.068
pair_coeff     6    37 thole    1.068
pair_coeff     6    38 thole    1.108
pair_coeff     6    39 thole    1.108
pair_coeff     6    40 thole    1.033
pair_coeff     6    41 thole    0.781
pair_coeff     6    42 thole    1.214
pair_coeff     6    43 thole    1.175
pair_coeff     6    44 thole    0.713
pair_coeff     7     7 thole    1.016
pair_coeff     7     8 thole    1.016
pair_coeff     7     9 thole    1.068
pair_coeff     7    10 thole    1.068
pair_coeff     7    21 thole    1.108
pair_coeff     7    22 thole    1.108
pair_coeff     7    23 thole    1.033
pair_coeff     7    24 thole    0.781
pair_coeff     7    25 thole    1.214
pair_coeff     7    26 thole    1.175
pair_coeff     7    27 thole    0.713
pair_coeff     7    28 thole    1.016
pair_coeff     7    29 thole    1.016
pair_coeff     7    30 thole    1.016
pair_coeff     7    31 thole    1.016
pair_coeff     7    32 thole    1.068
pair_coeff     7    33 thole    1.016
pair_coeff     7    34 thole    1.016
pair_coeff     7    35 thole    1.016
```

```
pair_coeff     7    36 thole    1.068
pair_coeff     7    37 thole    1.068
pair_coeff     7    38 thole    1.108
pair_coeff     7    39 thole    1.108
pair_coeff     7    40 thole    1.033
pair_coeff     7    41 thole    0.781
pair_coeff     7    42 thole    1.214
pair_coeff     7    43 thole    1.175
pair_coeff     7    44 thole    0.713
pair_coeff     8     8 thole    1.016
pair_coeff     8     9 thole    1.068
pair_coeff     8    10 thole    1.068
pair_coeff     8    21 thole    1.108
pair_coeff     8    22 thole    1.108
pair_coeff     8    23 thole    1.033
pair_coeff     8    24 thole    0.781
pair_coeff     8    25 thole    1.214
pair_coeff     8    26 thole    1.175
pair_coeff     8    27 thole    0.713
pair_coeff     8    28 thole    1.016
pair_coeff     8    29 thole    1.016
pair_coeff     8    30 thole    1.016
pair_coeff     8    31 thole    1.016
pair_coeff     8    32 thole    1.068
pair_coeff     8    33 thole    1.016
pair_coeff     8    34 thole    1.016
pair_coeff     8    35 thole    1.016
pair_coeff     8    36 thole    1.068
pair_coeff     8    37 thole    1.068
pair_coeff     8    38 thole    1.108
pair_coeff     8    39 thole    1.108
pair_coeff     8    40 thole    1.033
```

```
pair_coeff     8    41 thole    0.781
pair_coeff     8    42 thole    1.214
pair_coeff     8    43 thole    1.175
pair_coeff     8    44 thole    0.713
pair_coeff     9     9 thole    1.122
pair_coeff     9    10 thole    1.122
pair_coeff     9    21 thole    1.164
pair_coeff     9    22 thole    1.164
pair_coeff     9    23 thole    1.085
pair_coeff     9    24 thole    0.820
pair_coeff     9    25 thole    1.275
pair_coeff     9    26 thole    1.235
pair_coeff     9    27 thole    0.749
pair_coeff     9    28 thole    1.068
pair_coeff     9    29 thole    1.068
pair_coeff     9    30 thole    1.068
pair_coeff     9    31 thole    1.068
pair_coeff     9    32 thole    1.122
pair_coeff     9    33 thole    1.068
pair_coeff     9    34 thole    1.068
pair_coeff     9    35 thole    1.068
pair_coeff     9    36 thole    1.122
pair_coeff     9    37 thole    1.122
pair_coeff     9    38 thole    1.164
pair_coeff     9    39 thole    1.164
pair_coeff     9    40 thole    1.085
pair_coeff     9    41 thole    0.820
pair_coeff     9    42 thole    1.275
pair_coeff     9    43 thole    1.235
pair_coeff     9    44 thole    0.749
pair_coeff    10    10 thole    1.122
pair_coeff    10    21 thole    1.164
```

```
pair_coeff    10    22 thole    1.164
pair_coeff    10    23 thole    1.085
pair_coeff    10    24 thole    0.820
pair_coeff    10    25 thole    1.275
pair_coeff    10    26 thole    1.235
pair_coeff    10    27 thole    0.749
pair_coeff    10    28 thole    1.068
pair_coeff    10    29 thole    1.068
pair_coeff    10    30 thole    1.068
pair_coeff    10    31 thole    1.068
pair_coeff    10    32 thole    1.122
pair_coeff    10    33 thole    1.068
pair_coeff    10    34 thole    1.068
pair_coeff    10    35 thole    1.068
pair_coeff    10    36 thole    1.122
pair_coeff    10    37 thole    1.122
pair_coeff    10    38 thole    1.164
pair_coeff    10    39 thole    1.164
pair_coeff    10    40 thole    1.085
pair_coeff    10    41 thole    0.820
pair_coeff    10    42 thole    1.275
pair_coeff    10    43 thole    1.235
pair_coeff    10    44 thole    0.749
pair_coeff    21    21 thole    1.208
pair_coeff    21    22 thole    1.208
pair_coeff    21    23 thole    1.126
pair_coeff    21    24 thole    0.851
pair_coeff    21    25 thole    1.323
pair_coeff    21    26 thole    1.282
pair_coeff    21    27 thole    0.777
pair_coeff    21    28 thole    1.108
pair_coeff    21    29 thole    1.108
```

pair_coeff 21 30 thole 1.108
pair_coeff 21 31 thole 1.108
pair_coeff 21 32 thole 1.164
pair_coeff 21 33 thole 1.108
pair_coeff 21 34 thole 1.108
pair_coeff 21 35 thole 1.108
pair_coeff 21 36 thole 1.164
pair_coeff 21 37 thole 1.164
pair_coeff 21 38 thole 1.208
pair_coeff 21 39 thole 1.208
pair_coeff 21 40 thole 1.126
pair_coeff 21 41 thole 0.851
pair_coeff 21 42 thole 1.323
pair_coeff 21 43 thole 1.282
pair_coeff 21 44 thole 0.777
pair_coeff 22 22 thole 1.208
pair_coeff 22 23 thole 1.126
pair_coeff 22 24 thole 0.851
pair_coeff 22 25 thole 1.323
pair_coeff 22 26 thole 1.282
pair_coeff 22 27 thole 0.777
pair_coeff 22 28 thole 1.108
pair_coeff 22 29 thole 1.108
pair_coeff 22 30 thole 1.108
pair_coeff 22 31 thole 1.108
pair_coeff 22 32 thole 1.164
pair_coeff 22 33 thole 1.108
pair_coeff 22 34 thole 1.108
pair_coeff 22 35 thole 1.108
pair_coeff 22 36 thole 1.164
pair_coeff 22 37 thole 1.164
pair_coeff 22 38 thole 1.208

```
pair_coeff    22    39 thole    1.208
pair_coeff    22    40 thole    1.126
pair_coeff    22    41 thole    0.851
pair_coeff    22    42 thole    1.323
pair_coeff    22    43 thole    1.282
pair_coeff    22    44 thole    0.777
pair_coeff    23    23 thole    1.050
pair_coeff    23    24 thole    0.794
pair_coeff    23    25 thole    1.234
pair_coeff    23    26 thole    1.195
pair_coeff    23    27 thole    0.725
pair_coeff    23    28 thole    1.033
pair_coeff    23    29 thole    1.033
pair_coeff    23    30 thole    1.033
pair_coeff    23    31 thole    1.033
pair_coeff    23    32 thole    1.085
pair_coeff    23    33 thole    1.033
pair_coeff    23    34 thole    1.033
pair_coeff    23    35 thole    1.033
pair_coeff    23    36 thole    1.085
pair_coeff    23    37 thole    1.085
pair_coeff    23    38 thole    1.126
pair_coeff    23    39 thole    1.126
pair_coeff    23    40 thole    1.050
pair_coeff    23    41 thole    0.794
pair_coeff    23    42 thole    1.234
pair_coeff    23    43 thole    1.195
pair_coeff    23    44 thole    0.725
pair_coeff    24    24 thole    0.600
pair_coeff    24    25 thole    0.933
pair_coeff    24    26 thole    0.903
pair_coeff    24    27 thole    0.548
```

```
pair_coeff    24    28 thole    0.781
pair_coeff    24    29 thole    0.781
pair_coeff    24    30 thole    0.781
pair_coeff    24    31 thole    0.781
pair_coeff    24    32 thole    0.820
pair_coeff    24    33 thole    0.781
pair_coeff    24    34 thole    0.781
pair_coeff    24    35 thole    0.781
pair_coeff    24    36 thole    0.820
pair_coeff    24    37 thole    0.820
pair_coeff    24    38 thole    0.851
pair_coeff    24    39 thole    0.851
pair_coeff    24    40 thole    0.794
pair_coeff    24    41 thole    0.600
pair_coeff    24    42 thole    0.933
pair_coeff    24    43 thole    0.903
pair_coeff    24    44 thole    0.548
pair_coeff    25    25 thole    1.450
pair_coeff    25    26 thole    1.404
pair_coeff    25    27 thole    0.851
pair_coeff    25    28 thole    1.214
pair_coeff    25    29 thole    1.214
pair_coeff    25    30 thole    1.214
pair_coeff    25    31 thole    1.214
pair_coeff    25    32 thole    1.275
pair_coeff    25    33 thole    1.214
pair_coeff    25    34 thole    1.214
pair_coeff    25    35 thole    1.214
pair_coeff    25    36 thole    1.275
pair_coeff    25    37 thole    1.275
pair_coeff    25    38 thole    1.323
pair_coeff    25    39 thole    1.323
```

```
pair_coeff    25    40 thole    1.234
pair_coeff    25    41 thole    0.933
pair_coeff    25    42 thole    1.450
pair_coeff    25    43 thole    1.404
pair_coeff    25    44 thole    0.851
pair_coeff    26    26 thole    1.360
pair_coeff    26    27 thole    0.825
pair_coeff    26    28 thole    1.175
pair_coeff    26    29 thole    1.175
pair_coeff    26    30 thole    1.175
pair_coeff    26    31 thole    1.175
pair_coeff    26    32 thole    1.235
pair_coeff    26    33 thole    1.175
pair_coeff    26    34 thole    1.175
pair_coeff    26    35 thole    1.175
pair_coeff    26    36 thole    1.235
pair_coeff    26    37 thole    1.235
pair_coeff    26    38 thole    1.282
pair_coeff    26    39 thole    1.282
pair_coeff    26    40 thole    1.195
pair_coeff    26    41 thole    0.903
pair_coeff    26    42 thole    1.404
pair_coeff    26    43 thole    1.360
pair_coeff    26    44 thole    0.825
pair_coeff    27    27 thole    0.500
pair_coeff    27    28 thole    0.713
pair_coeff    27    29 thole    0.713
pair_coeff    27    30 thole    0.713
pair_coeff    27    31 thole    0.713
pair_coeff    27    32 thole    0.749
pair_coeff    27    33 thole    0.713
pair_coeff    27    34 thole    0.713
```

```
pair_coeff    27    35 thole    0.713
pair_coeff    27    36 thole    0.749
pair_coeff    27    37 thole    0.749
pair_coeff    27    38 thole    0.777
pair_coeff    27    39 thole    0.777
pair_coeff    27    40 thole    0.725
pair_coeff    27    41 thole    0.548
pair_coeff    27    42 thole    0.851
pair_coeff    27    43 thole    0.825
pair_coeff    27    44 thole    0.500
pair_coeff    28    28 thole    1.016
pair_coeff    28    29 thole    1.016
pair_coeff    28    30 thole    1.016
pair_coeff    28    31 thole    1.016
pair_coeff    28    32 thole    1.068
pair_coeff    28    33 thole    1.016
pair_coeff    28    34 thole    1.016
pair_coeff    28    35 thole    1.016
pair_coeff    28    36 thole    1.068
pair_coeff    28    37 thole    1.068
pair_coeff    28    38 thole    1.108
pair_coeff    28    39 thole    1.108
pair_coeff    28    40 thole    1.033
pair_coeff    28    41 thole    0.781
pair_coeff    28    42 thole    1.214
pair_coeff    28    43 thole    1.175
pair_coeff    28    44 thole    0.713
pair_coeff    29    29 thole    1.016
pair_coeff    29    30 thole    1.016
pair_coeff    29    31 thole    1.016
pair_coeff    29    32 thole    1.068
pair_coeff    29    33 thole    1.016
```

pair_coeff 29 34 thole 1.016
pair_coeff 29 35 thole 1.016
pair_coeff 29 36 thole 1.068
pair_coeff 29 37 thole 1.068
pair_coeff 29 38 thole 1.108
pair_coeff 29 39 thole 1.108
pair_coeff 29 40 thole 1.033
pair_coeff 29 41 thole 0.781
pair_coeff 29 42 thole 1.214
pair_coeff 29 43 thole 1.175
pair_coeff 29 44 thole 0.713
pair_coeff 30 30 thole 1.016
pair_coeff 30 31 thole 1.016
pair_coeff 30 32 thole 1.068
pair_coeff 30 33 thole 1.016
pair_coeff 30 34 thole 1.016
pair_coeff 30 35 thole 1.016
pair_coeff 30 36 thole 1.068
pair_coeff 30 37 thole 1.068
pair_coeff 30 38 thole 1.108
pair_coeff 30 39 thole 1.108
pair_coeff 30 40 thole 1.033
pair_coeff 30 41 thole 0.781
pair_coeff 30 42 thole 1.214
pair_coeff 30 43 thole 1.175
pair_coeff 30 44 thole 0.713
pair_coeff 31 31 thole 1.016
pair_coeff 31 32 thole 1.068
pair_coeff 31 33 thole 1.016
pair_coeff 31 34 thole 1.016
pair_coeff 31 35 thole 1.016
pair_coeff 31 36 thole 1.068

```
pair_coeff    31    37 thole    1.068
pair_coeff    31    38 thole    1.108
pair_coeff    31    39 thole    1.108
pair_coeff    31    40 thole    1.033
pair_coeff    31    41 thole    0.781
pair_coeff    31    42 thole    1.214
pair_coeff    31    43 thole    1.175
pair_coeff    31    44 thole    0.713
pair_coeff    32    32 thole    1.122
pair_coeff    32    33 thole    1.068
pair_coeff    32    34 thole    1.068
pair_coeff    32    35 thole    1.068
pair_coeff    32    36 thole    1.122
pair_coeff    32    37 thole    1.122
pair_coeff    32    38 thole    1.164
pair_coeff    32    39 thole    1.164
pair_coeff    32    40 thole    1.085
pair_coeff    32    41 thole    0.820
pair_coeff    32    42 thole    1.275
pair_coeff    32    43 thole    1.235
pair_coeff    32    44 thole    0.749
pair_coeff    33    33 thole    1.016
pair_coeff    33    34 thole    1.016
pair_coeff    33    35 thole    1.016
pair_coeff    33    36 thole    1.068
pair_coeff    33    37 thole    1.068
pair_coeff    33    38 thole    1.108
pair_coeff    33    39 thole    1.108
pair_coeff    33    40 thole    1.033
pair_coeff    33    41 thole    0.781
pair_coeff    33    42 thole    1.214
pair_coeff    33    43 thole    1.175
```

```
pair_coeff    33    44 thole    0.713
pair_coeff    34    34 thole    1.016
pair_coeff    34    35 thole    1.016
pair_coeff    34    36 thole    1.068
pair_coeff    34    37 thole    1.068
pair_coeff    34    38 thole    1.108
pair_coeff    34    39 thole    1.108
pair_coeff    34    40 thole    1.033
pair_coeff    34    41 thole    0.781
pair_coeff    34    42 thole    1.214
pair_coeff    34    43 thole    1.175
pair_coeff    34    44 thole    0.713
pair_coeff    35    35 thole    1.016
pair_coeff    35    36 thole    1.068
pair_coeff    35    37 thole    1.068
pair_coeff    35    38 thole    1.108
pair_coeff    35    39 thole    1.108
pair_coeff    35    40 thole    1.033
pair_coeff    35    41 thole    0.781
pair_coeff    35    42 thole    1.214
pair_coeff    35    43 thole    1.175
pair_coeff    35    44 thole    0.713
pair_coeff    36    36 thole    1.122
pair_coeff    36    37 thole    1.122
pair_coeff    36    38 thole    1.164
pair_coeff    36    39 thole    1.164
pair_coeff    36    40 thole    1.085
pair_coeff    36    41 thole    0.820
pair_coeff    36    42 thole    1.275
pair_coeff    36    43 thole    1.235
pair_coeff    36    44 thole    0.749
pair_coeff    37    37 thole    1.122
```

```
pair_coeff    37    38 thole    1.164
pair_coeff    37    39 thole    1.164
pair_coeff    37    40 thole    1.085
pair_coeff    37    41 thole    0.820
pair_coeff    37    42 thole    1.275
pair_coeff    37    43 thole    1.235
pair_coeff    37    44 thole    0.749
pair_coeff    38    38 thole    1.208
pair_coeff    38    39 thole    1.208
pair_coeff    38    40 thole    1.126
pair_coeff    38    41 thole    0.851
pair_coeff    38    42 thole    1.323
pair_coeff    38    43 thole    1.282
pair_coeff    38    44 thole    0.777
pair_coeff    39    39 thole    1.208
pair_coeff    39    40 thole    1.126
pair_coeff    39    41 thole    0.851
pair_coeff    39    42 thole    1.323
pair_coeff    39    43 thole    1.282
pair_coeff    39    44 thole    0.777
pair_coeff    40    40 thole    1.050
pair_coeff    40    41 thole    0.794
pair_coeff    40    42 thole    1.234
pair_coeff    40    43 thole    1.195
pair_coeff    40    44 thole    0.725
pair_coeff    41    41 thole    0.600
pair_coeff    41    42 thole    0.933
pair_coeff    41    43 thole    0.903
pair_coeff    41    44 thole    0.548
pair_coeff    42    42 thole    1.450
pair_coeff    42    43 thole    1.404
pair_coeff    42    44 thole    0.851
```

```
pair_coeff     43    43 thole     1.360
pair_coeff     43    44 thole     0.825
pair_coeff     44    44 thole     0.500


# ------------- Variables definitions -------------- #

# basic

variable  TEMP    equal 300.0
variable  TEMP_D equal 1.0
variable  PRESS   equal 1.0
variable  dt          equal 1.0
variable  dthero   equal 100
variable  dtdump equal 100

variable  Tdamp  equal ${dt}*100
variable  Tdamp_D equal ${dt}*20
variable  Pdamp  equal ${dt}*1000
variable  Nevery equal 1000
variable  Nrepeat equal 10
variable  Nfreq    equal 10000
variable  cutoff    equal 12.0

# equilibration  variables

variable  TEMP_nvt      equal 1000.0
variable  TEMP_npt      equal 600.0
variable  PRESS_npt_1  equal 100.0
variable  PRESS_npt_2  equal 1.0
variable  tnvt             equal 100000.0    # 0.1  ns
variable  tnpt             equal 10000.0     # 0.01  ns
```

```
variable  tpro          equal 1000000.0  # 1 ns

velocity  ATOMS create ${TEMP} 12345
velocity  DRUDES create ${TEMP_D} 12345

# ------------- Energy Minimization -------------- #

# flag  for  each atom type: [C]ore,  [D]rude, [N]on-polarizable
fix  DRUDE all drude C C C C C C C C C C N N N N N N N N N N C C C C C C C D D D D
     D D D D D D D D D D D D D

min_style       cg
minimize        1e-45 1e-45 10000 10000

write_data      PolyBMIM_TFSI_1-p_minimized.data

# use  fix  shake to  constrain  the  C-H bonds for the Drude core atoms
fix  SHAKE ATOMS shake 0.0001 20 0 b 2 6 8 9 10 14 16 17 19 21


# ------------- Equilibration Stage 1 (NVT dynamics at 1000 K for 0.1 ns)
     -------------- #

print  "loop 1  starts   ......  "

fix             DIRECT all drude/transform/direct
fix             NVT_ATOMS ATOMS nvt temp ${TEMP_nvt} ${TEMP_nvt} ${Tdamp}
fix       NVT_DRUDES DRUDES nvt temp ${TEMP_D} ${TEMP_D} ${Tdamp_D}
fix             INVERSE all drude/transform/inverse
fix       MOMENTUM all momentum 100 linear 1 1 1
```

```
thermo_style    custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
    vol fmax
thermo          ${dthero}
timestep ${dt}

run     ${tnvt}    # 0.1  ns

unfix       DIRECT
unfix     INVERSE
unfix           NVT_ATOMS
unfix     NVT_DRUDES

# -------------- Equilibration Stage 2 (NPT dynamics at 600 K 100 bar for 0.1 ns)
    --------------- #

compute         TEMP_ATOMS_1 ATOMS temp
fix             DIRECT all drude/transform/direct
fix       NPT_ATOMS ATOMS npt temp ${TEMP_npt} ${TEMP_npt} ${Tdamp} iso
    ${PRESS_npt_1} ${PRESS_npt_1} ${Pdamp}
fix_modify  NPT_ATOMS temp TEMP_ATOMS_1 press thermo_press
fix       NVT_DRUDES DRUDES nvt temp ${TEMP_D} ${TEMP_D} ${Tdamp_D}
fix             INVERSE all drude/transform/inverse

thermo_style    custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
    vol fmax c_TEMP_ATOMS_1
thermo          ${dthero}
timestep        ${dt}

run     5000
run     5000
run     5000
run     5000
```

```
run         5000
run         5000
run         5000
run         5000
run         5000
run         5000
run         5000
run         5000
run         5000
run         5000
run         5000
run         5000
run         5000
run         5000
run         5000
run         5000
run         5000      # 0.1 ns

unfix         DIRECT
unfix       INVERSE
unfix               NPT_ATOMS
unfix       NVT_DRUDES

# ---------------- Equilibration Stage 3 (NPT dynamics at 600 K 1 bar for 0.1 ns)
      ---------------- #

compute           TEMP_ATOMS_2 ATOMS temp
fix               DIRECT all drude/transform/direct
fix         NPT_ATOMS ATOMS npt temp ${TEMP_npt} ${TEMP_npt} ${Tdamp} iso
      ${PRESS_npt_2} ${PRESS_npt_2} ${Pdamp}
fix_modify   NPT_ATOMS temp TEMP_ATOMS_1 press thermo_press
fix         NVT_DRUDES DRUDES nvt temp ${TEMP_D} ${TEMP_D} ${Tdamp_D}
fix               INVERSE all drude/transform/inverse
```

```
thermo_style    custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
    vol fmax c_TEMP_ATOMS_2
thermo          ${dthero}
timestep        ${dt}

run     ${tnpt}
run     ${tnpt}
run     ${tnpt}
run     ${tnpt}
run     ${tnpt}
run     ${tnpt}
run     ${tnpt}
run     ${tnpt}
run     ${tnpt}
run     ${tnpt}    # 0.1  ns

unfix       DIRECT
unfix    INVERSE
unfix           NPT_ATOMS
unfix    NVT_DRUDES

# ------------- Equilibration Stage 1 (NVT dynamics at 1000 K for 0.1 ns)
    -------------- #

print  "loop 2  starts   ......  "

fix             DIRECT all drude/transform/direct
fix             NVT_ATOMS ATOMS nvt temp ${TEMP_nvt} ${TEMP_nvt} ${Tdamp}
fix      NVT_DRUDES DRUDES nvt temp ${TEMP_D} ${TEMP_D} ${Tdamp_D}
fix             INVERSE all drude/transform/inverse
fix      MOMENTUM all momentum 100 linear 1 1 1
```

```
thermo_style    custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
    vol fmax
thermo          ${dthero}
timestep ${dt}

run     ${tnvt}    # 0.1  ns

unfix       DIRECT
unfix     INVERSE
unfix           NVT_ATOMS
unfix     NVT_DRUDES

# -------------- Equilibration Stage 2 (NPT dynamics at 600 K 100 bar for 0.1 ns)
    --------------- #

compute         TEMP_ATOMS_3 ATOMS temp
fix             DIRECT all drude/transform/direct
fix       NPT_ATOMS ATOMS npt temp ${TEMP_npt} ${TEMP_npt} ${Tdamp} iso
    ${PRESS_npt_1} ${PRESS_npt_1} ${Pdamp}
fix_modify   NPT_ATOMS temp TEMP_ATOMS_1 press thermo_press
fix       NVT_DRUDES DRUDES nvt temp ${TEMP_D} ${TEMP_D} ${Tdamp_D}
fix             INVERSE all drude/transform/inverse

thermo_style    custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
    vol fmax c_TEMP_ATOMS_3
thermo          ${dthero}
timestep        ${dt}

run     ${tnpt}
run     ${tnpt}
run     ${tnpt}
```

```
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}    # 0.1  ns

unfix          DIRECT
unfix       INVERSE
unfix               NPT_ATOMS
unfix       NVT_DRUDES

# ---------------- Equilibration Stage 3 (NPT dynamics at 600 K 1 bar for 0.1 ns)
      ---------------- #

compute             TEMP_ATOMS_4 ATOMS temp
fix                 DIRECT all drude/transform/direct
fix         NPT_ATOMS ATOMS npt temp ${TEMP_npt} ${TEMP_npt} ${Tdamp} iso
      ${PRESS_npt_2} ${PRESS_npt_2} ${Pdamp}
fix_modify   NPT_ATOMS temp TEMP_ATOMS_1 press thermo_press
fix         NVT_DRUDES DRUDES nvt temp ${TEMP_D} ${TEMP_D} ${Tdamp_D}
fix                 INVERSE all drude/transform/inverse

thermo_style      custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
      vol fmax c_TEMP_ATOMS_4
thermo              ${dthero}
timestep            ${dt}

run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
```

```
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}    # 0.1  ns

unfix         DIRECT
unfix      INVERSE
unfix               NPT_ATOMS
unfix      NVT_DRUDES

# ---------------- Equilibration Stage 1 (NVT dynamics at 1000 K for 0.1 ns)
      ---------------- #

print  "loop 3  starts   ......  "

fix                 DIRECT all drude/transform/direct
fix                 NVT_ATOMS ATOMS nvt temp ${TEMP_nvt} ${TEMP_nvt} ${Tdamp}
fix         NVT_DRUDES DRUDES nvt temp ${TEMP_D} ${TEMP_D} ${Tdamp_D}
fix                 INVERSE all drude/transform/inverse
fix         MOMENTUM all momentum 100 linear 1 1 1

thermo_style     custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
      vol fmax
thermo              ${dthero}
timestep ${dt}

run         ${tnvt}    # 0.1  ns

unfix         DIRECT
```

```
unfix       INVERSE
unfix               NVT_ATOMS
unfix       NVT_DRUDES

# ------------- Equilibration Stage 2 (NPT dynamics at 600 K 100 bar for 0.1 ns)
    -------------- #

compute             TEMP_ATOMS_5 ATOMS temp
fix                 DIRECT all drude/transform/direct
fix         NPT_ATOMS ATOMS npt temp ${TEMP_npt} ${TEMP_npt} ${Tdamp} iso
    ${PRESS_npt_1} ${PRESS_npt_1} ${Pdamp}
fix_modify   NPT_ATOMS temp TEMP_ATOMS_1 press thermo_press
fix         NVT_DRUDES DRUDES nvt temp ${TEMP_D} ${TEMP_D} ${Tdamp_D}
fix                 INVERSE all drude/transform/inverse

thermo_style      custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
    vol fmax c_TEMP_ATOMS_5
thermo              ${dthero}
timestep            ${dt}

run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}    # 0.1 ns

unfix           DIRECT
```

```
unfix       INVERSE
unfix               NPT_ATOMS
unfix       NVT_DRUDES

# ------------- Equilibration Stage 3 (NPT dynamics at 600 K 1 bar for 0.1 ns)
    ------------- #

compute             TEMP_ATOMS_6 ATOMS temp
fix                 DIRECT all drude/transform/direct
fix         NPT_ATOMS ATOMS npt temp ${TEMP_npt} ${TEMP_npt} ${Tdamp} iso
    ${PRESS_npt_2} ${PRESS_npt_2} ${Pdamp}
fix_modify   NPT_ATOMS temp TEMP_ATOMS_1 press thermo_press
fix         NVT_DRUDES DRUDES nvt temp ${TEMP_D} ${TEMP_D} ${Tdamp_D}
fix                 INVERSE all drude/transform/inverse

thermo_style      custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
    vol fmax c_TEMP_ATOMS_6
thermo              ${dthero}
timestep            ${dt}

run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}    # 0.1 ns

unfix           DIRECT
```

```
unfix       INVERSE
unfix               NPT_ATOMS
unfix       NVT_DRUDES


# ------------- Equilibration Stage 1 (NVT dynamics at 1000 K for 0.1 ns)
      ------------- #

print "loop 4 starts ..... "

fix                 DIRECT all drude/transform/direct
fix                 NVT_ATOMS ATOMS nvt temp ${TEMP_nvt} ${TEMP_nvt} ${Tdamp}
fix         NVT_DRUDES DRUDES nvt temp ${TEMP_D} ${TEMP_D} ${Tdamp_D}
fix                 INVERSE all drude/transform/inverse
fix         MOMENTUM all momentum 100 linear 1 1 1

thermo_style    custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
      vol fmax
thermo              ${dthero}
timestep ${dt}

run         ${tnvt}    # 0.1 ns

unfix           DIRECT
unfix       INVERSE
unfix               NVT_ATOMS
unfix       NVT_DRUDES

# ------------- Equilibration Stage 2 (NPT dynamics at 600 K 100 bar for 0.1 ns)
      ------------- #

compute             TEMP_ATOMS_7 ATOMS temp
```

```
fix                 DIRECT all drude/transform/direct
fix         NPT_ATOMS ATOMS npt temp ${TEMP_npt} ${TEMP_npt} ${Tdamp} iso
    ${PRESS_npt_1} ${PRESS_npt_1} ${Pdamp}
fix_modify  NPT_ATOMS temp TEMP_ATOMS_1 press thermo_press
fix         NVT_DRUDES DRUDES nvt temp ${TEMP_D} ${TEMP_D} ${Tdamp_D}
fix                 INVERSE all drude/transform/inverse

thermo_style    custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
    vol fmax c_TEMP_ATOMS_7
thermo          ${dthero}
timestep        ${dt}

run     ${tnpt}
run     ${tnpt}
run     ${tnpt}
run     ${tnpt}
run     ${tnpt}
run     ${tnpt}
run     ${tnpt}
run     ${tnpt}
run     ${tnpt}
run     ${tnpt}   # 0.1 ns

unfix       DIRECT
unfix     INVERSE
unfix       NPT_ATOMS
unfix     NVT_DRUDES

# ------------- Equilibration Stage 3 (NPT dynamics at 600 K 1 bar for 0.1 ns)
    ------------- #

compute         TEMP_ATOMS_8 ATOMS temp
```

```
fix                 DIRECT all drude/transform/direct
fix         NPT_ATOMS ATOMS npt temp ${TEMP_npt} ${TEMP_npt} ${Tdamp} iso
    ${PRESS_npt_2} ${PRESS_npt_2} ${Pdamp}
fix_modify  NPT_ATOMS temp TEMP_ATOMS_1 press thermo_press
fix         NVT_DRUDES DRUDES nvt temp ${TEMP_D} ${TEMP_D} ${Tdamp_D}
fix                 INVERSE all drude/transform/inverse

thermo_style    custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
    vol fmax c_TEMP_ATOMS_8
thermo          ${dthero}
timestep        ${dt}

run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}
run         ${tnpt}    # 0.1 ns

unfix           DIRECT
unfix       INVERSE
unfix           NPT_ATOMS
unfix       NVT_DRUDES

write_restart     restart .PolyBMIM_TFSI_1-p.equilibrated
write_data        data.PolyBMIM_TFSI_1-p.equilibrated

# ------------- Production Run (NPT dynamics at 300 K for 2 ns) -------------- #
```

```
print "NPT Production Run"

compute         TEMP_ATOMS_9 ATOMS temp
fix             DIRECT all drude/transform/direct
fix       NPT_ATOMS ATOMS npt temp ${TEMP} ${TEMP} ${Tdamp} iso ${PRESS}
    ${PRESS} ${Pdamp}
fix_modify  NPT_ATOMS temp TEMP_ATOMS_2 press thermo_press
fix       NVT_DRUDES DRUDES nvt temp ${TEMP_D} ${TEMP_D} ${Tdamp_D}
fix             INVERSE all drude/transform/inverse

thermo_style    custom step temp pe ke etotal density evdwl ecoul epair ebond emol press
    vol fmax c_TEMP_ATOMS_9
thermo          ${dthero}

run       ${tpro}    # 1 ns

# ------------- record data for the last 1 ns ------------- #

dump            1 ATOMS custom ${dtdump} PolyBMIM_TFSI_1-p.xyz id type mol x y z vx
    vy vz
dump    2 ATOMS custom ${dtdump} PolyBMIM_TFSI_1-p.dump id type mol x y z vx vy vz
dump    3 ATOMS custom ${dtdump} PolyBMIM_TFSI_1-p.pdb id type mol x y z vx vy vz

# ------------ Calculate RDFs / Rg / msd ------------ #

compute TFSI_TFSI all rdf 500 23*27 23*27 cutoff ${cutoff}
compute PolyBMIM_PolyBMIM all rdf 500 1*22 1*22 cutoff ${cutoff}
compute TFSI_PolyBMIM all rdf 500 23*27 1*22 cutoff ${cutoff}
compute PolyBMIM_TFSI all rdf 500 1*22 23*27 cutoff ${cutoff}

compute PolyBMIM_Rg PolyBMIM gyration
```

```
compute     TFSI_msd_1 TFSI msd com yes
compute     TFSI_msd_2 TFSI msd com yes
compute     TFSI_msd_3 TFSI_and_dp msd com yes

fix         TFSI_TFSI all  ave/time ${Nevery} ${Nrepeat} ${Nfreq} c_TFSI_TFSI[*] file
    TFSI_TFSI.rdf mode vector
fix         PolyBMIM_PolyBMIM all ave/time ${Nevery} ${Nrepeat} ${Nfreq}
    c_PolyBMIM_PolyBMIM[*] file PolyBMIM_PolyBMIM.rdf mode vector
fix         TFSI_PolyBMIM all ave/time ${Nevery} ${Nrepeat} ${Nfreq} c_TFSI_PolyBMIM[*]
    file TFSI_PolyBMIM.rdf mode vector
fix         PolyBMIM_TFSI all ave/time ${Nevery} ${Nrepeat} ${Nfreq} c_PolyBMIM_TFSI[*]
    file PolyBMIM_TFSI.rdf mode vector

fix         PolyBMIM_Rg_v PolyBMIM ave/time ${Nevery} ${Nrepeat} ${Nfreq}
    c_PolyBMIM_Rg file data.PolyBMIM_Rg_v mode vector
fix         PolyBMIM_Rg_s PolyBMIM ave/time ${Nevery} ${Nrepeat} ${Nfreq}
    c_PolyBMIM_Rg[*] file data.PolyBMIM_Rg_s mode scalar
fix         TFSI_msd_1 TFSI ave/time ${Nevery} ${Nrepeat} ${Nfreq} c_TFSI_msd_1 file
    data.TFSI_msd_v mode vector
fix             TFSI_msd_2 TFSI ave/time ${Nevery} ${Nrepeat} ${Nfreq} c_TFSI_msd_2[*]
    file data.TFSI_msd_s mode scalar
fix             TFSI_msd_3 TFSI ave/time ${Nevery} ${Nrepeat} ${Nfreq} c_TFSI_msd_3[*]
    file data.TFSI_and_dp_msd_s mode scalar

run         ${tpro}    # 1 ns

unfix       DIRECT
unfix       INVERSE
unfix           NPT_ATOMS
unfix       NVT_DRUDES

write_restart     restart .PolyBMIM_TFSI_1-p.final
```

```
write_data        data.PolyBMIM_TFSI_1-p.final

print "All done!"
```

A.6.3 ANI-2x input script

```
from ase.io import read, write
from ase.io.trajectory import Trajectory
from ase.md.npt import NPT
from ase.io.lammpsdata import write_lammps_data
from ase.io.lammpsdata import read_lammps_data
from ase.md.velocitydistribution import MaxwellBoltzmannDistribution
from ase.md import MDLogger
from ase import units
import torch
import torchani
from ase import *
from ase import Atoms

from ase.md.analysis import DiffusionCoefficient # possible for msd calculation

# read data from lammps data file

atoms = read_lammps_data('data.PolyBMIM_TFSI_1.npt2', style='full',
                         units='real', Z_of_type={1: 6, 2: 6, 3: 6, 4: 6, 5: 6, 6: 6, 7: 6,
                             8: 6, 9: 6, 10: 6,
                         11: 1, 12: 1, 13: 1, 14: 1, 15: 1, 16: 1, 17: 1, 18: 1, 19: 1, 20:
                             1, 21: 7, 22: 7, 23: 6, 24: 9, 25: 7, 26: 8, 27: 16})

print(len(atoms), "atoms in the cell")          # number of atoms in the data file
atoms.set_pbc((True, True, True))               # impose the periodic boundary
    conditions
print(atoms.get_cell().volume, 'Angstrom^3')    # size of the box

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

```
calculator = torchani.models.ANI2x(periodic_table_index=True).to(device).ase()      #
    import ANI-2x
atoms.set_calculator(calculator)                                                    #
    Specify ANI-2x as the calculator

# define some variables

T = 600 # temperature in Kelvin
timestep = 1 * units.fs   # timestep set to 1 fs
pressure = 1 * units.bar # pressure = 1 bar during NPT
interval = 100  # how many timesteps print one row of data
tpro = 2000000 # run this many of timestep in npt production run
call_count = 0 # global variable to keep track of the function printsystem

MaxwellBoltzmannDistribution(atoms, temperature_K=T, communicator=None,
    force_temp=True, rng=None) # initialize the velocity of the atoms

def printenergy(a=atoms, file_name="PolyBMIM_TFSI_1_ANI_my.log"):

    global call_count   # we need to declare the variable as global to change it
    call_count += 1  # increment the call count

    """Function to print the potential, kinetic and total energy."""
    epot = a.get_potential_energy() * units.kcal / units.mol
    ekin = a.get_kinetic_energy() / len(atoms) * units.kcal / units.mol
    ekin_all = a.get_kinetic_energy() * units.kcal / units.mol
    volume = atoms.get_volume()
    mass = atoms.get_masses().sum()
    density = mass/volume * 1.66053906660e-24 / (1e-8)**3

    system_info = 'Simulation Info: Epot = %.3f kcal/mol Ekin = %.3f kcal/mol (T=%3.0f K)
        ' \
```

```
                'Etot = %.3f kcal/mol Volume = %.3f Angstrom^3 Density = %.3f g/cm^3'\
                % (epot, ekin_all, ekin / (units.kcal / units.mol) / (1.5 * units.kB),
                    epot + ekin, volume, density)

    print(system_info)

    with open(file_name, "a") as f:  # "a" means append mode
        f.write(f'{call_count * interval}: {system_info}\n')  # write call_count and
            energy_info to the file

print("Beginning NPT production run...")

dyn3 = NPT(atoms, timestep=1 * units.fs, externalstress=(-1 * units.bar, -1 * units.bar, -1
    * units.bar, 0, 0, 0),
            ttime=2 * units.fs, pfactor=2 * units.fs, temperature_K=T, mask=([1, 0, 0], [0,
                1, 0], [0, 0, 1]), append_trajectory=True)

logger_production = MDLogger(dyn3, atoms, 'PolyBMIM_TFSI_1_ANI.log', header=True,
    stress=True, mode="a")
traj = Trajectory('PolyBMIM_TFSI_1_ANI.traj', 'a', atoms)
dyn3.attach(traj.write, interval=interval)
dyn3.attach(logger_production, interval=interval)
dyn3.attach(printenergy, interval=interval)
printenergy()
dyn3.run(tpro)

write_lammps_data('data.PolyBMIM_TFSI_1_ANI.final', atoms, velocities=False,
    specorder=None, force_skew=False, prismobj=None,
                units='real', atom_style='full') # write final structure to data file
```

BIBLIOGRAPHY

[1] Simon Axelrod, Daniel Schwalbe-Koda, Somesh Mohapatra, James Damewood, Kevin P. Greenman, and Rafael Gómez-Bombarelli. Learning matter: Materials design with machine learning and atomistic simulations. *Accounts of Materials Research*, 3(3):343–357, 2022.

[2] Albert P. Bartók, Mike C. Payne, Risi Kondor, and Gábor Csányi. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Phys. Rev. Lett.*, 104:136403, Apr 2010.

[3] Dmitry Bedrov, Jean-Philip Piquemal, Oleg Borodin, Alexander D. Jr. MacKerell, Benoît Roux, and Christian Schröder. Molecular dynamics simulations of ionic liquids and electrolytes using polarizable force fields. *Chemical Reviews*, 119(13):7940–7995, 2019. PMID: 31141351.

[4] Jörg Behler and Michele Parrinello. Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces. *Phys. Rev. Lett.*, 98(14):146401, April 2007.

[5] B. R. Brooks, C. L. Brooks III, A. D. Mackerell Jr., L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A. R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R. W. Pastor, C. B. Post, J. Z. Pu, M. Schaefer, B. Tidor, R. M. Venable, H. L. Woodcock, X. Wu, W. Yang, D. M. York, and M. Karplus. Charmm: The biomolecular simulation program. *Journal of Computational Chemistry*, 30(10):1545–1614, 2009.

[6] José N. Canongia Lopes and Agílio A. H. Pádua. CL&P: A generic and systematic force field for ionic liquids modeling. *Theor Chem Acc*, 131(3):1129, March 2012.

[7] Chi Chen and Shyue Ping Ong. A universal graph deep learning interatomic potential for the periodic table. *Nat Comput Sci*, 2(11):718–728, November 2022.

[8] Alain Dequidt, Julien Devémy, and Agílio A. H. Pádua. Thermalized drude oscillators with the lammps molecular dynamics simulator. *Journal of Chemical Information and Modeling*, 56(1):260–268, 2016. PMID: 26646769.

[9] Christian Devereux, Justin S. Smith, Kate K. Huddleston, Kipton Barros, Roman Zubatyuk, Olexandr Isayev, and Adrian E. Roitberg. Extending the

Applicability of the ANI Deep Learning Molecular Potential to Sulfur and Halogens. *J. Chem. Theory Comput.*, 16(7):4192–4202, July 2020.

[10] Leela S. Dodda, Israel Cabeza de Vaca, Julian Tirado-Rives, and William L. Jorgensen. LigParGen web server: an automatic OPLS-AA parameter generator for organic ligands. *Nucleic Acids Research*, 45(W1):W331–W336, 04 2017.

[11] Leela S. Dodda, Jonah Z. Vilseck, Julian Tirado-Rives, and William L. Jorgensen. 1.14*cm1a-lbcc: Localized bond-charge corrected cm1a charges for condensed-phase simulations. *The Journal of Physical Chemistry B*, 121(15):3864–3870, 2017. PMID: 28224794.

[12] Brian Doherty, Xiang Zhong, Symon Gathiaka, Bin Li, and Orlando Acevedo. Revisiting opls force field parameters for ionic liquid simulations. *Journal of Chemical Theory and Computation*, 13(12):6131–6145, 2017. PMID: 29112809.

[13] Peter Eastman, Jason Swails, John D. Chodera, Robert T. McGibbon, Yutong Zhao, Kyle A. Beauchamp, Lee-Ping Wang, Andrew C. Simmonett, Matthew P. Harrigan, Chaya D. Stern, Rafal P. Wiewiora, Bernard R. Brooks, and Vijay S. Pande. Openmm 7: Rapid development of high performance algorithms for molecular dynamics. *PLOS Computational Biology*, 13(7):1–17, 07 2017.

[14] Tobias Fink and Jean-Louis Reymond. Virtual Exploration of the Chemical Universe up to 11 Atoms of C, N, O, F: Assembly of 26.4 Million Structures (110.9 Million Stereoisomers) and Analysis for New Ring Systems, Stereochemistry, Physicochemical Properties, Compound Classes, and Drug Discovery. *J. Chem. Inf. Model.*, 47(2):342–353, March 2007.

[15] Thomas E. Gartner and Arthi Jayaraman. Modeling and Simulations of Polymers: A Roadmap. *Macromolecules*, 52(3):755–786, February 2019.

[16] A. Gaulton, L. J. Bellis, A. P. Bento, J. Chambers, M. Davies, A. Hersey, Y. Light, S. McGlinchey, D. Michalovich, B. Al-Lazikani, and J. P. Overington. ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic Acids Research*, 40(D1):D1100–D1107, January 2012.

[17] Kateryna Goloviznina, José N. Canongia Lopes, Margarida Costa Gomes, and Agílio A. H. Pádua. Transferable, polarizable force field for ionic liquids. *Journal of Chemical Theory and Computation*, 15(11):5858–5871, 2019. PMID: 31525922.

[18] Jean-Pierre Hansen and Ian Ranald McDonald. *Theory of simple liquids: with applications to soft matter*. Academic press, 2013.

[19] Ask Hjorth Larsen, Jens Jørgen Mortensen, Jakob Blomqvist, Ivano E Castelli, Rune Christensen, Marcin Dułak, Jesper Friis, Michael N Groves, Bjørk Hammer, Cory Hargus, Eric D Hermes, Paul C Jennings, Peter Bjerre Jensen, James Kermode, John R Kitchin, Esben Leonhard Kolsbjerg, Joseph Kubal, Kristen Kaasbjerg, Steen Lysgaard, Jón Bergmann Maronsson, Tristan Maxson, Thomas Olsen, Lars Pastewka, Andrew Peterson, Carsten Rostgaard, Jakob Schiøtz, Ole Schütt, Mikkel Strange, Kristian S Thygesen, Tejs Vegge, Lasse Vilhelmsen, Michael Walter, Zhenhua Zeng, and Karsten W Jacobsen. The atomic simulation environment—a Python library for working with atoms. *J. Phys.: Condens. Matter*, 29(27):273002, July 2017.

[20] Tran Doan Huan, Arun Mannodi-Kanakkithodi, Chiho Kim, Vinit Sharma, Ghanshyam Pilania, and Rampi Ramprasad. A polymer dataset for accelerated property prediction and design. *Sci Data*, 3(1):160012, March 2016.

[21] Jonathan G. Huddleston, Ann E. Visser, W. Matthew Reichert, Heather D. Willauer, Grant A. Broker, and Robin D. Rogers. Characterization and comparison of hydrophilic and hydrophobic room temperature ionic liquids incorporating the imidazolium cation. *Green Chem.*, 3:156–164, 2001.

[22] William Humphrey, Andrew Dalke, and Klaus Schulten. VMD: Visual molecular dynamics. *Journal of Molecular Graphics*, 14(1):33–38, February 1996.

[23] Pieter J. in 't Veld and Gregory C. Rutledge. Temperature-dependent elasticity of a semicrystalline interphase composed of freely rotating chains. *Macromolecules*, 36(19):7358–7365, 2003.

[24] William L. Jorgensen, David S. Maxwell, and Julian Tirado-Rives. Development and testing of the opls all-atom force field on conformational energetics and properties of organic liquids. *Journal of the American Chemical Society*, 118(45):11225–11236, 1996.

[25] William L. Jorgensen and Julian Tirado-Rives. Potential energy functions for atomic-level simulations of water and organic and biomolecular systems. *Proceedings of the National Academy of Sciences*, 102(19):6665–6670, 2005.

[26] Sanket Kadulkar, Zachary W. Brotherton, Anna L. Lynch, Gabriel Pohlman,

Zidan Zhang, Rudy Torres, Arumugam Manthiram, Nathaniel A. Lynd, Thomas M. Truskett, and Venkat Ganesan. The importance of morphology on ion transport in single-ion, comb-branched copolymer electrolytes: Experiments and simulations. *Macromolecules*, 56(7):2790–2800, 2023.

[27] Manoj K. Kesharwani, Amir Karton, Nitai Sylvetsky, and Jan M. L. Martin. The S66 Non-Covalent Interactions Benchmark Reconsidered Using Explicitly Correlated Methods Near the Basis Set Limit. *Aust. J. Chem.*, 71(4):238, 2018.

[28] M. H. Kowsari, Saman Alavi, Mahmud Ashrafizaadeh, and Bijan Najafi. Molecular dynamics simulation of imidazolium-based ionic liquids. I. Dynamics and diffusion coefficient. *The Journal of Chemical Physics*, 129(22), 12 2008. 224508.

[29] Piotr Kubisiak, Piotr Wróbel, and Andrzej Eilmes. Molecular dynamics investigation of correlations in ion transport in metfsi/emim–tfsi (me = li, na) electrolytes. *The Journal of Physical Chemistry B*, 124(2):413–421, 2020. PMID: 31850757.

[30] Guillaume Lamoureux and Benoit Roux. Modeling induced polarization with classical Drude oscillators: Theory and molecular dynamics simulation algorithm. *The Journal of Chemical Physics*, 119(6):3025–3039, August 2003.

[31] J E Lennard-Jones. Cohesion. *Proceedings of the Physical Society*, 43(5):461, sep 1931.

[32] L. Martínez, R. Andrade, E. G. Birgin, and J. M. Martínez. Packmol: A package for building initial configurations for molecular dynamics simulations. *Journal of Computational Chemistry*, 30(13):2157–2164, 2009.

[33] Jesse G. McDaniel, Eunsong Choi, Chang Yun Son, J. R. Schmidt, and Arun Yethiraj. *Ab Initio* Force Fields for Imidazolium-Based Ionic Liquids. *J. Phys. Chem. B*, 120(28):7024–7036, July 2016.

[34] Glen McHale, Chris Hardacre, Rile Ge, Nicola Doy, Ray W. K. Allen, Jordan M. MacInnes, Mark R. Bown, and Michael I Newton. Density viscosity product of small volume ionic liquid samples using quartz crystal impedance analysis. *Analytical Chemistry*, 80(15):5806–5811, 2008. PMID: 18611039.

[35] David Mecerreyes. Polymeric ionic liquids: Broadening the properties and applications of polyelectrolytes. *Progress in Polymer Science*, 36(12):1629–1648, December 2011. Number: 12.

[36] Naveen Michaud-Agrawal, Elizabeth J. Denning, Thomas B. Woolf, and Oliver Beckstein. Mdanalysis: A toolkit for the analysis of molecular dynamics simulations. *Journal of Computational Chemistry*, 32(10):2319–2327, 2011.

[37] Santosh Mogurampelly, Jordan R. Keith, and Venkat Ganesan. Mechanisms Underlying Ion Transport in Polymerized Ionic Liquids. *J. Am. Chem. Soc.*, 139(28):9511–9514, July 2017.

[38] Naomi Nishimura and Hiroyuki Ohno. 15th anniversary of polymerised ionic liquids. *Polymer*, 55(16):3289–3297, 2014. Polymerized Ionic Liquids.

[39] Noel M O'Boyle, Michael Banck, Craig A James, Chris Morley, Tim Vandermeersch, and Geoffrey R Hutchison. Open Babel: An open chemical toolbox. *J Cheminform*, 3(1):33, December 2011.

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[41] James C. Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D. Skeel, Laxmikant Kalé, and Klaus Schulten. Scalable molecular dynamics with namd. *Journal of Computational Chemistry*, 26(16):1781–1802, 2005.

[42] Anne-Laure Rollet, Patrice Porion, Michel Vaultier, Isabelle Billard, Michael Deschamps, Catherine Bessada, and Laurence Jouvensal. Anomalous diffusion of water in [bmim][tfsi] room-temperature ionic liquid. *The Journal of Physical Chemistry B*, 111(41):11888–11891, 2007. PMID: 17887669.

[43] J. R. Sangoro, C. Iacob, A. L. Agapov, Y. Wang, S. Berdzinski, H. Rexhausen, V. Strehmel, C. Friedrich, A. P. Sokolov, and F. Kremer. Decoupling of ionic

conductivity from structural dynamics in polymerized ionic liquids. *Soft Matter*, 10(20):3536–3540, 2014.

[44] Yifan Sha, Tianhao Yu, Tao Dong, Xing-long Wu, Haoyu Tao, and Haitao Zhang. *In Situ* Network Electrolyte Based on a Functional Polymerized Ionic Liquid with High Conductivity toward Lithium Metal Batteries. *ACS Appl. Energy Mater.*, 4(12):14755–14765, December 2021.

[45] J. S. Smith, O. Isayev, and A. E. Roitberg. ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chem. Sci.*, 8(4):3192–3203, 2017.

[46] Justin S. Smith, Benjamin T. Nebgen, Roman Zubatyuk, Nicholas Lubbers, Christian Devereux, Kipton Barros, Sergei Tretiak, Olexandr Isayev, and Adrian E. Roitberg. Approaching coupled cluster accuracy with a general-purpose neural network potential through transfer learning. *Nat Commun*, 10(1):2903, December 2019.

[47] Justin S. Smith, Roman Zubatyuk, Benjamin Nebgen, Nicholas Lubbers, Kipton Barros, Adrian E. Roitberg, Olexandr Isayev, and Sergei Tretiak. The ANI-1ccx and ANI-1x data sets, coupled-cluster and density functional theory properties for molecules. *Sci Data*, 7(1):134, December 2020.

[48] Alexander Stukowski. Visualization and analysis of atomistic simulation data with OVITO-the Open Visualization Tool. *MODELLING AND SIMULATION IN MATERIALS SCIENCE AND ENGINEERING*, 18(1), JAN 2010.

[49] J. Tersoff. Modeling solid-state chemistry: Interatomic potentials for multicomponent systems. *Phys. Rev. B*, 39:5566–5568, Mar 1989.

[50] Aidan P. Thompson, H. Metin Aktulga, Richard Berger, Dan S. Bolintineanu, W. Michael Brown, Paul S. Crozier, Pieter J. in 't Veld, Axel Kohlmeyer, Stan G. Moore, Trung Dac Nguyen, Ray Shan, Mark J. Stevens, Julien Tranchida, Christian Trott, and Steven J. Plimpton. LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Computer Physics Communications*, 271:108171, February 2022.

[51] Adri C. T. van Duin, Siddharth Dasgupta, Francois Lorant, and William A. Goddard. ReaxFF: A Reactive Force Field for Hydrocarbons. *J. Phys. Chem. A*, 105(41):9396–9409, October 2001.

[52] Josh V. Vermaas, David J. Hardy, John E. Stone, Emad Tajkhorshid, and Axel Kohlmeyer. Topogromacs: Automated topology conversion from charmm to gromacs within vmd. *Journal of Chemical Information and Modeling*, 56(6):1112–1116, 2016. PMID: 27196035.

[53] Han Wang, Linfeng Zhang, Jiequn Han, and Weinan E. Deepmd-kit: A deep learning package for many-body potential energy representation and molecular dynamics. *Computer Physics Communications*, 228:178–184, 2018.

[54] Junmei Wang, Romain M. Wolf, James W. Caldwell, Peter A. Kollman, and David A. Case. Development and testing of a general amber force field. *Journal of Computational Chemistry*, 25(9):1157–1174, 2004.

[55] Wujie Wang, Tzuhsiung Yang, William H. Harris, and Rafael Gómez-Bombarelli. Active learning and neural network potentials accelerate molecular screening of ether-based solvate ionic liquids. *Chem. Commun.*, 56:8920–8923, 2020.

[56] David Weininger. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Model.*, 28(1):31–36, February 1988.

[57] Jonathan T. Willman, Kien Nguyen-Cong, Ashley S. Williams, Anatoly B. Belonoshko, Stan G. Moore, Aidan P. Thompson, Mitchell A. Wood, and Ivan I. Oleynik. Machine learning interatomic potential for simulations of carbon at extreme conditions. *Phys. Rev. B*, 106:L180101, Nov 2022.

[58] Zidan Zhang, Jakub Krajniak, and Venkat Ganesan. A Multiscale Simulation Study of Influence of Morphology on Ion Transport in Block Copolymeric Ionic Liquids. *Macromolecules*, 54(11):4997–5010, June 2021.

[59] Zidan Zhang, Amir T. Nasrabadi, Dipak Aryal, and Venkat Ganesan. Mechanisms of Ion Transport in Lithium Salt-Doped Polymeric Ionic Liquid Electrolytes. *Macromolecules*, 53(16):6995–7008, August 2020.

[60] Zidan Zhang, Bill K. Wheatle, Jakub Krajniak, Jordan R. Keith, and Venkat Ganesan. Ion Mobilities, Transference Numbers, and Inverse Haven Ratios of Polymeric Ionic Liquids. *ACS Macro Lett.*, 9(1):84–89, January 2020.

[61] Zidan Zhang, Everett Zofchak, Jakub Krajniak, and Venkat Ganesan. Influence of Polarizability on the Structure, Dynamic Characteristics, and Ion-Transport Mechanisms in Polymeric Ionic Liquids. *J. Phys. Chem. B*, 126(13):2583–2592, April 2022.