Variationally Enhanced Sampling with Permutationally Invariant Collective Variables

Dissertation
Zur Erlangung des Grades
„Doktor der Naturwissenschaften“
im Promotionasfach Chemie

am Fachbereich Chemie, Pharmazie,
Geographie und Geowissenschaften
der Johnanes Gutenberg-Universität Mainz

Bin Song

geb. in Shanxi, China

Mainz, 2022

1. Bericherstatter:
2. Bericherstatter:
Tag der mündlichen Prüfung:

ABSTRACT

Molecular dynamics (MD) simulations have become an indispensable tool in understanding the physical world at a resolution of molecular details. Currently, MD simulations are still limited to the microsecond timescale in most circumstances. When there is a high free energy barrier between metastable states, sampling is easily trapped in a local free energy minimum. A variety of enhanced sampling methods and algorithms have been developed to overcome such issues. In this thesis, we present our work on both developments of algorithms and methods. In terms of algorithm development, we have created an interface between PLUMED 2, a software package that has implemented some of the most prominent enhanced sampling methods, and the MD engine ESPResSo++. We used the combination of the two packages to study the first-order phase transition of the 128 monomer single-chain smooth square-well polymer. In terms of method development, based on the variationally enhanced sampling method, we have created the variationally enhanced sampling with permutationally invariant collective variables method so that such local collective variables can be used in biased simulations. We have demonstrated the effectiveness of the new method in phase transition studies of seven Lennard-Jones particles in two-dimensional space and crystallization of bulk sodium. We have also explored crystallization of ice and urea from melt with the new method and discussed the limitations of the current implementation encountered in these works.

ZUSAMMENFASSUNG

Molekulardynamiksimulationen (MD) sind zu einem unverzichtbaren Instrument für das Verständnis der physikalischen Welt auf molekularer Ebene geworden. Derzeit sind MD-Simulationen in den meisten Fällen noch auf den Mikrosekundenbereich beschränkt. Wenn eine hohe Freie-Energie-Barriere zwischen metastabilen Zuständen existiert, ist das System oft in einem der lokalen Minima der freien Energie gefangen. Es wurden verschiedene Enhanced-Sampling-Methoden und Algorithmen entwickelt, um solche Schwierigkeiten zu überwinden. In dieser Thesis präsentieren wir unsere Forschung welche sowohl Algorithmus- als auch Methodenentwicklung umfasst. Bei der Algorithmusentwicklug haben wir eine Schnittstelle zwischen PLUMED 2, einem Softwarepaket das einige der bekanntesten Enhanced-Sampling-Methoden implementiert, und der MD-Engine ESPResSo++ geschaffen. Die Kombination der beiden Pakete wurde zur Untersuchung des Phasenübergangs erster Ordnung des 128 monomeren, einkettigen, gleichmäßig Square-Well-Polymers genutzt. Auf der Grundlage der Variationally-Enhanced-Sampling-Methode haben wir Variationally-Enhanced-Sampling mit permutationsinvarianten kollektiven Variablen entwickelt, so dass solche lokalen kollektiven Variablen in Enhanced-Sampling-Simulationen verwendet werden können. Wir haben die Wirksamkeit der neuen Methode in Phasenübergangsstudien von sieben Lennard-Jones-Partikeln im zweidimensionalen Raum und Kristallisation von Natrium gezeigt. Wir haben auch die Kristallisation von Eis und Harnstoffs aus der Schmelze mit der neuen Methode untersucht und die Grenzen der derzeitigen Implementierung aufgezeigt.

ACKNOWLEDGMENTS

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

CV	collective variable
EAM	embedded-atom method
FES	free energy surface
FS	Finnis-Sinclair
PICV	permutationally invariant collective variable
MD	molecular dynamics
SCP	single-chain smooth square-well polymer
PBC	periodic boundary condition
US	umbrella sampling
VES	variationally enhanced sampling
WTMetaD	well-tempered metadynamics
WHAM	weighted histogram analysis method

1 INTRODUCTION

1.1 THE ADVENT OF COMPUTER SIMULATIONS

The advent of modern computers towards end of the Second World War has paved the way for high performance computing as a new tool in physical science research in addition to the traditional theoretical and experimental approaches. By expression of the fundamental laws of physics into computational instructions, chemists and physicists alike are able to conduct their experiments in a virtual environment through computer modeling and observe collective behaviors of atoms, ions, molecules or a combination of these species at a particular state of their choosing. To address different scientific questions, various techniques have been developed for different timescales and length-scales as illustrated in Figure 1.1. In this thesis we focus on techniques in empirical MD regime that can be used to investigate physical phenomena such as conformational changes of biomacromolecules, ice crystallization, growth of clathrate hydrates, ligand binding and so forth.

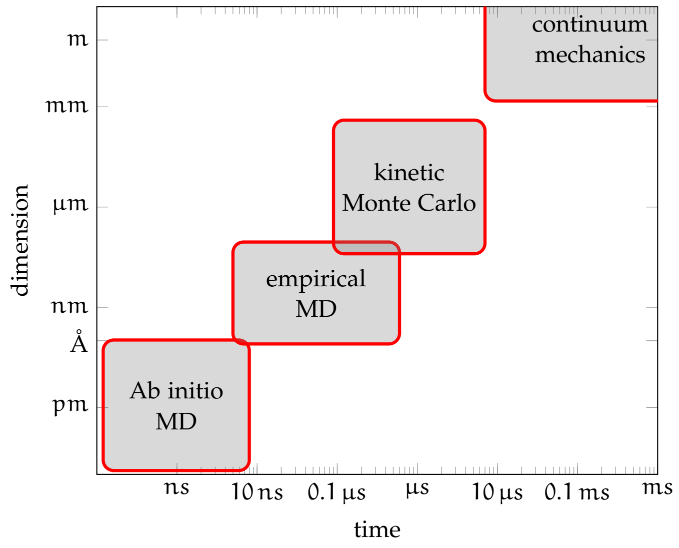


Figure 1.1: The timescale and length-scale of computer simulations of different computer modeling techniques are illustrated by the block in which they reside. The boundary shows the general use cases of a technique of a block rather than a strict limitation of a technique.

With easy controls of the conditions afforded by computer simulations, researchers gain unrivaled resolution of these processes. We can not only glean a peek into mechanisms on a molecular level from trajectories of simulations, but also obtain free energy landscapes and kinetics by employing theories of statistical mechanics and chemical kinetics. The term *in silico*[1] quite aptly reflects the important role that computation increasingly plays.

1.1.1 Monte Carlo

Metropolis *et al.* performed the first computer simulation of a liquid in 1953 on Los Alamos National Laboratory's computer MANIAC using the algorithm that came to be known as Metropolis-Monte-Carlo method[2]. We give a simple example of a Monte Carlo simulation in Figure 1.2. In this example the N particles in a simulation box interact with each other according to certain force fields, of which examples are given in Section 1.3. The potential energy U_o of the system of the original configuration $\mathbf{r}_o = \{\mathbf{r}_1(t), \ldots, \mathbf{r}_N(t)\}$ can be computed based on the chosen force fields and associated parameters. A random particle is then selected, and a move of its position is proposed to create a new configuration $\mathbf{r}_n$. The potential energy U_n of the new configuration is then computed so that the change of the potential energy $\Delta U_{o\to n}$ of the proposed move is known. The proposed move is either accepted or rejected by according to the criteria given in Equation 1.1 and Equation 1.2. The configuration space at time t is $\mathbf{r}(t) = \mathbf{r}_o$. The new configuration is accepted with probability of 1 when the new potential energy is reduced or stay unchanged, i. e.,

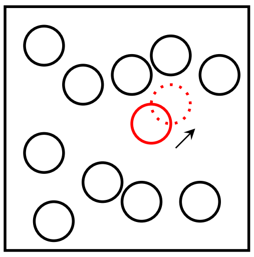


Figure 1.2: An illustration of a simple Monte Carlo simulation, where each solid circle represents a particle in a two-dimensional box and the red circle is randomly selected to move to a new location marked by the dotted red circle. Such a move is either accepted or rejected with a probability per prescription of Equation 1.1 or Equation 1.2.

$$\mathbf{r}(t+1) = \mathbf{r}_n \text{ when } \Delta U_{o\to n} \le 0. \tag{1.1}$$

Otherwise, a random number x in the range from 0 to 1 needs to be drawn. If the $\exp(-\beta\Delta U_{o\to n}) \ge x$, the new state $\mathbf{r}_n$ is accepted with the probability of $\exp(-\beta\Delta U_{o\to n})$. Otherwise, the old state $\mathbf{r}_o$ is retained with the probability of $\exp(-\beta\Delta U_{o\to n})$. This process can be summarized in Equation 1.2:

$$\mathbf{r}(t+1) = \begin{cases} \mathbf{r}_n, & \exp(-\beta\Delta U_{o\to n}) \ge x; \\ \mathbf{r}_o, & \exp(-\beta\Delta U_{o\to n}) < x. \end{cases} \tag{1.2}$$

By repeating aforementioned process we can obtain a trajectory of the configurations that have been visited by a Monte Carlo simulation. These configurations are sampled under the *Boltzmann distribution*, which is the probability distribution of the microscopic states, and to which we give more detailed introductions in Section 1.2.1. The Monte-Carlo-Metropolis algorithm takes samples under the Boltzmann distribution because it follows the requirement of “detailed balance”:

$$p(\mathbf{r}_i)T_{ij} = p(\mathbf{r}_j)T_{ji}. \tag{1.3}$$

In Equation 1.3 $p(\mathbf{r}_i)$ and $p(\mathbf{r}_j)$ are the probabilities of states i and j according to the Boltzmann distribution. T_{ij} is the transition probability from state i to j, and conversely T_{ji} is the probability of the reverse transition.

One can envision that an observable value $\mathcal{A}$ is defined as a function of the configuration space $\mathbf{r}$, i. e., $\mathcal{A}(\mathbf{r}) \equiv f(\mathbf{r})$. $\mathcal{A}(\mathbf{r})$, therefore, can be computed for the entire trajectory, and its probability distribution follows the Boltzmann distribution. The trajectory average of $\mathcal{A}(\mathbf{r})$ corresponds to the experimental value of $\mathcal{A}$.

The design of the moving step is critical for the success of a Monte Carlo simulation. When the moving step (or other types of changes in the configuration) is too little, the acceptance rate will be high. However, the generated trajectory is plagued by correlations between successive frames. In other words, the generated configurations are too similar between each other, and the configuration space is not explored sufficiently. On the other hand, if the moving step is too large, the acceptance rate will be reduced, and the simulation suffers from a grave slowdown as a result. A practitioner of Monte Carlo simulations must find the balance between the acceptance rate and the size of proposed moves.

There are two benefits to Monte Carlo methods. First, there is no need to compute the forces of interactions which saves considerable computation time. Secondly, the Boltzmann distribution can be sampled comparatively easily without the need of specialized algorithms, when the temperature is kept constant.

Monte Carlo methods have several drawbacks too. For example, in Monte Carlo simulations no real dynamics are involved in the process, which is necessary to study diffusion behaviors. Neither are Monte Carlo simulations suitable for studying mechanical properties of a system, where forces of interactions are essential but absent in Monte Carlo simulations. In the next section we introduce another simulation method that is free of these shortcomings and is the main simulation method used in this thesis.

1.1.2 Molecular Dynamics

Four years after the Metropolis-Monte-Carlo method was introduced, molecular dynamics (MD) was developed to study a system of hard spheres by Alder & Wainwright.[3,4] In a MD simulation a box of N particles are initialized with both their positions $\mathbf{r} = \{\mathbf{r}_1(t), \ldots, \mathbf{r}_N(t)\}$

and their velocities or momenta $\mathbf{p} = \{\mathbf{p}_1(t), \ldots, \mathbf{p}_N(t)\}$. The particles in the simulation box move according to classical mechanics in short time steps, which are determined by the fastest vibrational motions in a molecule. They are usually in the femtosecond scale for atomistic simulations. The equations of motions are integrated through short time steps Δt to update the positions and momenta of the system, as shown in Equation 1.4:

$$\begin{aligned} \mathbf{r}(t) &\to \mathbf{r}(t+\Delta t), \\ \mathbf{p}(t) &\to \mathbf{p}(t+\Delta t). \end{aligned} \tag{1.4}$$

Unlike with Monte Carlo simulations, with MD simulations we obtain a trajectory of configurations $\mathbf{r}(t)$ and momenta $\mathbf{p}(t)$, which together form the phase space $(\mathbf{r}, \mathbf{p})$. In MD simulations both the potential energy $U(\mathbf{r}_i)$ and the force $F_i(\mathbf{r}_i) = -\nabla U(\mathbf{r}_i)$ are computed at each time step to update the positions and velocities of all the particles in the system step by step. After the completion of a MD simulation the trajectory that we obtain is a time series of snapshots of the configuration space, from which we can also compute a time series of the observable $\mathcal{A}(\mathbf{r})$.

It is far more complex to achieve correctness and efficiency in a MD simulation than what has been described above. For example, the non-bonded interactions including Lennard-Jones interactions usually are truncated, i. e., a pair of particles interact only when they are separated within a cutoff distance. The consequence of the truncation of force fields is that all pairwise interactions do *not* need to be computed and significant savings of computation power. In a MD simulation the number of particles typically range from a few hundred to a few million atoms, which is nowhere near to the size of a bulk system. Hence, periodic boundary conditions are deployed to emulate a bulk environment. The algorithm to propagate the dynamics of a MD simulation is called the integrator algorithm. A requirement for an integrator is that it conserves the total energy of the system. An algorithm that keeps the temperature constant is called a thermostat, and analogously a barostat keeps the pressure constant in a MD simulation. Before we dive into the topics of these algorithms of molecular dynamics, we first take a detour into statistical thermodynamics in Section 1.2. After that we visit the topic of force fields in Section 1.3, and then we move onto topics of the integrator, periodic boundary conditions, and finally the thermostat and the barostat in later sections.

Before we get into statistical thermodynamics, we briefly mention a few open source software packages that are most relevant to this thesis. They are LAMMPS[5], GROMACS[6–13], ESPResSo++[14,15]. LAMMPS is written mostly in high performance C++, which is developed and maintained by Sandia National Laboratories. In addition to the common force fields, it has also implemented three-body interactions such as the Stillinger-Weber potential and embedded atom models, so that LAMMPS is popular in material science research. GROMACS is another highly performant MD package, which has implemented atomistic force fields AMBER, CHARMM, and the popular coarse-grained MARTINI[16] force fields. It is extensively used in simulations of biomolecules. ESPResSo++ is developed and maintained by the Theory Group of the Max Planck Institute for Polymer Research, Mainz. It has a flexible Python interface that exposes positions, forces and other thermodynamics parameters to users for them to easily monitor or manipulate the simulations. The performance sensitive algorithms in ESPResSo++ are still implemented with C++. Besides these MD engines another important software library is PLUMED[17–19], which can not only be used for enhanced sampling with MD engines in the real time of the simulations, but for the analysis of simulation trajectories in postprocessing as well.

1.2 STATISTICAL THERMODYNAMICS

MD simulations can be easily parallelized to either achieve a massive simulation with millions of atoms in size or to create a large amount of trajectories of smaller sizes simultaneously. The longest simulations have reached millisecond in time.[20] As a result, a large amount of data can be generated from the simulations. We rely on the principles of statistical thermodynamics to extract relevant information from the large amount of trajectories generated from MD simulations. Fortunately the development of statistical thermodynamics predates the development of computer simulations. Statistical thermodynamics connects the laws of classical thermodynamics to its microscopic origins, and macroscopic observables to the probability distributions of microscopic states.

1.2.1 Thermodynamic Ensembles and Partition Functions

To build a connection between the microscopic states and the macroscopic states, we first define the term "ensemble". An ensemble is an imaginary collection of atoms and molecules with a certain set of constraints of the thermodynamic parameters applied to the collection. The constrained thermodynamic parameters could include the number of particles N, the volume V, the temperature T, the pressure P, the chemical potential μ, the internal energy E and so on. The most common ensembles include microcanonical ensemble (or the NVE ensemble), the canonical ensemble (or the NVT ensemble), the isobaric-isothermal ensemble (or the NPT ensemble), and the grand canonical ensemble (or the μVT ensemble). The letters in the parentheses indicate the constrained thermodynamic parameters in each ensemble. Once the set of the constrained thermodynamic parameters and their values are decided, a hypersurface of the microscopic states that are permitted in the phase space by the constrained parameters is then determined for the ensemble. Moreover, the probability distribution of the permitted microscopic states on the hypersurface is also determined. The Boltzmann distribution states that the probability density of a microscopic state p is related to the internal energy E and temperature T as in that $p \propto \exp(-\frac{E}{k_B T})$. The internal energy of a system or the Hamiltonian $\mathcal{H}$ in Hamiltonian dynamics is expressed as the sum of the kinetic energy K and the potential energy U:

$$\begin{aligned} \mathcal{H}(\mathbf{r}_1, \ldots, \mathbf{r}_N, \mathbf{p}_1, \ldots, \mathbf{p}_N) &= \sum_{i=1}^{N} \frac{\mathbf{p}_i^2}{2m_i} + U(\mathbf{r}_1, \ldots, \mathbf{r}_N) \\ &= K + U \\ &= E. \end{aligned} \tag{1.5}$$

Therefore the Boltzmann distribution can also be written as $p \propto \exp[-\frac{\mathcal{H}(\mathbf{r},\mathbf{p})}{k_B T}]$.

The microcanonical ensemble (NVE) is conceptually the simplest ensemble, with the number of particles N, the volume V and the internal energy E kept constant. According to the Boltzmann distribution, each state is equally probable as all microscopic states have the same internal energy, or their internal energy is within a very narrow range of each other in practice.[21] The total number of microscopic states of an NVE ensemble can be computed with a Kronecker delta function:

$$Q_{NVE} = \sum_{(\mathbf{r},\mathbf{p})} \delta[\mathcal{H}(\mathbf{r}, \mathbf{p}) - E]. \tag{1.6}$$

Q_{NVE} is called the partition function of a microcanonical ensemble. The entropy S, which describes the order or disorder of the system, of a microcanonical ensemble thus according to Boltzmann is

$$S = -k_B \ln Q_{NVE}. \tag{1.7}$$

A more relevant ensemble is the canonical ensemble (NVT), in which the following thermodynamic parameters, the number of particles N, the volume V and the temperature T are kept constant. The Hamiltonian $\mathcal{H}$ is no longer a constant. The canonical partition function is

$$Q_{NVT} = \sum_{(\mathbf{r},\mathbf{p})} \exp[-\frac{\mathcal{H}(\mathbf{r}, \mathbf{p})}{k_B T}]. \tag{1.8}$$

The probability of a microscopic state $(\mathbf{r}, \mathbf{p})$ can be computed with the partition function according to the Boltzmann distribution:

$$p(\mathbf{r}, \mathbf{p}) = \frac{\exp\left(-\frac{\mathcal{H}(\mathbf{r},\mathbf{p})}{k_B T}\right)}{Q_{NVT}}. \tag{1.9}$$

The macroscopic energy E is the ensemble average of the Hamiltonian ($\langle\mathcal{H}\rangle$):

$$\begin{aligned} E &= \frac{\int \mathcal{H} \exp\left(-\frac{\mathcal{H}}{k_B T}\right)}{Q_{NVT}} \\ &= -\left(\frac{\partial}{\partial\beta} \ln Q_{NVT}\right)_{N,V}, \end{aligned} \tag{1.10}$$

in which the inverse temperature $\beta = (k_B T)^{-1}$. The detailed derivations of Equation 1.10 can be found in Tuckerman [22, section 4.3]. Again without going into detailed derivations, the Helmholtz free energy A, not to be confused with an observable $\mathcal{A}$, and the entropy S, the pressure P can be computed using the following equations:

$$A = -k_B T \ln Q_{NVT}, \tag{1.11}$$

$$S = k_B \ln Q_{NVT} + k_B T \left(\frac{\partial \ln Q_{NVT}}{\partial T}\right)_{N,V}, \tag{1.12}$$

$$P = k_B T \left(\frac{\partial \ln Q_{NVT}}{\partial V}\right)_{N,T}. \tag{1.13}$$

Another common ensemble is the isobaric-isothermal ensemble (NPT), where the pressure P is also kept constant. The probability distribution is now proportional to $\exp(-\frac{\mathcal{H}+PV}{k_B T})$. The partition function of the NPT ensemble is

$$Q_{NPT} = \frac{1}{V_0} \int_0^\infty dV\, e^{-\beta PV} Q_{NVT}. \tag{1.14}$$

The Gibbs free energy G, and average volume $\langle V\rangle$, enthalpy H, which should not be confused with the Hamiltonian $\mathcal{H}$, are:

$$G = -k_B T \ln Q_{NPT}, \tag{1.15}$$

$$\langle V\rangle = -k_B T \left(\frac{\partial \ln Q_{NVT}}{\partial P}\right)_{N,T}, \tag{1.16}$$

$$H = \langle\mathcal{H}\rangle + P\langle V\rangle. \tag{1.17}$$

The relationships of macroscopic thermodynamic parameters and the partition functions for other ensembles can be found in Tuckerman [22], McQuarrie [23] and Allen & Tildesley [24]. These relationships demonstrate the microscopic origins of macroscopic behaviors. However, to compute the partition functions analytically is impractical but for the simplest systems. Sampling through computer simulations has become indispensable in finding numerical solutions to these equations.

1.2.2 Observables and Ensemble Averages

An Observable $\mathcal{A}$, strictly speaking, is a function of the microscopic state or the phase space $(\mathbf{r}, \mathbf{p})$ of the system. The average value of $\mathcal{A}$, which corresponds to the experimental value of $\mathcal{A}$, can be computed as the ensemble average of $\mathcal{A}$. In the case of a canonical ensemble:

$$\langle \mathcal{A} \rangle = \frac{\iint_{\mathbf{r},\mathbf{p}} d\mathbf{r} d\mathbf{p}\, \mathcal{A}(\mathbf{r},\mathbf{p}) \exp\left(-\frac{\mathcal{H}(\mathbf{r},\mathbf{p})}{k_B T}\right)}{Q_{NVT}} \tag{1.18}$$

More often than not, though, $\mathcal{A}$ is defined as a function only of the configuration space $\mathbf{r}$. The ensemble average $\langle \mathcal{A} \rangle$ of a canonical ensemble, for example, can be computed with the configurational integral Z instead of the full partition function Q_{NVT}. The configurational integral Z, also called the configurational partition function, is computed according to

$$Z = \int d\mathbf{r} \exp\left(-\frac{U(\mathbf{r})}{k_B T}\right). \tag{1.19}$$

Now the ensemble average of observable $\mathcal{A}$ is

$$\langle \mathcal{A} \rangle = \frac{\int d\mathbf{r}\, \mathcal{A}(\mathbf{r}) \exp\left(-\frac{U(\mathbf{r})}{k_B T}\right)}{Z} = \int d\mathbf{r}\, \mathcal{A}(\mathbf{r}) p(\mathbf{r}), \tag{1.20}$$

in which $p(\mathbf{r}) = \frac{e^{-\frac{U(\mathbf{r})}{k_B T}}}{Z}$ is the Boltzmann distribution of the configuration $\mathbf{r}$.

In the case of Monte Carlo simulations, the Boltzmann distribution is directly sampled by the simulations due to *detailed balance*, and the ensemble average of an observable is the probability-weighted average of the observable from the simulations. In the case of MD simulations, as long as ergodicity is upheld in the simulations, in other words, a possible microscopic state can be visited given enough time in the simulation regardless of the initial state of a simulation, an ensemble average can be computed as the time average in an MD simulation provided enough sampling. However, due to the limitation of the timescale of MD simulations, the full phase space can not be exhaustively sampled. This situation is referred to as "quasi-nonergodicity".[25] In Chapter 2 we will introduce enhanced sampling methods that tackle this issue.

The difference in the Helmholtz free energy A (or the Gibbs free energy G) between two thermodynamic states is the reversible work done to or by the system when it transitions between the two states reversibly. However, we are often more interested in the free energy landscape or free energy surface (FES) of a thermodynamic state rather than the overall free energy in a computer simulation.

To characterize a free energy surface, we first define the term collective variable (CV). A collective variable is a smooth differentiable function of the coordinates or the configuration space, whose derivative with respect to the coordinates is also smooth. Throughout this thesis, we use the small letter s for a single CV and the bold small $\mathbf{s}$ for multiple CVs. In short,

$$f \in C^1(\mathbf{r}, s). \tag{1.21}$$

A CV must be able to distinguish different metastable states for it to be useful in describing the free energy landscape of a thermodynamic state. The free energy surface $F(s)$ is a function of the CVs, and it is related to the probability distribution of s, $p(s)$ by

$$F(s) = -\frac{1}{\beta} \log p(s). \tag{1.22}$$

The probability distribution $p(s)$ is related to $p(\mathbf{r})$ by

$$p(s) = \int d\mathbf{r}\, \delta[s - s(\mathbf{r})] p(\mathbf{r}), \tag{1.23}$$

where the delta function $\delta[s - s(\mathbf{r})] = 0$, when $s(\mathbf{r}) = s$. Otherwise, it is equal to 0. Similarly with multiple CVs, the probability density $p(\mathbf{s})$ is

$$p(\mathbf{s}) = \int d\mathbf{r}\, \delta[\mathbf{s} - \mathbf{s}(\mathbf{r})] p(\mathbf{r}). \tag{1.24}$$

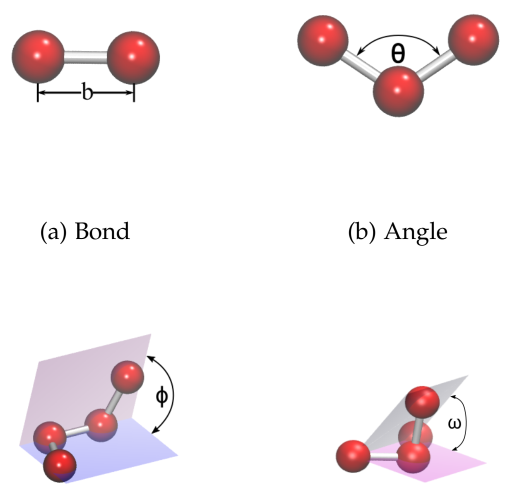
1.3 FORCE FIELDS

Force fields used in MD simulations should accurately represent the potential energy surface and usually do so with pairwise additive functions. A force field defines both the function form and related parameters. The functions of the force fields must be differentiable and smooth so that the forces acting on the atoms are continuous and smooth. For example, in Section 1.3.2 a smooth square-well potential is developed for MD simulations.

1.3.1 Force Fieds for Biomolecules

The interactions of molecular systems, which are mainly bonded interactions and long-ranged nonbonded interactions in MD simulations, are modeled by some of most commonly used force fields. In atomistic simulations of biomolecules, the force fields typically are comprised of the following terms:

$$\begin{aligned} U = &\sum_{\textit{bonds}} k_b (b - b_0)^2 + \sum_{\textit{angles}} k_\theta (\theta - \theta_0)^2 \\ &+ \sum_{\textit{dihedrals}} k_\phi [1 + \cos(n\phi - \delta)] + \sum_{\textit{impropers}} k_\omega (\omega - \omega_0)^2 \\ &+ \sum_{\textit{nonbonded}} \left(4\varepsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right] + \frac{q_i q_j}{\varepsilon r_{ij}} \right), \end{aligned} \tag{1.25}$$



(a) Bond (b) Angle

(c) Dihedral (d) Improper

Figure 1.3: Four types of bonded interactions.

in which the bonded interactions include the two types of vibrations — bonds and angles, and the two types of torsions — dihedral angles and improper angles. In Equation 1.25, three of the four bonded interactions are modeled by a harmonic potential. And the dihedral angle is modeled by the CHARMM style dihedral potential. Several other types of potentials for bonded interactions are available for MD practitioners as well. The four types of bonded interactions are illustrated in Figure 1.3. The nonbonded interactions include the van der Waals interaction, which is modeled by the 12-6 style Lennard-Jones potential here, and the electrostatic coulombic interaction. The Lennard-Jones potential is plotted in Figure 1.4.

The costliest computations in MD simulations are the computations of nonbonded interactions due to their long-ranged nature. Both Lennard-Jones interactions and electrostatic interactions can be expressed as power series of the separation r^{-1} between two interacting atoms, and they have a long tail before approaching zero.[26] Therefore, both Lennard-Jones interactions and electrostatic interactions are truncated in practice, i. e., beyond a cutoff distance the Lennard-Jones interaction between two atoms is either left not computed or shifted to zero. Electrostatic interactions beyond the cutoff can be treated with particle mesh approaches or Ewald summation.[27]

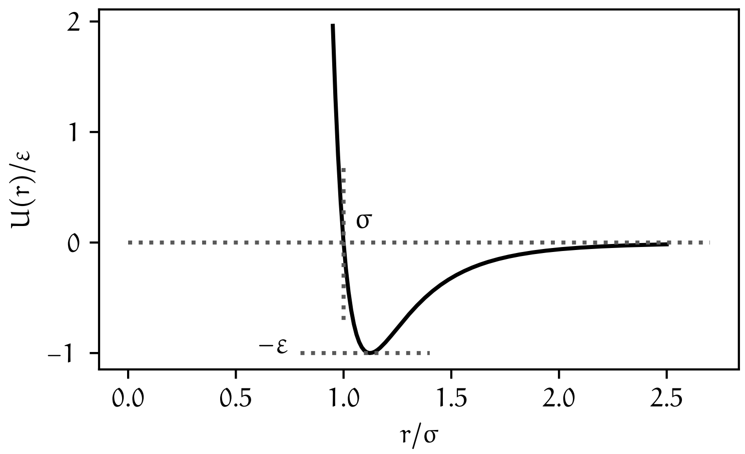


Figure 1.4: The shape of the 12-6 style Lennard-Jones potential.

1.3.2 Smooth Square-Well Potential

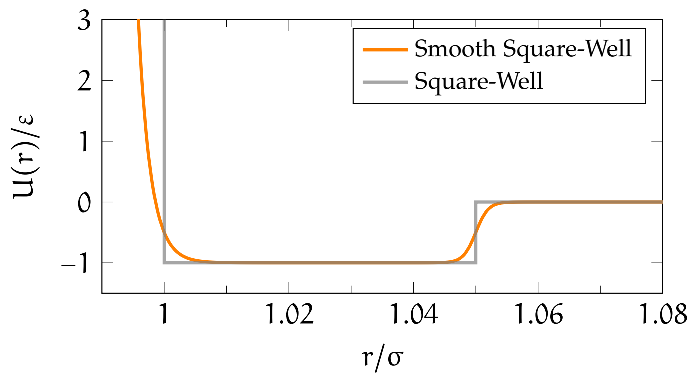
The smooth square-well potential is a specialty potential designed by Leitold & Dellago to represent the square-well interaction between nonbonded pairs of a square-well polymer for MD simulations.[28] In Section 3.2 we will have more detailed discussions of the phase behaviors of such a polymer. The function form of the smooth square-well potential is

$$U(r) = \frac{\varepsilon}{2}\left\{\exp\left[\frac{-(r-\sigma)}{a}\right] + \tanh\left[\frac{r-\lambda\sigma}{a}\right] - 1\right\}, \tag{1.26}$$

in which r is the separation between a pair of nonbonded subunits, and ε controls the depth of the well, and σ and λ control the width of the well, and a is related to the steepness of the repulsive part of the potential. The smooth square-well potential with the parameters listed in Table 1.1 is shown in the figure below. The smooth square-well potential in orange is juxtaposed to the square-well potential in gray, illustrating their resemblance.

ε	σ	λ	a
1	1	1.05	0.002

Table 1.1: Parameters used in the smooth square-well potential.



The smooth square-well potential and the square-well potential.

1.3.3 the Embedded-Atom Method Potential

The embedded-atom method (EAM) potentials are used in MD simulations to model bulk metal alloys. Wilson *et al.* developed an EAM potential based on Finnis-Sinclair (FS) scheme for bulk sodium to study the solid-liquid interface free energy,[29] which we use to study the free energy of crystallization of bulk sodium. The function form of the EAMs potential of Finnis-Sinclair scheme is shown here:

$$E_i = F_\alpha\left(\sum_{i\neq j}\rho_{\alpha\beta}(r_{ij})\right) + \frac{1}{2}\sum_{i\neq j}\phi_{\alpha\beta}(r_{ij}), \tag{1.27}$$

where r_{ij} is the distance between two atoms i and j, and $\phi_{\alpha\beta}(r_{ij})$ is a two-body potential describing the repulsive interaction between two atoms. F_α is the required energy to embed an atom i in the electron cloud at its current location. The electron density of the electron cloud is described by $\rho_{\alpha\beta}$, which has contributions from all atoms within a cutoff distance of atom i but the atom i itself. Furthermore, the contributions are dependent on the species of both atom i and j in the FS scheme. The values for the parameters in Equation 1.27 are listed in a tabularized form for MD engines to look up the during the simulations. An abridged table of Wilson's sodium potential[29] is shown in Figure 1.5, where the first five lines and last two lines are shown. When the distance r_{ij}, or the electron density $\rho_{\alpha\beta}$ fall between values in the table, the cubic spline interpolation is used to compute the values of repulsions $\phi_{\alpha\beta}(r_{ij})$ and embedding energy F_α.

```
1  Na
10000   1.00000000000000E-0002  10000   9.20000000000000E-0004   9.20000000000000E+0000
11   2.29897700000000E+0001   4.22786798098572E+0000  bcc
0  -1.00000000000000E-0001  -1.41421356237310E-0001  -1.73205080756888E-0001  -2.00000000000000E-0001
-2.23606797749979E-0001  -2.44948974278318E-0001  -2.64575131106459E-0001  -2.82842712474619E-0001  -3.00000000000000E-0001
                                     ⋮                           ⋮                           ⋮
                                     ⋮                           ⋮                           ⋮
4.25498791185477E-0010   2.79906382168121E-0010   1.75204912166188E-0010   1.02972484952037E-0010   5.57283103367888E-0011
2.69458489853916E-0011   1.10660324784449E-0011   3.51055884905364E-0012   6.95263818452957E-0013   4.35679574858516E-0014
```

Figure 1.5: Embedded-atom method potential of sodium in FS scheme. The first line lists the number of elements and their symbols. The second line lists number of $\rho_{\alpha\beta}$ values, $d\rho_{\alpha\beta}$, number of r_{ij}, dr_{ij}, and the cutoff distance. The third line includes atomic number, mass, lattice constant, and lattice type. The rest of file are lines of five values in non-delineated blocks of embedding energy F_α corresponding to different electron density, electron density ρ_α corresponding to the distance r_{ij}, and finally the pairwise repulsions to distance.

1.4 THE VELOCITY-VERLET INTEGRATOR

In MD simulations all particles move in short time steps (Δt) according to the Newton's equations of motion. An ideal integrator would be both speedy and accurate, however, compromises often have to be made between the two requirements. To obtain a scheme to integrate Newton's equations of motion, we start with a Taylor expansion of the positions of a particle i about time t,

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t)\Delta t + \frac{\mathbf{F}_i(t)}{2m_i}\Delta t^2 + \frac{\Delta t^3}{3!}\dddot{\mathbf{r}} + \mathcal{O}(\Delta t^4). \quad (1.28)$$

With a higher order of Taylor expansion the integrator is more accurate, but the computation becomes more complex with higher orders of derivatives, which also slows down the computation and requires higher usage of the memory.[30]

It has been found that the second order of the Taylor expansion in Equation 1.28 is sufficient to achieve the conservation of energy and linear momentum. we rewrite it up to the second order Taylor expansion and drop the index i for the benefit of conciseness as:

$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{\mathbf{F}(t)}{2m}\Delta t^2. \quad (1.29)$$

Now with Equation 1.29 we can already propagate the positions $\mathbf{r}$ of the particles, and we just need a way to propagate the velocities $\mathbf{v}$. It should be noted that the Taylor expansion is reversible in time, therefore $r(t)$ can be rewritten as

$$\begin{aligned}\mathbf{r}(t) &= \mathbf{r}([t+\Delta t]-\Delta t) \\ &= \mathbf{r}(t+\Delta t) + \mathbf{v}(t+\Delta t)(-\Delta t) + \frac{\mathbf{F}(t+\Delta t)}{2m}\Delta t^2.\end{aligned} \quad (1.30)$$

Replacing the $\mathbf{r}(t+\Delta t)$ in Equation 1.30 with Equation 1.29 and after some rearrangements, we can update the velocities $\mathbf{v}$ through:

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \Delta t\left(\frac{\mathbf{F}(t)+\mathbf{F}(t+\Delta t)}{2m}\right). \quad (1.31)$$

Even though we can use Equation 1.31 directly to update the velocities, it means that we have to keep two copies of the forces in memory, at time t and $t+\Delta t$, respectively. Instead we first define $\mathbf{v}(t+\frac{1}{2}\Delta t)$

$$\mathbf{v}(t+\frac{1}{2}\Delta t) \equiv \mathbf{v}(t) + \frac{\mathbf{F}(t)}{2m}\Delta t, \quad (1.32)$$

and plug it into Equation 1.31 and Equation 1.29, and we obtain the most common form of the Verlet integrator — the velocity-Verlet integrator[31] as follows,

$$\mathbf{v}(t+\frac{1}{2}\Delta t) = \mathbf{v}(t) + \Delta t\frac{\mathbf{F}(t)}{2m}, \quad (1.33)$$

$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \Delta t\cdot\mathbf{v}(t+\frac{1}{2}\Delta t), \quad (1.34)$$

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t+\frac{1}{2}\Delta t) + \Delta t\frac{\mathbf{F}(t+\Delta t)}{2m}. \quad (1.35)$$

The velocity-Verlet integrator advances the velocities first by half a step and then the positions by a full step, and finally the velocities are updated again by half a step with the current forces, eliminating the need to maintain forces of two time steps in the memory. It has been found that although in short time the velocity-Verlet integrator produces somewhat energy fluctuations, in long time the drift in energy is quite minimal.[30]

There are other forms of Verlet integrators, for example, the classical form of Verlet integrator and "leap frog" form of Verlet integrator. Despite the different forms they are essentially the same.

1.5 PERIODIC BOUNDARY CONDITIONS

Periodic boundary conditions (PBCs) are a set of conditions for the simulation box in a MD simulation to mimic a homogeneous bulk environment and eliminate boundaries or surfaces. To illustrate this concept, a two-dimensional example is shown in Figure 1.6. In this case the central cell containing four water molecules is replicated in both directions to create eight periodic images with the water molecules occupying the same locations in their respective image cell as they do in the central cell. Similarly in a three-dimensional system with periodic boundary conditions the central cell has 26 periodic images. With PBCs the computation of the potential energy and the

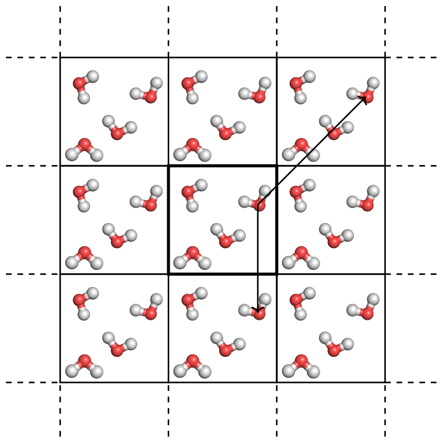


Figure 1.6: A two-dimensional illustration of periodic boundary conditions. The central cell in the bold rectangle is the initial image containing four water molecules, which is surrounded by its eight periodic images. Two arrows point to the periodic images of the water molecule.

forces must take into account the particles in the periodic images:

$$U = \frac{1}{2} \sum_{i,j,n} {}' u(|\mathbf{r}_{ij} + \mathbf{n}L|), \tag{1.36}$$

where L is the length of the cubic box, and $\mathbf{n}$ is a vector designating the periodic image, and i and j are the indices of atoms. The prime in Equation 1.36 indicates that $i \neq j$ when $\mathbf{n} = 0$.

Usually the force fields are truncated when distance between two atoms are outside the cutoff distance r_c, which means that computing all pairwise interactions is inefficient. Instead, potentials and forces are computed according to the *minial-image* convention, where only atoms that fall within the cutoff distance are accounted.

1.6 THERMOSTATS AND BAROSTATS

1.6.1 the Stochastic Rescaling Thermostat

The thermostat controls the temperature of a system so that MD simulations can sample the NVT and the NPT ensembles. The thermostat used in most simulations in this work is the stochastic rescaling thermostat developed by Bussi & Parrinello.[32] The velocities are modified by a scaling factor every step after the integration step performed by an integrator e. g., the velocity-Verlet integrator. For the target inverse temperature β, the average kinetic energy $\bar{K} = \frac{N_f}{2\beta}$, and N_f is the degree of freedom. The dynamics of kinetic energy K of the stochastic rescaling thermostat can be written as the following equation:

$$dK = (\bar{K} - K)\frac{dt}{\tau} + 2\sqrt{\frac{K\bar{K}}{N_f}}\frac{dW}{\sqrt{\tau}}, \tag{1.37}$$

in which dt is the time step, τ is a relaxation parameter chosen by the user with the dimension of time, and lastly dW is a Wiener noise. The stochastic rescaling thermostat modifies the velocities with the same scaling factor, no additonal self-consistency procedures are needed to enforce rigid bond constraints.

1.6.2 the Parrinello-Rahman Barostat

The real world experiments are carried out under a constant pressure rather than a constant volume. To simulate the constant pressure, the volume of the system box has to be adjusted in the process. In other words, the volume V is a dynamic variable in constant-pressure simulations. A three-dimensional unit cell can be represented by the vectors $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$, as illustrated by Figure 1.7. The volume $V = \mathbf{a} \cdot \mathbf{b} \times \mathbf{c}$. The three components can be written in a 3×3 cell matrix $\mathbf{h}$ as

$$\mathbf{h} = \begin{pmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{pmatrix}. \tag{1.38}$$

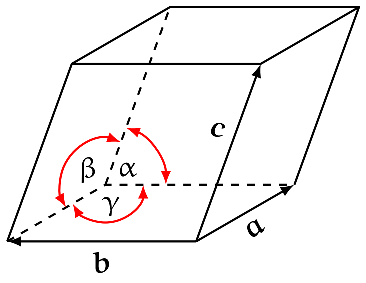


Figure 1.7: An example of a triclinic simulation box.

The volume V is equal to det($\mathbf{h}$), the determinant of the cell matrix $\mathbf{h}$. With cell matrix $\mathbf{h}$, the absolute coordinates of atoms can be replaced with a product of relative coordinates with the cell matrix so that the box size volume can evolve in the dynamics. In the Parrinello-Rahman barostat[33] the cell matrix evolves according to

$$\frac{d\mathbf{h}^2}{dt^2} = V\mathbf{W}^{-1}\mathbf{h}'^{-1}(\mathbf{P} - \mathbf{P}_{tg}), \tag{1.39}$$

where $\mathbf{W}$ is a matrix parameter that determines the strength of the coupling. The matrices $\mathbf{P}$ and $\mathbf{P}_{tg}$ are the current pressure and the target pressure, respectively. With Parrinello-Rahman barostat the dimensions of the unit cell can be evolved isotropically or anisotropically.

REFERENCES

1. Miramontes, P. Un Modelo de Autómata Celular Para La Evolución de Los Ácidos Nucleicos [A Cellular Automaton Model for the Evolution of Nucleic Acids] (1992).
2. Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics* **21,** 1087–1092 (June 1, 1953).
3. Alder, B. J. & Wainwright, T. E. Phase Transition for a Hard Sphere System. *The Journal of Chemical Physics* **27,** 1208–1209 (Nov. 1, 1957).
4. Alder, B. J. & Wainwright, T. E. Studies in Molecular Dynamics. I. General Method. *The Journal of Chemical Physics* **31,** 459–466 (Aug. 1, 1959).
5. Plimpton, S. Fast Parallel Algorithms for Short-Range Molecular Dynamics. *Journal of Computational Physics* **117,** 1–19 (Mar. 1, 1995).

6. Abraham, M. J. (, Murtola, T., Schulz, R., Páll, S., Smith, J. C., Hess, B. & Lindahl, E. GROMACS: High Performance Molecular Simulations through Multi-Level Parallelism from Laptops to Supercomputers. *SoftwareX* (Sept. 1, 2015).

7. Berendsen, H., van der Spoel, D. & van Drunen, R. GROMACS: A Message-Passing Parallel Molecular Dynamics Implementation. *Computer Physics Communications* **91,** 43–56 (Sept. 2, 1995).

8. Hess, B., Kutzner, C., van der Spoel, D. & Lindahl, E. GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation. *Journal of Chemical Theory and Computation* **4,** 435–447 (Mar. 1, 2008).

9. Lindahl, E., Hess, B. & van der Spoel, D. GROMACS 3.0: A Package for Molecular Simulation and Trajectory Analysis. *Molecular modeling annual* **7,** 306–317 (Aug. 1, 2001).

10. Páll, S., Abraham, M. J., Kutzner, C., Hess, B. & Lindahl, E. *Tackling Exascale Software Challenges in Molecular Dynamics Simulations with GROMACS* in *Solving Software Challenges for Exascale* (Springer International Publishing, Cham, 2015), 3–27.

11. Pronk, S. *et al.* GROMACS 4.5: A High-Throughput and Highly Parallel Open Source Molecular Simulation Toolkit. *Bioinformatics (Oxford, England)* **29,** 845–854 (Apr. 1, 2013).

12. Van Der Spoel, D., Lindahl, E., Hess, B., Groenhof, G., Mark, A. E. & Berendsen, H. J. C. GROMACS: Fast, Flexible, and Free. *Journal of Computational Chemistry* **26,** 1701–1718 (Dec. 2005).

13. BEKKER, H., BERENDSEN, H., DIJKSTRA, E., ACHTEROP, S., VONDRUMEN, R., VANDERSPOEL, D., SIJBERS, A., Keegstra, H. & RENARDUS, M. GROMACS - A PARALLEL COMPUTER FOR MOLECULAR-DYNAMICS SIMULATIONS: 4th International Conference on Computational Physics (PC 92). *PHYSICS COMPUTING '92,* 252–256 (1993).

14. Guzman, H. V., Tretyakov, N., Kobayashi, H., Fogarty, A. C., Kreis, K., Krajniak, J., Junghans, C., Kremer, K. & Stuehn, T. ESPResSo++ 2.0: Advanced Methods for Multiscale Molecular Simulation. *Computer Physics Communications* **238,** 66–76 (May 1, 2019).

15. Halverson, J. D., Brandes, T., Lenz, O., Arnold, A., Bevc, S., Starchenko, V., Kremer, K., Stuehn, T. & Reith, D. ESPResSo++: A Modern Multiscale Simulation Package for Soft Matter Systems. *Computer Physics Communications* **184,** 1129–1149 (Apr. 1, 2013).

16. Marrink, S. J., de Vries, A. H. & Mark, A. E. Coarse Grained Model for Semiquantitative Lipid Simulations. *The Journal of Physical Chemistry B* **108,** 750–760 (Jan. 1, 2004).

17. Bonomi, M. *et al.* PLUMED: A Portable Plugin for Free-Energy Calculations with Molecular Dynamics. *Computer Physics Communications* **180,** 1961–1972 (Oct. 1, 2009).
18. Bonomi, M. *et al.* Promoting Transparency and Reproducibility in Enhanced Molecular Simulations. *Nature Methods* **16,** 670–673 (Aug. 1, 2019).
19. Tribello, G. A., Bonomi, M., Branduardi, D., Camilloni, C. & Bussi, G. PLUMED 2: New Feathers for an Old Bird. *Computer Physics Communications* **185,** 604–613 (Feb. 2014).
20. Shaw, D. E. *et al.* *Anton 3: Twenty Microseconds of Molecular Dynamics Simulation before Lunch* in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Association for Computing Machinery, New York, NY, USA, Nov. 14, 2021), 1–11.
21. Chandler, D. & Wu, D. *Introduction to Modern Statistical Mechanics* (Oxford University Press, 1987).
22. Tuckerman, M. *Statistical Mechanics: Theory and Molecular Simulation* (OUP Oxford, 2010).
23. McQuarrie, D. *Statistical Mechanics* (University Science Books, 2000).
24. Allen, M. P. & Tildesley, D. J. *Computer Simulation of Liquids: Second Edition* 2nd ed. 640 pp. (Oxford University Press, Oxford, 2017).
25. Hénin, J., Lelièvre, T., Shirts, M. R., Valsson, O. & Delemotte, L. *Enhanced Sampling Methods for Molecular Dynamics Simulations* Feb. 8, 2022. arXiv: 2202.04164 [cond-mat, physics:physics].
26. Stone, A. *The Theory of Intermolecular Forces* 2nd ed. 352 pp. (Oxford University Press, Oxford, 2013).
27. Frenkel, D. & Smit, B. in *Understanding Molecular Simulation (Second Edition)* 291–320 (Academic Press, San Diego, Jan. 1, 2002).
28. Leitold, C. & Dellago, C. Folding Mechanism of a Polymer Chain with Short-Range Attractions. *The Journal of Chemical Physics* **141,** 134901 (Oct. 7, 2014).
29. Wilson, S. R., Gunawardana, K. G. S. H. & Mendelev, M. I. Solid-Liquid Interface Free Energies of Pure Bcc Metals and B2 Phases. *The Journal of Chemical Physics* **142,** 134705 (Apr. 7, 2015).
30. Frenkel, D. & Smit, B. in *Understanding Molecular Simulation (Second Edition)* 63–107 (Academic Press, San Diego, Jan. 1, 2002).
31. Swope, W. C., Andersen, H. C., Berens, P. H. & Wilson, K. R. A Computer Simulation Method for the Calculation of Equilibrium Constants for the Formation of Physical Clusters of Molecules: Application to Small Water Clusters. *The Journal of Chemical Physics* **76,** 637–649 (Jan. 1, 1982).

32. Bussi, G. & Parrinello, M. Accurate Sampling Using Langevin Dynamics. *Physical Review E* **75** (May 25, 2007).

33. Parrinello, M. & Rahman, A. Polymorphic Transitions in Single Crystals: A New Molecular Dynamics Method. *Journal of Applied Physics* **52,** 7182–7190 (Dec. 1, 1981).

2 ENHANCED SAMPLING METHODS

Although the computing capabilities of modern CPUs and GPUs have increased tremendously over the last few decades, MD simulations are still limited to microseconds in timescale, and rarely do they reach milliseconds without specially designed hardware.[1] While microseconds are enough for many topics of research, such as properties of simple liquids or dynamics associated with nonhydrodynamic modes, many stochastic phenomena occur at much longer timescales and demands an approach other than the conventional brute-force molecular dynamics simulations.[2] When two local free energy minima are separated by a free energy barrier reaching a few k_BT in height, the transitions between the two states quickly become so slow that the transitions are now known as so-called *rare events*.

One example of a rare event, which is often used as a test study case for enhanced sampling development, is the interconversion of two conformers of alanine dipeptide in vacuum, which differ from each other in the conformation that the C_β methyl group adopts. One conformer has an equatorial C_β relative to the backbone atoms, while the other has an axial C_β. Conventional molecular dynamics simulations can no longer sufficiently sample the free energy landscape of the conversion of the conformers, owing to a free energy barrier up to $20k_BT$ between the conformers. To address the issue of rare events, researchers have developed various enhanced sampling methods to overcome challenges in timescale in MD simulations.

Various enhanced sampling methods have been developed for different purposes, some for the computation of the kinetic rate constant of a process, and others for sampling the free energy landscape. Several methods that are based the transition state theory include transition path sampling (TPS),[3] forward flux sampling (FFS),[4] milestoning,[5] to name just a few. In transition path sampling, a Monte Carlo sampling of the transition paths is conducted by generating new trajectories from a configuration from an existing trajectory that connects the two metastable states by shooting forwards and backwards with modified velocities, as illustrated by Figure 2.1. The transition rate is then computed with the mean first passage time of barrier crossing from the collection of generated trajectories known as "Transition Path Ensemble". Transition path sampling uses order parameters, which are functions of the configurations, to distinguish the initial state and final state and as signals to terminate simulations with successful trajectories linking the initial state and the final state.

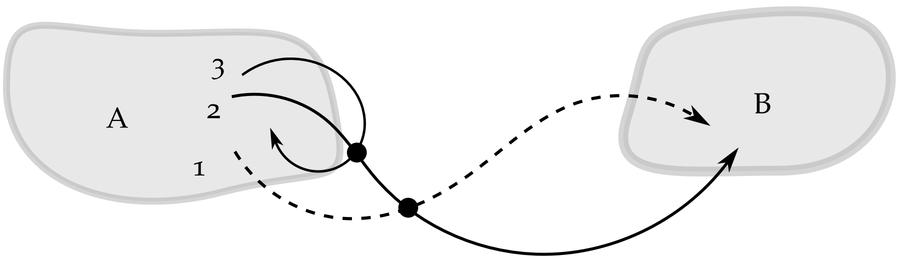


Figure 2.1: Schematic illustration of the shooting moves from existing trajectories connecting the initial state (A) and the final state (B) in transition path sampling. The figure is inspired by Figure 3 of Reference Bolhuis *et al.* [3].

In forward flux sampling, an order parameter λ is used to delineate interfaces along the transition pathway from the initial state to the final state. Simulations are started from the initial state to collect configurations that have crossed the first interface, from which new simulations are launched to cross the next interface, so on and so forth until the final state has been reached. A Transition Path Ensemble is eventually collected. The probability of crossing from one interface to another can be computed, and the rate of transition from the initial state to the final state is obtained. Similarly in milestoning milestones are demarcated along the transition path, however short trajectories are launched from all milestones instead of only starting from the initial state. In the end the probability of crossing the milestones and the mean first passage time are also obtained. In milestoning equilibrium around the milestones is assumed, whereas no such assumption is made in forward flux sampling. Schematic illustrations of forward flux sampling and milestoning are given in Figure 2.2. Notice that these methods harvest short trajectories instead of long trajectories connecting both the initial and final states.

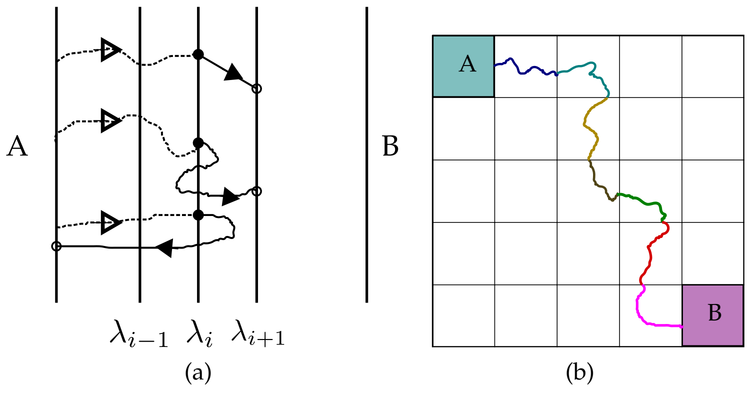


Figure 2.2: A schematic illustration of forward flux sampling (a) and milestoning (b). The different line types or colors represent different short trajectories.

In another type of enhanced sampling methods, multiple replicas or walkers of the same system are run in parallel simulations at increasing temperatures. The spacing of the temperatures normally follow the geometric rule. During the simulation periodic swaps of temperatures between neighboring replicas are attempted. Neighbors with the difference in energy conforming to the Metropolis criterion exchange their temperatures by rescaling their velocities to the target temperature; those that do not conform to the criterion do not exchange their temperatures. This method is known parallel tempering, or replica-exchange molecular dynamics (REMD). In parallel tempering simulations, systems climb a ladder of elevated temperatures may eventually overcome the free energy barrier at high temperatures and visit regions of the configuration space that is previously inaccessible at lower temperatures. In the end a more complete picture of the free energy landscape is achieved.

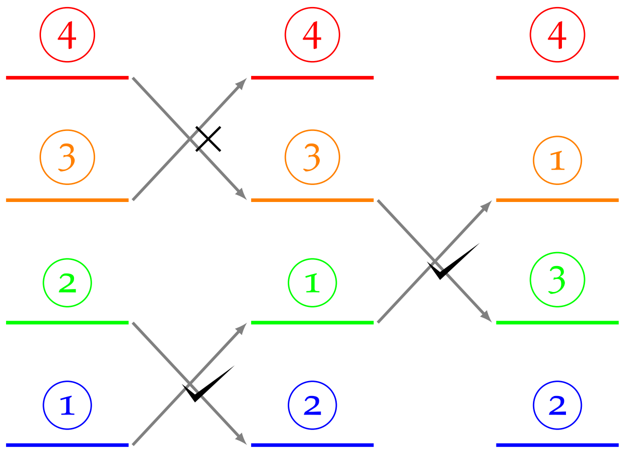


Figure 2.3: An illustration of replica-exchange molecular dynamics simulations with four walkers. Warmer colors indicate higher temperatures, and cooler colors indicate lower temperatures. The check mark indicates a successful exchange, and the cross means an unsuccessful exchange.

The final group of enhanced sampling methods that we introduce employs an external bias potential, which includes umbrella sampling, metadynamics and its various variants, and the variationally enhanced sampling method, only to give a few examples that are most relevant to this work. It is not meant to be a complete list. The bias potential effectively reduces the height of the free energy barrier in order to induce much faster transitions across it. The bias potential is defined as a function of CVs. Similar to previously mentioned order parameters, collective variables are also functions of the configuration space $\mathbf{r}$, which can discriminate different states. More often than not, one single CV does not suffice and several are needed so that different metastable states can be distinguished. In essence the high dimensional configuration space $\mathbf{r}$ is now effectively represented by a low dimensional CV space $\mathbf{s}$. The low dimensionality of the CV space also aids us in visualizing the free energy surface. Another requirement for CVs is that they must be differentiable and smooth so that forces derived from the bias potential are smooth. The choice of CVs plays a crucial role in the success of this group of enhanced sampling methods. Poor choices of CVs can result in a low efficiency of the sampling, and lead the system to long stays in irrelevant regions of the configuration space as they ultimately contribute very little to the unbiased free energy landscape. However, how to choose proper CVs would not be a focal point of this thesis. The function form of the bias potential can be quite flexible. In umbrella sampling, a harmonic potential is commonly used for the bias potential, while in metadynamics the bias potential is comprised of a sum of Gaussian kernel functions. In comparison, the bias potential is expressed as a linear combination of bias functions, for which there are many choices, in variationally enhanced sampling. More details of these methods are presented in the next few sections of this chapter.

Ideally collective variables should represent the slowest degrees of freedom so that enhanced sampling methods are most effective in driving barrier crossings.[6] While physical and chemical intuition is invaluable in designing or finding the right collective variables, more and more machine learning methods have been developed to automatically identify these slow modes of motions. Some of these methods can be found in the introduction of reference [6].

2.1 UMBRELLA SAMPLING

Umbrella sampling (US)[7] is one of the first enhanced sampling methods with an external bias potential. Torrie & Valleau developed umbrella sampling to compute the free energy of Lennard-Jones clusters relative to that of the soft-sphere clusters in Monte Carlo simulations, which had previously been computed[8]. The Lennard-Jones potential and soft-sphere potential are

$$U_{LJ}(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right], \tag{2.1}$$

$$U_{soft}(r) = 4\varepsilon\left(\frac{\sigma}{r}\right)^{12}, \tag{2.2}$$

respectively. The Helmholtz free energy A of Lennard-Jones clusters is related to the Helmholtz free energy A_0 of soft-sphere clusters:

$$\begin{aligned}\frac{A}{k_B T} - \frac{A_0}{k_B T_0} &= -\ln\left\langle \exp\left(-\frac{U}{k_B T} + \frac{U_0}{k_B T_0}\right)\right\rangle_0 \\ &= -\ln\langle \exp(-\Delta U^*)\rangle_0, \\ &= -\ln\int_{-\infty}^{\infty} f_0(\Delta U^*)\exp(-\Delta U^*)d\Delta U^*.\end{aligned} \tag{2.3}$$

in which $\langle \cdot \rangle_0$ represents the canonical ensemble average of the reference system, i. e., the soft-sphere cluster. $U^* = \frac{U}{k_B T}$ is the reduced energy, and $f_0(\Delta U^*)$ is the probability density of U^* in the reference system. In order to use Equation 2.3 to compute A, the sampling is conducted on the reference system (soft-sphere clusters) rather than the system of interest (Lennard-Jones clusters) to obtain $f_0(\Delta U^*)$. The probability density of the system of interest $f(\Delta U^*)$, after some rearrangement of Equation 2.3, is

$$f(\Delta U^*) = f_0(\Delta U^*) \exp(-\Delta U^*) Q_0/Q, \tag{2.4}$$

in which Q and Q_0 are the configurational partition function of the system of interest and the reference system. What Equation 2.4 informs us is that the accuracy of Equation 2.3 depends on the overlap of the two distributions $f(\Delta U^*)$ and $f_0(\Delta U^*)$, both of which are sharply peaked. In consequence, conventional Monte Carlo simulations of either the reference system or the system of interest cannot sample the probability density of ΔU^* of the other system sufficiently. To overcome this issue, the authors added a bias potential $V(\Delta U^*)$ to sample as broad a distribution of ΔU^* as possible on the reference system. However, with the external bias potential $V(\Delta U^*)$, the biased simulations sample a biased distribution $f_v(\Delta U^*)$ instead of the unbiased $f_0(\Delta U^*)$. To recover the unbiased distribution:

$$f_0(\Delta U^*) = \frac{f_v(\Delta U^*)/V(\Delta U^*)}{\langle 1/V \rangle_V}, \tag{2.5}$$

where $\langle \cdot \rangle_V$ represents an average of the biased ensemble.

Since its initial introduction, umbrella sampling has gone through many developments. For example, the most recognizable form of umbrella sampling draws on running biased simulations in multiple windows, each window with their harmonic bias potential and target CV values.[9] Newer developments include the adaptive bias umbrella sampling and weighted histogram analysis method (WHAM), which is a postprocessing method to estimate free energy surface from the umbrella sampling simulations. Readers are referred to the review by Kästner [10] on these topics.

2.2 METADYNAMICS AND WELL-TEMPERED METADYNAMICS

In the conventional forms of umbrella sampling the bias potential is in a harmonic function form such as $V(s) = K_i(s - s_i^0)^2$. It is chosen in advance of the biased simulation, and is stationary. In other words, the bias potential is the function only of the chosen CVs and independent of time. For example, the target value of CV s_i^0 for each window and the prefactors K_i for the harmonic potentials are chosen ahead of the simulations and do not change during the course of the simulations. Fixing the bias potential in advance of the simulations can be less optimal, when all metastable states are not known prior to the simulations, making it extremely difficult to select the windows for those states. Finding the proper prefactors for the harmonic potentials is also an issue. When the prefactors are too low, transitions across free energy barriers are not significantly boosted; when they are too large, the biased distributions between windows do not have enough overlap for users to accurately estimate the FES.

Laio & Parrinello introduced metadynamics in 2002.[11] Metadynamics is a technique that combines free energy surface exploration and enhanced sampling. Similar to umbrella sampling, metadynamics adds an external bias potential on the CV space to accelerate transitions between states, it does so by gradually building up the bias, rather than setting a fixed one in advance. Gaussian kernels are added to the CV space that has been visited by the system. The bias potential nudges the system away from its current region in the CV space into unexplored regions. The gradually growing bias potential eventually pushes the system over the free energy barrier into other metastable states. Unlike umbrella sampling, where users have to manually set the windows along the transition path, a challenging task especially in a multidimensional CV space, the system is allowed much more freedom to explore different paths and states. Also similar to umbrella sampling, the quality of the CVs is critical for the success of methods.

This initial development of metadynamics is usually referred to as the conventional metadynamics. A newer and more popular variant of metadynamics dubbed well-tempered metadynamics (WTMetaD) has been developed by Barducci, Bussi & Parrinello.[12,13] In both variants, the bias potential is evolved by periodically adding to it a Gaussian kernel. In the conventional metadynamics the height of the Gaussian kernels stays constant, while it decreases with a preceding scaling factor related to the previous bias potential in WTMetaD. We give more detailed introductions to metadynamics and well-tempered metadynamics in the next two subsections.

2.2.1 the Conventional Metadynamics

As stated above, metadynamics is an enhanced sampling method with a history-dependent bias potential. Periodically Gaussian kernels on the point of the CV space at the deposition step are added. The system are gradually moved into unexplored regions of the CV space. The Gaussian kernel being deposited is:

$$G(\mathbf{s},\mathbf{s}') = We^{-\|\mathbf{s}-\mathbf{s}'\|^2}, \tag{2.6}$$

where W is the height of the Gaussian kernel, and $\|\mathbf{s}-\mathbf{s}'\|$ is metric of the distance between $\mathbf{s}$ and $\mathbf{s}'$. When $G(\mathbf{s},\mathbf{s}')$ is a multivariate Gaussian, as often it is

$$\|\mathbf{s}-\mathbf{s}'\| = \frac{1}{2}\sum_{i,j}(s_i - s_i')\Sigma_{i,j}^{-1}(s_j - s_j'), \tag{2.7}$$

in which $\Sigma_{i,j}$ is the covariance matrix, and $\Sigma_{i,j}^{-1}$ is its inverse. Here $\Sigma_{i,j}$ is only taken with the diagonal members of the matrix, i. e., $\Sigma_{i,j} = \delta_{i,j}\sigma_i^2$, in which σ_i is the width of the Gaussian kernels for the ith CV, which is chosen by the user often according to the fluctuations of the CV in unbiased simulations. Equation 2.6 can be rewritten as:

$$G(\mathbf{s},\mathbf{s}') = W\exp\left[-\frac{1}{2}\sum_{i=1}^{d}\frac{(s_i - s_i')^2}{\sigma_i^2}\right], \tag{2.8}$$

in which d is the dimension of the CV space. In metadynamics every τ_G steps a new Guassian kernel is deposited, so the current bias potential $V(\mathbf{s},t)$ at time t is:

$$V(\mathbf{s},t) = \sum_{\substack{t'=\tau_G,2\tau_G,\cdots \\ t'<t}}^{n\tau_G} W\exp\left\{-\frac{1}{2}\sum_{i=1}^{d}\frac{[s_i(t) - s_i(t')]^2}{\sigma_i^2}\right\}, \tag{2.9}$$

where W is the height of the Gaussian kernels and a constant in the conventional metadynamics. The integer n on the top of summation is the number of iterations of depositions of Gaussian kernels that has been completed by the time t.

As a metadynamics simulation progresses, the height of the bias potential grows with every addition of the Gaussian kernel. Eventually it fills up the valley of the local free energy minima, and the free energy barrier is no longer an impediment to fast transitions between states. The dynamics of such transitions has become quite diffusive in the CV space. In the long time limit the bias potential fluctuates around the negative of the free energy surface, as illustrated in Figure 2.4.

In conventional metadynamics, the underlying free energy surface can be estimated as the time-average of the negative bias potentials after a time t_{fill} when the local free energy has been filled[14]:

$$F(\mathbf{s}) = -\frac{1}{t - t_{fill}}\sum_{t'=t_{fill}}^{t} V_{t'}(\mathbf{s}). \tag{2.10}$$

2.2.2 Well-tempered Metadynamics

As shown in Figure 2.4 and Section 2.2.1, in the conventional metadynamics the bias potential grows constantly. Therefore, it is difficult to know when to terminate a conventional metadynamics simulation. It is also unsatisfactory and aesthetically displeasing that the bias

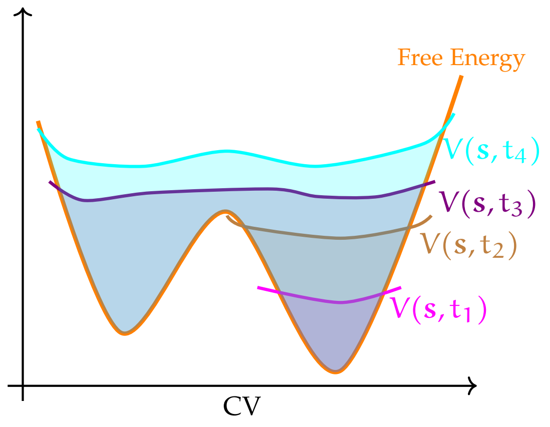


Figure 2.4: A schematic illustration of the growth of the time dependent bias potential $V(\mathbf{s},t)$ with the time in a metadynamics simulation.

potential does *not* reach a stationary state in the long time limit. This situation however has been remedied with the development of well-tempered metadynamics.[12] In WTMetaD, the height of the Gaussian kernel is not constant anymore. A scaling factor that modifies its height is included to take into account the history of past visited states. Hence, both the height and the location of the Gaussian are history dependent. Each iteration of the bias potential is:

$$V_n(\mathbf{s}) = V_{n-1}(\mathbf{s}) + G(\mathbf{s},\mathbf{s}_n)e^{-\frac{1}{\gamma-1}\beta V_{n-1}(\mathbf{s}_n)}, \tag{2.11}$$

where $V_0(\mathbf{s}) = 0$, and β is the inverse temperature. γ is called the bias factor.

Thus the real time bias potential at time t in the CV space $\mathbf{s}$ is:

$$V(\mathbf{s},t) = \sum_{\substack{t'=\tau_G,2\tau_G,\cdots \\ t'<t}}^{n\tau_G} We^{-\frac{1}{2}\sum_{i=1}^{d}\frac{\left[s_i(t)-s_i(t')\right]^2}{\sigma_i^2}} e^{-\frac{1}{\gamma-1}\beta V_{t'-\tau_G}(\mathbf{s}(t'))}, \tag{2.12}$$

or we can rewrite $V(\mathbf{s},t)$ using the iteration of Guassian deposition k instead of the time of deposition t' for simplicity:

$$V(\mathbf{s},t) = \sum_{k=1}^{n} We^{-\|\mathbf{s}-\mathbf{s}_k\|^2} e^{-\frac{1}{\gamma-1}\beta V_{k-1}(\mathbf{s}_k)}. \tag{2.13}$$

Equation 2.11 describes the stochastic iteration of the bias potential $V(\mathbf{s},t)$, as a result of which, the evolution of the bias potential can be described asymptotically by an ordinary differential equation, which reads

$$\frac{dV(\mathbf{s},t)}{dt} = \int d\mathbf{s}' G(\mathbf{s},\mathbf{s}')\exp\left[-\frac{1}{\gamma-1}\beta V(\mathbf{s}',t)\right]p_V(\mathbf{s}',t), \tag{2.14}$$

in which $p_V(\mathbf{s},t)$ is the biased distribution of the CV space, and it is given by

$$p_V(\mathbf{s},t) = \frac{e^{-\beta[F(\mathbf{s})+V(\mathbf{s},t)]}}{\int d\mathbf{s}e^{-\beta[F(\mathbf{s})+V(\mathbf{s},t)]}}. \tag{2.15}$$

The formal derivation of Equation 2.14 and Equation 2.15 can be found in references [15, 16]. Equation 2.15 has the asymptotic solution

$$V(s,t) = -\left(1 - \frac{1}{\gamma}\right) F(s) + c(t), \tag{2.16}$$

where $c(t)$ is a time-dependent constant defined as

$$c(t) = \frac{1}{\beta} \log \frac{\int ds e^{-\beta F(s)}}{\int ds e^{-\beta[F(s)+V(s,t)]}}. \tag{2.17}$$

$c(t)$ is a time-dependent constant. The convergence property of the bias potential described in Equation 2.16 has contributed to the higher popularity of well-tempered metadynamics compared to the conventional metadynamics. It has also provided a way to estimate the free energy surface from the bias potential.

The biased probability distribution that WTMetaD simulations sample is called the well-tempered distribution. It is related to the underlying probability distribution $p(s)$ by:

$$p_V(s) = \frac{[p(s)]^{1/\gamma}}{\int ds [p(s)]^{1/\gamma}}, \tag{2.18}$$

where γ is a real number greater than 1. This relationship is illustrated in Figure 2.5. The free energy barrier in the well-tempered distribution is significantly lower than that of the unbiased distribution. By taking the logarithm on both sides of Equation 2.18, one obtains that the biased free energy $F_V(s) = \frac{F(s)}{\gamma}$. It is evident that free energy barriers are reduced by the same bias factor. The bias factor γ can also be written as $\gamma = (T + \Delta T)/T$. It can be viewed that well-tempered metadynamics simulations sample the CV space at a higher temperature $T + \Delta T$.

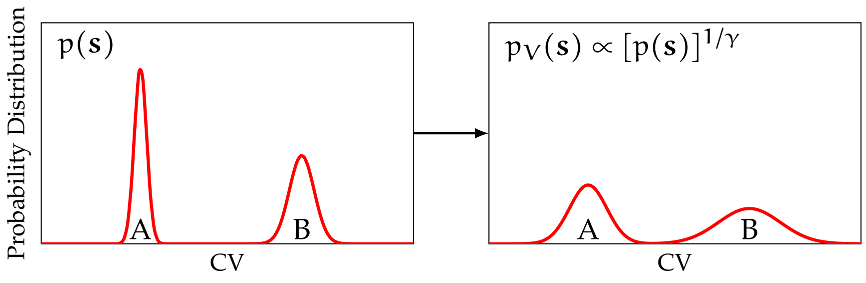


Figure 2.5: The relationship between the well-tempered distribution and the unbiased distribution.

2.3 REWEIGHTING OF WELL-TEMPERED METADYNAMICS SIMULATIONS

Although Equation 2.16 already provides a way to obtain the free energy surface, a reweighting method is still needed for well-tempered metadynamics when we want to estimate the free energy surface projected to other CVs that are not directly biased in the well-tempered metadynamics simulations. As the bias potential in the well-tempered metadynamics simulation is not static, it would be incorrect to reweight the well-tempered metadynamics simulations the same way as it is done in Equation 2.5 for umbrella sampling. Two ways have been developed to address the need to reweight well-tempered metadynamics simulations, which are introduced in the next two subsections.

2.3.1 Residual Bias Reweighting

Residual bias reweighting, which is also called $c(t)$ reweighting, was developed by Tiwary & Parrinello.[17] It uses the same time-dependent constant $c(t)$ already mentioned in Equation 2.17, which is an estimator of the reversible work done by the bias potential. There are a number of ways to compute $c(t)$. First we introduce the free energy estimator of the well-tempered metadynamics simulations:

$$F(\mathbf{s}) = -\left(\frac{\gamma}{\gamma-1}\right)V(\mathbf{s},t) + \frac{1}{\beta}\log\int d\mathbf{s}\,\exp\left[\frac{\gamma}{\gamma-1}\beta V(\mathbf{s},t)\right]. \tag{2.19}$$

By plugging Equation 2.19 into Equation 2.17, we obtain

$$c(t) = \frac{1}{\beta}\log\frac{\int d\mathbf{s}\,\exp\left[\frac{\gamma}{\gamma-1}\beta V(\mathbf{s},t)\right]}{\int d\mathbf{s}\,\exp\left[\frac{1}{\gamma-1}\beta V(\mathbf{s},t)\right]}. \tag{2.20}$$

Using Equation 2.20 one can calculate the average of the observable $O(\mathbf{r})$ from a well-tempered metadynamics trajectory with

$$\langle O(\mathbf{r})\rangle = \left\langle O(\mathbf{r})e^{\beta[V(\mathbf{s}(\mathbf{r}),t)-c(t)]}\right\rangle_V. \tag{2.21}$$

The reweighted free energy surface projected onto the CV space $\mathbf{s}'$, which could be the same as $\mathbf{s}$, can be computed as

$$F(\mathbf{s}') = -\frac{1}{\beta}\log\int d\mathbf{r}\,\delta[\mathbf{s}'-\mathbf{s}'(\mathbf{r})]e^{\beta[V(\mathbf{s}(\mathbf{r}),t)-c(t)]} + C. \tag{2.22}$$

2.3.2 Last Bias Reweighting

Last bias reweighting[18] uses the weights calculated from the final form of a well-tempered metadynamics simulation to compute the average of an observable:

$$\langle O(\mathbf{r})\rangle = \frac{\langle O(\mathbf{r})e^{\beta V(\mathbf{s}(\mathbf{r}),t_{last})}\rangle_V}{\langle e^{\beta V(\mathbf{s}(\mathbf{r}),t_{last})}\rangle_V}. \tag{2.23}$$

And the reweighted free energy surface $F(\mathbf{s}')$ is

$$F(\mathbf{s}') = -\frac{1}{\beta}\log\int d\mathbf{r}\,\delta[\mathbf{s}'-\mathbf{s}'(\mathbf{r})]e^{\beta V(\mathbf{s}(\mathbf{r}),t_{last})} + C. \tag{2.24}$$

2.4 VARIATIONALLY ENHANCED SAMPLING

In 2014 Valsson & Parrinello developed the variationally enhanced sampling (VES) method as a new way to find a bespoke bias potential for enhanced sampling.[19] First a functional Ω of the bias potential $V(s)$ is defined in VES as:

$$\Omega[V] = \frac{1}{\beta} \log \frac{\int ds\, e^{-\beta[F(s)+V(s)]}}{\int ds\, e^{-\beta F(s)}} + \int ds\, p_{tg}(s)V(s), \qquad (2.25)$$

where s are the biased CVs, and $F(s)$ is the free energy surface projected to the s, $p_{tg}(s)$ is a probability distribution of s predefined by a user and referred to as the target distribution. It has been proven that Ω is a convex functional. In other words, Ω has a global minimum, the only point where the gradient of Ω, $\frac{\partial \Omega[V(s)]}{\partial V(s)}$ is equal to 0. Furthermore, at the global minimum of Ω the bias potential $V(s)$ is related to the FES $F(s)$ and the target distribution $p_{tg}(s)$ by:

$$V(s) = -F(s) - \frac{1}{\beta} \log p_{tg}(s) + C, \qquad (2.26)$$

in which C is a nonconsquential constant that can be ignored. Finally, $p_V(s)$, the biased distribution of s is equal to the user-defined target distribution $p_{tg}(s)$ at the global minimum of Ω. A simplistic choice for $p_{tg}(s)$ would be the uniform distribution. Clearly, it would not be the most efficient choice. More commonly the well-tempered distribution has been used for VES.[20] The relationship between the well-tempered distribution and the unbiased distribution has been illustrated in Figure 2.5. As the unbiased distribution ($p(s)$) is unknown prior to the simulations, the well-tempered distribution must be updated iteratively during the run of the simulation so that it can more accurately reflect the true probability distribution.[20]

The variationally enhanced sampling is related to both relative entropy[21,22] and the Kullback-Leibler divergence[23]. The function Ω can be rewritten as

$$\beta\Omega[V(s)] = D_{KL}(p_{tg}(s)||p_V(s)) - D_{KL}(p_{tg}(s)||p(s)), \qquad (2.27)$$

in which $D_{KL}(\cdot||\cdot)$, the Kullback-Leibler divergence is a metric of the difference between two probability distributions. Minimizing $\Omega[V(s)]$ is akin to the minimization of the Kullback-Leibler divergence between the biased distribution $p_V(s)$ and the target distribution $p_{tg}(s)$.

2.4.1 Variational Optimization Process

Variationally enhanced sampling prescribes a variational approach of finding the bias potential $V(s)$ that minimizes Ω. We start first by representing the bias potential $V(\mathbf{s})$ as a linear combinations of bias functions $f_k(\mathbf{s})$:

$$V(\mathbf{s};\boldsymbol{\alpha})=\sum_k \alpha_k f_k(\mathbf{s}). \tag{2.28}$$

The bias functions $f_k(\mathbf{s})$ are differentiable functions of the CVs. Currently the choice of basis functions include Legendre polynomials, Chebyshev polynomials, Fourier cosine or sine series and so on. The coefficients $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ are the variational parameters that are optimized during the biased simulation. To find the global minimum of Ω where $\frac{\partial\Omega[V(\mathbf{s})]}{\partial V(\mathbf{s})} = 0$, we need to compute the gradient and the Hessian of $\Omega(\boldsymbol{\alpha})$. An element of the gradient $\nabla\Omega(\boldsymbol{\alpha})$ can be computed:

$$\frac{\partial\Omega(\boldsymbol{\alpha})}{\partial\alpha_i} = -\left\langle\frac{\partial V(\mathbf{s};\boldsymbol{\alpha})}{\partial\alpha_i}\right\rangle_{V(\boldsymbol{\alpha})} + \left\langle\frac{\partial V(\mathbf{s};\boldsymbol{\alpha})}{\partial\alpha_i}\right\rangle_{p_{tg}}, \tag{2.29}$$

and similarly the elements of the Hessian $\mathbb{H}\Omega(\boldsymbol{\alpha})$ can be computed as:

$$\begin{aligned}\frac{\partial^2\Omega(\boldsymbol{\alpha})}{\partial\alpha_i\partial\alpha_j} = \beta\,\mathrm{Cov}\left[\frac{\partial V(\mathbf{s};\boldsymbol{\alpha})}{\partial\alpha_j}, \frac{\partial V(\mathbf{s};\boldsymbol{\alpha})}{\partial\alpha_i}\right]_{V(\boldsymbol{\alpha})} \\ -\left\langle\frac{\partial^2 V(\mathbf{s};\boldsymbol{\alpha})}{\partial\alpha_j\partial\alpha_i}\right\rangle_{V(\boldsymbol{\alpha})} + \left\langle\frac{\partial^2 V(\mathbf{s};\boldsymbol{\alpha})}{\partial\alpha_j\partial\alpha_i}\right\rangle_{p_{tg}}.\end{aligned} \tag{2.30}$$

In Equation 2.29 and Equation 2.30, $\langle\cdot\rangle_{V(\boldsymbol{\alpha})}$ denotes an average over the biased ensemble, and $\langle\cdot\rangle_{p_{tg}}$ denotes an average over the target distribution. Both equations can be simplified when we employ a linear expression for the bias potential:

$$\frac{\partial\Omega(\boldsymbol{\alpha})}{\partial\alpha_i} = -\langle f_i(\mathbf{s})\rangle_{V(\boldsymbol{\alpha})} + \langle f_i(\mathbf{s})\rangle_{p_{tg}}, \tag{2.31}$$

$$\frac{\partial^2\Omega(\boldsymbol{\alpha})}{\partial\alpha_j\partial\alpha_i} = \beta\,\mathrm{Cov}[f_j(\mathbf{s}), f_i(\mathbf{s})]_{V(\boldsymbol{\alpha})}. \tag{2.32}$$

$\langle f_i(\mathbf{s})\rangle_{V(\boldsymbol{\alpha})}$ can be computed by sampling during the biased simulation, while $\langle f_i(\mathbf{s})\rangle_{p_{tg}}$ can be computed analytically, as $p_{tg}(\mathbf{s})$ and $f_i(\mathbf{s})$ are both chosen by the users. The covariance $\mathrm{Cov}[f_j(\mathbf{s}), f_i(\mathbf{s})]_{V(\boldsymbol{\alpha})}$ can also be computed from the biased simulations. As both Equation 2.31 and Equation 2.32 are computed statistically, they are noisy in nature. As a result, the variational parameters can display large fluctuations between updates, making it difficult to reach a stable bias potential. It has been found that the averaged stochastic gradient descent-based optimization method[24] works well to address the issue of large fluctuations. The variational parameters at iteration n, $\boldsymbol{\alpha}^{(n)}$, are updated according to

$$\boldsymbol{\alpha}^{(n+1)} = \boldsymbol{\alpha}^{(n)} - \mu\left[\nabla\Omega(\bar{\boldsymbol{\alpha}}^{(n)}) + \mathbb{H}\Omega(\bar{\boldsymbol{\alpha}}^{(n)})[\boldsymbol{\alpha}^{(n)} - \bar{\boldsymbol{\alpha}}^{(n)}]\right], \tag{2.33}$$

in which μ is the step size of the update, a parameter chosen by the user prior to the simulation, and the average variational parameters $\bar{\boldsymbol{\alpha}}^{(n)} = (n+1)^{-1} \sum_{k=0}^{n} \boldsymbol{\alpha}^{(k)}$. Only the diagonal elements of $\mathbb{H}\Omega(\bar{\boldsymbol{\alpha}}^{(n)})$ are used in the update. The evolution of $\bar{\boldsymbol{\alpha}}$ is considerably smoother with Equation 2.33, as illustrated in Figure 2.6. The update is performed between every a number of time steps per iteration. The length between updates is called the *stride* in VES. The free energy surface can

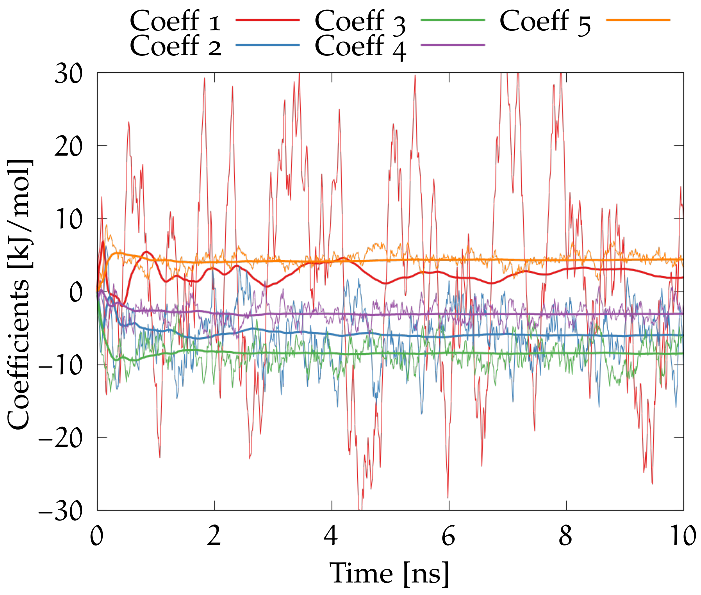


Figure 2.6: The time series of five coefficients of $\boldsymbol{\alpha}$ (thin) and their average coefficients $\bar{\boldsymbol{\alpha}}$ (thick) of VES simulations of association of a pair of Na^+ and Cl^- solutes in explicit water.

be estimated by Equation 2.26, if the bias potential and the sampling have converged. Otherwise reweighting can be called upon for free energy estimation.

2.4.2 Algorithmic Implementation of VES

Already mentioned in section Molecular Dynamics, PLUMED 2 implements various variants of metadynamics and variationally enhanced sampling,[25–27] and it has patches to many commonly used MD engines such as LAMMPS, GROMACS, Amber and so on, so they that work together in real time simulations in addition to in postprocessing. The cooperation between the PLUMED 2 library and a MD engine is illustrated in Figure 2.7. For metadynamics, the bias potential $V(\mathbf{s})$ is updated through depositions of Gaussian kernels. For variationally enhanced sampling, it is achieved by updating the linear coefficients or the variational parameters $\boldsymbol{\alpha}$ in the bias potential of $V(\mathbf{s};\boldsymbol{\alpha})$. Additionally, the website of PLUMED-NEST (`https://www.plumed-nest.org/`) hosts voluntarily contributed PLUMED input files and other relevant files of publications that are listed with their Digital Object Identifiers

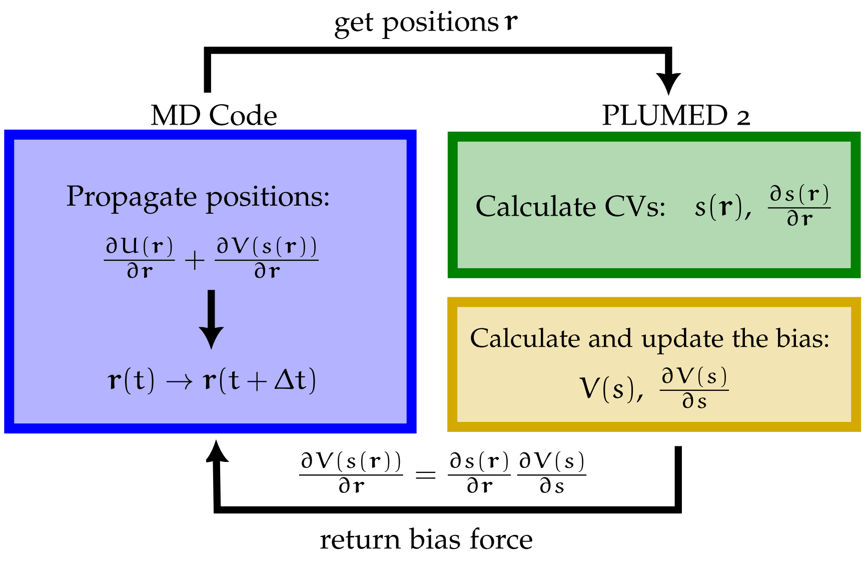

get positions $\mathbf{r}$
MD Code
PLUMED 2
Propagate positions:
$\frac{\partial U(\mathbf{r})}{\partial \mathbf{r}} + \frac{\partial V(s(\mathbf{r}))}{\partial \mathbf{r}}$
$\mathbf{r}(t) \to \mathbf{r}(t+\Delta t)$
Calculate CVs: $s(\mathbf{r}),\ \frac{\partial s(\mathbf{r})}{\partial \mathbf{r}}$
Calculate and update the bias:
$V(s),\ \frac{\partial V(s)}{\partial s}$
$\frac{\partial V(s(\mathbf{r}))}{\partial \mathbf{r}} = \frac{\partial s(\mathbf{r})}{\partial \mathbf{r}} \frac{\partial V(s)}{\partial s}$
return bias force

Figure 2.7: An illustration of how PLUMED 2 and a MD engine work together in a simulation.

(DOIs), which are open to all that are interested in realizing these simulations.

REFERENCES

1. Shaw, D. E. *et al. Anton 3: Twenty Microseconds of Molecular Dynamics Simulation before Lunch* in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Association for Computing Machinery, New York, NY, USA, Nov. 14, 2021), 1–11.
2. Frenkel, D. & Smit, B. in *Understanding Molecular Simulation (Second Edition)* 431–464 (Academic Press, San Diego, Jan. 1, 2002).
3. Bolhuis, P. G., Chandler, D., Dellago, C. & Geissler, P. L. TRANSITION PATH SAMPLING: Throwing Ropes Over Rough Mountain Passes, in the Dark. *Annual Review of Physical Chemistry* **53,** 291–318 (Oct. 1, 2002).
4. Allen, R. J., Valeriani, C. & Rein ten Wolde, P. Forward Flux Sampling for Rare Event Simulations. *Journal of Physics: Condensed Matter* **21,** 463102 (Oct. 26, 2009).
5. Bello-Rivas, J. M. & Elber, R. Exact Milestoning. *The Journal of Chemical Physics* **142,** 094102 (Mar. 7, 2015).
6. Rydzewski, J. & Valsson, O. Multiscale Reweighted Stochastic Embedding: Deep Learning of Collective Variables for Enhanced Sampling. *J. Phys. Chem. A* **125,** 6286–6302 (July 22, 2021).

7. Torrie, G. M. & Valleau, J. P. Nonphysical Sampling Distributions in Monte Carlo Free-Energy Estimation: Umbrella Sampling. *Journal of Computational Physics* **23,** 187–199 (Feb. 1, 1977).

8. Torrie, G. M. & Valleau, J. P. Monte Carlo free energy estimates using non-Boltzmann sampling: Application to the sub-critical Lennard-Jones fluid. *Chemical Physics Letters* **28,** 578–581 (1974).

9. Virnau, P. & Müller, M. Calculation of Free Energy through Successive Umbrella Sampling. *The Journal of Chemical Physics* **120,** 10925–10930 (June 15, 2004).

10. Kästner, J. Umbrella Sampling. *WIREs Computational Molecular Science* **1,** 932–942 (2011).

11. Laio, A. & Parrinello, M. Escaping Free-Energy Minima. *Proceedings of the National Academy of Sciences* **99,** 12562–12566 (Oct. 1, 2002).

12. Barducci, A., Bussi, G. & Parrinello, M. Well-Tempered Metadynamics: A Smoothly Converging and Tunable Free-Energy Method. *Physical Review Letters* **100,** 020603 (Jan. 18, 2008).

13. Valsson, O., Tiwary, P. & Parrinello, M. Enhancing Important Fluctuations: Rare Events and Metadynamics from a Conceptual Viewpoint. *Annual Review of Physical Chemistry* **67,** 159–184 (May 27, 2016).

14. Bussi, G. & Laio, A. Using Metadynamics to Explore Complex Free-Energy Landscapes. *Nature Reviews Physics* **2,** 200–212 (4 Apr. 2020).

15. Dama, J. F., Parrinello, M. & Voth, G. A. Well-Tempered Metadynamics Converges Asymptotically. *Physical Review Letters* **112** (June 18, 2014).

16. Tiwary, P., Dama, J. F. & Parrinello, M. A Perturbative Solution to Metadynamics Ordinary Differential Equation. *The Journal of Chemical Physics* **143,** 234112 (Dec. 21, 2015).

17. Tiwary, P. & Parrinello, M. A Time-Independent Free Energy Estimator for Metadynamics. *The Journal of Physical Chemistry B* **119,** 736–742 (Jan. 22, 2015).

18. Branduardi, D., Bussi, G. & Parrinello, M. Metadynamics with Adaptive Gaussians. *Journal of Chemical Theory and Computation* **8,** 2247–2254 (July 10, 2012).

19. Valsson, O. & Parrinello, M. Variational Approach to Enhanced Sampling and Free Energy Calculations. *Physical Review Letters* **113,** 090601 (Aug. 27, 2014).

20. Valsson, O. & Parrinello, M. Well-Tempered Variational Approach to Enhanced Sampling. *Journal of Chemical Theory and Computation* **11,** 1996–2002 (May 12, 2015).

21. Chaimovich, A. & Shell, M. S. Coarse-Graining Errors and Numerical Optimization Using a Relative Entropy Framework. *The Journal of Chemical Physics* **134,** 094112 (Mar. 7, 2011).

22. De Boer, P.-T., Kroese, D. P., Mannor, S. & Rubinstein, R. Y. A Tutorial on the Cross-Entropy Method. *Annals of Operations Research* **134,** 19–67 (Feb. 1, 2005).

23. Bilionis, I. & Koutsourelakis, P. Free Energy Computations by Minimization of Kullback–Leibler Divergence: An Efficient Adaptive Biasing Potential Method for Sparse Representations. *Journal of Computational Physics* **231,** 3849–3870 (May 1, 2012).

24. Bach, F. & Moulines, E. *Non-strongly-convex smooth stochastic approximation with convergence rate O(1/n)* in *Advances in Neural Information Processing Systems* **26** (Curran Associates, Inc., 2013).

25. Bonomi, M. *et al.* PLUMED: A Portable Plugin for Free-Energy Calculations with Molecular Dynamics. *Computer Physics Communications* **180,** 1961–1972 (Oct. 1, 2009).

26. Bonomi, M. *et al.* Promoting Transparency and Reproducibility in Enhanced Molecular Simulations. *Nature Methods* **16,** 670–673 (Aug. 1, 2019).

27. Tribello, G. A., Bonomi, M., Branduardi, D., Camilloni, C. & Bussi, G. PLUMED 2: New Feathers for an Old Bird. *Computer Physics Communications* **185,** 604–613 (Feb. 2014).

3 THE EXTPLUMED EXTENSION AND THE SINGLE-CHAIN SMOOTH SQUARE-WELL POLYMER

ESPResSo++, developed and maintained by the theory group of Max Planck Institute of Polymer Research, Mainz, is a software package for MD simulations, that has been developed mainly for studies of phase behaviors of polymers such as polymer melts and rheology of polymers. PLUMED 2 is a open-source software library that provides many functionalities to compute collective variables and performs enhanced sampling methods on these collective variables. PLUMED 2 provides a well-documented application programming interface (API) for multiple programming languages such as C, C++, FORTRAN, and Python. So far PLUMED already works with many popular MD engines, for example, Amber, DL_POLY, GROMACS, LAMMPS, NAMD, OpenMM to name a few, but not yet with ESPResSo++. Here, we create an interface between two codes in the form of an extension to the integrator of ESPResSo++ so that information such as coordinates and forces can be passed between the two codes, as illustrated in Figure 2.7, and that they can be used together in an enhanced sampling MD simulation. We provide details of our implementation of the extension and demonstrate both software packages working together in a study of the phase transition of a single chain polymer through a pairwise smooth square-well interaction potential with well-tempered metadynamics.

3.1 ESPRESSO++ AND EXTPLUMED EXTENSION

ESPResSo++ combines a flexible user-facing Python interface and a C++ core responsible for the intensive computational workload. This design takes advantage of the easy scriptability of the Python language and high performance of native C++ code. For ESPResSo++, the *Parallel Method Invocation* (PMI) Python module has been developed.[1] It possesses features from both of the two models of parallelism, namely the shared-memory thread model and the distributed-memory message-passing model. As its name suggested, PMI module can execute function calls in parallel. Moreover, it can also create parallel Python class instances and make parallel calls to the class methods. ESPResSo++ heavily leverages the versatile Boost library. In

ESPResSo++ Python instances and its corresponding C++ objects interact through the Boost.Python library. Communications of messages between the C++ class objects are realized with the boost.MPI library. Figure 3.1 illustrates the architecture and interactions between the Python interface and the C++ core of ESPResSo++.

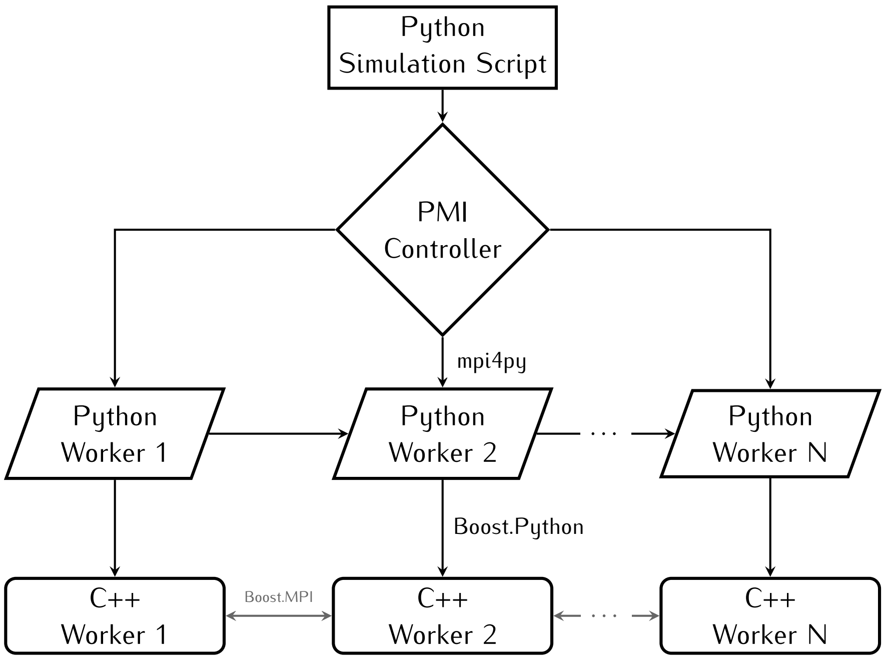


Figure 3.1: An illustration of the binding of Python and C++ in ESPResSo++.

In ESPResSo++ the velocity-Verlet integrator is implemented, which we have introduced in Section 1.4. It propagates the dynamics in the following three steps:

1. Update the velocities by half a time step;
2. Update the positions by a full step with updated velocities, and update the forces at the new positions;
3. Complete the update of velocities by another half a time step with the new forces.

ESPResSo++ places signals throughout the steps of the integrator, as shown in Figure 3.2, so that calls to extensions to the integrator can be made at proper time in order to, for example, compute the virial matrix or energy. Users can also modify the forces acting on the atoms through the already available extension ExtForce. We therefore have chosen to implement the interface between ESPResSo++ and PLUMED as an extension to the integrator. We name the extension ExtPlumed. The code structure of the integrator and the ExtPlumed extension is illustrated in Figure 3.2. The source code of the ExtPlumed extension is included in Appendix A.

PLUMED 2 provides two files, Plumed.h and Plumed.c, in which a stable API to PLUMED 2 is located. Software developers can include the two files in their own MD code to call PLUMED 2 through a few function calls that are defined in the two files. PLUMED 2 can be built as a static library or a shared library to a MD code. Or more conveniently, a user can supply the PLUMED library by defining an environment variable `PLUMED_KERNEL` pointing to the location of the library `libplumedkernel.so` is stored.

The MD engine must pass the global MPI communicator to PLUMED so that it can take advantage of all available processors during computation. Also passed are the total number of atoms and dimensions of the box. The positions are passed from the MD engine to PLUMED every step so that CVs can be computed and a bias potential on the CV can be applied. The forces and the energy derived from these bias potentials are then passed back to the MD engine in order that they are taken into account by the integrator when updating the velocities and positions. PLUMED provides a stable interface to perform aforementioned functions, and they are coordinated with the signals within the integrator (the VelocityVerlet class, Figure 3.2) of ESPResSo++ to ensure both correctness and efficiency.

3.2 THE SINGLE-CHAIN SMOOTH SQUARE-WELL POLYMER

3.2.1 the Square-Well Potential and the Smooth Square-Well Potential

The square-well potential can be written as:

$$U(r_{ij}) = \begin{cases} \infty & 0 < r_{ij} < \sigma, \\ -\varepsilon & \sigma \leq r_{ij} \leq \lambda\sigma, \\ 0 & r_{ij} > \lambda\sigma. \end{cases} \tag{3.1}$$

In Equation 3.1 ε is the depth of the square-well, and λ is related to the width of the square-well. The interaction sites i and j have a strong repulsion when their distance is shorter than σ; and when their distance is within $[\sigma, \lambda\sigma]$, they have a favorable interaction inside the square well. When two interaction sites are further apart than $\lambda\sigma$, they do not interact with each other. The thermodynamics of simple liquids has been studied in the form of square-well liquids, and thermodynamics of alkanes has been studied in simulations in the form of freely-joined square-well chain fluids[2]. In these studies the parameter λ was often chosen to be 1.5. A freely-joined single-chain square-well polymers can have complex phase behaviors depending on the width of the well, temperature and the length of the polymer.

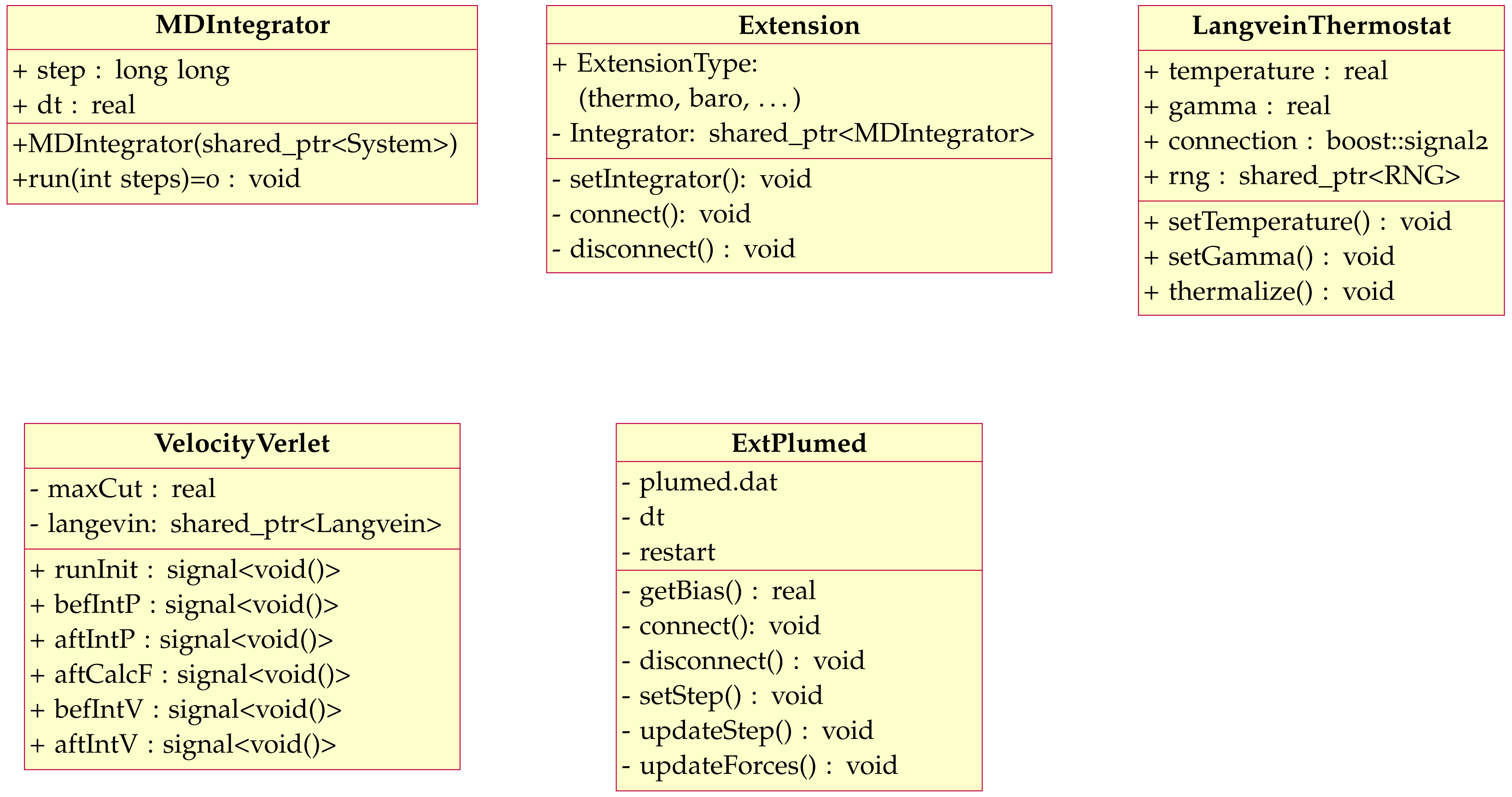


Figure 3.2: The ExtPlumed extension is the interface between PLUMED and ESPResSo++. Method functions in the ExtPlumed class, which calls PLUMED, are executed upon the notifications of the signals in the VelocityVerlet class, which inherits from the virtual base class MDIntegrator. Similarly the LangevinThermostat is also implemented as an Extension so that method calls are executed upon receiving a signal from the integrator.

Taylor *et al.* have constructed an extensive phase diagram of the single-chain square-well polymer with Monte Carlo simulations and Wang-Landau sampling.[3–6] Růžička *et al.* used Monte Carlo simulations and forward flux sampling to obtain the phase diagram of single-chain square-well polymer based on kinetic data.[7,8] They have also showed that the nonbonded contacts are a good indicator of the phase that the square-well polymer is in. Wicks *et al.* used Monte Carlo simulations with parallel tempering and a bias potential to construct free energy surface at multiple temperatures.[9]

The aforementioned previous studies have concluded that the phase behavior of the single-chain square-well polymer depends on various parameters such as the length of the polymer, the width of the square well. In the case of the 128 monomer long square-well polymer, it has three metastable states when $\lambda > 1.05$. The polymer has a random coil state, which can freeze into a crystalline state of body-centered cubic (BCC) packing after first forming a compact globule state. When the width $\lambda \leq 1.05$, the freezing of the single-chain square-well polymer goes directly from the random coil to the crystalline state without going through another state. An example of a random coil and the crystalline state of the single-chain square-well polymer comprised of 128 monomers can be found Figure 3.3. The phase transition between

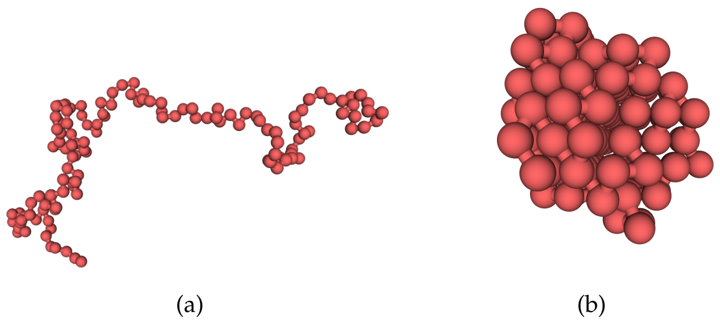


Figure 3.3: Conformations of the random coil state (a) and the crystalline state (b) of the single-chain square-well polymer. The crystalline state adopts the BCC packing.

the two states is a first-order process, i. e., a significant free energy barrier lies between them. In this work we investigate the latter case with $\lambda = 1.05$.

Leitold & Dellago designed the smooth square-well potential, which we have introduced in Section 1.3.2, so that the square-well polymer can be studied with MD simulations.[10] Its function form is:

$$U(r) = \frac{\varepsilon}{2}\left\{\exp\left[\frac{-(r-\sigma)}{a}\right] + \tanh\left[\frac{r-\lambda\sigma}{a}\right] - 1\right\}, \tag{3.2}$$

where ε and σ are equal to 1, and $\lambda = 1.05$, and $a = 0.002$. The rigid bonds of the single-chain square-well polymer are replaced with harmonic bonds: $U(b) = K(b - b_0)^2$, with $K = 10000$. The smooth square-well potential and the harmonic bonding potential are illustrated in Figure 3.4. Leitold & Dellago have used transition path sampling to study the phase transition of the smooth square-well polymer with 128 monomers at its coexistence temperature $T_{co} = 0.428$, which value was determined with Wang-Landau sampling.[10] The authors have determined the reaction coordinate of the first-order phase transition between the random coil and crystalline state to be a linear combination of the potential energy and Steinhardt order parameter Q_6. In a followup study Leitold *et al.* obtained the reaction coordinate that is nonlinear combination of potential energy and Q_6.[11] They have found that in this work that the initial stages of crystallization is dominated by Q_6, while changes in the potential energy dominate the latter stage of the crystallization.

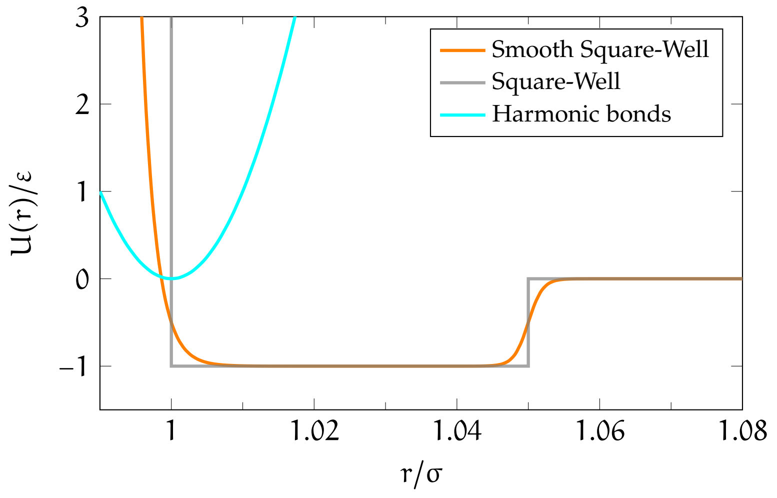


Figure 3.4: The square-well potential, the smooth square-well potential and the harmonic bond potential.

In this work we study the phase transition of the 128-unit single-chain smooth square-well polymer (SCP) with well-tempered metadynamics by directly biasing the linear combination of potential energy and Steinhardt order parameter Q_6 at three temperatures. We not only demonstrate ESPResSo++ working in tandem with PLUMED, but also the effectiveness of the linear combination of the potential energy U and Q_6 as CV in a WTMetaD simulation of single-chain smooth square-well polymer. We have obtained the reweighted free energy surface estimation $F(U, Q_6)$ with two reweighting methods. We have obtained the energy profile $U(Q_6)$ and entropy profile $S(Q_6)$ at $T = 0.438$, with which we estimate the free energy surfaces at two other temperatures by extrapolation. The coexistence temperature of the 128-monomer single-chain smooth square-well polymer has been determined with the free energy of crystallization at three temperatures.

3.2.2 Simulations and Methods

We have performed our simulations with the Langevin thermostat, with the frictional coefficient γ equal to 0.5. A developmental version based on ESPResSo++ 2.02 was used in our simulation. The time step is $0.0002\sqrt{\frac{\varepsilon}{m\sigma^2}}$. The extremely short time step is necessary because of the steepness of the repulsive region of the smooth square-well potential. The single-chain polymer occupies a cubic box with a dimension of 1000σ. The cutoff distance for pairwise interactions is 1.5σ. We ran the simulations at three temperatures, one at previously determined $T_{co} = 0.438$, one at 0.418 with 4.6% supercooling, and the final one at 0.458 which is 4.6% above the coexistence temperature. PLUMED version 2.5.2 was used in the biased simulations. We ran all three simulations for $400000\sqrt{\frac{\varepsilon}{m\sigma^2}}$. We have implemented the smooth square-well potential in ESPResSo++, whose source code can be found in Appendix B.

STEINHARDT ORDER PARAMETER The Steinhardt order parameter is based on spherical harmonic functions Y_{lm}, which is used to describe the relative orientations of neighboring sites. The Steinhardt order parameter is regularly used in distinguishing different crystal structures and liquids or the glass phase.[12] First we compute $q_{lm}(i)$ for each atom, which is the weighted average spherical harmonic functions Y_{lm} between atom i and its closest neighbors, by:

$$q_{lm}(i) = \frac{\sum_j \sigma(r_{ij}) Y_{lm}(\mathbf{r}_{ij})}{\sum_j \sigma(r_{ij})}, \tag{3.3}$$

in which the index l is an even integer greater than 0, and m is also an integer that runs from $-l$ to $+l$. The weigh $\sigma(r_{ij})$ is a switching function that is computed as:

$$\sigma(r_{ij}) = \frac{1 - \left(\frac{r_{ij}-d_0}{r_0}\right)^N}{1 - \left(\frac{r_{ij}-d_0}{r_0}\right)^M}, \tag{3.4}$$

where usually the exponent $M = 2N$. The switching function $\sigma(r_{ij})$ ranges from 0 to 1. An example of the switching function $\sigma(r_{ij})$ is shown in Figure 3.5.

The average vector $\bar{q}_{6m}$ is averaged over all the atoms:

$$\bar{q}_{lm} = \frac{\sum_i^N q_{lm}(i)}{N}. \tag{3.5}$$

And finally Q_l is computed as:

$$Q_l = \left(\sum_{m=-l}^{l} \bar{q}_{lm}^* \bar{q}_{lm}\right)^{1/2}, \tag{3.6}$$

where $\bar{q}_{lm}^*$ is the conjugated vector of $\bar{q}_{lm}$.

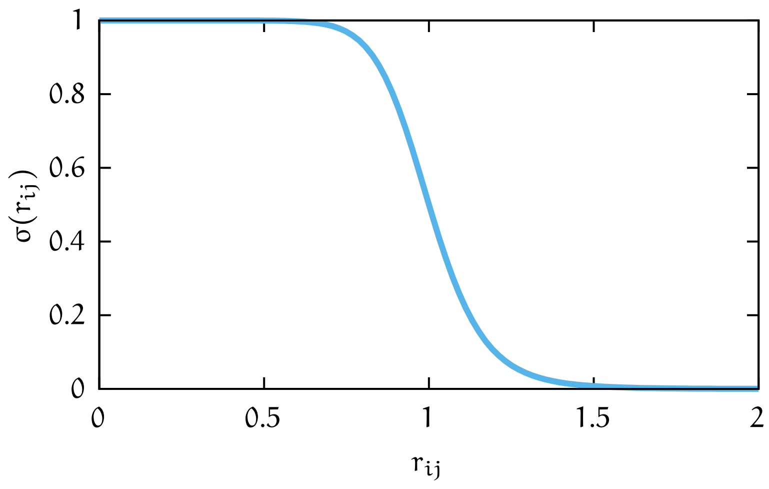


Figure 3.5: An example of the switching function $\sigma(r_{ij})$. In this example, $d_0 = 0$, $r_0 = 1.0$, $N = 12$, $M = 24$. These parameters are used in the calculation of Q_6 of the single-chain smooth square-well polymer.

In this case $l = 6$ and Q_6 is computed for the square-well polymer. Another commonly used Steinhardt order parameter is Q_4, which has been used as CV in metadynamics simulations of ice nucleation.[13] Q_4 and Q_6 ranges from 0 to 1. Generally speaking, a higher value indicates a higher degree of crystallinity. In the case of the single-chain square-well polymer, when $Q_6 > 0.20$, the polymer is in the crystalline state. Otherwise it is a random coil.

As discussed in Section 3.2.1, nonbonded contacts are a good indicator of the phase of the single-chain polymer, which can be counted through the potential energy by proxy, as the pairwise interaction is only favorable in the very narrow well. When the potential energy $U < -179.5$, we say the polymer is in the crystalline state.

WELL-TEMPERED METADYNAMICS The well-tempered metadynamics is used in the biased sampling. The CV is a linear of combination of the potential energy U and Q_6:

$$s = 4.07430109214 Q_6 - 0.0187482044056 U, \tag{3.7}$$

which is obtained from Leitold & Dellago [10]. In our simulations we have used these coefficients faithfully. We do not contend that the results of the simulations are susceptible to changes to numbers after a few decimal points. We consider the polymer is in the crystalline state when $s > 3.95$.

The bias factor is chosen to be 30. The width σ of the Gaussian kernel is set to 0.1. The initial height of the Gaussian kernel for each of three simulations is set to equal to $k_B T$ for each temperature. Gaussian kernels are deposited onto a grid to accelerate the computation of the bias potential and the derived forces. We chose 1200 bins for s from -1 to 20. A new Gaussian kernel is deposited every 2000 steps.

UPPER WALL When the polymer is in the coil state, there is no effective interaction between monomers in the majority range of the separation. Nonboned interactions contribute very little to the potential energy U, the bias potential exerted by the well-metadynamics can lead to nonphysical bond stretching. Therefore, we apply an upper wall on the potential energy:

$$V(U) = K(U - U_0)^2, \tag{3.8}$$

in which $U_0 = 5\varepsilon$, and $K = 2.5$. We find that with this upper wall we can effectively prohibit the biased simulations from visiting nonphysical states.

The plumed.dat file, which contains the commands and parameters for the upper wall and well-tempered metadynamics, and the Python script that sets up simulations with ESPResSo++ are include in Appendix C.

3.2.3 Methods of Analysis

FREE ENERGY SURFACE We have performed the estimation of the free energy surface for the biased simulation at $T = 0.438$ in three ways. First, we have done so directly from the bias potential $V(\mathbf{s}, t)$ using Equation 2.16. We have used the `sum_hills` utility provided by the PLUMED library as a command line tool to perform the calculation. With this method, the free energy estimation can only be done with respect to the biased CV, namely the linear reaction coordinate s in this case. The other two methods are the two reweighting methods introduced in Section 2.3. We discarded the initial 20% of the data of the biased simulations, as we have found that spurious features emerged in the free energy surface due to higher fluctuations of the bias potential at the beginning of the simulation. We have obtained three one-dimensional free energy curves, $F(s)$, $F(Q_6)$, and $F(U)$. We also have obtained the two-dimensional $F(U, Q_6)$, which will be used to compute the entropy profile $S(Q_6)$ at $T = 0.438$.

In the reweighting procedures, we used the kernel density estimation (KDE) to obtain an accurate and smooth curve. For each CV a bandwidth parameter is used with KDE. Specifically, The bandwidths are 0.1, 0.012 and 5 for the CV s, the Steinhardt order parameter Q_6 and the potential energy U, respectively.

BLOCK ANALYSIS The standard error along the free energy surface can be computed using the block analysis method.[14] The entire simulation is divided into five consecutive blocks of equal length so that the unbiased probability distribution of the CV s can be estimated independently in each block. We used both the residual bias reweighting method and the last bias reweighting method so that we can compare the convergence of the sampling from two the reweighting methods. We used the `histogram` command with the discrete kernel inside PLUMED to estimate the probability distribution. For each method we computed standard errors (σ_p) of the probability distribution. According to the rule of error propagation for a logarithmic function:

$$y = a \ln(bx), \tag{3.9}$$

$$\sigma_y^2 = \left(a\frac{\sigma_x}{x}\right)^2. \tag{3.10}$$

As the free energy surface $F(s) = -\frac{1}{\beta}\ln p(s)$, σ_F the standard error of the free energy F is related to the probability distribution p by

$$\sigma_F = \frac{1}{\beta} \times \frac{\sigma_p}{p}. \tag{3.11}$$

The standard errors of the free energy is a measure of the convergence of the biased simulations.

FREE ENERGY OF CRYSTALLIZATION The free energy of crystallization ΔG_{cry} has also been computed. The probability distribution of the linear reaction coordinate s is integrated in two regions, separated by $s = 3.95$, below which is the crystalline region, and above is the random coil region, to obtain the probability of crystalline state $\mathcal{P}_{cry}$ and the probability of the coil state $\mathcal{P}_{coil}$. The free energy crystallization $\Delta G_{cry} = -\frac{1}{\beta}\ln\frac{\mathcal{P}_{cry}}{\mathcal{P}_{coil}}$. By computing the free energy of crystallization along the time of the simulation, we can evaluate the convergence of the sampling. We have computed ΔG_{cry} for all three temperatures so that we can estimate the coexistence temperature for the single-chain smooth square-well polymer.

ENTROPY The entropy profile $S_{T_0}(Q_6)$ at $T = 0.438$ is computed with the aid of the two-dimensional free energy projection to (U, Q_6). We use the same method introduced by Michel *et al.*[15] to do so. In their metadynamics simulation study of the interaction of a water dimer, the potential energy U is one of the collective variables that were biased in the simulations, whereas in this work we have used reweighting to obtain $F(U, Q_6)$. It highlights the flexibility of reweighting to obtain an estimate of the free energy surface. First, we computed the potential energy profile $U_{T_0}(Q_6)$ at T_0:

$$U_{T_0}(Q_6) = \frac{\int dU\, U \exp(-\beta F_{T_0}(U, Q_6))}{\int dU\, \exp(-\beta F_{T_0}(U, Q_6))}, \tag{3.12}$$

and the free energy profile $F_{T_0}(Q_6)$ is obtained from reweighting. And we compute the entropy profile:

$$S_{T_0}(Q_6) = (U_{T_0}(Q_6) - F_{T_0}(Q_6))/T_0. \tag{3.13}$$

Assuming that the potential energy profile $U_{T_0}(Q_6)$ and entropy profile $S_{T_0}(Q_6)$ are insensitive to the change in temperature, one can compute the free energy $F_T(Q_6)$ at temperatures other than the one, at which the biased simulation was performed, through the following equation:

$$F_T(Q_6) = U_{T_0}(Q_6) - TS_{T_0}(Q_6). \tag{3.14}$$

As we did conduct enhanced sampling simulations of the single-chain smooth square-well polymer at two other temperatures, we can assess the feasibility of this method of estimating FES with simulations performed at different temperatures, at least for the single-chain smooth square-well polymer.

3.2.4 Results and Discussions

The time series of the linear reaction coordinate s, potential energy U, and Steinhardt order parameter Q_6 for the biased simulation at $T = 0.438$ are shown in Figure 3.6. There are about a dozen of transitions between the two states, as illustrated by Figure 3.6a. The times series of the potential energy U and Steinhardt order parameter Q_6 also display the same transitions between the crystalline state and the random coil state, as evidenced by Figure 3.6b and Figure 3.6c. For the well-tempered metadynamics simulations at the other two temperatures, the time series are shown in Figure 3.7. Based on these time series we conclude that enough number of transitions have occurred for us to estimate the FES from these simulations.

The free energy profile with respect to s is first estimated directly with the bias potential $V(s,t)$, as shown in Figure 3.8. The free energy profile at the random coil state at lower values of s is well converged from the early stage of the simulation, while the free energy profile of the crystalline state at high values of s is not yet converged with large fluctuations between different time. Similarly, the height of free energy separating the crystalline state and random coil state is neither well converged, making it difficult to estimate the true value of the height of the free energy barrier.

The reweighted free energy profile with respect to s is shown in Figure 3.9. Both residual bias reweighting and last bias reweighting have been employed. We show later that these reweighted free energy profiles have much better convergence than that estimated from that directly estimated from the bias potential. As already stated, we have excluded the initial 20% of the trajectory from the reweighting process. The two free energy profiles show a quantitative agreement on the height of the free energy barrier ($\sim 20 k_B T$) for transitions from the random coil state, however, they differ in the depth of the local minimum at the crystalline state. Previous studies[10,11] have reported

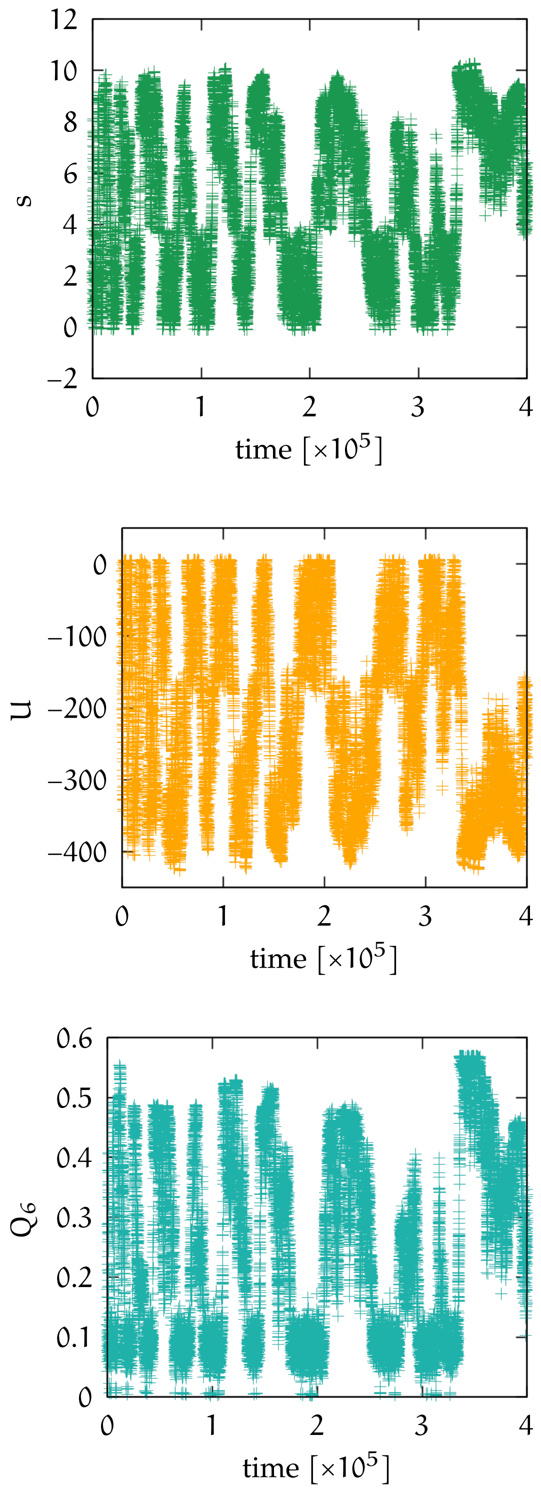
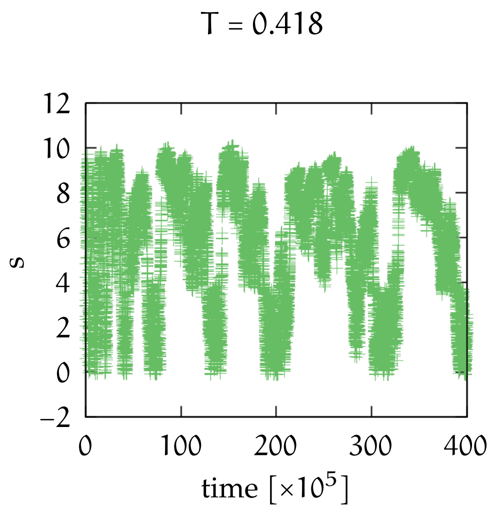
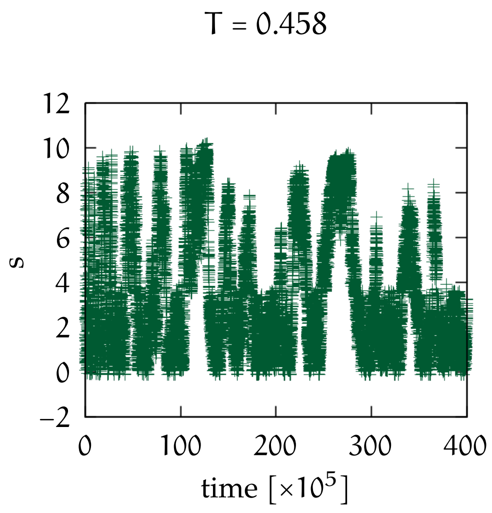


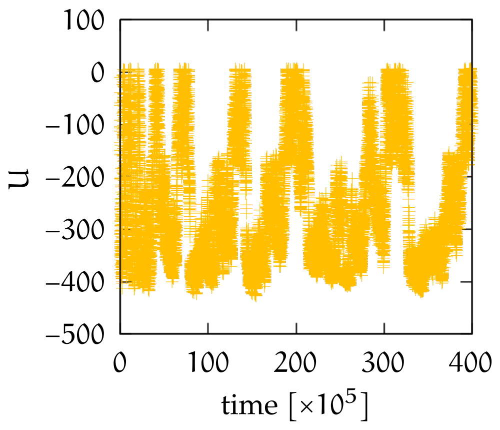
Figure 3.6: The time series of the biased CV s, potential energy U, and Steinhardt order parameter Q_6 of the biased simulations at $T = 0.438$.



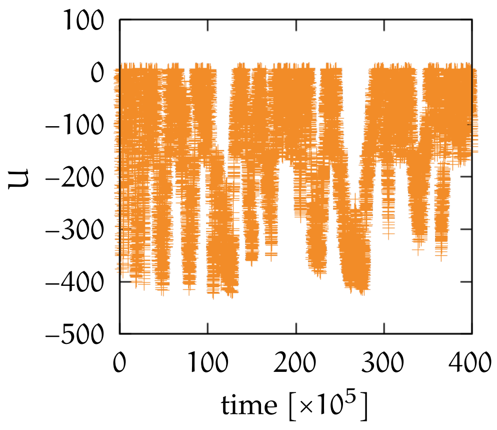
(a)



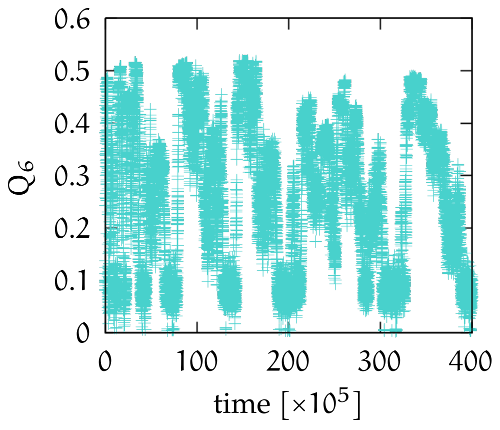
(b)



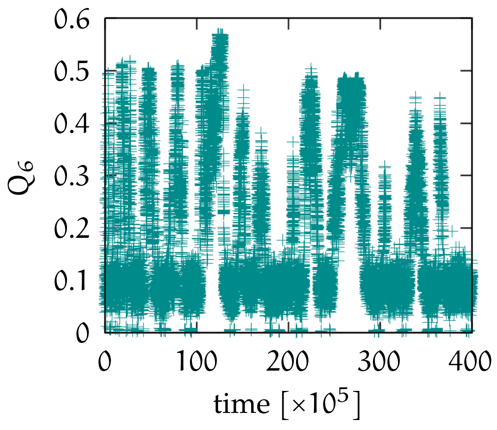
(c)



(d)



(e)



(f)

Figure 3.7: The time series of the biased CV s, potential energy U, and Steinhardt order parameter Q_6 of the biased simulations at T = 0.418 and T = 0.458.

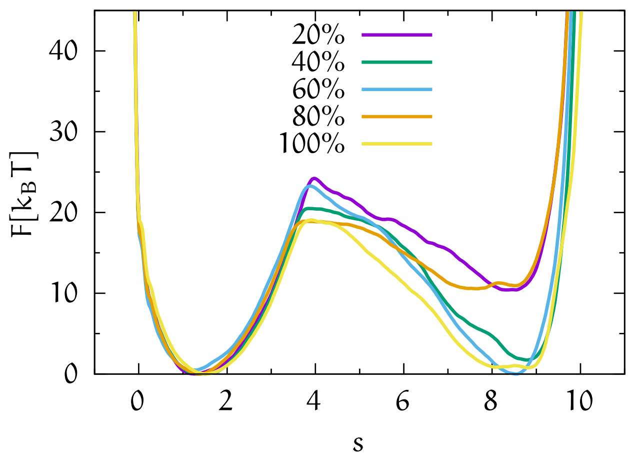


Figure 3.8: Free energy profiles F with respect to the linear reaction coordinate s at five stages evenly spaced over the course of the well-tempered metadynamics simulation. The free energy estimation was performed using the relationship $V(\mathbf{s}, t) = -(1 - \frac{1}{\gamma})F(\mathbf{s}) + c(t)$.

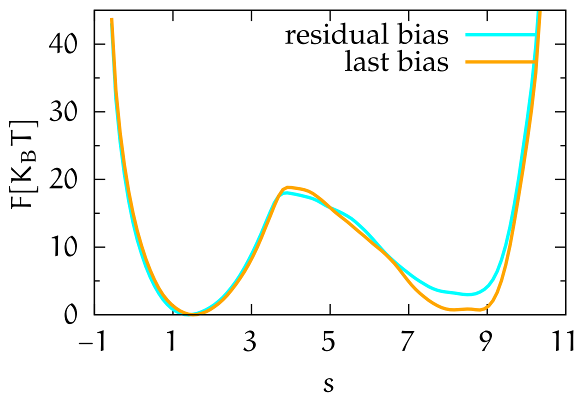


Figure 3.9: Reweighted free energy surface with respect to the linear reaction coordinate s with the last bias reweighting method and residual bias reweighting method.

the height the free energy barrier at $T = 0.438$ to be about $20k_BT$, in agreement with our results.

We examine the convergence of the reweighted free energy surfaces by their statistical standard errors. The standard errors for the two reweighted probability distributions and free energy profiles about the linear reaction coordinate s were computed using the block analysis method, as shown in Figure 3.10. We have divided the simulation

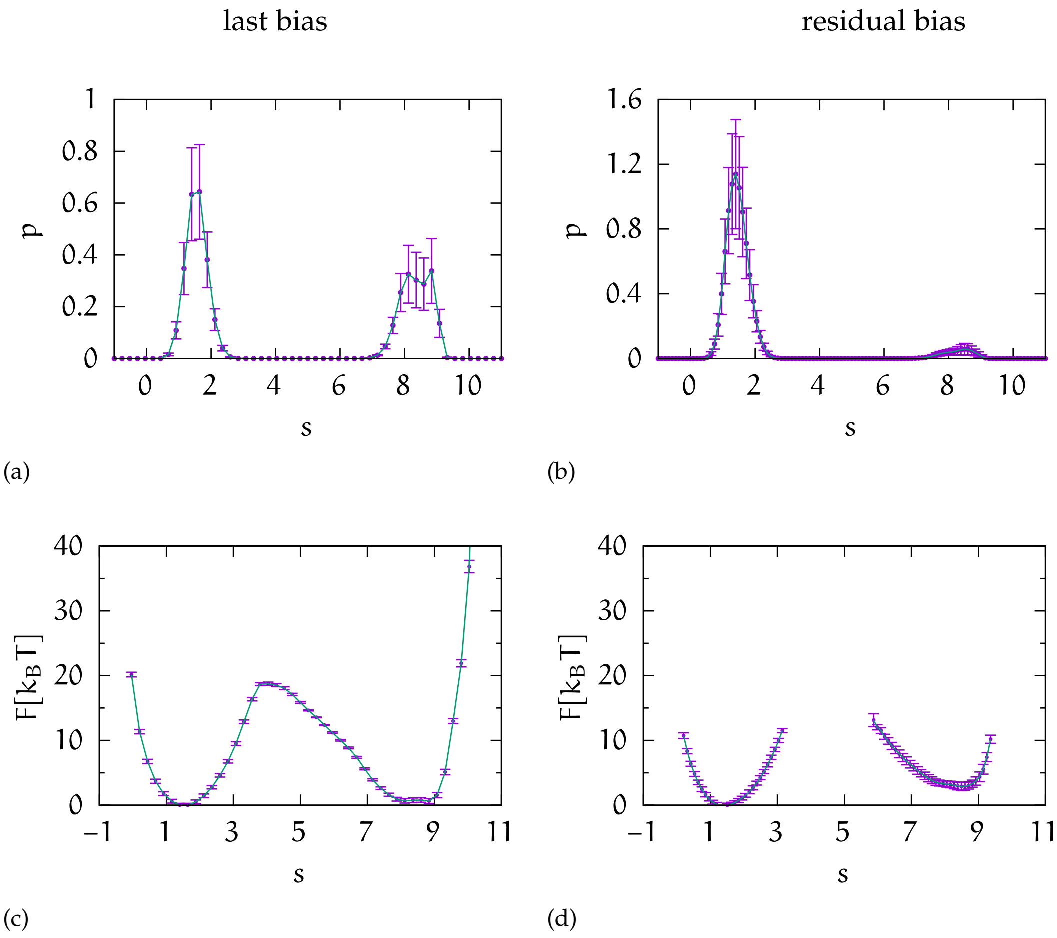
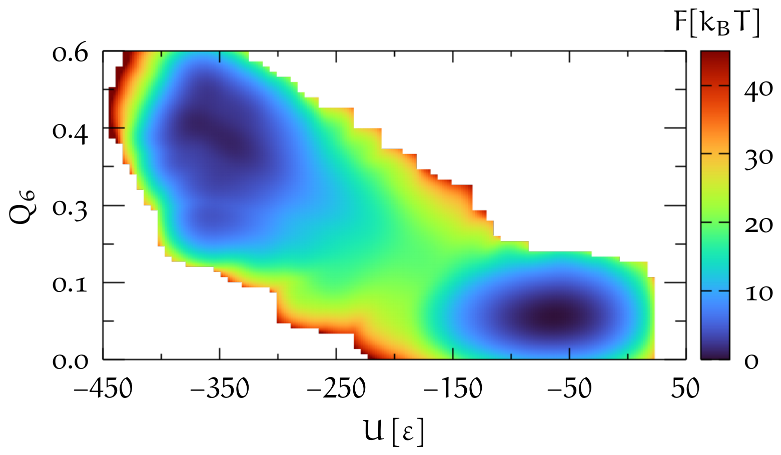


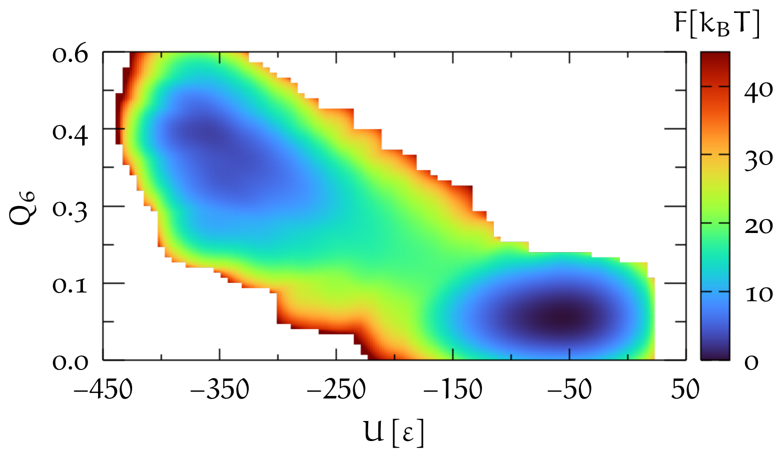
Figure 3.10: The standard errors of the reweighted histograms and free energy profiles of the linear reaction coordinate s at $T = 0.438$. (a) and (c) are the reweighted histogram and free energy profile of s with last bias reweighting. (b) and (d) are the reweighted histogram and free energy surface of s obtained with residual bias reweighting. The reweighted histograms were computed by binning instead of the kernel density estimation as previously.

without the initial 20% into five blocks of equal length to obtain independent reweighted histograms with both residual bias reweighting and last bias reweighting. In Figure 3.10, the average histograms with their standard errors are shown instead of the five independent histograms. The average free energy profiles were computed with the average histograms, according to $F = -1/\beta \ln p$. Although in Figure 3.10d the standard errors at the portion of the free energy barrier cannot be computed, as the probability distribution near the barrier is close to zero. The figures convincingly demonstrate that the reweighted free energy surfaces have converged to a quantitative degree to our satisfaction.

To better visualize and understand the transition path between the crystalline state and random coil state, we have obtained the two-dimensional reweighted FES projected onto U and Q_6 at $T = 0.438$ with the two reweighting methods, and given the results in Figure 3.11. In



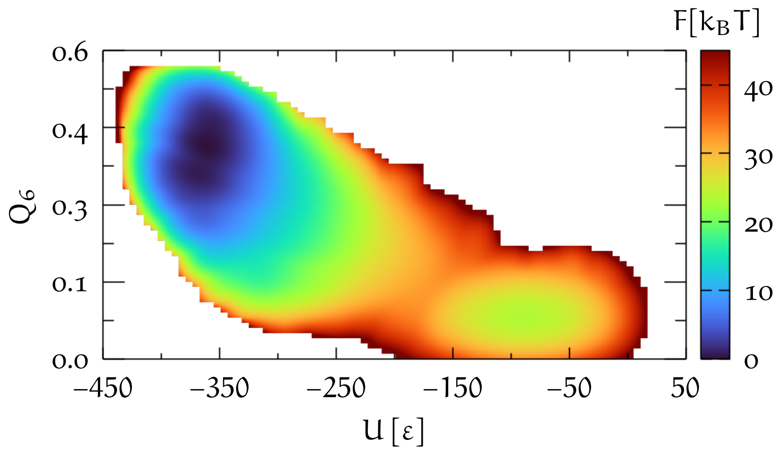
(a) last bias



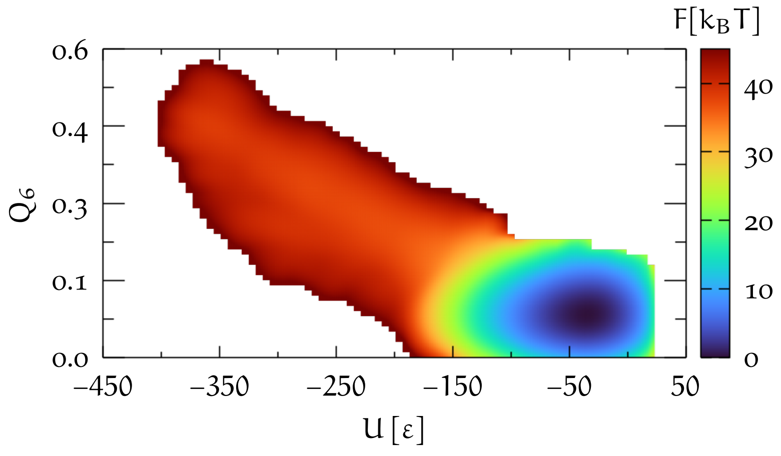
(b) residual bias

Figure 3.11: The reweighted 2D free energy surface $F(U, Q_6)$ for the single-chain smooth square-well polymer at $T = 0.438$, estimated with residual bias reweighting and last bias reweighting.

these two-dimensional FESs, the crystalline state resides in the top left corner of the FES, and the random coil state resides in bottom right corner. Both reweighted free energy surfaces suggest that the linear combination of U and Q_6 is a decent choice for the collective variable, even though it seems that at the random coil state the fluctuation of the potential energy U is greater than that of Q_6. One can observe on the FES of last bias reweighting (Figure 3.11a) a narrow faint metastable state tightly below the crystalline state at about $Q_6 = 0.2$ and $U = -370$, which is invisible in the FES by residual bias reweighting. However, we have not observed a structurally different crystalline form in this narrow metastable state. Furthermore, we have found that excluding more data of the initial stage of the simulations from the last bias reweighting process does not eliminate this metastable state. We conjecture that it is an artifact from the last bias reweighting process. $F(U, Q_6)$ for the other two temperatures can be found in Figure 3.12.



(a) T=0.418

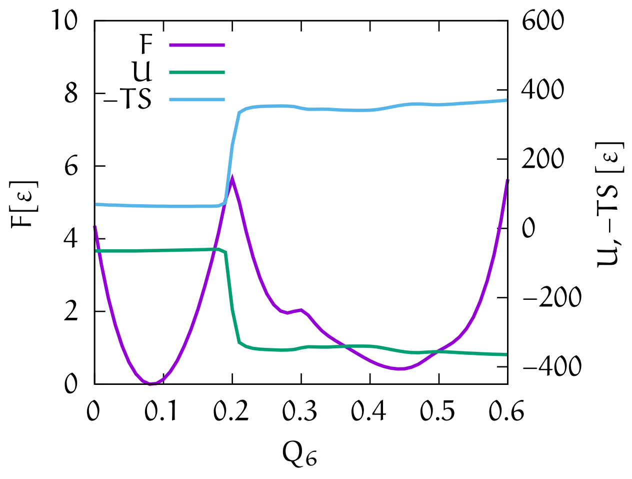


(b) T=0.458

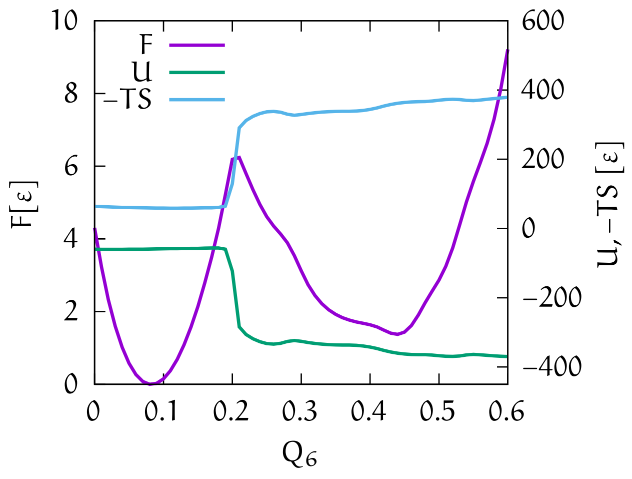
Figure 3.12: $F(U, Q_6)$ at T=0.418 and T=0.458, estimated by last bias reweighting.

The reweighted two-dimensional FESs are integrated along Steinhardt order parameter Q_6 to obtain the potential energy profile $U(Q_6)$ at $T = 0.438$. The entropy profile $S(Q_6)$ can also be computed for

$T = 0.438$ according to earlier introduced method. We show the results in Figure 3.13. The left y axis indicates the values of $F(Q_6)$, and the right y axis specifies the values of $U(Q_6)$ and $-TS(Q_6)$, which are two orders of magnitude higher than that of $F(Q_6)$.



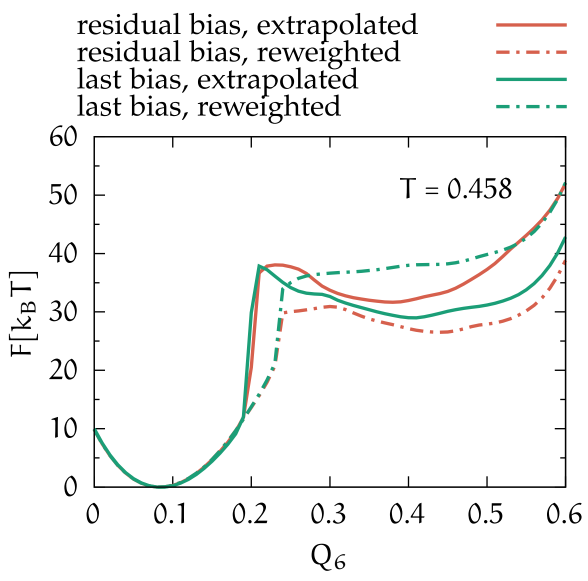
(a) last bias



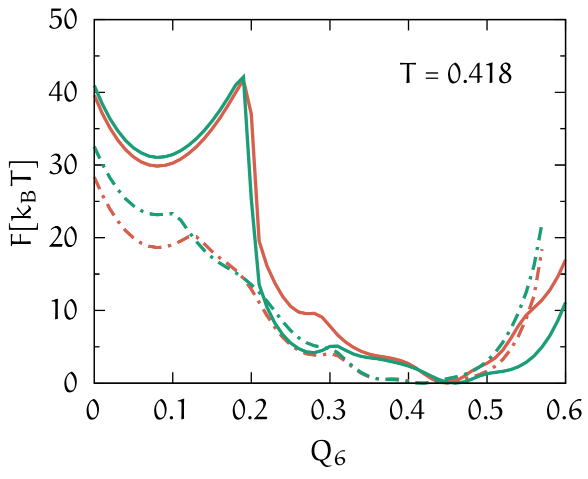
(b) residual bias

Figure 3.13: Free energy profile $F(Q_6)$, the potential energy profile $U(Q_6)$, and entropy $-TS(Q_6)$ with respect to Q_6 at $T = 0.438$. As labeled, the left axis is the scale of F in the unit of ε, and the right axis is the scale of U and $-TS$ also in the unit of ε.

Assuming that both entropy $S(Q_6)$ and the potential energy profile $U(Q_6)$ are insensitive to temperature, we can estimate the free energy surface $F(Q_6)$ at the other two temperatures. As we have conducted WTMetaD simulations for the single-chain square-well polymer at two other temperatures, we can compare the reweighted results with the FES obtained from extrapolation. The results are shown in Figure 3.14. As can be seen in Figure 3.14a, at the higher temperature $T = 0.458$, extrapolation does a good job reproducing the free energy surface



(a)

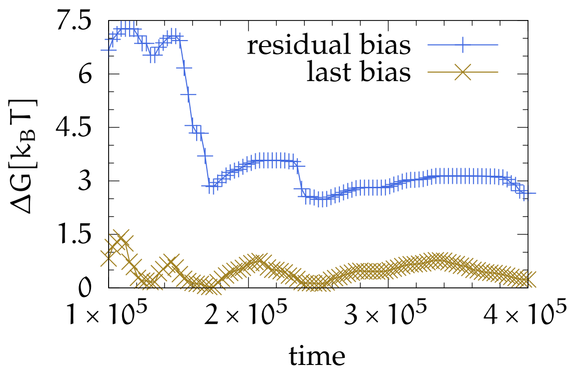


(b)

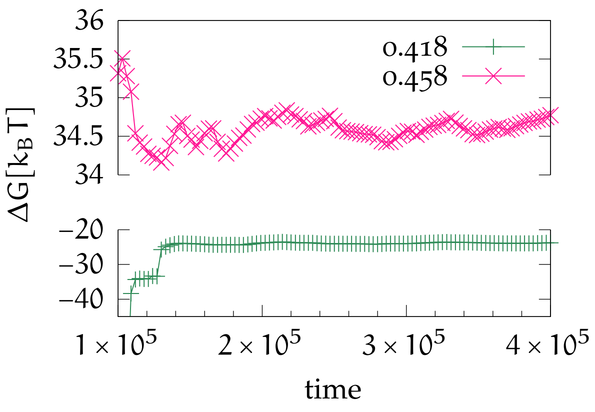
Figure 3.14: Extrapolation of free energy profiles to $T = 0.418$ or 0.438 from the potential energy profile U and entropy S obtained at $T = 0.438$.

at the random coil state, regardless that the data are obtained from residual bias reweighting or last bias reweighting. However, it does poorly at the free energy barrier and crystalline sate. The results of extrapolation fails completely across the board for $T = 0.418$ when compared with the results of the both reweighting methods (Figure 3.14b). The poor performance of the extrapolation method to estimate the FES may be due to the large values of the potential energy profile $U(Q_6)$ and entropy profile $-TS(Q_6)$ (Figure 3.13) and the relative low values of $F(Q_6)$. Even though there is compensation between $U(Q_6)$ and $-TS(Q_6)$, their statistical errors and fluctuations could overwhelm $F(Q_6)$ and drown out the features of $F(Q_6)$, considering that they are two orders of magnitude larger. However, we can not exclude that it is unsound to assume that that the potential energy profile and entropy profile are insensitive to temperature in the case of single-chain smooth square-well. Nevertheless, the results of extrapolation at $T = 0.458$ still gave a good estimation of $F(Q_6)$ at low values of Q_6 at a fraction of the cost of simulations.

Leitold and Dellago have previously determined the coexistence temperature for the 128-monomer smooth square-well polymer with $\lambda = 1.05$.[10] Here we estimate it by using the free energy of crystallization. We have computed ΔG_{cry} for all three biased simulations at three temperatures. The results for $T = 0.438$ are given in Figure 3.15a. And the figures for the other two temperatures can be found in Figure 3.15b.



(a) $T = 0.438$



(b) $T = 0.418$ and 0.458

Figure 3.15: (a) Free energy of crystallization ΔG_{cry} at $T = 0.438$ obtained through last bias reweighting and residual bias reweighting. (b) Free energy of crystallization ΔG_{cry} at $T = 0.418$ and $T = 0.458$ using data from last bias reweighting.

Figure 3.15a shows that the free energy of crystallization from the last bias reweighting has converged much sooner, and that the fluctuations of ΔG_{cry} are much lower than those from the residual bias reweighting. Therefore, we have chosen to use the results of the last bias reweighting to estimate the coexistence temperature. And the curves in Figure 3.15b are also from last bias reweighting. Instead of using the final values of ΔG_{cry} at each temperature, we have used the average values of ΔG_{cry} after initial large fluctuations in our estimation of the coexistence temperature. The average values of ΔG_{cry} and their standard errors are listed in Table 3.1. We have

T	0.418	0.438	0.458
$\langle \Delta G_{cry} \rangle$	-10.0141	0.1865	15.88478
σ_G	0.01345	0.009204	0.009152

Table 3.1: The averages and standard errors of free energy of crystallization ΔG_{cry} in unit of ε.

plotted the average free energy of crystallization $\langle \Delta G_{cry} \rangle$ and a linear fit by least-squares-regression in Figure 3.16, in which the unit of $\langle \Delta G_{cry} \rangle$ is ε rather than $k_B T$ as in other figures. The R^2 value of the linear regression is 0.985. The coexistence temperature is 0.435 with a standard error 0.002, based on the linearly fitted line. The previously reported 0.438 fails narrowly outside one standard error of 0.435. The R script used to compute the standard error for the estimated coexistence temperature can be found in Listing C.3. The standard errors of the free energy of crystallization σ_G (Table 3.1) is too little compared to $\langle \Delta G_{cry} \rangle$, and thus not plotted in Figure 3.16.

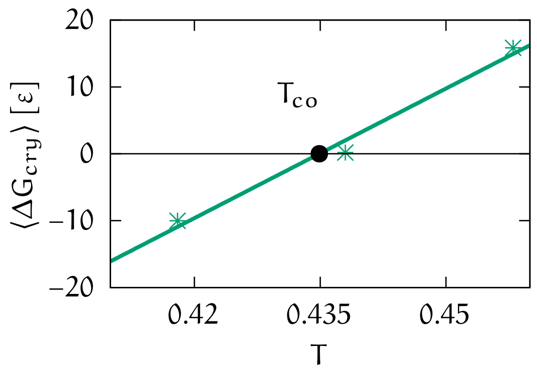


Figure 3.16: The coexistence temperature T_{co} of the single-chain smooth square-well polymer estimated using free energy of crystallization.

3.2.5 Conclusions

ESPResSo++, a MD simulation engine developed and maintained by researchers from Max Planck Institute for Polymer Research and University of Mainz, is often used in soft matter simulations. PLUMED 2 is a software plugin that be used with MD engines concurrently to calculate CVs and conducts enhanced sampling in the real time. We have created an interface between the two software packages in the form of an extension, which inherits from the `MDIntegrator` class inside ESPResSo++. As an example, we have studied the first-order phase transition of single-chain smooth square-well polymer with 128 monomers. A prototypical singe-chain square-well polymer has rigid bonds and square-well interactions between nonbonded monomers. Here, we model the rigid bonds with harmonic bonding interactions, the square-well potential with a smooth square-well potential. In this way we are able to study its phase transition with well-tempered metadynamics simulations at three temperatures, including the previously reported coexistence temperature, about 5% above it, and about 5% below it. The 128-monomer single-chain smooth square-well polymer has a random coil state and a crystalline state that are separated by a significant free energy barrier. We have found in agreement with previous studies that the height of the free energy at $T = 0.438$ is approximately $20k_BT$. We also have obtained the free energy profiles at the other two temperatures by extrapolation from the potential energy profile and the entropy profile with respect to the Steinhardt order parameter Q_6 at $T = 0.438$. When compared with the reweighted free energy profiles, the extrapolated free energy profiles performed generally poorly at predicting the reweighted free energy profiles. For the single-chain smooth square-well polymer enhanced sampling simulations cannot be dispensed with by extrapolation. We have also computed free energy of crystallization at the three temperatures, based on which we have determined the coexistence temperature T_{co} of the single-chain smooth square-well polymer to be 0.435, with a standard error of 0.002. The previously reported 0.438 value is narrowly outside one standard error of our result.

REFERENCES

1. Lenz, O. *Pmi - Parallel Method Invocation* in *Proceedings of the 8th Python in Science Conference* (Pasadena, CA USA, 2009), 48–50.
2. Yethiraj, A. & Hall, C. K. Generalized Flory Equations of State for Square-well Chains. *The Journal of Chemical Physics* **95,** 8494–8506 (Dec. 1, 1991).
3. Taylor, M. P., Paul, W. & Binder, K. All-or-None Proteinlike Folding Transition of a Flexible Homopolymer Chain. *Physical Review E* **79** (May 7, 2009).
4. Taylor, M. P., Paul, W. & Binder, K. Phase Transitions of a Single Polymer Chain: A Wang–Landau Simulation Study. *The Journal of Chemical Physics* **131,** 114907 (Sept. 21, 2009).
5. Taylor, M. P., Paul, W. & Binder, K. Two-State Protein-like Folding of a Homopolymer Chain. *Physics Procedia* **4,** 151–160 (2010).

6. Taylor, M. P. & Lipson, J. E. G. Collapse of a Polymer Chain: A Born–Green–Yvon Integral Equation Study. *The Journal of Chemical Physics* **104,** 4835–4841 (Mar. 22, 1996).

7. Růžička, Š., Quigley, D. & Allen, M. P. Folding Kinetics of a Polymer. *Physical Chemistry Chemical Physics* **14,** 6044–6053 (Apr. 4, 2012).

8. Vorselaars, B., Růžička, Š., Quigley, D. & Allen, M. P. Correction: Folding Kinetics of a Polymer. *Physical Chemistry Chemical Physics* **19,** 5674–5676 (Feb. 15, 2017).

9. Wicks, T. J., Wattis, J. A. D. & Graham, R. S. Monte–Carlo Simulation of Crystallization in Single-Chain Square-Well Homopolymers. *POLYMER CRYSTALLIZATION* **4,** e10146 (Feb. 1, 2021).

10. Leitold, C. & Dellago, C. Folding Mechanism of a Polymer Chain with Short-Range Attractions. *The Journal of Chemical Physics* **141,** 134901 (Oct. 7, 2014).

11. Leitold, C., Lechner, W. & Dellago, C. A String Reaction Coordinate for the Folding of a Polymer Chain. *Journal of Physics: Condensed Matter* **27,** 194126 (May 20, 2015).

12. Steinhardt, P. J., Nelson, D. R. & Ronchetti, M. Bond-Orientational Order in Liquids and Glasses. *Phys. Rev. B* **28,** 784–805 (July 15, 1983).

13. Quigley, D. & Rodger, P. M. Metadynamics Simulations of Ice Nucleation and Growth. *J. Chem. Phys.* **128,** 154518 (Apr. 18, 2008).

14. Bussi, G. & Tribello, G. A. *Analyzing and Biasing Simulations with PLUMED* 2019. arXiv: 1812.08213 [cond-mat, physics:physics, q-bio].

15. Michel, C., Laio, A. & Milet, A. Tracing the Entropy along a Reactive Pathway: The Energy As a Generalized Reaction Coordinate. *Journal of Chemical Theory and Computation* **5,** 2193–2196 (Sept. 8, 2009).

4 VARIATIONALLY ENHANCED SAMPLING WITH PERMUTATIONALLY INVARIANT COLLECTIVE VARIABLES

In Chapter 2, we have presented two enhanced sampling methods, well-tempered metadynamics and the variationally enhanced sampling method. Both well-tempered metadynamics and the variationally enhanced sampling method build a bias potential that is customized to the underlying the free energy surface. The former does so through periodic depositions of Gaussian kernels to the bias potential at prior visited CV space, and the latter through iteratively minimizing the functional Ω of the bias potential. In our study of the single-chain smooth square-well polymer in Chapter 3, we have demonstrated a real world application of well-tempered metadynamics.

In both methods the bias potential is applied to a low dimensional CV space $\mathbf{s} = [s_1, s_2, \ldots, s_n]$, in which each CV s is a smooth differentiable function of the configuration space $\mathbf{r}$. Furthermore, collective variables must also be able to discriminate among different metastable states of a system and describe different transition pathways between metastable states in a complex free energy landscape. It means that often multiple CVs are deployed in enhanced sampling. As the number of CVs in the CV space $\mathbf{s}$ increases, quickly the efficiency of enhanced sampling methods diminishes. For example, the rate of convergence of the sampling in well-tempered metadynamics is inversely proportional to the dimensionality of the CV space. Although it can pose a challenge to enhanced sampling methods if the dimensionality of the CV space is greater than two, it is sometimes inevitable to employ a few CVs.

A collective variable can be roughly characterized as global collective variables or local collective variables, based on whether it describes the entire simulation box or only the local molecular environments. A global collective variable could be the system density, or the system potential energy. The system potential energy is the primary biased CV in well-tempered ensemble simulations.[1,2] Two examples of local collective variables are the dihedral angles of alanine dipeptide, as shown in Figure 4.1, which have been used in enhanced sampling simulations with well-tempered dynamics[3] and variationally enhanced sampling[4] to study the free energy landscape of its conformers in vacuum. It is evident that the dihedral angles Φ and Ψ are not equivalent.

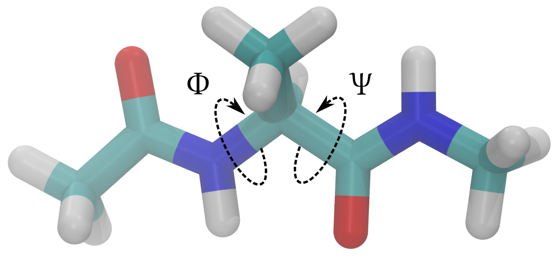


Figure 4.1: The molecular structure of alanine dipeptide and the two dihedral angles Φ and Ψ are the two often biased local collective variables in enhanced simulations.

In contrast, other types of local collective variables whose indices can be arbitrarily exchanged without altering the system, are equivalent CVs. The free energy surface projected onto any one of a group of equivalent local collective variables should be the same as that onto any of the others. It is not true in the case of the two dihedral angles Ψ and Φ in alanine dipeptide for their different chemical environment. We call these equivalent local collective variables permutationally invariant collective variables (PICVs). Examples of PICVs include averaged local bond order parameter $\bar{q}_l(i)$ (e. g., $l = 4$ or 6)[5], CHILL+[6], coordination numbers[7], kernel CVs[8–10], and the individual contributions to SMAC[11]. We introduce the coordination number and the kernel CV later in this chapter, and SMAC in Chapter 5.

Similar to the Steinhardt order parameters Q_l (e. g., $l = 4$ or 6) introduced in Chapter 3, the averaged local bond order parameter $\bar{q}_l(i)$ is also based spherical harmonic functions. We have used Q_6 to characterize the crystalline state of single-chain smooth square-well polymer in the previous chapter. When Q_l is computed, a global average vector $\bar{q}_{lm}$ is computed according to Equation 3.5. Whereas when $\bar{q}_l(i)$ is computed, a local average vector $\bar{q}_{lm}(i)$ is computed as

$$\bar{q}_{lm}(i) = \frac{1}{\tilde{N}_b(i)} \sum_{k=0}^{\tilde{N}_b(i)} q_{lm}(k), \tag{4.1}$$

where $\tilde{N}_b(i)$ is the number of closest neighbors of atom i within a cutoff distance. Then $\bar{q}_l(i)$ for atom i can be computed according to

$$\bar{q}_l(i) = \sqrt{\frac{4\pi}{2l+1} \sum_{m=-l}^{l} |\bar{q}_{lm}(i)|^2}. \tag{4.2}$$

The averaged bond order parameters can be computed for every atom in a bulk sodium crystal in a BCC lattice as the one shown in Figure 4.2. In fact, the averaged bond order parameters were developed for identifying various packing orders such as BCC, HCP and FCC, of Lennard-Jones particles by the probability distributions of $\bar{q}_4(i)$ and $\bar{q}_6(i)$.[5] In contrast the global Steinhardt order parameters Q_4 and Q_6 do not sufficiently distinguish the HCP and BCC packing.[5] However, we do not use $\bar{q}_4(i)$ and $\bar{q}_6(i)$ for our simulations of bulk sodium in this work due to their high computation cost. Nevertheless, they are a good instrument to illustrate PICVs when contrasted with the related global CV, the Steinhardt order parameters.

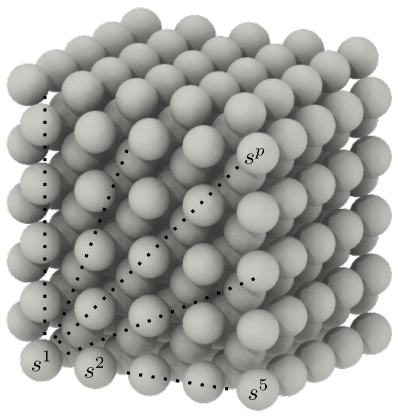


Figure 4.2: An illustration of permutationally invariant collective variables on each atom in a block of sodium crystal. The superscript on s^p indicates the index of the Na atom. For each sodium one $\bar{q}_l(i)$ can be computed, while one Q_l is computed for entire system.

Despite the usefulness of permutationally invariant collective variables, especially in distinguishing different crystalline polymorphs, they present a unique challenge to enhanced sampling methods. First, biasing all PICVs as separate CVs would mean a large dimension in the CV space, which radically diminishes the sampling efficiency. Secondly, a pathway could be artificially imposed if equivalent collective variables are treated as distinct ones. To properly leverage PICVs in enhanced sampling simulations, a group of PICVs should be treated as a family of the same collective variable, and each value is a member of the same family. Specifically in the case of VES, we have adapted its optimization process and the bias potential for PICVs and refer to the adapted method as variationally enhanced sampling with permutationally enhanced sampling method (VES-PICV). In the next few sections, we give detailed accounts of our method, and then present two examples of its application in simulation studies of seven particles in a 2-dimensional (2D) space and bulk sodium. The phase transitions and free energy surface of the seven Lennard-Jones particles in a 2D space were first studied by Dellago and coworkers due to its complex behaviors,[12] and later by Valsson and Parrinello in their paper that introduced VES[4]. We use VES-PICV to sample the landscape of this system. The melting temperature (T_m) of bulk sodium has been reported to be ~340 K and 360 K in two computer simulation studies using the same EAM potential.[13,14] In this work the free energy surface of bulk sodium at 380.0K is sampled with VES-PICV multiple-walker simulations. Finally we discuss the VES-PICV method in the context of other parallel bias enhanced sampling methods.

4.1 A SHORT REVIEW OF VES

We first go over an overview of the variationally enhanced sampling method to refresh our memory before we dove into VES-PICV. In VES a functional Ω is defined of the bias potential V as:

$$\Omega[V] = \frac{1}{\beta} \log \frac{\int ds\, e^{-\beta[F(\mathbf{s})+V(\mathbf{s})]}}{\int ds\, e^{-\beta F(\mathbf{s})}} + \int ds p_{tg}(\mathbf{s})V(\mathbf{s}), \tag{4.3}$$

where $\mathbf{s}$ is the CV space, and $p_{tg}(\mathbf{s})$ is the target distribution of $\mathbf{s}$, predefined by the user. The $F(\mathbf{s})$ is the underlying free energy surface. The functional Ω is a convex functional with a global minimum, at which point the free energy surface $F(\mathbf{s})$ and the bias potential $V(\mathbf{s})$ are related by $V(\mathbf{s}) = F(\mathbf{s}) - {}^1/_\beta \log p_{tg}(\mathbf{s}) + C$. Furthermore, $p_V(\mathbf{s})$, the biased distribution of $\mathbf{s}$ that is sampled by the VES simulation at the global minimum of Ω is the same as the target distribution $p_{tg}(\mathbf{s})$. To find the bias potential $V(\mathbf{s})$ that minimizes $\Omega[V]$, we first express it as $V(\mathbf{s};\boldsymbol{\alpha})$, a linear combination of a series of basis functions $f_k(\mathbf{s})$:

$$V(\mathbf{s};\boldsymbol{\alpha}) = \sum_k \alpha_k f_k(\mathbf{s}), \tag{4.4}$$

in which the linear coefficients $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ are the variational parameters that are updated iteratively by a variational method, e. g., the average stochastic gradient descent method. The gradient $\nabla\Omega(\boldsymbol{\alpha})$ and Hessian $\mathbb{H}\Omega(\boldsymbol{\alpha})$ of the functional $\Omega(\boldsymbol{\alpha})$ with respect to the variational parameters $\boldsymbol{\alpha}$ are computed to update $\boldsymbol{\alpha}$. An element of the gradient $\nabla\Omega(\boldsymbol{\alpha})$ is computed by

$$\frac{\partial\Omega(\boldsymbol{\alpha})}{\partial\alpha_i} = -\left\langle \frac{\partial V(\mathbf{s};\boldsymbol{\alpha})}{\partial\alpha_i} \right\rangle_{V(\boldsymbol{\alpha})} + \left\langle \frac{\partial V(\mathbf{s};\boldsymbol{\alpha})}{\partial\alpha_i} \right\rangle_{p_{tg}}. \tag{4.5}$$

An element of the Hessian $\mathbb{H}\Omega(\boldsymbol{\alpha})$ can be computed by

$$\begin{aligned} \frac{\partial^2\Omega(\boldsymbol{\alpha})}{\partial\alpha_i\partial\alpha_j} = \beta \operatorname{Cov}\left[\frac{\partial V(\mathbf{s};\boldsymbol{\alpha})}{\partial\alpha_j}, \frac{\partial V(\mathbf{s};\boldsymbol{\alpha})}{\partial\alpha_i}\right]_{V(\boldsymbol{\alpha})} \\ - \left\langle \frac{\partial^2 V(\mathbf{s};\boldsymbol{\alpha})}{\partial\alpha_j\partial\alpha_i} \right\rangle_{V(\boldsymbol{\alpha})} + \left\langle \frac{\partial^2 V(\mathbf{s};\boldsymbol{\alpha})}{\partial\alpha_j\partial\alpha_i} \right\rangle_{p_{tg}}. \end{aligned} \tag{4.6}$$

In the case of average stochastic gradient descent optimizer, the coefficients are updated by

$$\boldsymbol{\alpha}^{(n+1)} = \boldsymbol{\alpha}^{(n)} - \mu\left[\nabla\Omega(\bar{\boldsymbol{\alpha}}^{(n)}) + \mathbb{H}\Omega(\bar{\boldsymbol{\alpha}}^{(n)})[\boldsymbol{\alpha}^{(n)} - \bar{\boldsymbol{\alpha}}^{(n)}]\right], \tag{4.7}$$

in which μ is the step size of the update, a parameter set by the user, in strides of time steps.

4.2 THE VES-PICV METHOD

Now we introduce the adaptation of VES for permutationally invariant collective variables. First, we present a change to the notations. So far we have used the bold $\mathbf{s} = [s_1, s_2, \dots, s_n]$ to denote an n-dimensional CV space that consist of multiple distinct CVs. The bias potential $V(\mathbf{s}; \boldsymbol{\alpha})$ is defined over the multidimensional CV space and expressed as $V(\mathbf{s}; \boldsymbol{\alpha}) = \sum_k \alpha_k f_k(\mathbf{s})$, where the subscript k indicates the number of basis functions. To better differentiate them from the multidimensional CV space $\mathbf{s}$, a family of PICVs is written as the small s. A member of a PICV family is written as s^p with the superscript p corresponding to the index of the relevant atom or the molecule. The bias potential $V(s; \boldsymbol{\alpha})$ is now written as a sum of the unit bias potential $v(s; \boldsymbol{\alpha})$:

$$V(s; \boldsymbol{\alpha}) = \sum_p^N v(s^p; \boldsymbol{\alpha}), \tag{4.8}$$

$$v(s; \boldsymbol{\alpha}) = \sum_k \alpha_k f_k(s). \tag{4.9}$$

$V(s; \boldsymbol{\alpha})$ is the applied bias in the VES-PICV simulations, and the unit bias potential $v(s; \boldsymbol{\alpha})$ is the amount of bias on each member of a family of PICVs. In other words, every member of a family of PICVs is concurrently biased in a VES-PICV simulations. And the variational parameters $\boldsymbol{\alpha}$ are the same for all members of a family of PICVs.

The optimization process of updating the variational parameters $\boldsymbol{\alpha}$ inevitably also has to change due to the PICVs. First of all, we no longer compute the gradient $\nabla\Omega(\boldsymbol{\alpha})$ and Hessian $\mathbb{H}\Omega(\boldsymbol{\alpha})$ as presented for Equation 4.7. In their stead, we compute $g(\boldsymbol{\alpha})$ and $h(\boldsymbol{\alpha})$ with the unit bias potential $v(s; \boldsymbol{\alpha})$:

$$g_i(\boldsymbol{\alpha}) = -\left\langle \frac{\partial v(s; \boldsymbol{\alpha})}{\partial \alpha_i} \right\rangle_{V(\boldsymbol{\alpha})} + \left\langle \frac{\partial v(s; \boldsymbol{\alpha})}{\partial \alpha_i} \right\rangle_{p_{tg}}, \tag{4.10}$$

$$\begin{aligned} h_{ij}(\boldsymbol{\alpha}) = {} & \beta \operatorname{Cov}\left[\frac{\partial v(s; \boldsymbol{\alpha})}{\partial \alpha_i}, \frac{\partial v(s; \boldsymbol{\alpha})}{\partial \alpha_j} \right]_{V(\boldsymbol{\alpha})} \\ & - \left\langle \frac{\partial^2 v(s; \boldsymbol{\alpha})}{\partial \alpha_j \partial \alpha_i} \right\rangle_{V(\boldsymbol{\alpha})} + \left\langle \frac{\partial^2 v(s; \boldsymbol{\alpha})}{\partial \alpha_j \partial \alpha_i} \right\rangle_{p_{tg}}. \end{aligned} \tag{4.11}$$

One can see that they have similar forms to the gradient $\nabla\Omega[\boldsymbol{\alpha}]$ and Hessian $\mathbb{H}\Omega(\boldsymbol{\alpha})$. Plugging Equation 4.9 into the two equations above, $g(\boldsymbol{\alpha})$ and $h(\boldsymbol{\alpha})$ can be simplified into

$$g_i(\boldsymbol{\alpha}) = -\langle f_i(s) \rangle_{V(\boldsymbol{\alpha})} + \langle f_i(s) \rangle_{p_{tg}}, \tag{4.12}$$

and

$$h_{ij}(\boldsymbol{\alpha}) = \beta \operatorname{Cov}[f_j(s), f_i(s)]_{V(\boldsymbol{\alpha})}. \tag{4.13}$$

When computing Equation 4.12 and Equation 4.13, all members of a family of PICVs accumulated during a stride between updates are used. Now the average stochastic gradient descent optimizer changes to

$$\boldsymbol{\alpha}^{(n+1)} = \boldsymbol{\alpha}^{(n)} - \mu\left[g(\bar{\boldsymbol{\alpha}}^{(n)}) + h(\boldsymbol{\alpha})(\bar{\boldsymbol{\alpha}}^{(n)})[\boldsymbol{\alpha}^{(n)} - \bar{\boldsymbol{\alpha}}^{(n)}]\right]. \tag{4.14}$$

One configuration from a VES-PICV simulation provide all members of a family of PICV for the optimizer, wheres in a regular VES simulation one configuration supplies a single data point for a collective variable. From Equations 4.10-4.14, it is clear that the total bias is only used for the bias, while the unit bias potential is used in the optimization process. It results in a breakdown of the relationship between the underlying free energy surface $F(\mathbf{s})$ and the bias $V(\mathbf{s})$ at the global minimum of Ω, which has been useful to test the convergence of VES simulations. However, it is still feasible to gain insight into the convergence of the simulations by examining the convergence of the unit bias potential $v(s;\boldsymbol{\alpha})$. The free energy landscape of the system, therefore, has to be estimated through reweighting in a VES-PICV simulation.

The VES-PICV method is currently only available in an in-house customized implementation of PLUMED 2. The future release and upstreaming of our implementation are intended.

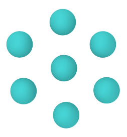
4.3 SIMULATIONS AND METHODS

4.3.1 Seven Lennard-Jones Particles in Two-Dimensional Space

The seven Lennard-Jones particles are modeled by the 12-6 style Lennard-Jones potential in a two-dimensional space, as seen in Equation 4.15 where ε and σ are equal to 1 and in natural units:

$$U_{LJ}(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right], \tag{4.15}$$

in which ε and σ are equal to 1, and r is the distance between two particles. We employed a cutoff distance in calculation of the Lennard-Jones potential, which is 2.5σ. The mass m of a particle is 1. The time step is $0.002\sqrt{\frac{\varepsilon}{m\sigma^2}}$. The temperature is $0.2k_BT/\varepsilon$, which is enforced by the Langevin thermostat with a friction coefficient equal to 1. All simulations of this system were performed with the simple MD engine as part of the PLUMED package. The simulations have been run for a total of 10^9 steps for both the well-tempered metadynamics simulation and the VES-PICV simulation. An illustration of seven Lennard-Jones particles in two-dimensional space can be found in the margin.



COORDINATION NUMBERS. The pairwise coordination number s_{ij} between particles i and j is calculated using the following equation[7]:

$$s_{ij} = \frac{1-\left(\frac{r_{ij}}{d_0}\right)^8}{1-\left(\frac{r_{ij}}{d_0}\right)^{16}}, \tag{4.16}$$

where parameters $d_0 = 1.0$, and r_{ij} is the pairwise distance between particles i and j. The coordination number of particle i, s_i, is the sum of all the pairwise coordination numbers s_{ij} involving particle i, i. e., $s_i = \sum s_{ij}$. The function form of s_{ij} ensures the differentiability of the coordination number s_i as a collective variable.

The second moment (μ_2^2) and third moment (μ_3^3) of the coordination numbers for the seven particles is computed using the following equation:

$$\mu_m^m = \frac{1}{7}\sum_{i=1}^{7}(s_i - \bar{s})^m. \tag{4.17}$$

They are the variance and skewness of the coordination numbers s_i, respectively, in statistical terms.

WELL-TEMPERED METADYNAMICS In the well-tempered metadynamics simulation of the seven particles, the biased CVs are the second moment of the coordination numbers (μ_2^2) and the third moment (μ_3^3). The bias factor γ was 20. The height of the bias potential was 0.02ε, one tenth of the thermal energy. The stride of depositing Gaussian kernels was 500 time steps, and the width σ for μ_2^2 and μ_3^3 in the Gaussian kernel was 0.02 and 0.02, respectively. To speed up the computation of the bias potential, Gaussian kernels were deposited on a two-dimensional grid with 5600 bins in the dimension of μ_2^2 and 2800 bins in the dimension of μ_3^3. The range for the grid in μ_2^2 is $[0.1, 1.5]$, and $[-0.9, 1.9]$ for μ_3^3.

The estimation of free energy surface was performed through the two methods introduced in Section 2.3, namely directly through the bias potential and residual bias reweighting. In the residual reweighting, the kernel density estimation has been used with the bandwidths for μ_2^2 and μ_3^3 are 0.012 and 0.015, respectively.

VES-PICV In the VES-PICV simulations, the PICVs were the coordination numbers s_i. Legendre functions up to 20th order were selected as the basis functions. The stride between updates of the bias potential was 500 time steps. The average stochastic gradient descent optimizer was employed in the update of the bias potentials. The step size μ for update was 0.1. The well-tempered distribution is used as the target distribution, and the bias factor γ for the well-tempered distribution is 3. The well-tempered distribution was update every 500 iterations, i. e., 250000 time steps. As stated previously, the free energy surface from VES-PICV simulations can only be estimated through reweighting. The bandwidths for μ_2^2 and μ_3^3 in kernel density estimation were the same as those for the well-tempered metadynamics. The plumed.dat file for the VES-PICV simulation is in Listing D.1.

4.3.2 Bulk Sodium

SIMULATION DETAILS Bulk sodium was simulated using the embed atom potential developed by Wilson and coworkers.[13] We have introduced the embed atom model in Section 1.3.3, including the function form and an abridged table of the EAM potential for Na. The MD engine for these simulations was LAMMPS.[15] The velocity-Verlet integrator was used to propagate the dynamics with a time step of $2\ \mathrm{fs}$. The simulations were carried out in a canonical ensemble at $380.0\ \mathrm{K}$ and $1.0\ \mathrm{bar}$. The thermostat of choice was the velocity-rescaling thermostat[16] with the relaxation parameter τ equal to 100 fs. The three dimensions were coupled in the Nosé-Hoover barostat[17] and its damping constant was $2\ \mathrm{ps}$. There were a total number of 250 sodium atoms in a periodic cubic box, with a dimension around $2.14\ \mathrm{nm}$. The total simulation time was $22\ \mathrm{ns}$ for each walker, and $176\ \mathrm{ns}$ in total for the eight walkers. The crystalline sodium had a body-centered cubic unit cell. The cell constant determined for the Wilson's EAM potential for sodium was 4.29 Å.[13]

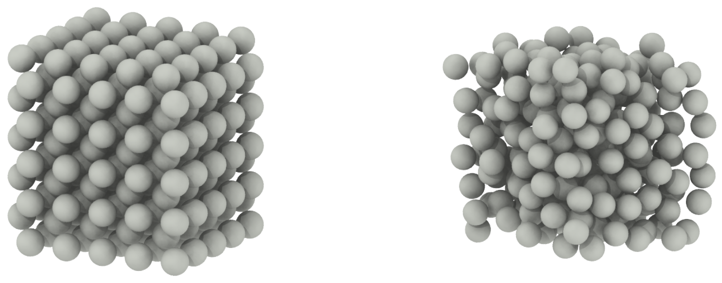

(a) A perfect BCC crystal (b) A liquid configuration

Figure 4.3: The liquid phase and crystalline phase of bulk sodium.

KERNEL CV The kernel CVs were the biased PICVs in the simulations of sodium. They were developed for studies such as crystallization and solid-solid phase transitions by comparing local similarities between a local environment of the system of interest and a reference template[8–10] The similarity of two local environments is defined as the inner product of two local densities ρ_χ and ρ_{χ_0},

$$k_{\chi_0}(\rho_\chi, \rho_{\chi_0}) = \int \rho_\chi(\mathbf{r})\rho_{\chi_0}(\mathbf{r})d\mathbf{r}, \tag{4.18}$$

in which the subscript χ and χ_0 refer to the environment of the system of interest and the reference template, respectively. And $\rho_\chi(\mathbf{r})$ is defined as

$$\rho_\chi(\mathbf{r}) = \sum_{i\in\chi} \exp\left(-\frac{|\mathbf{r}_i - \mathbf{r}|^2}{2\sigma^2}\right), \tag{4.19}$$

where $\mathbf{r}_i$ is the relative vector of atom i to the center atom, and σ is the width of the Gaussian kernel. The density of the reference template $\rho_{\chi_0}(\mathbf{r})$ is computed similarly. After plugging Equation 4.19 into Equation 4.18, and then taking its weighed average, we obtain the kernel CV for the center atom as

$$\tilde{k}_{\chi_0}(\chi) = \frac{\sum_{i\in\chi}\sum_{j\in\chi_0} s(\mathbf{r}_i)\exp\left(-\frac{|\mathbf{r}_i-\mathbf{r}_j^0|^2}{4\sigma^2}\right)}{\sum_{i\in\chi} s(\mathbf{r}_i)}, \tag{4.20}$$

where the weight $s(\mathbf{r}_i)$ is a switching function computed as

$$s(\mathbf{r}_i) = \frac{1-\left(\frac{r_i}{r_0}\right)^{12}}{1-\left(\frac{r_i}{r_0}\right)^{24}}, \tag{4.21}$$

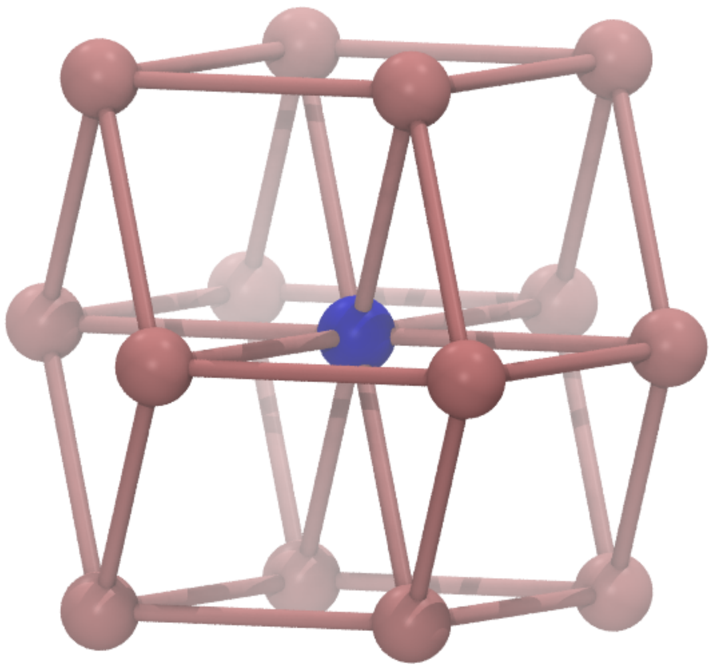
where r_i is the norm of vector $\mathbf{r}_i$, and r_0 is equal to 0.5 nm. We have chosen $\sigma = 0.65$ nm, the same as in reference [10], in which $\tilde{k}_{\chi_0}(\chi)$ is an average over the number of neighbor atoms rather than a weighted average. Fourteen nearest neighbors were used in calculation of $\tilde{k}_{\chi_0}(\chi)$ for our simulations of bulk sodium, compared to the reference template as shown in Figure 4.4a. The histograms of $\tilde{k}_{\chi_0}(\chi)$ for sodium atoms in both liquid state and the solid state at 380.0 K have been computed and given in Figure 4.4b. The figure shows a modest overlap of the two histograms of the two states. It suggests that one can use the kernel CV as the PICV for our simulations of crystallization of sodium.

Sodium atoms with $\tilde{k}_{\chi_0}(\chi) \geq 0.45$ are categorized as in the solid state. We count the number of solid atoms n_s by

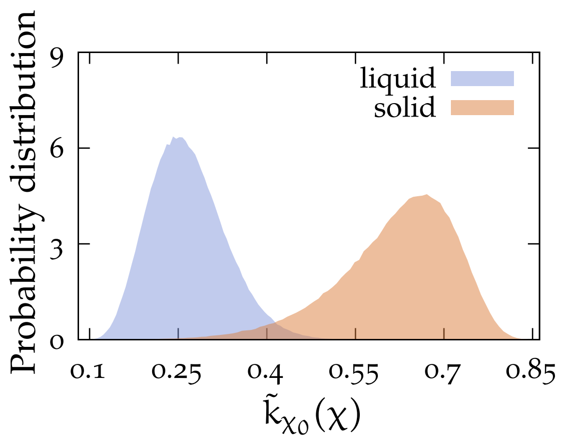
$$n_s = \sum_{i}^{N}\left[1 - \frac{1-\left(\frac{\tilde{k}_{\chi_0}(\chi_i)}{0.45}\right)^{12}}{1-\left(\frac{\tilde{k}_{\chi_0}(\chi_i)}{0.45}\right)^{24}}\right], \tag{4.22}$$

and N is the total number of atoms in the system. The n_s is a global CV to characterize the overall degree of crystallinity of the system.

Q_6 Another global CV used to characterize the degree of crystallinity of the system is the Steinhardt order parameter Q_6. One can find the formulas to compute it in Section 3.2.2. The parameters in use were $N = 12$, $M = 24$, $r_0 = 0.5$ nm, and $d_0 = 0$.



(a)



(b)

Figure 4.4: (a) The reference system for bulk sodium with the fourteen closest neighbors in pink and the center atom in blue. The bond in the figure illustrates distance between the atoms, and are not real bonds. (b) The sampled probability distribution of the kernel CV $\tilde{k}_{\chi_0}(\chi)$ of the solid form and the liquid form of sodium at 380 K and 1 bar sampled over two brute-force simulations for 20 ns. The slight difference between (Figure 4.4b) and Figure 1b in reference [10] is mainly attributed to the two different ways of taking the average when calculating $\tilde{k}_{\chi_0}(\chi)$, and in minor part to the 5 K difference in temperature.

VES-PICV In the multiple-walker VES-PICV simulations of bulk sodium, the ADAM optimizer has been used instead of the average stochastic gradient descent optimizer, which is commonly used with VES. Readers can find out more about the details of the implementation of the ADAM optimizer in the VES module of PLUMED 2 in Pampel & Valsson [18]. The choice of the ADAM optimizer is due to its superior performance of converging the bias potential to a stable value in this case. Later we show that the bias potential indeed rapidly converges in the simulations. We have found that the quality and speed of the convergence of the bias potential both affect the quality of the free energy estimation of reweighting. Legendre functions up to the 20th order from 0.1 to 0.85 are chosen as the basis functions. The target distribution is the well-tempered distribution with the bias factor $\gamma = 1.2$. The stride between updates of the bias potential is 500 time steps, and the step size for the update is 0.0005. The frequency of updating the well-tempered distribution is 500 iterations or 250000 time steps. The plumed.dat for the multiple-walker VES-PICV simulations of bulk sodium can found in Listing D.2.

The free energy surface is estimated by reweighting with kernel density estimation. The free energy surface from the multiple-walker simulations are reweighted excluding the initial 4 ns of the simulations as the large fluctuations in the bias potential at the beginning of the simulations degrade the quality of free energy estimation by reweighting.

4.4 RESULTS

4.4.1 Seven Lennard-Jones Particles

The stable state and three metastable states of the seven particles are shown in Figure 4.5, and their locations in the free energy landscape with respect to the second moment (μ_2^2) and the third moment (μ_3^3) of the coordination numbers (s) are given in Figure 4.6.

Just to remind the readers, both a well-tempered metadynamics simulation and a VES-PICV simulation have been performed on the seven particles in a 2D space. In the WTMetaD simulation the biased CV space $\mathbf{s}$ is $[\mu_2^2, \mu_3^3]$. In the VES-PICV simulation the biased CVs are the coordination numbers (s), however, we have estimated the FES with respect to μ_2^2 and μ_3^3 by reweighting. In Figure 4.6a the estimation of FES is performed with $V(s,t) = \frac{1-\gamma}{\gamma}F(s) + c(t)$, while in Figure 4.6b it is obtained with the residual bias reweighting method[19] developed by Tiwary and Parrinello. It is clear that these two free energy surfaces do not agree with each other quantitatively, although qualitatively the locations of the stable state and metastable states are in agreement. As the convergence of the well-tempered metadynamics is hard to reach

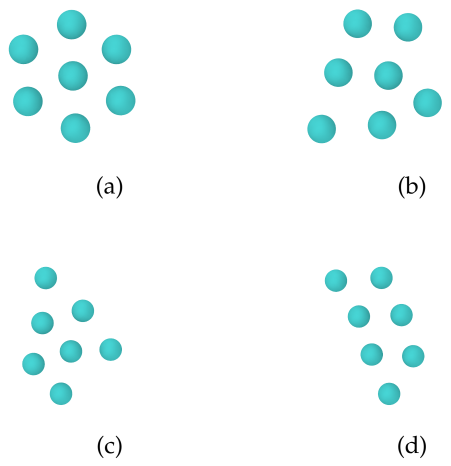
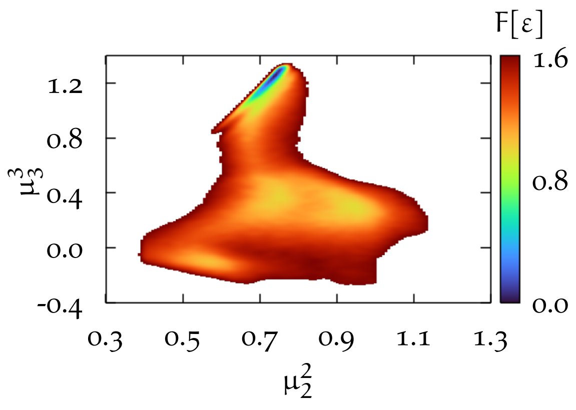


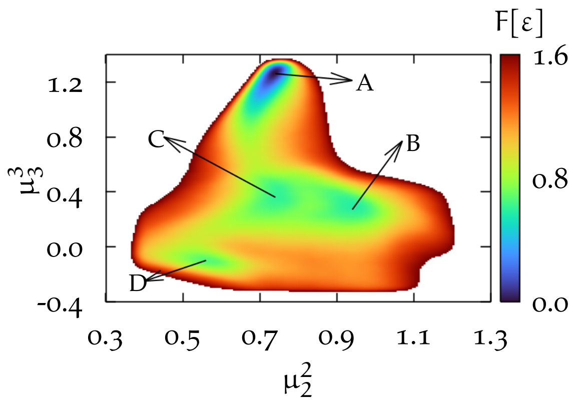
Figure 4.5: The most stable configuration (a) and the three metastable configurations (b, c, d) of seven Lennard-Jones particles in two dimensions.

even with the current computing power and advanced algorithms, reweighting is generally preferred to estimate the FES. For the VES-PICV simulations, the free energy surface with respect to μ_2^2 and μ_3^3 has been obtained through reweighting, although they were not biased in the simulations. As shown in Figure 4.6c and Figure 4.6b, the reweighed free energy surface from the VES-PICV simulations displays an almost quantitative agreement with the reweighted FES from WTMetaD simulations. The four metastable states present in the FES from WTMetaD simulations are all present in the reweighted FES from VES-PICV simulations. To best illustrate the agreement of the reweighed FES between WTMetaD and VES-PICV simulations, the free energy profiles projected to μ_2^2 and μ_3^3 are shown in Figure 4.7.

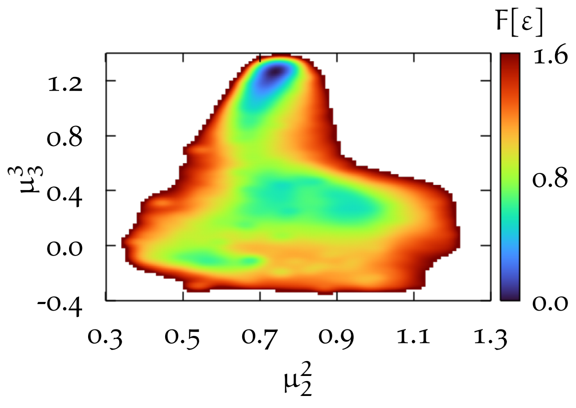
The assumption of VES-PICV is that in the long time regime the free energy surface reweighted to each of the PICVs is equivalent. However, it is unclear whether it is necessary to achieve for a well working VES-PICV simulation. In the current case, the sampled distributions of the coordination numbers have tightly converged, as demonstrated by Figure 4.8a, which are histograms of the coordination numbers of each particle of entire duration of the VES-PICV simulation. The unit bias potential $v(s; \boldsymbol{\alpha})$, as shown in Figure 4.8b, has very rapidly converged as well, which is important in ensuring the quality of reweighted FES. Certainly the converged sampled distributions of the PICVs could only help the convergence of the unit bias potential $v(s; \boldsymbol{\alpha})$.



(a) WTMetaD



(b) WTMetaD



(c) VES-PICV

Figure 4.6: The free energy surface $F(\mu_2^2, \mu_3^3)$ of seven Lennard-Jones particles in two dimensions, which is obtained through the bias potential (a) of well-tempered metadynamics, through reweighting of well-tempered metadynamics trajectories (b), through VES-PICV (c). In (b) the labeled regions correspond to the figures in Figure 4.5

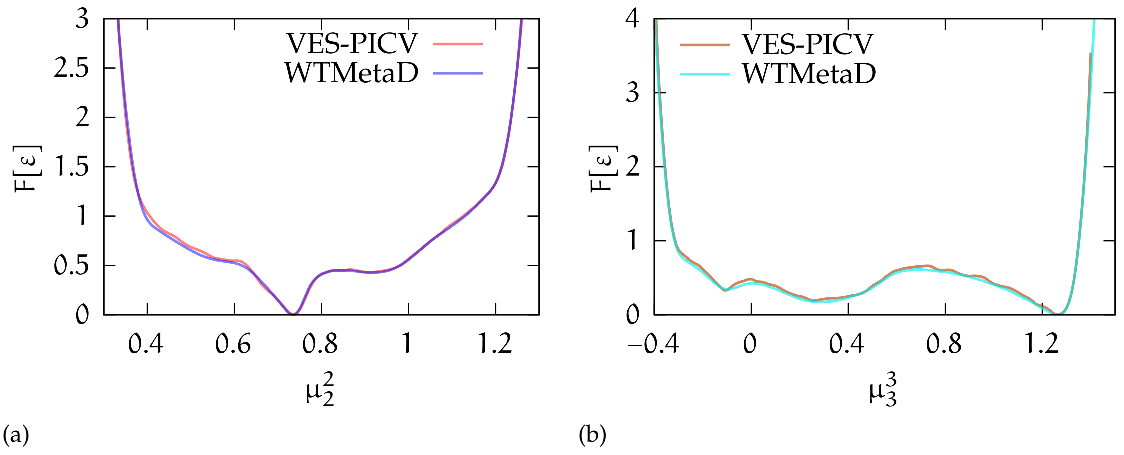


Figure 4.7: Reweighted free energy profiles with respect to μ_2^2 and μ_3^3 by well-tempered metadynamics and VES-PICV simulations.

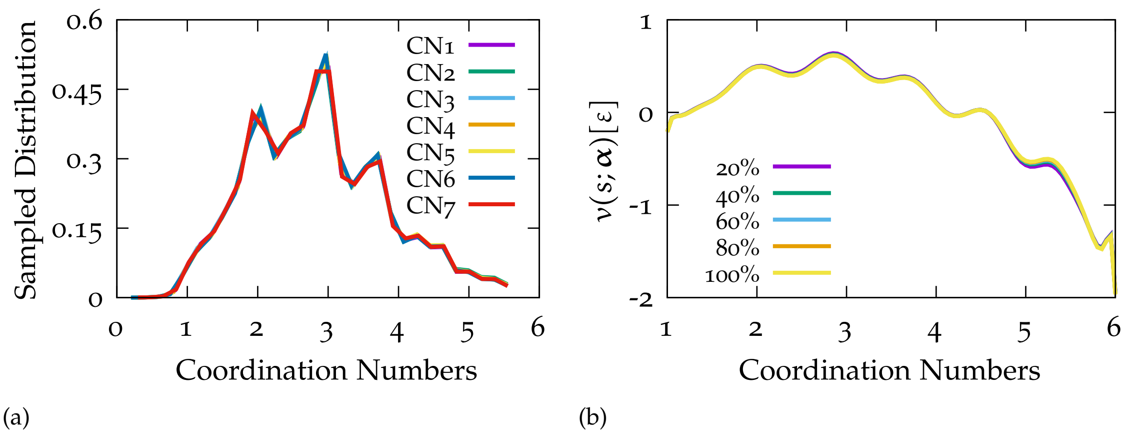
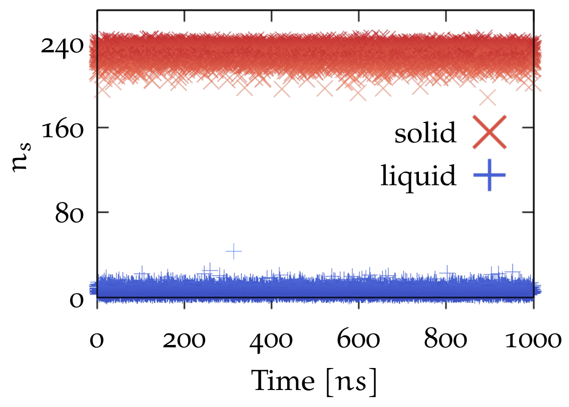


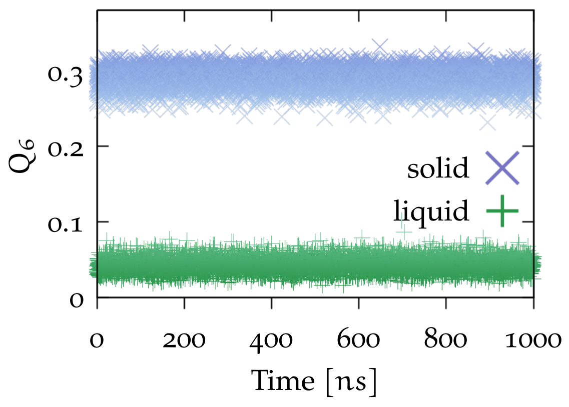
Figure 4.8: (a) The sampled distribution of coordination numbers of each particle. (b) The unit bias potential $v(s;\boldsymbol{\alpha})$ in the progression of VES-PICV simulations.

4.4.2 Bulk Sodium

Now we look at the second example in the phase transition of the bulk sodium. The phase transition between the liquid phase and the solid phase, which has a body-centered cubic packing is a first-order phase transition. A high free energy barrier that separates the two phases makes accurately determining the T_m for bulk sodium by simulations a nontrivial task. We have found by way of brute force simulations, even at 380 K, 14 – 40 K above the previously reported melting temperatures T_m, phase transitions do not spontaneously occur in 1 μs long simulation, as shown in Figure 4.9. It shows the time series of the Steinhardt order parameter Q_6 and the number of atoms in the solid phase n_s of the two brute force simulations starting from either a solid phase or a liquid phase. It is evident that the systems do not cross over to the other state.



(a)



(b)

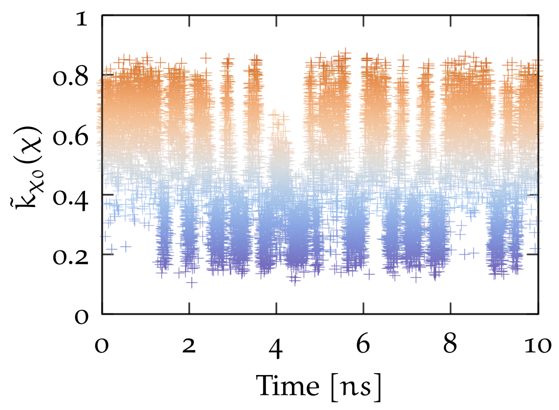
Figure 4.9: The times series of CV n_s (a) and Q_6 (b) of bulk sodium starting from the solid phase and liquid phase, respectively. The initial phases of sodium are labeled with the legends in the figures. Both CVs indicate no phase transition in the brute-force simulations.

Contrary to the brute force simulations, the multiple-walker VES-PICV simulations at $T = 380$ K have driven rapid transitions in all eight walkers between the liquid phase and solid phase. To illustrate these transitions, we show in Figure 4.10 the time series of the biased kernel CV $\tilde{k}_{\chi_0}(\chi)$ of a typical sodium atom of the first walker during the initial 10 ns simulation, and the time series of the number of solid atoms n_s and the Steinhardt order parameter Q_6 of the first walker during the entire simulation. The other walkers display similar diffusive dynamics between the two states. The time series of n_s and Q_6 of the other seven walkers can be found in Figure D.1.

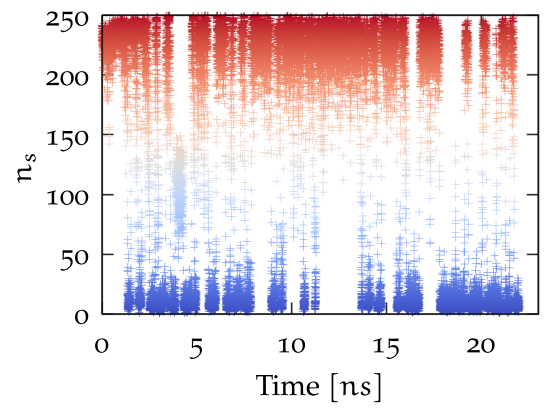
In the VES-PICV simulations of the seven particles in a 2D space, the sampled distributions of the coordination numbers of the seven particles have tightly converged. In the case of the multiple-walker simulations of the bulk sodium, it is impractical to present the sampled distribution of all atoms. Instead, we have arbitrarily chosen one sodium atom from each walker and given their sampled distributions of the biased kernel CV $\tilde{k}_{\chi_0}(\chi)$ in Figure 4.11. Even though all eight atoms have sampled the both liquid state and solid state, their distributions have not reached full convergence.

The unit bias potential $v(\tilde{k}_{\chi_0};\boldsymbol{\alpha})$, as the result of the optimization process of the VES-PICV simulations, has however converged to a stable value very early on in the simulations, as shown in Figure 4.12. The convergence is rather swift considering the short time span of the simulations. It is made possible by the rapid transitions between the two phases of bulk sodium in the multiple-walker simulations. The yet to be fully converged sampled distributions of the kernel CVs $\tilde{k}_{\chi_0}(\chi)$ seem to have not been detrimental to the convergence of the unit bias potential, suggesting that sufficient sampling can be achieved without reaching full convergence of the biased PICVs. It is also supported by the reweighted free energy surfaces.

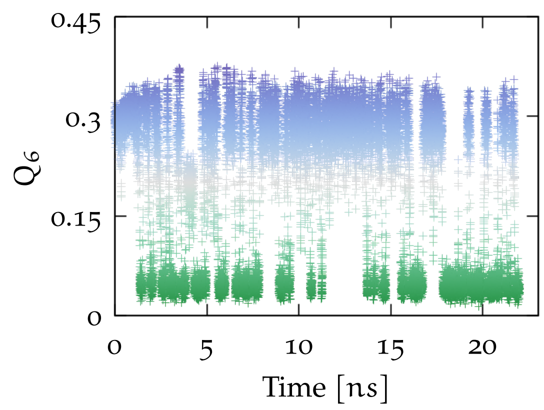
The free energy surface estimated using reweighting is shown in Figure 4.13. The FES of the eight walkers show excellent agreement between them, and the standard errors among them follows tightly the mean free energy surface, even though the sampled distribution of the kernel CVs have not reached full convergence. The free energy barrier between the solid state at $Q_6 = 0.29$ and the liquid state at $Q_6 = 0.04$ reaches about 60 kJ/mol. The figure also shows that F_{liquid} is lower than F_{solid}, indicating a relative more stable liquid state compared to the solid state, and a melting temperature T_m below 380K.



(a)



(b)



(c)

Figure 4.10: The time series of (a) the kernel CV $\tilde{k}_{\chi_0}(\chi)$ of a typical Na atom of the first walker during first 10 ns of the simulation, (b) the number of solid atoms n_s of the first walker , and (c) Q_6 of the first walker of the multiple-walker simulations of sodium. The kernel CV $\tilde{k}_{\chi_0}(\chi)$ was directly biased in the simulations, it covers the entire range of values of the solid state and liquid state. We show the first 10 ns of $\tilde{k}_{\chi_0}(\chi)$ to have a better visualization.

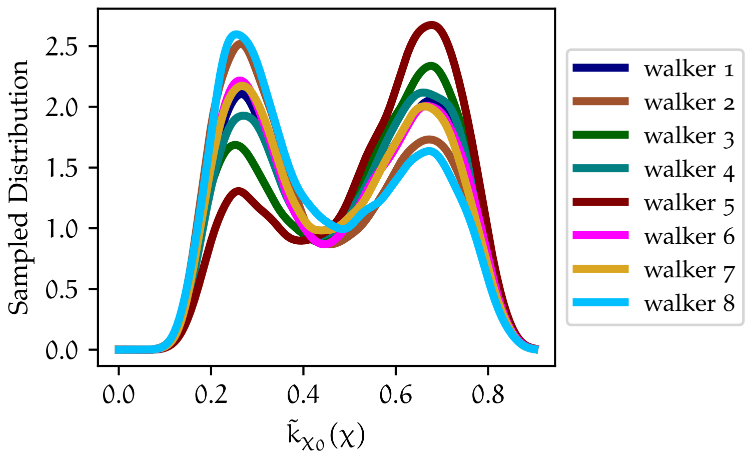


Figure 4.11: The sampled distributions of kernel CV $\tilde{k}_{\chi_0}(\chi)$ on an arbitrarily chosen sodium from each of the eight walkers.

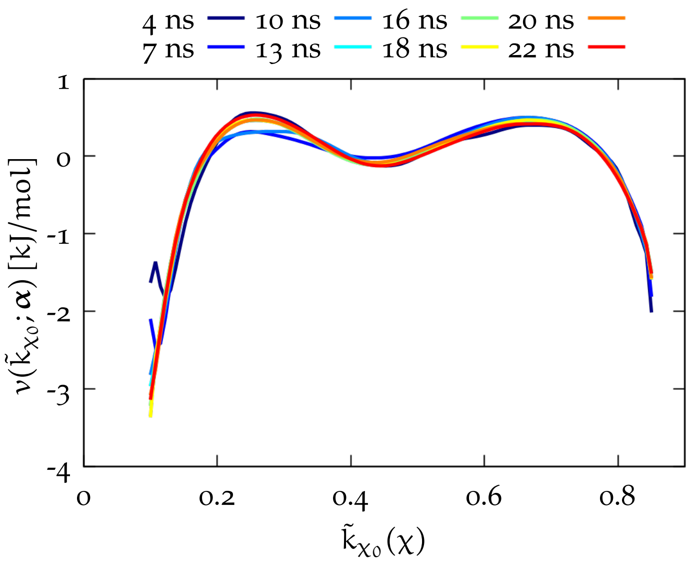
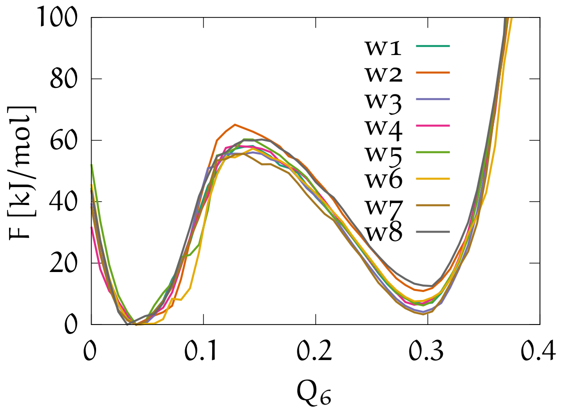
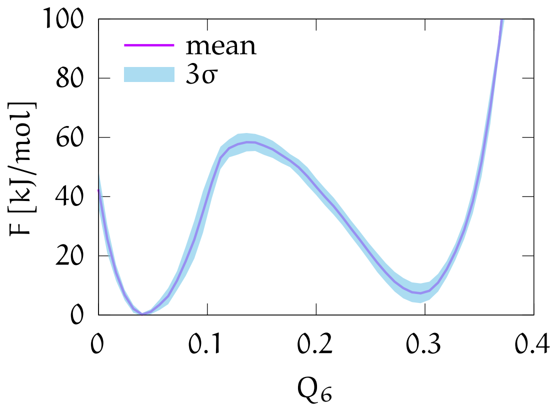


Figure 4.12: The unit bias potential $v(\tilde{k}_{\chi_0};\boldsymbol{\alpha})$ at different time of VES-PICV simulations.



(a)



(b)

Figure 4.13: (a) The free energy $F(Q_6)$ for all eight walkers in VES-PICV simulations through reweighting. The minimum on the left corresponds to the liquid state, and the right well corresponds to the solid state. (b) The mean free energy surface of the eight walkers and the blue shaded areas are within three times the standard errors of the mean.

4.5 DISCUSSIONS

We have shown that by taking advantage of the local permutation invariance collective variables, new opportunities are opened for applications of enhanced sampling in computer simulations of phase transitions of atomic crystals, especially in cases where global collective variables are not feasible. VES-PICV is based on the simple assumption that in the long time limit the free energy profiles projected to a family of PICVs are the same.

Treating PICVs as a family of the same CV has profound impacts on the scheme of VES. In an optimization step, each step of simulations contributes all members of PICVs to the calculations of $g(\boldsymbol{\alpha})$ and $h(\boldsymbol{\alpha})$. Therefore, a large of amount of statistics is accumulated between optimization steps to optimize the unit bias potential $v(s;\boldsymbol{\alpha})$. Second, in the VES-PICV scheme, the relationship $V(\mathbf{s}) = F(\mathbf{s}) - 1/\beta \log p_{tg}(\mathbf{s}) + C$ no longer holds true. As a result, the free energy surface can only be estimated by reweighting. Despite this change, we have shown in the two examples that the unit bias potential converged very early during the VES-PICV simulations, which gives us confidence in the reweighted free energy surfaces.

Although we have seen in the two applications of the VES-PICV method that the unit bias potentials have quickly converged to a stable form, the sampled distributions of the PICVs have only converged for the coordination numbers in the case of the seven Lennard-Jones particles. The sampled distributions of the kernel CVs have not fully converged in the VES-PICV simulations of the bulk sodium, even though the sampled distributions show that both the liquid state and the solid state have been visited by the Na atoms. The reweighted free energy surface from VES-PICV simulations of the seven Lennard-Jones has an excellent agreement with the well-tempered metadynamics simulation. The reweighted free energy profiles of the eight walkers of the bulk sodium fall within a relatively tight range, evidenced by their low standard errors. The yet-to-converge sampled distributions of the kernel CVs seem to have not degraded the quality of the reweighted free energy profiles.

VES-PICV is a versatile method that can be further extended. Even though in the two cases reported here only one family of PICVs has been used in each case, one can bias more than one family of PICVs in the simulations if it is needed. It is also possible to simultaneously bias PICVs and global CVs in one simulation. One such scenario could be a simulation of a system comprised of multiple copies of the same molecule, where a phase change that is manifested in a global reordering of the molecules that is accompanied by local internal conformation changes in such molecules.

Finally, we would like to note that the method dubbed parallel bias metadynamics with partitioned families can also use PICVs as biased collective variables,[20] however, it does so by biasing a random member of the PICV every step instead of biasing all members of PICVs.

REFERENCES

1. Bonomi, M. & Parrinello, M. Enhanced Sampling in the Well-Tempered Ensemble. *Phys. Rev. Lett.* **104,** 190601 (May 10, 2010).
2. Valsson, O. & Parrinello, M. Thermodynamical Description of a Quasi-First-Order Phase Transition from the Well-Tempered Ensemble. *Journal of Chemical Theory and Computation* **9,** 5267–5276 (Dec. 10, 2013).
3. Barducci, A., Bussi, G. & Parrinello, M. Well-Tempered Metadynamics: A Smoothly Converging and Tunable Free-Energy Method. *Physical Review Letters* **100,** 020603 (Jan. 18, 2008).
4. Valsson, O. & Parrinello, M. Variational Approach to Enhanced Sampling and Free Energy Calculations. *Physical Review Letters* **113,** 090601 (Aug. 27, 2014).
5. Lechner, W. & Dellago, C. Accurate Determination of Crystal Structures Based on Averaged Local Bond Order Parameters. *The Journal of Chemical Physics* **129,** 114707 (Sept. 17, 2008).
6. Nguyen, A. H. & Molinero, V. Identification of Clathrate Hydrates, Hexagonal Ice, Cubic Ice, and Liquid Water in Simulations: The CHILL+ Algorithm. *J. Phys. Chem. B* **119,** 9369–9376 (July 23, 2015).
7. White, A. D. & Voth, G. A. Efficient and Minimal Method to Bias Molecular Simulations with Experimental Data. *J. Chem. Theory Comput.* **10,** 3023–3030 (Aug. 12, 2014).
8. Bartók, A. P., Kondor, R. & Csányi, G. On Representing Chemical Environments. *Physical Review B* **87** (May 28, 2013).
9. Martelli, F., Ko, H.-Y., Oğuz, E. C. & Car, R. Local-Order Metric for Condensed-Phase Environments. *Phys. Rev. B* **97,** 064105 (Feb. 12, 2018).
10. Piaggi, P. M. & Parrinello, M. Calculation of Phase Diagrams in the Multithermal-Multibaric Ensemble. *J. Chem. Phys.* **150,** 244119 (June 28, 2019).
11. Giberti, F., Salvalaglio, M., Mazzotti, M. & Parrinello, M. Insight into the Nucleation of Urea Crystals from the Melt. *Chemical Engineering Science* **121,** 51–59 (Jan. 2015).
12. Dellago, C., Bolhuis, P. G. & Chandler, D. Efficient Transition Path Sampling: Application to Lennard-Jones Cluster Rearrangements. *J. Chem. Phys.* **108,** 9236–9245 (June 1, 1998).

13. Wilson, S. R., Gunawardana, K. G. S. H. & Mendelev, M. I. Solid-Liquid Interface Free Energies of Pure Bcc Metals and B2 Phases. *The Journal of Chemical Physics* **142,** 134705 (Apr. 7, 2015).
14. Piaggi, P. M., Valsson, O. & Parrinello, M. Enhancing Entropy and Enthalpy Fluctuations to Drive Crystallization in Atomistic Simulations. *Phys. Rev. Lett.* **119,** 015701 (July 6, 2017).
15. Plimpton, S. Fast Parallel Algorithms for Short-Range Molecular Dynamics. *Journal of Computational Physics* **117,** 1–19 (Mar. 1, 1995).
16. Bussi, G., Donadio, D. & Parrinello, M. Canonical Sampling through Velocity Rescaling. *The Journal of Chemical Physics* **126,** 014101 (Jan. 3, 2007).
17. Tuckerman, M. E., Alejandre, J., López-Rendón, R., Jochim, A. L. & Martyna, G. J. A Liouville-operator Derived Measure-Preserving Integrator for Molecular Dynamics Simulations in the Isothermal–Isobaric Ensemble. *Journal of Physics A: Mathematical and General* **39,** 5629–5651 (Apr. 24, 2006).
18. Pampel, B. & Valsson, O. *Improving the Efficiency of Variationally Enhanced Sampling with Wavelet-Based Bias Potentials* Feb. 27, 2022. arXiv: 2202.13459 [cond-mat, physics:physics].
19. Tiwary, P. & Parrinello, M. A Time-Independent Free Energy Estimator for Metadynamics. *The Journal of Physical Chemistry B* **119,** 736–742 (Jan. 22, 2015).
20. Prakash, A., Fu, C. D., Bonomi, M. & Pfaendtner, J. Biasing Smarter, Not Harder, by Partitioning Collective Variables into Families in Parallel Bias Metadynamics. *J. Chem. Theory Comput.* **14,** 4985–4990 (Oct. 9, 2018).

5 CRYSTALLIZATION OF ICE AND UREA

In this chapter we discuss application of VES-PICV in atomistic simulations of crystallization of ice and urea. Although our efforts are ultimately unsuccessful, we summarize our work and findings here.

5.1 ICE

Understanding the growth and melting of ice has a direct impact on our understanding of the weather and climate. For example, heterogeneous ice nucleation mediated by biogenic aerosols or soot plays an important role in cloud formation.[1–3] In view of climate change, better understanding the melting rate of ice in Greenland is crucial for accurate estimation of the rate of sea level rise. Ice nucleation, especially homogeneous ice nucleation, is still a hard problem for atomistic computer simulations to tackle without resolving to deep supercooling of the system up to twenty degrees or more. Several groups have applied enhanced sampling methods to ice nucleation and gained insights to the process. Forward flux sampling[4] and transition path sampling[5] with aimless shooting have been used in the studies of heterogeneous nucleation of ice on graphite surfaces with at least 30 degrees of supercooling to compute the size of the critical nucleus and examine the validity of the classical nucleation theory, and both studies used the coarse-grained monatomic water model.[6] Quigley and Roger employed metadynamics with Steinhardt order parameters in their study of homogeneous ice nucleation under extreme high supercooling at 180 K with the four-site TIP4P water model.[7] Niu and coworkers also used metadynamics but with the TIP4P/ice water model and a collective variable that was biased on x-ray diffraction peak intensities to study homogeneous ice nucleation.[8] Piaggi and Car investigated the homogeneous nucleation of the Ih phase of ice under atmospheric pressure at 270 K or higher temperatures, in which the kernel CV is used in conjunction with variationally enhanced sampling.[9]

In this work we study homogeneous ice nucleation at 280 K, seven degrees above the melting temperature of ice Ih with VES-PICV enhanced sampling. We bias the kernel CVs in our simulations to drive the transitions between liquid water and ice Ih.

5.1.1 Details of Simulations

We used the LAMMPS[10] package (version 10Feb21) as the MD engine in the simulations of ice in the isobaric-isothermal ensemble (NPT). The time step is 2 fs. The Nosé-Hoover chain thermostat[11] is used at the target temperature 280.0 K with the length of the chain equal to three. The damping constant for the thermostat is 100 fs. The Nosé-Hoover chain barostat of length equal to three is used.[12] The target pressure is 1.0 bar. The damping constant for the barostat is 1000 fs. The three dimensions of the simulation box is coupled during the relaxation by the barostat. The simulation box was not allowed to deform during relaxation. We have conducted both single walker simulations and multiple walker simulations. In the single walker simulations, we have used both liquid water configuration and hexagonal ice state to start the simulations. In the multiple walker simulations there were eight walkers, which were divided into two groups of four walkers that share the same initial configuration either in the liquid water state or in the hexagonal ice state. Initial velocities for each walker are generated a with different random seed in a Gaussian distribution, in order that a broader phase space could be sampled by the simulations. We started the multiple simulations with both the liquid water state and the hexagonal ice state so as to accelerate the convergence of the biased potential in the enhanced sampling. There are 288 water molecules or 864 atoms in the simulation box, which measures $18.2 \times 23.6 \times 22.2$ Å^3 in size with periodic boundary conditions applied in all three dimensions. The water molecules are modeled by the four-site TIP4P/ice water model,[13] in which the positive partial charges are placed on the hydrogen atoms, and the negative charge is on a ghost site (also known as the M site) 0.1577 Å from the oxygen atom bisecting the H-O-H angle in the same plane, as shown in Figure 5.1.

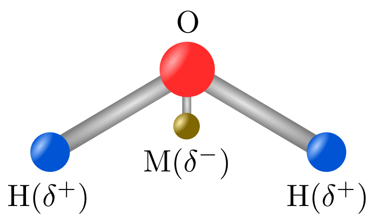


Figure 5.1: A schematic illustration of the TIP4P/ice water model.

The Lennard-Jones interactions are only computed between the oxygen atoms. The cutoff distances of both the electrostatic interaction and Lennard-Jones interaction are both 8.5 Å. The long-range electrostatic interaction is computed with a particle-particle particle-mesh solver with relative accuracy of 10^{-5}. A tail correction is applied to the long range Lennard-Jones interaction. The O-H bond length and the H-O-H bond angle are restrained using the shake algorithm[14] with the energy tolerance of 1.0×10^4 kcal/mol in maximal 200 iterations.

5.1.2 the Kernel CV

The kernel CVs were biased in the VES-PICV simulations of ice, which we have first introduced in the simulations of bulk sodium in Section 4.3.2. It is a measure of local structural similarity of a system of interest when compared with a reference template. The structural similarity is computed with a Gaussian kernel. More information about the kernel CV about the kernel CV can be found in Section 4.3.2. The kernel CV is computed as

$$\tilde{k}_{\chi_0}(\chi) = \frac{\sum_{i\in\chi}\sum_{j\in\chi_0} s(\mathbf{r}_i)\exp\left(-\frac{|\mathbf{r}_i-\mathbf{r}_j^0|^2}{4\sigma^2}\right)}{\sum_{i\in\chi} s(\mathbf{r}_i)}, \tag{5.1}$$

in which χ and χ_0 denote the environments of the system of interest and the reference template, respectively. The number of neighbors is n, which is 17 for ice Ih. The σ controls the spread of the Gaussian kernel, is chosen to be 0.055 nm. $\mathbf{r}_i$ is the vector of the *ith* neighbor to the center atom in the system of interest, while $\mathbf{r}_j^0$ is the vector the *jth* neighbor to the center atom in the reference template. The kernel CV $\tilde{k}_{\chi_0}(\chi)$ is computed as a weighted average, and the weights are computed with a switching function $s(\mathbf{r}_i)$. The function form of $s(\mathbf{r}_i)$ is given below.

$$s(\mathbf{r}_i) = \frac{1-\left(\frac{r_i}{r_0}\right)^{12}}{1-\left(\frac{r_i}{r_0}\right)^{24}}, \tag{5.2}$$

where r_i is the norm of vector $\mathbf{r}_i$, and $r_0 = 1$ nm.

Only the oxygen atoms are considered when computing the kernel CVs. Naturally occurring ice or ice obtained from MD simulations has a disordered hydrogen arrangement, and has an overall zero dipole moment. Water molecules famously form the tetrahedral structures in the ice Ih phase, in which a water molecule has four nearest neighbors on the vertex corners of the tetrahedron. Due to the symmetry of the tetrahedron, four orientations are possible inside the simulation box.[9] Therefore, four templates correspond to the four orientations are used in the calculation of $\tilde{k}_{\chi_0}$, one of which is shown in Figure 5.2d. The set of four templates is $X = \{\chi_1, \chi_2, \chi_3, \chi_4\}$. $k_X(\chi)$ is the maximum among the set $\{\tilde{k}_{\chi_1}(\chi), \tilde{k}_{\chi_2}(\chi), \tilde{k}_{\chi_3}(\chi), \tilde{k}_{\chi_4}(\chi)\}$, and it is computed using

$$k_X(\chi) = \frac{1}{t}\log\left(\sum_i^4 \exp\left[t\cdot\tilde{k}_{\chi_i}(\chi)\right]\right), \tag{5.3}$$

where $t = 100$. Equation 5.3 is used to obtain smooth bias forces.

The probability distribution of $k_X(\chi)$ for liquid water and hexagonal ice, and their snapshots are shown in Figure 5.2. The probability distribution were computed for 288 water molecules simulated at 270 K for the ice structure and at 400 K for liquid water. Figure 5.2c

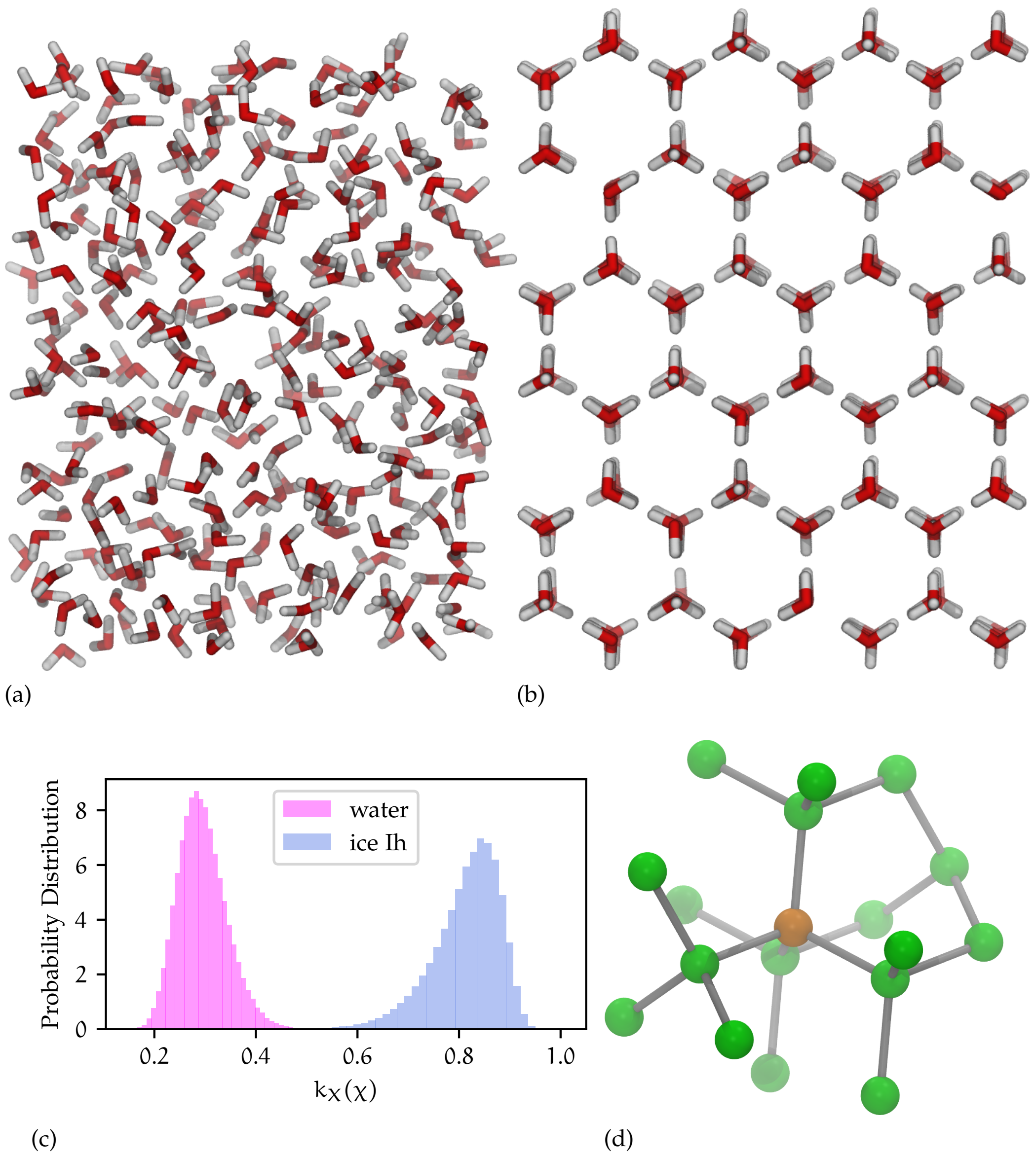


Figure 5.2: (a) A snapshot of liquid water. (b) A snapshot of hexagonal ice looked from the basal plane. (c) The histogram of $k_\chi(\chi)$ in liquid water and hexagonal ice. (d) The ice Ih template with the center oxygen atom in orange and the seventeen nearest neighbors in green. The hydrogen atoms are not shown.

shows the possibility of $k_X(\chi)$ as permutational invariant collective variables to study crystallization of hexagonal ice.

There are two other global collective variables that we used to characterize the state of the system. The first is the number of ice-like particles in the system n_{ice}. Based on Figure 5.2, we decide whether a water molecule is ice-like or water-like based on their values of kernel CVs. If the kernel CV is equal or greater than 0.5, we say the molecule is ice-like. Otherwise, the water molecule is water-like. The n_{ice} is the total number of water molecules that are ice-like, which is computed as

$$n_{nice} = \sum_{i}^{N} \left[1 - \frac{1 - \left(\frac{k_X(\chi_i)}{0.5}\right)^{12}}{1 - \left(\frac{k_X(\chi_i)}{0.5}\right)^{24}} \right]. \tag{5.4}$$

The Steinhardt order parameter Q_6 has been introduced previously in Section 3.2.2. For the switching function the 12th-24th power is used. To compute Q_6, $R_0 = 0.3$ nm and $D_0 = 0$ nm.

5.1.3 Parameters in VES-PICV

The Legendre polynomials up to 20th order are used as the basis functions of the bias potential to cover a range from 0.1 to 0.95 in $k_X(\chi)$. Well-tempered distribution with the bias factor equal to 2 is used as the target distribution $p_{tg}(s)$, and it is updated every 250 iterations. The average stochastic gradient descent optimizer is used to update the linear coefficients every 500 MD steps per iteration. The step size of the optimizer is 2.0. The plumed.dat file for the multiple-walker simulations can be found in Listing E.1.

5.1.4 Results

The time series of the kernel CV of a typical water molecule, and that of the global CVs Q_6 and n_{ice} are given in Figure 5.3. The difference between the single walker simulation and multiple walker simulations is immediately apparent from the time series of the biased kernel CV (Figure 5.3a vs 5.3b). The kernel CV $k_X(\chi)$ of the single walker simulations visits the entire range of values, while it is confined in the starting region of values in the multiple walker simulations. Furthermore, the unbiased global CVs Steinhardt order parameter Q_6 (Figure 5.3c) and the number of ice particles n_{ice} (Figure 5.3e) demonstrate that in the single walker simulations the system can move back forth between the ice Ih state and liquid state rapidly, driven by the bias potential.

In the multiple walker simulations, however, we do not see any effective transitions between the two states. As can been in Figure 5.3f

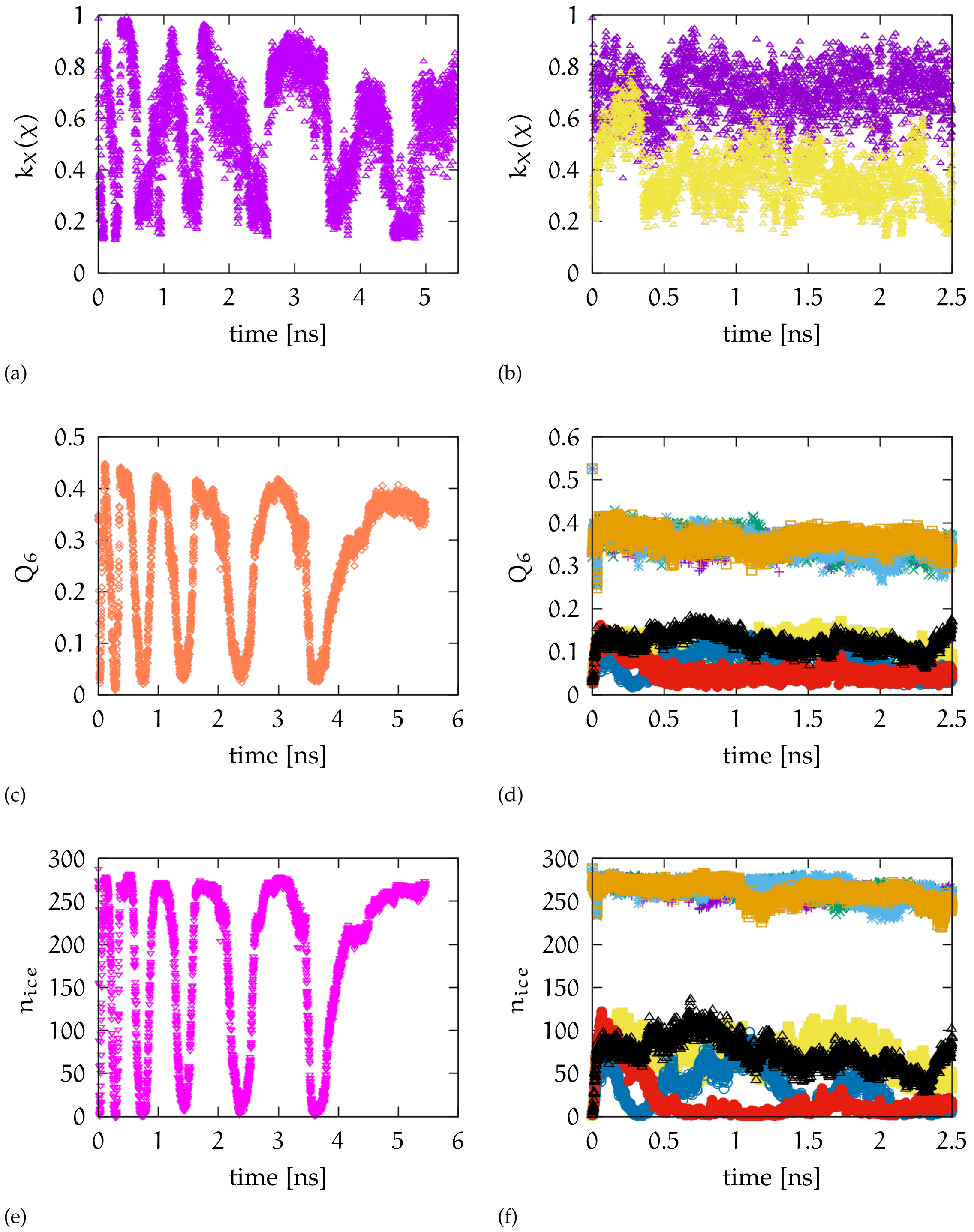


Figure 5.3: The time series of the kernel CV $k_X(\chi)$ (a) of a typical water molecule, Steinhardt order parameter Q_6 (c), number of ice-like water molecules n_{ice} (e) in the single walker simulation, and the kernel CV (b) of a typical water molecule in walker 1 (purple) and water 5 (yellow), Q_6 (d) and number of ice-like water molecules n_{ice} (f) of all eight walkers in the multiple walker simulations.

and Figure 5.3d, the walkers stay in their initial states during the simulations, despite the same CVs and same parameters in VES-PICV are used as in the single walker simulations. In our efforts to improve the sampling of the multiple walker simulations, we have tried parameters in the simulations, for example, different forms of basis functions, the same basis functions but with different ranges, the uniform target distribution instead of the well-tempered target distribution, different step sizes for the stochastic gradient descent optimizer, or the ADAM optimizer and so on. Despite these different parameters, we do not see marked improvements in the sampling of multiple walker simulations of ice with VES-PICV.

5.2 UREA

Urea is a simple molecule comprising of only eight atoms in a relatively rigid structure, which is shown in Figure 5.4. Nevertheless, urea has a rich polymorphism, with at least five metastable crystal forms reported previously.[15–17] Molecular simulations have been used in studies of polymorphism of urea crystals.[18] The form I urea crystal has been found to be the most stable polymorph under the atmospheric pressure. The nucleation of form I urea crystal has been studied by others previously using computer simulations, either from the melt[19], or from a solution[20]. Here, we also study the crystallization of urea from melt with the VES-PICV enhanced sampling method.

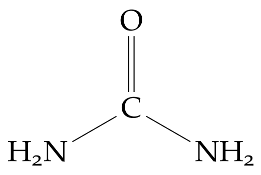


Figure 5.4: The chemical structure of a urea molecule.

5.2.1 Simulation details

GROMACS 2019 is used for the MD simulations. The simulations were conducted in the isobaric-isothermal ensemble (NPT). The Generalized Amber force field (GAFF) was chosen to describe the interactions of urea.[21,22] The cutoff distances for both electrostatic interaction and Lennard-Jones interaction are $0.9\ \mathrm{nm}$. Long range electrostatic interaction is computed with particle-mesh Ewald (PME) method.[23] The order of interpolation of the PME method is 4. The fourier spacing for the PME method is $0.15\ \mathrm{nm}$, and the relative tolerance is 1.0×10^{-5}. All bonds formed with a hydrogen atom are constrained with LINCS algorithm[24] so that the urea molecules are modeled as rigid molecules. The time step is $0.002\ \mathrm{ps}$. The stochastic velocity rescaling thermostat with damping constant equal to 0.1 ps is used for thermostating, and Parrinello-Rahman barostat for barostating. The three dimensions of the simulation box are relaxed isotropically with a time constant 1.0 ps. The cubic simulation box is $2.21 \times 2.21 \times 1.86\ \mathrm{nm}^3$ in size with 128 urea molecules and 1024 atoms in total. Periodic boundary conditions are applied to all three dimensions.

5.2.2 the SMAC Collective Variable

Giberti *et al.* introduced the SMAC collective variable ($\mathcal{S}$) to distinguish the different polymorphs of urea in the solid phase and the liquid phase.[19] $\mathcal{S}$ describes the global crystallinity of a molecular crystall, and it is the average of the individual contributions (Γ) from each molecule in the system, i. e.,

$$\mathcal{S} = \frac{1}{N}\sum_{i=1}^{N}\Gamma_i. \tag{5.5}$$

The Γ_i has two major contributions, which describe the local packing density of molecule i, and the relative orientations of neighboring molecules to i, respectively. The local packing density is easily described by the coordination numbers of molecule i within a cutoff distance r_{cut}. The molecule j contributes to the coordination number of the neighboring molecule i by ϕ_{ij}, which is what we call the pairwise coordination number in the simulations of seven Lennard-Jones particles (Equation 4.16) and written as

$$\phi_{ij}(r_{ij}) = \frac{1-\left(\frac{r_{ij}}{r_{cut}}\right)^6}{1-\left(\frac{r_{ij}}{r_{cut}}\right)^{12}}. \tag{5.6}$$

r_{ij} is the distance between the centers of mass of molecules i and j. The coordination number of the molecule i is a sum of ϕ_{ij}, i. e., $n_i = \sum_{j\neq i}\phi_{ij}$, when $j \neq i$.

The orientations of neighboring molecules are computed for molecules with a sufficiently large coordination number. The switching function ψ_i is defined to filter out molecules with low coordination numbers:

$$\psi_i(n_i) = 1 - \exp\left(-\frac{n_i}{n_{cut}}\right), \tag{5.7}$$

where n_{cut} is the cutoff number of neighbors.

The relative orientation of two molecules are described the sum of a series of functions $\sum_n K_n(\theta_{ij})$ over all characteristic angles defined for neighboring molecules. For a specific characteristic angle n between molecule i and j:

$$K_n(\theta_{ij}) = \exp\left(-\frac{(\theta_{ij}-\bar{\theta}_n)^2}{2\sigma_n^2}\right), \tag{5.8}$$

where θ_{ij} is the torsional angle between the internal orientation vectors for molecules i and j, and $\bar{\theta}_n$ is the average value of θ_{ij} when the system in the crystalline state. Combining the functions above we have the function form of Γ_i:

$$\begin{aligned}\Gamma_i &= \frac{\psi_i}{n_i}\sum_{j\neq i}\phi_{ij}\sum_n K_n(\theta_{ij}) \\ &= \frac{\psi\left[\sum_{j\neq i}\phi(r_{ij})\right]\sum_{j\neq i}\phi(r_{ij})\sum_n \exp\left(-\frac{(\theta_{ij}-\bar{\theta}_n)^2}{2\sigma_n^2}\right)}{\sum_{j\neq i}\phi(r_{ij})}.\end{aligned} \tag{5.9}$$

In their work of well-tempered metadynamics simulations of urea, Giberti and coworkers defined two vectors m and n in a urea molecule as seen Figure 5.5a, and the two corresponding SMAC collective

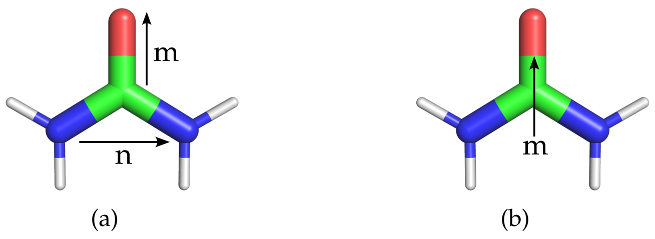


Figure 5.5: Vectors defined on a urea molecule used to computed the SMAC CV. In (a), the m vector is defined as the C-O bond, and the n vector is the two N atoms. In (b) the m vector is defined as the vector from the middle point between the two N atoms to the middle point of the C-O bond.

variables S_m and S_n. For each vector, two relative orientation angle K functions are defined. For example, for the m vector the $\bar{\theta}$ of the two kernel functions is 0 and π, respectively.

We would like to point out that in the introduction of SMAC collective variable,[19] the authors used the Fermi function as the two switching functions instead of the function forms used in Equation 5.6-5.7.

In our work here, we do not bias the global SMAC collective variable $\mathcal{S}$. Instead, we bias the individual molecular contributions Γ_i in the VES-PICV simulations. We have found that to distinguish the urea melt and urea form I crystal structure, the orientations of m vectors in Figure 5.5a are sufficient. The parameters used in the computation of Γ_i is listed in Table 5.1.

	$r_{cut}[nm]$	n_{cut}	$\bar{\theta}_1$	$\bar{\theta}_2$	σ_1	σ_2
Γ_i	0.6	4	0	π	0.8	0.7

Table 5.1: Parameters used in the computation of SMAC (Γ_i).

The probability distribution of Γ_i is given in Figure 5.6, in which snapshots of the urea melt and form I urea crystal structure are also shown. The form I urea crystal is obtained from Cambridge Structural Database its entry number is UREAXX02. Its structure was determined by Zavodnik *et al.*[25] The probability distribution of Γ_i for the solid state is computed for a trajectory of urea crystal simulated at 300 K. The probability distribution for the liquid state is computed for a trajectory of urea melt at 500 K. Although there is an overlap between two distributions, the two distributions are different enough for us to use Γ_i in our VES-PICV simulations.

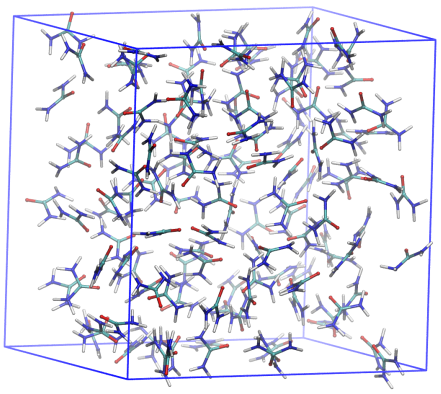
VES-PICV The 20th order Legendre functions were used as the basis functions over the range from 1 to 14. The target distribution was the well-tempered distribution. We have tried several values for the bias factor γ from 1.2 to 3. The average stochastic gradient optimizer was used with a stride of 1000 steps. The target distribution was updated every 100 iterations. The plumed.dat used for the simulations is included in Listing E.2.

5.2.3 Results

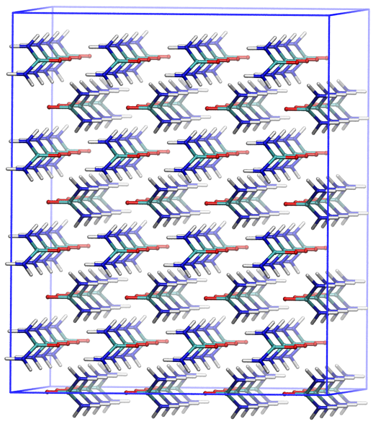
In our simulations of urea, we have encountered an issue with the LINCS algorithm, and the simulations would break down around 100 ps. To overcome this issue, we redefined the m vector as from the middle of the two nitrogen atoms to the middle of carbon and oxygen atoms, as shown in Figure 5.5b. Although this change slight extended simulations to around 200 ps, the VES-PICV simulations still broke down.

5.3 DISCUSSIONS

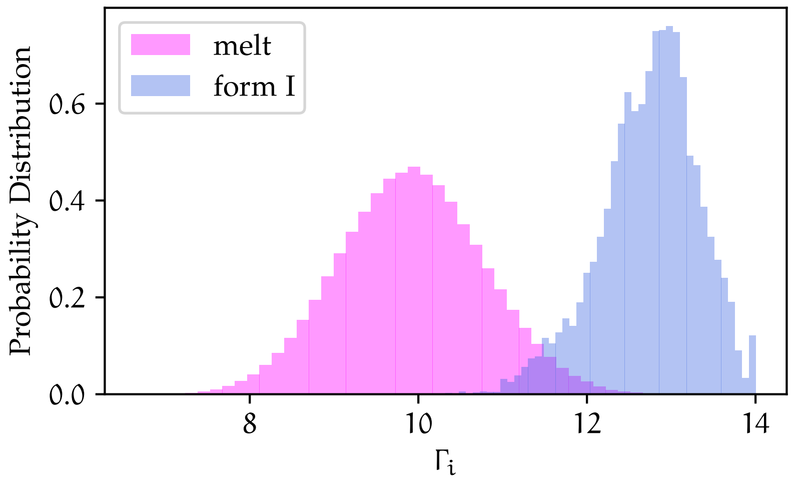
The results of VES-PICV have highlighted the need of deeper understanding of effects of biasing a group of permutational invariant collective variables so that it can be successfully applied to phase transitions. It is quite puzzling why the single walker simulations should be able to drive ice nucleation, while the multiple walker simulations are not. In the case of VES-PICV simulations of urea, the bias force seemed to have interfered with the LINCS algorithm that constrains the lengths of bonds involving hydrogen atoms, which regularly broke down during the simulations. Better distributions of the bias force among the atoms of a molecule can provide relieve to this issue, however, more improvements to the method are needed.



(a)



(b)



(c)

Figure 5.6: (a) A snapshot of urea melt. (b) A snapshot of urea crystal. (c) Histograms of Γ_i in liquid urea and in form I urea crystal.

REFERENCES

1. Schneider, J. *et al.* The Seasonal Cycle of Ice-Nucleating Particles Linked to the Abundance of Biogenic Aerosol in Boreal Forests. *Atmospheric Chemistry and Physics* **21,** 3899–3918 (2021).
2. Gorbunov, B., Baklanov, A., Kakutkina, N., Windsor, H. & Toumi, R. Ice Nucleation on Soot Particles. *Journal of Aerosol Science* **32,** 199–215 (Feb. 1, 2001).
3. Knopf, D. A., Alpert, P. A. & Wang, B. The Role of Organic Aerosol in Atmospheric Ice Nucleation: A Review. *ACS Earth Space Chem.* **2,** 168–202 (Mar. 15, 2018).
4. Cabriolu, R. & Li, T. Ice Nucleation on Carbon Surface Supports the Classical Theory for Heterogeneous Nucleation. *Phys. Rev. E* **91,** 052402 (May 13, 2015).
5. Lupi, L., Peters, B. & Molinero, V. Pre-Ordering of Interfacial Water in the Pathway of Heterogeneous Ice Nucleation Does Not Lead to a Two-Step Crystallization Mechanism. *J. Chem. Phys.* **145,** 211910 (Dec. 7, 2016).
6. Molinero, V. & Moore, E. B. Water Modeled As an Intermediate Element between Carbon and Silicon. *J. Phys. Chem. B* **113,** 4008–4016 (Apr. 2, 2009).
7. Quigley, D. & Rodger, P. M. Metadynamics Simulations of Ice Nucleation and Growth. *J. Chem. Phys.* **128,** 154518 (Apr. 18, 2008).
8. Niu, H., Piaggi, P. M., Invernizzi, M. & Parrinello, M. Molecular Dynamics Simulations of Liquid Silica Crystallization. *PNAS* **115,** 5348–5352 (May 22, 2018).
9. Piaggi, P. M. & Car, R. Phase Equilibrium of Liquid Water and Hexagonal Ice from Enhanced Sampling Molecular Dynamics Simulations. *J. Chem. Phys.* **152,** 204116 (May 27, 2020).
10. Plimpton, S. Fast Parallel Algorithms for Short-Range Molecular Dynamics. *Journal of Computational Physics* **117,** 1–19 (Mar. 1, 1995).
11. Hoover, W. G. Canonical Dynamics: Equilibrium Phase-Space Distributions. *Phys. Rev. A* **31,** 1695–1697 (Mar. 1, 1985).
12. Tuckerman, M. E., Alejandre, J., López-Rendón, R., Jochim, A. L. & Martyna, G. J. A Liouville-operator Derived Measure-Preserving Integrator for Molecular Dynamics Simulations in the Isothermal–Isobaric Ensemble. *Journal of Physics A: Mathematical and General* **39,** 5629–5651 (Apr. 24, 2006).
13. Abascal, J. L. F., Sanz, E., García Fernández, R. & Vega, C. A Potential Model for the Study of Ices and Amorphous Water: TIP4P/Ice. *The Journal of Chemical Physics* **122,** 234511 (June 15, 2005).

14. Ryckaert, J.-P., Ciccotti, G. & Berendsen, H. J. Numerical Integration of the Cartesian Equations of Motion of a System with Constraints: Molecular Dynamics of n-Alkanes. *Journal of Computational Physics* **23,** 327–341 (Mar. 1, 1977).

15. Dziubek, K., Citroni, M., Fanetti, S., Cairns, A. B. & Bini, R. High-Pressure High-Temperature Structural Properties of Urea. *J. Phys. Chem. C* **121,** 2380–2387 (Feb. 2, 2017).

16. Lamelas, F. J., Dreger, Z. A. & Gupta, Y. M. Raman and X-Ray Scattering Studies of High-Pressure Phases of Urea. *J. Phys. Chem. B* **109,** 8206–8215 (Apr. 1, 2005).

17. Olejniczak, A., Ostrowska, K. & Katrusiak, A. H-Bond Breaking in High-Pressure Urea. *J. Phys. Chem. C* **113,** 15761–15767 (Sept. 3, 2009).

18. Piaggi, P. M. & Parrinello, M. Predicting Polymorphism in Molecular Crystals Using Orientational Entropy. *PNAS* **115,** 10251–10256 (Oct. 9, 2018).

19. Giberti, F., Salvalaglio, M., Mazzotti, M. & Parrinello, M. Insight into the Nucleation of Urea Crystals from the Melt. *Chemical Engineering Science* **121,** 51–59 (Jan. 2015).

20. Salvalaglio, M., Mazzotti, M. & Parrinello, M. Urea Homogeneous Nucleation Mechanism Is Solvent Dependent. *Faraday Discuss.* **179,** 291–307 (2015).

21. Cornell, W. D., Cieplak, P., Bayly, C. I., Gould, I. R., Merz, K. M., Ferguson, D. M., Spellmeyer, D. C., Fox, T., Caldwell, J. W. & Kollman, P. A. A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules. *J. Am. Chem. Soc.* **117,** 5179–5197 (May 1, 1995).

22. Wang, J., Wolf, R. M., Caldwell, J. W., Kollman, P. A. & Case, D. A. Development and Testing of a General Amber Force Field. *Journal of Computational Chemistry* **25,** 1157–1174 (July 15, 2004).

23. Hockney, R. W. & Eastwood, J. W. *Computer Simulation Using Particles* 540 pp. (CRC Press, Boca Raton, Mar. 25, 2021).

24. Hess, B. P-LINCS: A Parallel Linear Constraint Solver for Molecular Simulation. *J. Chem. Theory Comput.* **4,** 116–122 (Jan. 1, 2008).

25. Zavodnik, V., Stash, A., Tsirelson, V., de Vries, R. & Feil, D. Electron Density Study of Urea Using TDS-corrected X-ray Diffraction Data: Quantitative Comparison of Experimental and Theoretical Results. *Acta Cryst B* **55,** 45–54 (1 Feb. 1, 1999).

6 CONCLUSIONS

In Chapter 1, we have introduced the basics of Monte Carlo simulations, molecular dynamics simulations, and the theoretical foundation for both methods — statistical thermodynamics. In MD simulations the velocity Verlet integrator can be used to propagate the dynamics of the system, and maintains a constant internal energy. Thermostats and barostats are used for simulations in the canonical ensemble (NVT) or the isobaric-isothermal ensemble (NPT).

In Chapter 2, we have discussed different flavors of enhanced sampling methods, including umbrella sampling, metadynamics and its popular variant well-tempered metadynamics, and the variationally enhanced sampling method. They build a customized bias potential in different processes over a representative collective variable space ($\mathbf{s}$) of reduced dimensions in comparison to the high dimensional phase space. The underlying free energy surface $F(\mathbf{s})$ can be estimated according to its relationship with the bias potential $V(\mathbf{s})$ when the convergence of sampling has been reached. Additionally, the free energy estimation can also be achieved by reweighting. Reweighting poses a less stringent demand on the convergence of the bias potential. Furthermore, users can obtain projections of the free energy surface to collective variables that are not biased in the simulations. Residual reweighting and last bias reweighting have been introduced in Chapter 2. PLUMED 2 is a plugin library that has implemented the enhanced sampling methods mentioned above and various collective variables. PLUMED 2 can be used together with many popular MD engines during real time simulations. Or it can be used in postprocessing.

The development of enhanced sampling methods has broadened role that computer simulations paly in tackling various scientific questions. To give ESPResSo++, a MD engine developed and maintained by the Theory group of Max Planck Institute for Polymer Research, Mainz, access to PLUMED 2, we have developed an extension that connects the two software packages. With PLUMED 2 and ESPResSo++, we have investigated the first order transition of the 128 monomer long single-chain square-well with well-tempered metadynamics. In the MD simulations, rigid bonds of the single-chain square-well polymer is replaced with harmonic bonds, and the square-well potential between nonbonded monomers are is replaced a smooth square-well potential so that a smooth force can be derived in the MD simulations. We have performed three biased simulations at three temperatures, at the previously reported coexistence temperature, one below it and one above it. Then we determined the free energy of crystallization from these three simulations by reweighting. And then the coexistence temperature is determined from the free energy of crystallization. Our result largely agree with previously reported result.

In Chapter 4 we have introduced a new development of variationally enhanced sampling so that permutationally invariant collective variables can be used with VES. Permutationally invariant collective variables are local collective variables with arbitrarily exchangeable indexes. They are particularly useful in distinguishing different polymorphs of molecular or atomic crystals. We have modified the process of updating the bias potential that is represented by a linear combinations of the basis functions. We dubbed the new method variationally enhanced sampling with permutationally invariant collective variables (VES-PICV). We have applied the new method to two systems: seven Lennard-Jones particles in a two dimensional space and the bulk sodium. The permutationally invariant collective variables are the coordinate numbers and kernel CVs in the two simulations, respectively. We have been able to estimate the free energy landscape for the cases by reweighting. We have also examined the sampled distributions of the coordination numbers of the seven particles, which have converged. In the case of the bulk sodium, the sampled distributions of the kernel CVs, however, did not converge fully in the simulation. In both cases, we have obtained free energy surface estimations in good qualities by reweighting.

In Chapter 5, we have applied the VES-PICV method to ice nucleation and crystallization of urea from the melt. We have biased the local kernel CVs for ice nucleation, and the local SMAC collective variables for crystallization of urea. In the multiple walker simulations of ice nucleation, VES-PICV methods could not drive transitions between the liquid state and the ice Ih state, although the simulations ran to completion. In the case of urea, however, the simulations would break down only 200 ps into the simulations. Other collective variables may be explored in the future in order for the simulations to proceed.

The difficulties that we have run into in the simulation studies of ice nucleation and crystallization of urea highlight the need of further development of VES-PICV methods so that it see wider applications.

APPENDIX

A SOURCE CODE OF THE EXTPLUMED EXTENSION

The source code of the ExtPlumed extension is in three files: ExtPlumed.py, ExtPlumed.hpp and ExtPlumed.cpp.

Listing A.1: ExePlumed.py

```
#  Copyright (C) 2018
#      Max Planck Institute for Polymer Research
#
#  This file is part of ESPResSo++.
#
#  ESPResSo++ is free software: you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation, either version 3 of the License, or
#  (at your option) any later version.
#
#  ESPResSo++ is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with this program.  If not, see <http://www.gnu.org/licenses/>.


r"""
**************************************************
espressopp.integrator.ExtPlumed
**************************************************

This extension serves as interface between ESPResSo++ and `PLUMED <http://www.plumed.org/home>`_. PLUMED can
be used to run realtime analysis and bias the simulation along chosen collective variables. Details of using
PLUMED and its input file can be found on its website.

To call PLUMED, set environment variable :envvar:`PLUMED_KERNEL` to the full path of libplumedKernel.so, and add
the path of the directory of libplumed.so to environment variable :envvar:`LD_LIBRARY_PATH`, when PLUMED is linked through runtime linking.
If PLUMED is linked by dynamic linking, i.e., it is linked as a shared library to ESPResSo++, :envvar:`PLUMED_KERNEL` needs not to be set.
If PLUMED is linked statically, neither environment variable needs to be provided.
Error messages regarding to "undefined symbol" can be ignored, as long as the simulation runs.
```

```
By default, PLUMED is linked by runtime linking. The user can change ↙
    ↳ the way of PLUMED linking to ESPResSo++ by changing CMake ↙
    ↳ variable :makevar:‘PLUMED_LINK_TYPE‘
to either "shared" or "static" from the default "runtime". If your ↙
    ↳ distribution of PLUMED is not installed in a default system ↙
    ↳ directory, you can define the root
path of PLUMED installation as CMake variable :makevar:‘PLUMED_HOME‘. ↙
    ↳ If you keep the default "runtime" linking, your installation of ↙
    ↳ PLUMED is not searched. Hence, you
do not need to worry about whether your PLUMED is not installed in a ↙
    ↳ default system directory or setting :makevar:‘PLUMED_HOME‘.

usage:

.. code:: python

    plumed = espressopp.integrator.ExtPlumed(system, "plumed.dat", ↙
        ↳ "log.plumed", 0.005)
    plumed.setNaturalUnits()
    plumed.Init()
    integrator.addExtension(plumed)

or:

.. code:: python

    plumed = espressopp.integrator.ExtPlumed(system, "plumed.dat", ↙
        ↳ "log.plumed", 0.005)
    plumed.setNaturalUnits()
    plumed.setRestart(1)
    plumed.Init()
    integrator.addExtension(plumed)

.. function:: espressopp.integrator.ExtPlumed(system, cmd, log, dt)

                :param system: The Espresso++ system object.
                :type system: espressopp.System
                :param cmd: input file for PLUMED
                :type cmd: ‘‘str‘‘
                :param log: log file for PLUMED
                :type log:  ‘‘str‘‘
                :param dt:  time step
                :type dt: ‘‘float‘‘ (default: 0.005)

"""
from espressopp.esutil import cxxinit
from espressopp import pmi
from espressopp.integrator.Extension import *
from _espressopp import integrator_ExtPlumed
import mpi4py.MPI as MPI

class ExtPlumedLocal(ExtensionLocal, integrator_ExtPlumed):

    def __init__(self, system, cmd, log, dt, restart=False):

        if not (pmi._PMIComm and pmi._PMIComm.isActive()) or ↙
            ↳ pmi._MPIcomm.rank in pmi._PMIComm.getMPIcpugroup():
```

```
            cxxinit(self, integrator_ExtPlumed, system, cmd, log, dt, ↙
                ↳ restart)

if pmi.isController :
    class ExtPlumed(Extension):
        __metaclass__ = pmi.Proxy
        pmiproxydefs = dict(
            cls = 'espressopp.integrator.ExtPlumedLocal',
            pmicall = ['getBias']
            )
```

Listing A.2: ExtPlumed.hpp

```
/*
  Copyright (C) 2018
  Max Planck Institute for Polymer Research

  This file is part of ESPResSo++.

  ESPResSo++ is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  ESPResSo++ is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/

// ESPP_CLASS
#ifndef _INTEGRATOR_ExtPlumed_HPP
#define _INTEGRATOR_ExtPlumed_HPP

#include "python.hpp"
#include "types.hpp"
#include "Extension.hpp"
#include "boost/signals2.hpp"
#include "mpi.hpp"
#include "Particle.hpp"
#include "Plumed.h"

namespace espressopp {
  namespace integrator {

    /** ExtPlumed */
    class ExtPlumed : public Extension {

    public:
      ExtPlumed(shared_ptr < System >, std::string, std::string, real, ↙
          ↳ bool);
      real getBias();

      virtual ~ExtPlumed();
      /** Register this class so it can be used from Python. */
      static void registerPython();

    private:
      PLMD::Plumed * p;
      real dt;
      long step;

      longint nreal; // total number of atoms (real & ghost) on the ↙
          ↳ processor
      longint natoms; // total number of atoms
      int *  gatindex;
      real * masses;
      real * charges;
      real * pos;
      real * f;
      real bias;
      bool particlesChanged;

      boost::signals2::connection _runInit, _aftCalcF, _aftIntP;
      boost::signals2::connection _onParticlesChanged;

      void onParticlesChanged();
      void connect();
      void disconnect();
      void setStep();
      void updateForces();
      void updateStep();
    };
  }
}

#endif
```

Listing A.3: ExtPlumed.cpp

```
/*
  Copyright (C) 2018
  Max Planck Institute for Polymer Research

  This file is part of ESPResSo++.

  ESPResSo++ is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  ESPResSo++ is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/

#include <string>
#include "python.hpp"
#include "System.hpp"
#include "storage/Storage.hpp"
```

```
#include "interaction/Interaction.hpp"
#include "iterator/CellListIterator.hpp"
#include "bc/BC.hpp"
#include "ExtPlumed.hpp"
#include "mpi.hpp"

namespace espressopp {

  using namespace iterator;
  using std::string;

  namespace integrator {

    ExtPlumed::ExtPlumed(shared_ptr<System> _system, string _dat, string _log, ↙
         ↳ real _dt, bool _restart):
      Extension(_system),
      dt(_dt),
      step(0),
      nreal(0),
      gatindex(NULL),
      masses(NULL),
      f(NULL),
      pos(NULL),
      charges(NULL),
      particlesChanged(false)
    {
      p=new PLMD::Plumed;
      int real_precision = sizeof(real);
      p->cmd("setRealPrecision",&real_precision);
      p->cmd("setMDEngine","ESPResSo++");
      MPI_Comm comm = MPI_Comm(*_system->comm);
      p->cmd("setMPIComm", &comm);
      bool dat_is_file = (_dat.find_first_of("\n") > _dat.size()); // test if ↙
           ↳ input is a multline string.
      if (dat_is_file) p->cmd("setPlumedDat", _dat.c_str());
      p->cmd("setLogFile", _log.c_str());
      p->cmd("setTimestep",&dt);
      longint tmp = _system->storage->getNRealParticles();
      boost::mpi::all_reduce(*_system->comm, tmp, natoms, std::plus<longint>());
      p->cmd("setNatoms",&natoms);
      if (_restart) {
        int res = 1;
        p->cmd("setRestart", &res);
      }
      p->cmd("init");
      if (!dat_is_file) {
        std::istringstream iss(_dat);
        std::string token;
        iss >> std::ws; // remove leading white spaces in a line
        while(std::getline(iss, token)) {
          iss >> std::ws; // remove leading white spaces in a line
          if (token[0] == '#') continue;
          p->cmd("readInputLine", token.c_str());
          token.clear();
        }
      }
      _onParticlesChanged = ↙
           ↳ _system->storage->onParticlesChanged.connect(boost::bind( ↙
           ↳ &ExtPlumed::onParticlesChanged, this));
    }

    ExtPlumed::~ExtPlumed() {
      _onParticlesChanged.disconnect();
      delete [] f;
      delete [] pos;
      delete [] charges;
      delete [] gatindex;
      delete [] masses;
      delete p;
    }

    real ExtPlumed::getBias() {
      return bias;
    }

    void ExtPlumed::onParticlesChanged() {
      particlesChanged = true;
    }

    void ExtPlumed::disconnect() {
      _runInit.disconnect();
      _aftCalcF.disconnect();
      _aftIntP.disconnect();
    }

    void ExtPlumed::connect() {
      _runInit = integrator->runInit.connect( boost::bind(&ExtPlumed::setStep, ↙
           ↳ this));
      _aftCalcF = integrator->aftCalcF.connect( ↙
              ↳ boost::bind(&ExtPlumed::updateForces, this));
      _aftIntP = integrator->aftIntP.connect( ↙
              ↳ boost::bind(&ExtPlumed::updateStep, this));
    }

    void ExtPlumed::setStep() {
      step = long(integrator->getStep());
    }

    void ExtPlumed::updateForces() {
      System& system = getSystemRef();
      CellList realCells = system.storage->getRealCells();

      if (nreal!=system.storage->getNRealParticles()) {
        nreal = system.storage->getNRealParticles();
        if(charges) delete [] charges;
        if(masses) delete [] masses;
        if(gatindex) delete [] gatindex;
        if(pos) delete [] pos;
        if(f) delete [] f;
        gatindex = new int [nreal];
        masses = new real [nreal];
        charges = new real [nreal];
        pos = new real[nreal*3];
        f = new real[nreal*3];

        for(auto tp=std::make_pair(0, CellListIterator(realCells));
            tp.first<nreal && !tp.second.isDone();
            ++tp.first, ++tp.second)
          {
            gatindex[tp.first] = tp.second->getId() - 1;
          }
        particlesChanged = false;

      } else if (particlesChanged) {
        for(auto tp=std::make_pair(0, CellListIterator(realCells));
            tp.first<nreal && !tp.second.isDone();
            ++tp.first, ++tp.second)
          {
            gatindex[tp.first] = tp.second->getId() - 1;
          }
        particlesChanged = false;
      }

      for(auto tp=std::make_pair(0, CellListIterator(realCells));
          tp.first<nreal && !tp.second.isDone();
          ++tp.first, ++tp.second)
        {
          masses[tp.first] = tp.second->mass();
          charges[tp.first] = tp.second->q();
          std::copy(tp.second->force().begin(), tp.second->force().end(), ↙
               ↳ &f[tp.first*3]);
          std::copy(tp.second->position().begin(), tp.second->position().end(), ↙
               ↳ &pos[tp.first*3]);
        }

      p->cmd("setStepLong",&step);
      p->cmd("setAtomsNlocal",&nreal);
      p->cmd("setAtomsGatindex",&gatindex[0]);

      real box[3][3];
      for(int i=0;i<3;i++) for(int j=0;j<3;j++) box[i][j]=0.0;
      Real3D L = system.bc->getBoxL();
      box[0][0]=L[0];
      box[1][1]=L[1];
      box[2][2]=L[2];

      real virial[3][3];
      for(int i=0; i<3; ++i) for(int j=0; j<3; ++j) virial[i][j]=0.0;

      p->cmd("setPositions", pos);
      p->cmd("setForces", f);
      p->cmd("setBox",&box[0][0]);
      p->cmd("setMasses",masses);
      p->cmd("setCharges",charges);
      p->cmd("setVirial", &virial[0][0]);
      p->cmd("getBias",&bias);
      p->cmd("prepareCalc");

      int plumedNeedsEnergy = 0;
      p->cmd("isEnergyNeeded", &plumedNeedsEnergy);
      if (plumedNeedsEnergy) {
        real pot_energy = 0.;
        const interaction::InteractionList& srIL = system.shortRangeInteractions;
        for (size_t j =0; j < srIL.size(); ++j) {
          pot_energy += srIL[j]->computeEnergy();
        }
        pot_energy /= system.comm->size(); // PLUMED defines PE this way.
        p->cmd("setEnergy", &pot_energy);
      }
      p->cmd("performCalc");

      for(auto tp=std::make_pair(0, CellListIterator(realCells));
          tp.first<nreal && !tp.second.isDone();
          ++tp.first, ++tp.second)
        {
          std::copy(&f[tp.first*3], &f[tp.first*3+3], tp.second->force().begin());
        }
    }

    void ExtPlumed::updateStep() {step++;}

    /****************************************************
     ** REGISTRATION WITH PYTHON
     ****************************************************/

    void ExtPlumed::registerPython() {

      using namespace espressopp::python;
```

```
      class_<ExtPlumed, shared_ptr<ExtPlumed>, bases<Extension> >

        ("integrator_ExtPlumed", init< shared_ptr< System >, string, string, ↙
             ↪ real, bool>())
        .def("getBias", &ExtPlumed::getBias)
        .def("connect", &ExtPlumed::connect)
        .def("disconnect", &ExtPlumed::disconnect)
        ;
    }
  }
}
```

B SOURCE CODE FOR THE SMOOTH SQUARE-WELL POTENTIAL

The implementation of the smooth square-well pairwise potential involves three files, SmoothSquareWell.py, SmoothSquareWell.hpp and SmoothSquareWell.cpp. Their contents are listed below.

Listing B.1: SmoothSquareWell.py

```python
#  Copyright (C) 2017,2018
#      Max Planck Institute for Polymer Research
#
#  This file is part of ESPResSo++.
#
#  ESPResSo++ is free software: you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation, either version 3 of the License, or
#  (at your option) any later version.
#
#  ESPResSo++ is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with this program.  If not, see <http://www.gnu.org/licenses/>.

r"""
***************************************
espressopp.interaction.SmoothSquareWell
***************************************
This is an implementation of the smoothed square-well potential from `Leitold ↙
     ↪ and Dellago JCP 141, 134901 (2014) ↙
     ↪ <https://doi.org/10.1063/1.4896560>`_ :

.. math::

    V(r) = \frac{\varepsilon}{2} \left\{ \exp\left[\frac{-(r-\sigma)}{a}\right] ↙
         ↪ + \tanh\left[\frac{r-\lambda\sigma}{a}\right] - 1 \right\},

of which :math:`a` dictates the steepness of the slope of the square well, and ↙
     ↪ :math:`{\lambda\sigma}` determines the width of the step, and ↙
     ↪ :math:`{\sigma}` is the bond length of the polymer.

To reproduce the potential in the prior reference, use the code below.

.. code:: python

   pot = espressopp.interaction.SmoothSquareWell(epsilon=1.0,sigma=1.0,cutoff=2.5)
   pot.a = 0.002
   pot.Lambda = 1.05

The SmoothSquareWell potential supports VerletListInteraction, ↙
     ↪ FixedPairListInteraction and FixedPairListTypesInteraction.
```

```
"""

from espressopp import pmi, infinity
from espressopp.esutil import *

from espressopp.interaction.Potential import *
from espressopp.interaction.Interaction import *
from _espressopp import interaction_SmoothSquareWell, ↙
     ↳ interaction_VerletListSmoothSquareWell, \
    interaction_FixedPairListSmoothSquareWell, ↙
         ↳ interaction_FixedPairListTypesSmoothSquareWell

class SmoothSquareWellLocal(PotentialLocal, interaction_SmoothSquareWell):

    def __init__(self, epsilon=1.0, sigma=0.0, cutoff=infinity, shift=0.0):
        """Initialize the local SmoothSquareWell object."""
        if not (pmi._PMIComm and pmi._PMIComm.isActive()) or pmi._MPIcomm.rank ↙
             ↳ in pmi._PMIComm.getMPIcpugroup():
            if shift == "auto":
                cxxinit(self, interaction_SmoothSquareWell, epsilon, sigma, ↙
                     ↳ cutoff)
            else:
                cxxinit(self, interaction_SmoothSquareWell, epsilon, sigma, ↙
                     ↳ cutoff, shift)

class VerletListSmoothSquareWellLocal(InteractionLocal, ↙
     ↳ interaction_VerletListSmoothSquareWell):

    def __init__(self, vl):
        if not (pmi._PMIComm and pmi._PMIComm.isActive()) or pmi._MPIcomm.rank ↙
             ↳ in pmi._PMIComm.getMPIcpugroup():
            cxxinit(self, interaction_VerletListSmoothSquareWell, vl)

    def setPotential(self, type1, type2, potential):
        if not (pmi._PMIComm and pmi._PMIComm.isActive()) or pmi._MPIcomm.rank ↙
             ↳ in pmi._PMIComm.getMPIcpugroup():
            self.cxxclass.setPotential(self, type1, type2, potential)

    def getPotential(self, type1, type2):
        if not (pmi._PMIComm and pmi._PMIComm.isActive()) or pmi._MPIcomm.rank ↙
             ↳ in pmi._PMIComm.getMPIcpugroup():
            return self.cxxclass.getPotential(self, type1, type2)

    def getVerletListLocal(self):
        if not (pmi._PMIComm and pmi._PMIComm.isActive()) or pmi._MPIcomm.rank ↙
             ↳ in pmi._PMIComm.getMPIcpugroup():
            return self.cxxclass.getVerletList(self)

class FixedPairListSmoothSquareWellLocal(InteractionLocal, ↙
     ↳ interaction_FixedPairListSmoothSquareWell):

    def __init__(self, system, fpl, potential):
        if not (pmi._PMIComm and pmi._PMIComm.isActive()) or pmi._MPIcomm.rank ↙
             ↳ in pmi._PMIComm.getMPIcpugroup():
            cxxinit(self, interaction_FixedPairListSmoothSquareWell, system, ↙
                 ↳ fpl, potential)

    def setPotential(self, potential):
        if not (pmi._PMIComm and pmi._PMIComm.isActive()) or pmi._MPIcomm.rank ↙
             ↳ in pmi._PMIComm.getMPIcpugroup():
            self.cxxclass.setPotential(self, potential)

    def getPotential(self):
        if not (pmi._PMIComm and pmi._PMIComm.isActive()) or pmi._MPIcomm.rank ↙
             ↳ in pmi._PMIComm.getMPIcpugroup():
            self.cxxclass.getPotential(self)
    def setFixedPairList(self, fpl):
        if not (pmi._PMIComm and pmi._PMIComm.isActive()) or pmi._MPIcomm.rank ↙
            ↳ in pmi._PMIComm.getMPIcpugroup():
            self.cxxclass.setFixedPairList(self, fpl)

    def getFixedPairList(self):
        if not (pmi._PMIComm and pmi._PMIComm.isActive()) or pmi._MPIcomm.rank ↙
            ↳ in pmi._PMIComm.getMPIcpugroup():
            return self.cxxclass.getFixedPairList(self)

class FixedPairListTypesSmoothSquareWellLocal(InteractionLocal, ↙
    ↳ interaction_FixedPairListTypesSmoothSquareWell):

    def __init__(self, system, fpl, potential):
        if not (pmi._PMIComm and pmi._PMIComm.isActive()) or pmi._MPIcomm.rank ↙
            ↳ in pmi._PMIComm.getMPIcpugroup():
            cxxinit(self, interaction_FixedPairListSmoothSquareWell, system, fpl)

    def setPotential(self, type1, type2, potential):
        if not (pmi._PMIComm and pmi._PMIComm.isActive()) or pmi._MPIcomm.rank ↙
            ↳ in pmi._PMIComm.getMPIcpugroup():
            self.cxxclass.setPotential(self, type1, type2, potential)

    def getPotential(self, type1, type2):
        if not (pmi._PMIComm and pmi._PMIComm.isActive()) or pmi._MPIcomm.rank ↙
            ↳ in pmi._PMIComm.getMPIcpugroup():
            self.cxxclass.getPotential(self, type1, type2)

    def setFixedPairList(self, fpl):
        if not (pmi._PMIComm and pmi._PMIComm.isActive()) or pmi._MPIcomm.rank ↙
            ↳ in pmi._PMIComm.getMPIcpugroup():
            self.cxxclass.setFixedPairList(self, fpl)

    def getFixedPairList(self):
        if not (pmi._PMIComm and pmi._PMIComm.isActive()) or pmi._MPIcomm.rank ↙
            ↳ in pmi._PMIComm.getMPIcpugroup():
            return self.cxxclass.getFixedPairList(self)

if pmi.isController:
    class SmoothSquareWell(Potential):
        'The SmoothSquareWell potential.'
        pmiproxydefs = dict(
            cls = 'espressopp.interaction.SmoothSquareWellLocal',
            pmiproperty = ['epsilon', 'sigma', 'Lambda', 'a']
        )

    class VerletListSmoothSquareWell(Interaction):
        __metaclass__ = pmi.Proxy
        pmiproxydefs = dict(
            cls = 'espressopp.interaction.VerletListSmoothSquareWellLocal',
            pmicall = ['setPotential', 'getPotential', 'getVerletList']
        )

    class FixedPairListSmoothSquareWell(Interaction):
        __metaclass__ = pmi.Proxy
        pmiproxydefs = dict(
            cls =  'espressopp.interaction.FixedPairListSmoothSquareWellLocal',
            pmicall = ['setPotential', 'getPotential', ↙
                ↳ 'setFixedPairList','getFixedPairList']
        )

    class FixedPairListTypesSmoothSquareWell(Interaction):
        __metaclass__ = pmi.Proxy
        pmiproxydefs = dict(
            cls = ↙
                ↳ 'espressopp.interaction.FixedPairListTypesSmoothSquareWellLocal',
            pmicall = ['setPotential', 'getPotential', 'setFixedPairList', ↙
                ↳ 'getFixedPairList']
        )
```

Listing B.2: SmoothSquareWell.hpp

```
/*
  Copyright (C) 2017,2018
  Max Planck Institute for Polymer Research

  This file is part of ESPResSo++.

  ESPResSo++ is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  ESPResSo++ is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/

// ESPP_CLASS
#ifndef _INTERACTION_SQUAREWELL_HPP
#define _INTERACTION_SQUAREWELL_HPP

#include "Potential.hpp"
#include "VerletListInteractionTemplate.hpp"
#include "FixedPairListInteractionTemplate.hpp"
#include "FixedPairListTypesInteractionTemplate.hpp"
#include <cmath>

namespace espressopp {

  namespace interaction {
    /** This class provides methods to compute forces and energies of
        a squarewell potential based on Leitold and Dellago J. Chem. ↙
            ↳ Phys. 141. 134901 (2014)
    */
    class SmoothSquareWell: public PotentialTemplate < ↙
        ↳ SmoothSquareWell > {
    private:
      real lambda;
      real epsilon;
      real halfepsilon;
      real a;
      real sigma;
      // rb=lambda*sigma. Rightside bounary of the squarewell
      real rb;
    public:
      static void registerPython();
```

```
SmoothSquareWell(): epsilon(0.0), sigma(0.0), a(0.0), ↙
    ↳ lambda(0.0), halfepsilon(0.0) {
  setShift(0.0);
  setCutoff(infinity);
}

SmoothSquareWell(real _epsilon, real _sigma, real _cutoff, real ↙
    ↳ _shift): epsilon(_epsilon), sigma(_sigma) {
  halfepsilon = _epsilon/2;
  setShift(_shift);
  setCutoff(_cutoff);
}

SmoothSquareWell(real _epsilon, real _sigma, real _cutoff): ↙
    ↳ epsilon(_epsilon), sigma(_sigma) {
  halfepsilon = _epsilon/2;
  autoShift = false;
  setCutoff(_cutoff);
  setAutoShift();
}

virtual ~SmoothSquareWell() {};

void setEpsilon(real _epsilon) {
  epsilon = _epsilon;
  halfepsilon = _epsilon/2;
  updateAutoShift();
}

real getEpsilon() const {return epsilon;}

void setLambda(real _lambda) {
  lambda = _lambda;
  rb = lambda * sigma;
  updateAutoShift();
}

real getLambda() const {return lambda;}

void setSigma(real _sigma) {
  sigma = _sigma;
  updateAutoShift();
}

real getSigma() const {return sigma;}

void setA(real _a) {
  a = _a * sigma;
  updateAutoShift();
}

real getA() const {return a;}

real _computeEnergySqrRaw(real distSqr) const {
  real r = sqrt(distSqr);
  real energy = halfepsilon*(exp((sigma-r)/a)+tanh((r-rb)/a)-1);
  return energy;
```

```
      }

      bool _computeForceRaw(Real3D& force, const Real3D& dist, real ↙
          ↳ distSqr) const {
        real r = sqrt(distSqr);
        real ffactor = ↙
            ↳ -halfepsilon/r/a*(-exp((sigma-r)/a)+pow(cosh((r-rb)/a), ↙
            ↳ -2));
        force = dist * ffactor;
        return true;
      }
    };

    // provide pickle support
    struct SmoothSquareWell_pickle : boost::python::pickle_suite
    {
      static
      boost::python::tuple
      getinitargs(SmoothSquareWell const& pot)
      {
        real eps;
        real sig;
        real rc;
        real sh;
        eps=pot.getEpsilon();
        sig=pot.getSigma();
        rc=pot.getCutoff();
        sh=pot.getShift();
        return boost::python::make_tuple(eps, sig, rc, sh);
      }
    };

  }
}

#endif
```

Listing B.3: SmoothSquareWell.cpp

```
/*
  Copyright (C) 2017,2018
  Max Planck Institute for Polymer Research

  This file is part of ESPResSo++.

  ESPResSo++ is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  ESPResSo++ is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

```
*/

#include "python.hpp"
#include "SmoothSquareWell.hpp"
#include "VerletListInteractionTemplate.hpp"
#include "FixedPairListInteractionTemplate.hpp"
#include "FixedPairListTypesInteractionTemplate.hpp"

namespace espressopp {
  namespace interaction {

    typedef class VerletListInteractionTemplate < SmoothSquareWell > ↙
         ↳ VerletListSmoothSquareWell;
    typedef class FixedPairListInteractionTemplate< SmoothSquareWell > ↙
         ↳ FixedPairListSmoothSquareWell;
    typedef class FixedPairListTypesInteractionTemplate< ↙
         ↳ SmoothSquareWell > FixedPairListTypesSmoothSquareWell;

    //////////////////////////////////////////////////
    // REGISTRATION WITH PYTHON
    //////////////////////////////////////////////////
    void SmoothSquareWell::registerPython() {
      using namespace espressopp::python;

      class_ <SmoothSquareWell, bases <Potential> >
        ("interaction_SmoothSquareWell", init< real, real, real >())
        .def(init< real, real, real, real >())
        .add_property("epsilon", &SmoothSquareWell::getEpsilon, ↙
             ↳ &SmoothSquareWell::setEpsilon)
        .add_property("sigma", &SmoothSquareWell::getSigma, ↙
             ↳ &SmoothSquareWell::setSigma)
        .add_property("Lambda", &SmoothSquareWell::getLambda, ↙
             ↳ &SmoothSquareWell::setLambda)
        .add_property("a", &SmoothSquareWell::getA, ↙
             ↳ &SmoothSquareWell::setA)
        .def_pickle(SmoothSquareWell_pickle())
        ;

      class_ <VerletListSmoothSquareWell, bases <Interaction> >
        ("interaction_VerletListSmoothSquareWell", init< ↙
             ↳ shared_ptr<VerletList> >())
        .def("getVerletList", &VerletListSmoothSquareWell::getVerletList)
        .def("setPotential", ↙
             ↳ &VerletListSmoothSquareWell::setPotential, ↙
             ↳ return_value_policy< reference_existing_object >())
        .def("getPotential", ↙
             ↳ &VerletListSmoothSquareWell::getPotential, ↙
             ↳ return_value_policy< reference_existing_object >())
        ;

      class_ <FixedPairListSmoothSquareWell, bases <Interaction> >
        ("interaction_FixedPairListSmoothSquareWell",
         init< shared_ptr<System>, shared_ptr<FixedPairList>,  ↙
              ↳ shared_ptr<SmoothSquareWell> >())
        .def(init< shared_ptr<System>, ↙
             ↳ shared_ptr<FixedPairListAdress>, ↙
             ↳ shared_ptr<SmoothSquareWell> >())
        .def("setPotential", ↙
             ↪ &FixedPairListSmoothSquareWell::setPotential)
        .def("getPotential", ↙
             ↪ &FixedPairListSmoothSquareWell::getPotential)
        .def("setFixedPairList", ↙
             ↪ &FixedPairListSmoothSquareWell::setFixedPairList)
        .def("getFixedPairList", ↙
             ↪ &FixedPairListSmoothSquareWell::getFixedPairList)
        ;

      class_ <FixedPairListTypesSmoothSquareWell, bases <Interaction> >
        ("interaction_FixedPairListTypesSmoothSquareWell",
         init< shared_ptr<System>, shared_ptr<FixedPairList> >())
        .def(init< shared_ptr<System>, shared_ptr<FixedPairListAdress> ↙
             ↪ >())
        .def("setPotential", ↙
             ↪ &FixedPairListTypesSmoothSquareWell::setPotential)
        .def("getPotential", ↙
             ↪ &FixedPairListTypesSmoothSquareWell::getPotentialPtr)
        .def("setFixedPairList", ↙
             ↪ &FixedPairListTypesSmoothSquareWell::setFixedPairList)
        .def("getFixedPairList", ↙
             ↪ &FixedPairListTypesSmoothSquareWell::getFixedPairList)
        ;
    }
  }
}
```

C SUPPORTING INFORMATION OF SINGLE-CHAIN SMOOTH SQUARE-WELL POLYMER

The following Python script is used to set up a MD simulation of the single-chain square-well polymer with ESPResSo++.

Listing C.1: Python script used to set up the ESPResSo++ simulation of SCP

```
import time
import json
import glob
import argparse
import numpy as np
import espressopp
from espressopp import Real3D

PARSER = argparse.ArgumentParser()
PARSER.add_argument("--warm", help="Warm up the simulation", ↙
    ↪ action="store_true", default=False)
PARSER.add_argument("--restart", help="Continue the prior simulation", ↙
    ↪ action="store_true", default=False)
PARSER.add_argument("--jobid", help="Append file name of SystemMonitorCSV with ↙
    ↪ the job ID", action="store")
ARGS = PARSER.parse_args()
WARM            = ARGS.warm     #
RESTART         = ARGS.restart  #
JOBID           = ARGS.jobid    #
RESTARTNUM      = 5             #
CUTOFF          = 1.5           #
SKIN            = 0.3           #
TOPOFILE        = "scp.conf"    #
STEPSIZE        = 0.0002        #
RUNSTEPS        = 22000000      #
RESTARTFREQ     = RUNSTEPS/RESTARTNUM
DUMPFREQ        = 500           #
WARMSTEPS       = 10000         #
WARMFREQ        = 5000          #
SEED            = 9487          #
TEMPERATURE     = 0.438         #
GAMMA           = 0.5           #
BOX             = []            #
PARTICLES       = []            #
BONDS           = []            #
PROPS           = ['id', 'type', 'mass', 'pos', 'v']
EPSILON         = 1.0
SIGMA           = 1.0
vel_zero        = Real3D(0.0, 0.0, 0.0)


def warmups():
    print "Warming up starts"
    for i in range(WARMSTEPS/WARMFREQ):
        espressopp.tools.analyse.info(system, integrator)
        integrator.run(WARMFREQ)
    espressopp.tools.analyse.info(system, integrator)
    print "Warming up ends"
    return

def runs():
    dump_conf_xyz = espressopp.io.DumpXYZ(system, integrator, ↙
        ↳ filename='scp.xyz', append=True)
    ext_dump = espressopp.integrator.ExtAnalyze(dump_conf_xyz, DUMPFREQ)
    integrator.addExtension(ext_dump)
    system_monitor_csv = ↙
        ↳ espressopp.analysis.SystemMonitorOutputCSV("thermo."+str(JOBID)+".csv")
    system_monitor = espressopp.analysis.SystemMonitor(system, integrator, ↙
        ↳ system_monitor_csv)
    system_monitor.add_observable('nonBond', ↙
        ↳ espressopp.analysis.PotentialEnergy(system, nonbondedInter))
    system_monitor.add_observable('Bond', ↙
        ↳ espressopp.analysis.PotentialEnergy(system, bondedInteraction))
    system_monitor.add_observable('temp', espressopp.analysis.Temperature(system))
    ext_analysis = espressopp.integrator.ExtAnalyze(system_monitor, DUMPFREQ)
    integrator.addExtension(ext_analysis)
    print "The run starts at", time.ctime()
    start_time = time.clock()
    integrator.run(RUNSTEPS)
    end_time = time.clock()
    espressopp.tools.analyse.final_info(system, integrator, verletList, ↙
        ↳ start_time, end_time)
    print "The run ends at", time.ctime()
    print "save files needed for restart"
    writerestart(integrator.step)
    system.rng.saveState(integrator.step)
    return


def readtopo():
    global BOX
    global PARTICLES
    global BONDS
    Lx, Ly, Lz, pids, ptypes, poss, vels, bonds, angels, dihedrals = ↙
        ↳ espressopp.tools.io_extended.read(TOPOFILE, readVelocities=True)
    BOX = [Lx, Ly, Lz]
    for i, pid in enumerate(pids):
        part = [pid, ptypes[i], 1.0, poss[i], vels[i]]
        PARTICLES.append(part)
    BONDS = [x[1] for x in bonds]
    return


def readrestart():
    global PARTICLES
    global BOX
    global BONDS
    restartfiles = glob.glob("./restart.*")
    steps = [int(f.rsplit(".", 1)[1]) for f in restartfiles]
    finalrestart = restartfiles[steps.index(max(steps))]
    with open(finalrestart, 'r') as F:
        restart = json.load(F)
    step = restart["step"]
    BOX = restart["box"]
    npart = restart["particles"]["number"]
    for i in range(npart):
        partList = []
        partList.append(restart["particles"][str(i+1)]["id"])
        partList.append(restart["particles"][str(i+1)]["type"])
        partList.append(restart["particles"][str(i+1)]["mass"])
        partList.append(Real3D(restart["particles"][str(i+1)]["pos"]))
        partList.append(Real3D(restart["particles"][str(i+1)]["v"]))
        partList.append(Real3D(restart["particles"][str(i+1)]["f"]))
        PARTICLES.append(partList)
    BONDS = restart["bonds"]
    return step


def writerestart(step):
    restart = {}
    restart["step"] = step
    restart["box"] = list(system.bc.boxL)
    maxid = int(espressopp.analysis.MaxPID(system).compute())
    restart["particles"] = {"number" : maxid}
    for i in range(maxid):
        p = system.storage.getParticle(i+1)
        restart["particles"].update({i+1:{"id":p.id, "type":p.type, ⤦
             ↪ "mass":p.mass, "pos":list(p.pos), "v":list(p.v), "f":list(p.f)}})
    restart["bonds"] = BONDS
    restartfile = "restart." + str(step)
    with open(restartfile, 'w') as fp:
        json.dump(restart, fp)
    return


if RESTART:
    STEP = readrestart()
else:
    STEP = 0
    readtopo()

system = espressopp.System()
system.rng = espressopp.esutil.RNG(SEED)
if RESTART:
    system.rng.loadState("rng."+str(STEP))
system.bc = espressopp.bc.OrthorhombicBC(system.rng, BOX)
system.skin = SKIN
nodeGrid = espressopp.tools.decomp.nodeGrid(espressopp.MPI.COMM_WORLD.size)
cellGrid = espressopp.tools.decomp.cellGrid(BOX, nodeGrid, CUTOFF, SKIN)
system.storage = espressopp.storage.DomainDecomposition(system, nodeGrid, ⤦
     ↪ cellGrid)
if RESTART:
    PROPS.append("f")
system.storage.addParticles(PARTICLES, *PROPS)
system.storage.decompose()

integrator = espressopp.integrator.VelocityVerlet(system)
integrator.step = STEP
integrator.dt = STEPSIZE
integrator.recalcForces = not RESTART
verletList = espressopp.VerletList(system, cutoff=CUTOFF)
verletList.exclude(BONDS)

if not WARM:
    plumed = espressopp.integrator.ExtPlumed(system, "plumed.dat", ⤦
         ↪ log="log.plumed", dt=integrator.dt, restart=RESTART)
    integrator.addExtension(plumed)

# thermostat
thermostat = espressopp.integrator.LangevinThermostat(system)
thermostat.gamma = GAMMA
thermostat.temperature = TEMPERATURE
integrator.addExtension(thermostat)

# non-bonded interactions
pot = ⤦
     ↪ espressopp.interaction.SmoothSquareWell(epsilon=EPSILON,sigma=SIGMA,cutoff=CUTOFF)
pot.a = 0.002
pot.Lambda = 1.05
```

```
nonbondedInter = espressopp.interaction.VerletListSmoothSquareWell(verletList)
nonbondedInter.setPotential(type1=1, type2=1, potential=pot)
system.addInteraction(nonbondedInter)

# bonded interactions
bondPairList = espressopp.FixedPairList(system.storage)
bondPairList.addBonds(BONDS)
bondedPot = espressopp.interaction.Harmonic(K=10000, r0=1.0)
bondedInteraction = espressopp.interaction.FixedPairListHarmonic(system, ↙
     ↳ bondPairList, bondedPot)
system.addInteraction(bondedInteraction)

if WARM:
    warmups()
    espressopp.tools.io_extended.write(TOPOFILE, system, writeVelocities=True)
else:
    runs()
```

Listing C.2: plumed.dat file for well-tempered metadynamics simulation of SCP

```
# vim:ft=plumed
UNITS NATURAL
FLUSH STRIDE=5000
ene: ENERGY
UPPER_WALLS ARG=ene AT=5.0 KAPPA=2.5 LABEL=uwall
Q6 SPECIES=1-128 SWITCH={RATIONAL R_0=1.0 NN=12 D_MAX=1.85} VMEAN ↙
     ↳ LABEL=q6
COMBINE ...
LABEL=rxc
PERIODIC=NO
ARG=ene,q6.vmean
COEFFICIENTS=-0.0187482044056,4.07430109214
... COMBINE
METAD ...
LABEL=metad
ARG=rxc
PACE=2000
HEIGHT=0.438
TEMP=0.438
BIASFACTOR=30
SIGMA=0.1
GRID_MIN=-1. GRID_MAX=20.0 GRID_BIN=1200
CALC_RCT
RCT_USTRIDE=1
FILE=HILLS
... METAD
PRINT ARG=ene,q6.vmean,rxc,uwall.bias,metad.* STRIDE=500 FILE=COLVAR
```

Listing C.3: R script for the standard error of T_{co}

```
# reference: https://stats.stackexchange.com/a/486563
mydata <- read.table("crys.txt", header=TRUE)
fit <- lm(temperature ~ deltag, data=mydata)
fit$coefficients
X <- model.matrix(fit)
S <- solve(t(X) %*% X)
RMS <- summary(fit)$sigma^2
x_h <- matrix(c(1, 0), ncol = 1)
```

```
y_h_se <- sqrt(RMS * (t(x_h) %*% S %*% x_h)); y_h_se
```

Listing C.4: crys.txt

```
"temperature" "deltag"
0.418   -10.01413485
0.438   0.186465697
0.458   15.84778002
```

D SUPPORTING INFORMATION FOR CHAPTER 4

D.1 SEVEN LENNARD-JONES PARTICLES

The following file is the plumed.dat file used for VES-PICV simulations of the seven Lennard Jones particles.

Listing D.1: plumed.dat file for the VES-PICV simulation of 7 Lennard-Jones Particles

```
# vim:ft=plumed

UNITS NATURAL
FLUSH STRIDE=500

energy: ENERGY

com: COM ATOMS=1-7
d1: DISTANCE ATOMS=1,com
d2: DISTANCE ATOMS=2,com
d3: DISTANCE ATOMS=3,com
d4: DISTANCE ATOMS=4,com
d5: DISTANCE ATOMS=5,com
d6: DISTANCE ATOMS=6,com
d7: DISTANCE ATOMS=7,com

uw_d1: UPPER_WALLS ARG=d1 AT=2.0 KAPPA=1000.
uw_d2: UPPER_WALLS ARG=d2 AT=2.0 KAPPA=1000.
uw_d3: UPPER_WALLS ARG=d3 AT=2.0 KAPPA=1000.
uw_d4: UPPER_WALLS ARG=d4 AT=2.0 KAPPA=1000.
uw_d5: UPPER_WALLS ARG=d5 AT=2.0 KAPPA=1000.
uw_d6: UPPER_WALLS ARG=d6 AT=2.0 KAPPA=1000.
uw_d7: UPPER_WALLS ARG=d7 AT=2.0 KAPPA=1000.

c1: COORDINATION GROUPA=1 GROUPB=2,3,4,5,6,7 SWITCH={RATIONAL R_0=1.5 ↙
    ↳ NN=8 MM=16}
c2: COORDINATION GROUPA=2 GROUPB=1,3,4,5,6,7 SWITCH={RATIONAL R_0=1.5 ↙
    ↳ NN=8 MM=16}
c3: COORDINATION GROUPA=3 GROUPB=1,2,4,5,6,7 SWITCH={RATIONAL R_0=1.5 ↙
    ↳ NN=8 MM=16}
c4: COORDINATION GROUPA=4 GROUPB=1,2,3,5,6,7 SWITCH={RATIONAL R_0=1.5 ↙
    ↳ NN=8 MM=16}
c5: COORDINATION GROUPA=5 GROUPB=1,2,3,4,6,7 SWITCH={RATIONAL R_0=1.5 ↙
    ↳ NN=8 MM=16}
c6: COORDINATION GROUPA=6 GROUPB=1,2,3,4,5,7 SWITCH={RATIONAL R_0=1.5 ↙
    ↳ NN=8 MM=16}
c7: COORDINATION GROUPA=7 GROUPB=1,2,3,4,5,6 SWITCH={RATIONAL R_0=1.5 ↙
    ↳ NN=8 MM=16}
```

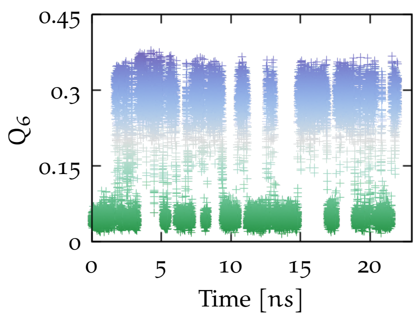
```
cn: COORDINATIONNUMBER SPECIES=1-7 MEAN MOMENTS=2-4 SWITCH={RATIONAL R_0=1.5 NN=8 MM=16}

lw_cn_m2: LOWER_WALLS ARG=cn.moment-2 AT=0.4  KAPPA=1000.
uw_cn_m2: UPPER_WALLS ARG=cn.moment-2 AT=1.2  KAPPA=1000.
lw_cn_m3: LOWER_WALLS ARG=cn.moment-3 AT=-0.3 KAPPA=1000.

bf1: BF_LEGENDRE ORDER=20 MINIMUM=1.0 MAXIMUM=6.0

td1: TD_WELLTEMPERED BIASFACTOR=3

VES_LINEAR_EXPANSION_MULTIPLE_IDENTICAL_CVS ...
 ARG_SET1=c1
 ARG_SET2=c2
 ARG_SET3=c3
 ARG_SET4=c4
 ARG_SET5=c5
 ARG_SET6=c6
 ARG_SET7=c7
 BASIS_FUNCTIONS=bf1
 TARGET_DISTRIBUTION=td1
 LABEL=ves1
 TEMP=0.2
 GRID_BINS=100
... VES_LINEAR_EXPANSION_MULTIPLE_IDENTICAL_CVS

OPT_AVERAGED_SGD ...
  BIAS=ves1
  STRIDE=500
  LABEL=o1
  STEPSIZE=0.1
  COEFFS_FILE=coeffs.data
  COEFFS_OUTPUT=100
  FES_OUTPUT=500
  BIAS_OUTPUT=500
  TARGETDIST_STRIDE=500
... OPT_AVERAGED_SGD


PRINT ...
ARG=energy,cn.*,ves1.*
STRIDE=500
FILE=colvar.data
... PRINT

PRINT ...
ARG=energy,c1,c2,c3,c4,c5,c6,c7,ves1.*
STRIDE=500
FILE=colvar.cn-particles.data
... PRINT

PRINT ...
ARG=cn.moment-2,cn.moment-3,lw_cn_m2.bias,uw_cn_m2.bias,lw_cn_m3.bias
STRIDE=500
FILE=colvar.cn-walls.data
... PRINT
```

D.2 BULK SODIUM

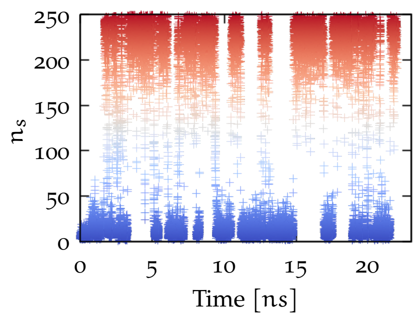
The plumed.dat file used for the VES-PICV simulations of bulk sodium is shown below.

Listing D.2: plumed.dat file for the VES-PICV simulation of bulk sodium

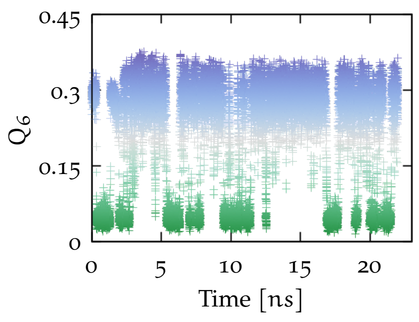
```
# vim:ft=plumed
UNITS LENGTH=A
FLUSH STRIDE=10000
ene: ENERGY

cmat: CONTACT_MATRIX GROUP=1-250 SWITCH={RATIONAL R_0=5 NN=12 ↙
    ↳ D_MAX=8.6} COMPONENTS

ee: ENVIRONMENT ...
    WEIGHT=cmat.w VECTORS1=cmat.x VECTORS2=cmat.y VECTORS3=cmat.z
    SIGMA=0.65 REFERENCE=ref2.pdb
...

q6: Q6 SPECIES=1-250 SWITCH={RATIONAL R_0=5 NN=12 D_MAX=8.6} VMEAN

bf1: BF_LEGENDRE ORDER=20 MINIMUM=0.1 MAXIMUM=0.85

td1: TD_WELLTEMPERED BIASFACTOR=1.2

VES_LINEAR_EXPANSION ...
    ARG=ee
    BASIS_FUNCTIONS=bf1
    TARGET_DISTRIBUTION=td1
    LABEL=ves1
    TEMP=380
    GRID_BINS=100
... VES_LINEAR_EXPANSION

OPT_ADAM ...
    BIAS=ves1
    STRIDE=500
    LABEL=o1
    STEPSIZE=0.0005
    COEFFS_FILE=coeffs.data
    COEFFS_OUTPUT=50
    FES_OUTPUT=500
    BIAS_OUTPUT=500
    TARGETDIST_STRIDE=500
    MULTIPLE_WALKERS
... OPT_ADAM

PRINT ARG=ene,q6_vmean,ves1.* STRIDE=500 FILE=colvar-en.data
PRINT ARG=ee STRIDE=500 FILE=colvar-ee.dat
DUMPATOMS STRIDE=500 FILE=run.xyz ATOMS=1-250
```



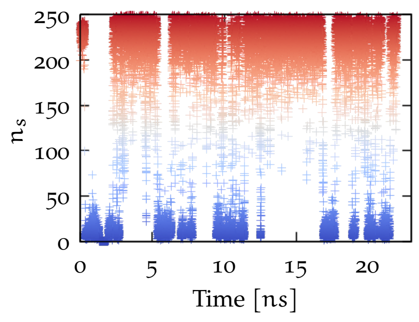
(a) walker 2



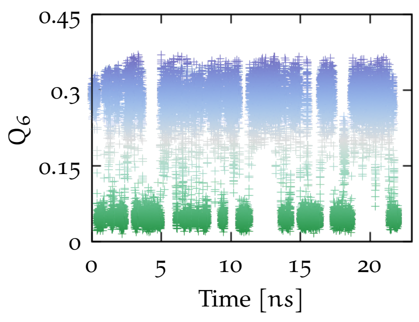
(b) walker 2



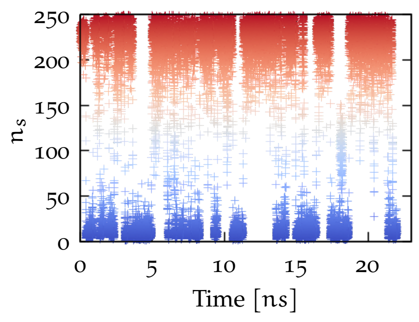
(c) walker 3



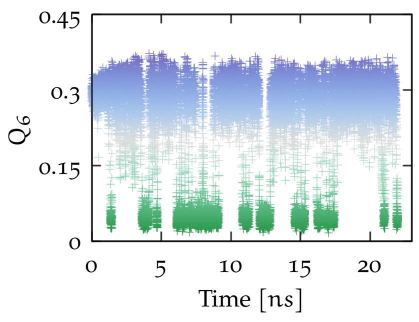
(d) walker 3



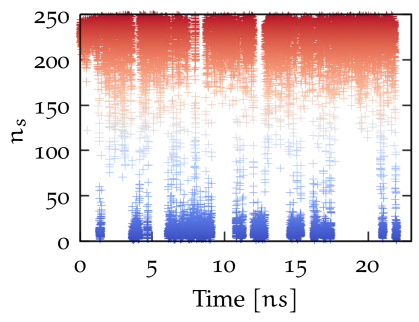
(e) walker 4



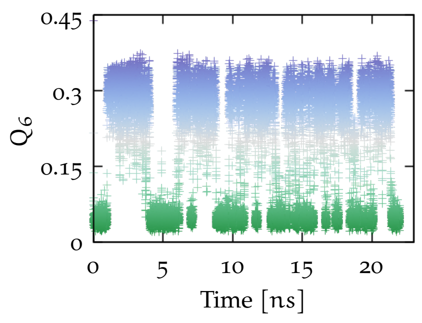
(f) walker 4



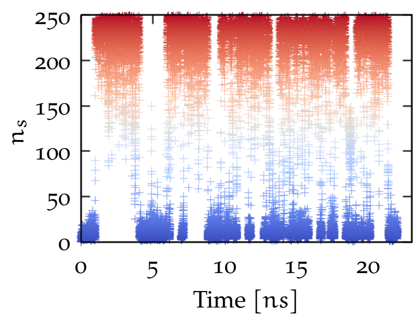
(g) walker 5



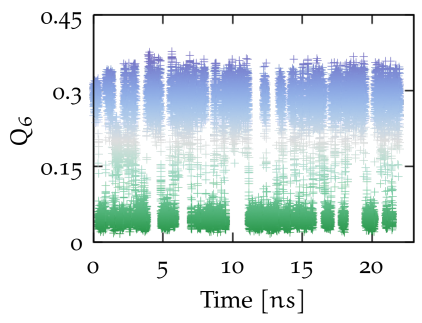
(h) walker 5



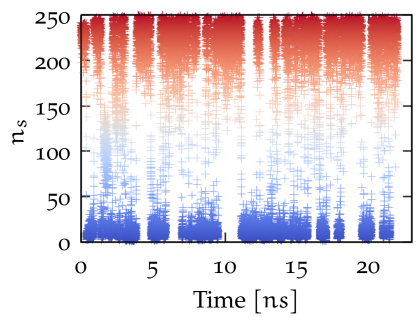
(i) walker 6



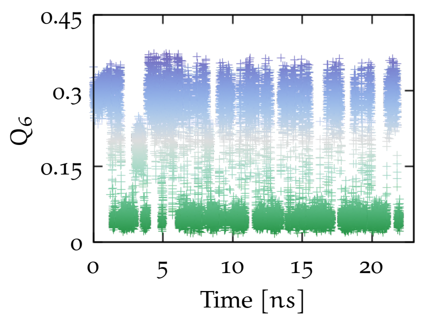
(j) walker 6



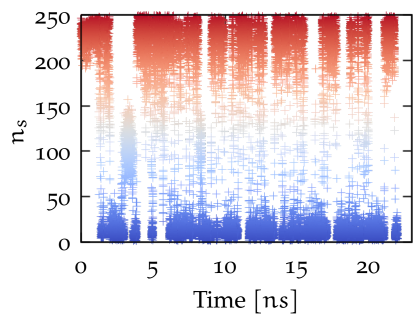
(k) walker 7



(l) walker 7



(m) walker 8



(n) walker 8

Figure D.1: The time series of the number of solid atoms n_s and Q_6 of walkers 2-8.

E INPUT FILES FOR PLUMED IN SIMULATIONS OF ICE AND UREA

E.1 ICE IH

One of the plumed.dat files used in the multiple-walker VES-PICV simulations of ice nucleation is listed below.

Listing E.1: plumed.dat file for the VES-PICV simulations of ice Ih

```
# vim:ft=plumed

vol: VOLUME
Q6 SPECIES=1-864:3 SWITCH={RATIONAL R_0=0.3 D_MAX=0.35} VMEAN LABEL=q6

cmat: CONTACT_MATRIX GROUP=1-864:3 SWITCH={RATIONAL R_0=1.000 NN=12 D_MAX=1.500} COMPONENTS

ENVIRONMENT ...
    LABEL=refcv1
    WEIGHT=cmat.w VECTORS1=cmat.x VECTORS2=cmat.y VECTORS3=cmat.z
    REFERENCE=./Environments/IceIhExtendedEnvironments/env1.pdb
    SIGMA=0.055
... ENVIRONMENT

ENVIRONMENT ...
    LABEL=refcv2
    WEIGHT=cmat.w VECTORS1=cmat.x VECTORS2=cmat.y VECTORS3=cmat.z
    REFERENCE=./Environments/IceIhExtendedEnvironments/env2.pdb
    SIGMA=0.055
... ENVIRONMENT

ENVIRONMENT ...
    LABEL=refcv3
    WEIGHT=cmat.w VECTORS1=cmat.x VECTORS2=cmat.y VECTORS3=cmat.z
    REFERENCE=./Environments/IceIhExtendedEnvironments/env3.pdb
    SIGMA=0.055
... ENVIRONMENT

ENVIRONMENT ...
    LABEL=refcv4
    WEIGHT=cmat.w VECTORS1=cmat.x VECTORS2=cmat.y VECTORS3=cmat.z
    REFERENCE=./Environments/IceIhExtendedEnvironments/env4.pdb
    SIGMA=0.055
... ENVIRONMENT

ENVIRONMENT ...
    LABEL=refcv5
    WEIGHT=cmat.w VECTORS1=cmat.x VECTORS2=cmat.y VECTORS3=cmat.z
    REFERENCE=./Environments/IceIcExtendedEnvironments/env1.pdb
    SIGMA=0.055
... ENVIRONMENT

ENVIRONMENT ...
    LABEL=refcv6
    WEIGHT=cmat.w VECTORS1=cmat.x VECTORS2=cmat.y VECTORS3=cmat.z
    REFERENCE=./Environments/IceIcExtendedEnvironments/env2.pdb
    SIGMA=0.055
... ENVIRONMENT

CUSTOM ...
    ARG1=refcv1
    ARG2=refcv2
    ARG3=refcv3
    ARG4=refcv4
    VAR=k1,k2,k3,k4
    FUNC=(0.01*log(exp(100*k1)+exp(100*k2)+exp(100*k3)+exp(100*k4)))
    PERIODIC=NO
    LABEL=kx1
... CUSTOM

CUSTOM ...
    ARG1=refcv4
    ARG2=refcv5
    VAR=k1,k2
    FUNC=(0.01*log(exp(100*k1)+exp(100*k2)))
    PERIODIC=NO
    LABEL=kx2
... CUSTOM

MORE_THAN ARG=kx1 R_0=0.5 NN=12 LABEL=si1
s1morethan: COMBINE ARG=si1 PERIODIC=NO
s1mean: COMBINE ARG=kx1 PERIODIC=NO NORMALIZE

MORE_THAN ARG=kx2 R_0=0.39 NN=12 LABEL=si2
s2morethan: COMBINE ARG=si2 PERIODIC=NO
s2mean: COMBINE ARG=kx2 PERIODIC=NO NORMALIZE

diff1: CUSTOM ARG1=s2mean ARG2=s1mean ↙
     ↳ FUNC=((x-0.26)/(0.58-0.26)-(y-0.29)/(0.80-0.29)) PERIODIC=NO
UPPER_WALLS ARG=diff1 AT=0.04 KAPPA=100000 EXP=2 LABEL=uwall1

diff2: CUSTOM ARG1=q6_vmean ARG2=s1mean ↙
     ↳ FUNC=((x-0.0668781995)/(0.39184059-0.0668781995) - ↙
     ↳ (y-0.2899390548628429)/(0.7838534089775562-0.2899390548628429)) ↙
     ↳ PERIODIC=NO
UPPER_WALLS ARG=diff2 AT=0.06 KAPPA=50000 EXP=2 LABEL=uwall2

# basis functions and the target distribution
bf1: BF_LEGENDRE ORDER=20 MINIMUM=0.15 MAXIMUM=0.95
td1: TD_WELLTEMPERED BIASFACTOR=2

VES_LINEAR_EXPANSION ...
  LABEL=ves1
  ARG=kx1
  BASIS_FUNCTIONS=bf1
  TARGET_DISTRIBUTION=td1
  TEMP=280
  GRID_BINS=100
... VES_LINEAR_EXPANSION

OPT_AVERAGED_SGD ...
  BIAS=ves1
  STRIDE=500
  LABEL=o1
  STEPSIZE=2.0
  FES_OUTPUT=500
  BIAS_OUTPUT=500
  COEFFS_FILE=coeffs.data
  COEFFS_OUTPUT=25
  TARGETDIST_STRIDE=250
  MULTIPLE_WALKERS
... OPT_AVERAGED_SGD
```

```
PRINT ARG=kx1 STRIDE=500 FILE=colvar-kx1.dat
PRINT ARG=kx2 STRIDE=500 FILE=colvar-kx2.dat
PRINT ↙
     ↳ ARG=vol,q6_vmean,s1mean,s1morethan,s2mean,s2morethan,diff1,uwall1.*,diff2,uwall2.*,ves1.* ↙
     ↳ STRIDE=500 FILE=colvar-ves.dat
```

E.2 UREA

The following file is one of the plumed.dat files used in the VES-PICV simulations of urea.

Listing E.2: plumed.dat file for the VES-PICV simulations of urea

```
# vim:ft=plumed

vol: VOLUME
ene: ENERGY

NC1: CENTER ATOMS=3,6 MASS
NC2: CENTER ATOMS=11,14 MASS
NC3: CENTER ATOMS=19,22 MASS
NC4: CENTER ATOMS=27,30 MASS
NC5: CENTER ATOMS=35,38 MASS
NC6: CENTER ATOMS=43,46 MASS
NC7: CENTER ATOMS=51,54 MASS
NC8: CENTER ATOMS=59,62 MASS
NC9: CENTER ATOMS=67,70 MASS
NC10: CENTER ATOMS=75,78 MASS
NC11: CENTER ATOMS=83,86 MASS
NC12: CENTER ATOMS=91,94 MASS
NC13: CENTER ATOMS=99,102 MASS
NC14: CENTER ATOMS=107,110 MASS
NC15: CENTER ATOMS=115,118 MASS
NC16: CENTER ATOMS=123,126 MASS
NC17: CENTER ATOMS=131,134 MASS
NC18: CENTER ATOMS=139,142 MASS
NC19: CENTER ATOMS=147,150 MASS
NC20: CENTER ATOMS=155,158 MASS
NC21: CENTER ATOMS=163,166 MASS
NC22: CENTER ATOMS=171,174 MASS
NC23: CENTER ATOMS=179,182 MASS
NC24: CENTER ATOMS=187,190 MASS
NC25: CENTER ATOMS=195,198 MASS
NC26: CENTER ATOMS=203,206 MASS
NC27: CENTER ATOMS=211,214 MASS
NC28: CENTER ATOMS=219,222 MASS
NC29: CENTER ATOMS=227,230 MASS
NC30: CENTER ATOMS=235,238 MASS
NC31: CENTER ATOMS=243,246 MASS
NC32: CENTER ATOMS=251,254 MASS
NC33: CENTER ATOMS=259,262 MASS
NC34: CENTER ATOMS=267,270 MASS
NC35: CENTER ATOMS=275,278 MASS
NC36: CENTER ATOMS=283,286 MASS
NC37: CENTER ATOMS=291,294 MASS
NC38: CENTER ATOMS=299,302 MASS
NC39: CENTER ATOMS=307,310 MASS
NC40: CENTER ATOMS=315,318 MASS
NC41: CENTER ATOMS=323,326 MASS
NC42: CENTER ATOMS=331,334 MASS
NC43: CENTER ATOMS=339,342 MASS
```

```
NC44: CENTER ATOMS=347,350 MASS
NC45: CENTER ATOMS=355,358 MASS
NC46: CENTER ATOMS=363,366 MASS
NC47: CENTER ATOMS=371,374 MASS
NC48: CENTER ATOMS=379,382 MASS
NC49: CENTER ATOMS=387,390 MASS
NC50: CENTER ATOMS=395,398 MASS
NC51: CENTER ATOMS=403,406 MASS
NC52: CENTER ATOMS=411,414 MASS
NC53: CENTER ATOMS=419,422 MASS
NC54: CENTER ATOMS=427,430 MASS
NC55: CENTER ATOMS=435,438 MASS
NC56: CENTER ATOMS=443,446 MASS
NC57: CENTER ATOMS=451,454 MASS
NC58: CENTER ATOMS=459,462 MASS
NC59: CENTER ATOMS=467,470 MASS
NC60: CENTER ATOMS=475,478 MASS
NC61: CENTER ATOMS=483,486 MASS
NC62: CENTER ATOMS=491,494 MASS
NC63: CENTER ATOMS=499,502 MASS
NC64: CENTER ATOMS=507,510 MASS
NC65: CENTER ATOMS=515,518 MASS
NC66: CENTER ATOMS=523,526 MASS
NC67: CENTER ATOMS=531,534 MASS
NC68: CENTER ATOMS=539,542 MASS
NC69: CENTER ATOMS=547,550 MASS
NC70: CENTER ATOMS=555,558 MASS
NC71: CENTER ATOMS=563,566 MASS
NC72: CENTER ATOMS=571,574 MASS
NC73: CENTER ATOMS=579,582 MASS
NC74: CENTER ATOMS=587,590 MASS
NC75: CENTER ATOMS=595,598 MASS
NC76: CENTER ATOMS=603,606 MASS
NC77: CENTER ATOMS=611,614 MASS
NC78: CENTER ATOMS=619,622 MASS
NC79: CENTER ATOMS=627,630 MASS
NC80: CENTER ATOMS=635,638 MASS
NC81: CENTER ATOMS=643,646 MASS
NC82: CENTER ATOMS=651,654 MASS
NC83: CENTER ATOMS=659,662 MASS
NC84: CENTER ATOMS=667,670 MASS
NC85: CENTER ATOMS=675,678 MASS
NC86: CENTER ATOMS=683,686 MASS
NC87: CENTER ATOMS=691,694 MASS
NC88: CENTER ATOMS=699,702 MASS
NC89: CENTER ATOMS=707,710 MASS
NC90: CENTER ATOMS=715,718 MASS
NC91: CENTER ATOMS=723,726 MASS
NC92: CENTER ATOMS=731,734 MASS
NC93: CENTER ATOMS=739,742 MASS
NC94: CENTER ATOMS=747,750 MASS
NC95: CENTER ATOMS=755,758 MASS
NC96: CENTER ATOMS=763,766 MASS
NC97: CENTER ATOMS=771,774 MASS
NC98: CENTER ATOMS=779,782 MASS
NC99: CENTER ATOMS=787,790 MASS
NC100: CENTER ATOMS=795,798 MASS
NC101: CENTER ATOMS=803,806 MASS
NC102: CENTER ATOMS=811,814 MASS
NC103: CENTER ATOMS=819,822 MASS
NC104: CENTER ATOMS=827,830 MASS
NC105: CENTER ATOMS=835,838 MASS
NC106: CENTER ATOMS=843,846 MASS
NC107: CENTER ATOMS=851,854 MASS
NC108: CENTER ATOMS=859,862 MASS
NC109: CENTER ATOMS=867,870 MASS
```

```
NC110: CENTER ATOMS=875,878 MASS
NC111: CENTER ATOMS=883,886 MASS
NC112: CENTER ATOMS=891,894 MASS
NC113: CENTER ATOMS=899,902 MASS
NC114: CENTER ATOMS=907,910 MASS
NC115: CENTER ATOMS=915,918 MASS
NC116: CENTER ATOMS=923,926 MASS
NC117: CENTER ATOMS=931,934 MASS
NC118: CENTER ATOMS=939,942 MASS
NC119: CENTER ATOMS=947,950 MASS
NC120: CENTER ATOMS=955,958 MASS
NC121: CENTER ATOMS=963,966 MASS
NC122: CENTER ATOMS=971,974 MASS
NC123: CENTER ATOMS=979,982 MASS
NC124: CENTER ATOMS=987,990 MASS
NC125: CENTER ATOMS=995,998 MASS
NC126: CENTER ATOMS=1003,1006 MASS
NC127: CENTER ATOMS=1011,1014 MASS
NC128: CENTER ATOMS=1019,1022 MASS

CO1: CENTER ATOMS=1,2 MASS
CO2: CENTER ATOMS=9,10 MASS
CO3: CENTER ATOMS=17,18 MASS
CO4: CENTER ATOMS=25,26 MASS
CO5: CENTER ATOMS=33,34 MASS
CO6: CENTER ATOMS=41,42 MASS
CO7: CENTER ATOMS=49,50 MASS
CO8: CENTER ATOMS=57,58 MASS
CO9: CENTER ATOMS=65,66 MASS
CO10: CENTER ATOMS=73,74 MASS
CO11: CENTER ATOMS=81,82 MASS
CO12: CENTER ATOMS=89,90 MASS
CO13: CENTER ATOMS=97,98 MASS
CO14: CENTER ATOMS=105,106 MASS
CO15: CENTER ATOMS=113,114 MASS
CO16: CENTER ATOMS=121,122 MASS
CO17: CENTER ATOMS=129,130 MASS
CO18: CENTER ATOMS=137,138 MASS
CO19: CENTER ATOMS=145,146 MASS
CO20: CENTER ATOMS=153,154 MASS
CO21: CENTER ATOMS=161,162 MASS
CO22: CENTER ATOMS=169,170 MASS
CO23: CENTER ATOMS=177,178 MASS
CO24: CENTER ATOMS=185,186 MASS
CO25: CENTER ATOMS=193,194 MASS
CO26: CENTER ATOMS=201,202 MASS
CO27: CENTER ATOMS=209,210 MASS
CO28: CENTER ATOMS=217,218 MASS
CO29: CENTER ATOMS=225,226 MASS
CO30: CENTER ATOMS=233,234 MASS
CO31: CENTER ATOMS=241,242 MASS
CO32: CENTER ATOMS=249,250 MASS
CO33: CENTER ATOMS=257,258 MASS
CO34: CENTER ATOMS=265,266 MASS
CO35: CENTER ATOMS=273,274 MASS
CO36: CENTER ATOMS=281,282 MASS
CO37: CENTER ATOMS=289,290 MASS
CO38: CENTER ATOMS=297,298 MASS
CO39: CENTER ATOMS=305,306 MASS
CO40: CENTER ATOMS=313,314 MASS
CO41: CENTER ATOMS=321,322 MASS
CO42: CENTER ATOMS=329,330 MASS
CO43: CENTER ATOMS=337,338 MASS
CO44: CENTER ATOMS=345,346 MASS
CO45: CENTER ATOMS=353,354 MASS
CO46: CENTER ATOMS=361,362 MASS
```

```
CO47: CENTER ATOMS=369,370 MASS
CO48: CENTER ATOMS=377,378 MASS
CO49: CENTER ATOMS=385,386 MASS
CO50: CENTER ATOMS=393,394 MASS
CO51: CENTER ATOMS=401,402 MASS
CO52: CENTER ATOMS=409,410 MASS
CO53: CENTER ATOMS=417,418 MASS
CO54: CENTER ATOMS=425,426 MASS
CO55: CENTER ATOMS=433,434 MASS
CO56: CENTER ATOMS=441,442 MASS
CO57: CENTER ATOMS=449,450 MASS
CO58: CENTER ATOMS=457,458 MASS
CO59: CENTER ATOMS=465,466 MASS
CO60: CENTER ATOMS=473,474 MASS
CO61: CENTER ATOMS=481,482 MASS
CO62: CENTER ATOMS=489,490 MASS
CO63: CENTER ATOMS=497,498 MASS
CO64: CENTER ATOMS=505,506 MASS
CO65: CENTER ATOMS=513,514 MASS
CO66: CENTER ATOMS=521,522 MASS
CO67: CENTER ATOMS=529,530 MASS
CO68: CENTER ATOMS=537,538 MASS
CO69: CENTER ATOMS=545,546 MASS
CO70: CENTER ATOMS=553,554 MASS
CO71: CENTER ATOMS=561,562 MASS
CO72: CENTER ATOMS=569,570 MASS
CO73: CENTER ATOMS=577,578 MASS
CO74: CENTER ATOMS=585,586 MASS
CO75: CENTER ATOMS=593,594 MASS
CO76: CENTER ATOMS=601,602 MASS
CO77: CENTER ATOMS=609,610 MASS
CO78: CENTER ATOMS=617,618 MASS
CO79: CENTER ATOMS=625,626 MASS
CO80: CENTER ATOMS=633,634 MASS
CO81: CENTER ATOMS=641,642 MASS
CO82: CENTER ATOMS=649,650 MASS
CO83: CENTER ATOMS=657,658 MASS
CO84: CENTER ATOMS=665,666 MASS
CO85: CENTER ATOMS=673,674 MASS
CO86: CENTER ATOMS=681,682 MASS
CO87: CENTER ATOMS=689,690 MASS
CO88: CENTER ATOMS=697,698 MASS
CO89: CENTER ATOMS=705,706 MASS
CO90: CENTER ATOMS=713,714 MASS
CO91: CENTER ATOMS=721,722 MASS
CO92: CENTER ATOMS=729,730 MASS
CO93: CENTER ATOMS=737,738 MASS
CO94: CENTER ATOMS=745,746 MASS
CO95: CENTER ATOMS=753,754 MASS
CO96: CENTER ATOMS=761,762 MASS
CO97: CENTER ATOMS=769,770 MASS
CO98: CENTER ATOMS=777,778 MASS
CO99: CENTER ATOMS=785,786 MASS
CO100: CENTER ATOMS=793,794 MASS
CO101: CENTER ATOMS=801,802 MASS
CO102: CENTER ATOMS=809,810 MASS
CO103: CENTER ATOMS=817,818 MASS
CO104: CENTER ATOMS=825,826 MASS
CO105: CENTER ATOMS=833,834 MASS
CO106: CENTER ATOMS=841,842 MASS
CO107: CENTER ATOMS=849,850 MASS
CO108: CENTER ATOMS=857,858 MASS
CO109: CENTER ATOMS=865,866 MASS
CO110: CENTER ATOMS=873,874 MASS
CO111: CENTER ATOMS=881,882 MASS
CO112: CENTER ATOMS=889,890 MASS
```

```
CO113: CENTER ATOMS=897,898 MASS
CO114: CENTER ATOMS=905,906 MASS
CO115: CENTER ATOMS=913,914 MASS
CO116: CENTER ATOMS=921,922 MASS
CO117: CENTER ATOMS=929,930 MASS
CO118: CENTER ATOMS=937,938 MASS
CO119: CENTER ATOMS=945,946 MASS
CO120: CENTER ATOMS=953,954 MASS
CO121: CENTER ATOMS=961,962 MASS
CO122: CENTER ATOMS=969,970 MASS
CO123: CENTER ATOMS=977,978 MASS
CO124: CENTER ATOMS=985,986 MASS
CO125: CENTER ATOMS=993,994 MASS
CO126: CENTER ATOMS=1001,1002 MASS
CO127: CENTER ATOMS=1009,1010 MASS
CO128: CENTER ATOMS=1017,1018 MASS

m1: DISTANCE ...
  ATOMS1=CO1,NC1 LOCATION1=CO1
  ATOMS2=CO2,NC2 LOCATION2=CO2
  ATOMS3=CO3,NC3 LOCATION3=CO3
  ATOMS4=CO4,NC4 LOCATION4=CO4
  ATOMS5=CO5,NC5 LOCATION5=CO5
  ATOMS6=CO6,NC6 LOCATION6=CO6
  ATOMS7=CO7,NC7 LOCATION7=CO7
  ATOMS8=CO8,NC8 LOCATION8=CO8
  ATOMS9=CO9,NC9 LOCATION9=CO9
  ATOMS10=CO10,NC10 LOCATION10=CO10
  ATOMS11=CO11,NC11 LOCATION11=CO11
  ATOMS12=CO12,NC12 LOCATION12=CO12
  ATOMS13=CO13,NC13 LOCATION13=CO13
  ATOMS14=CO14,NC14 LOCATION14=CO14
  ATOMS15=CO15,NC15 LOCATION15=CO15
  ATOMS16=CO16,NC16 LOCATION16=CO16
  ATOMS17=CO17,NC17 LOCATION17=CO17
  ATOMS18=CO18,NC18 LOCATION18=CO18
  ATOMS19=CO19,NC19 LOCATION19=CO19
  ATOMS20=CO20,NC20 LOCATION20=CO20
  ATOMS21=CO21,NC21 LOCATION21=CO21
  ATOMS22=CO22,NC22 LOCATION22=CO22
  ATOMS23=CO23,NC23 LOCATION23=CO23
  ATOMS24=CO24,NC24 LOCATION24=CO24
  ATOMS25=CO25,NC25 LOCATION25=CO25
  ATOMS26=CO26,NC26 LOCATION26=CO26
  ATOMS27=CO27,NC27 LOCATION27=CO27
  ATOMS28=CO28,NC28 LOCATION28=CO28
  ATOMS29=CO29,NC29 LOCATION29=CO29
  ATOMS30=CO30,NC30 LOCATION30=CO30
  ATOMS31=CO31,NC31 LOCATION31=CO31
  ATOMS32=CO32,NC32 LOCATION32=CO32
  ATOMS33=CO33,NC33 LOCATION33=CO33
  ATOMS34=CO34,NC34 LOCATION34=CO34
  ATOMS35=CO35,NC35 LOCATION35=CO35
  ATOMS36=CO36,NC36 LOCATION36=CO36
  ATOMS37=CO37,NC37 LOCATION37=CO37
  ATOMS38=CO38,NC38 LOCATION38=CO38
  ATOMS39=CO39,NC39 LOCATION39=CO39
  ATOMS40=CO40,NC40 LOCATION40=CO40
  ATOMS41=CO41,NC41 LOCATION41=CO41
  ATOMS42=CO42,NC42 LOCATION42=CO42
  ATOMS43=CO43,NC43 LOCATION43=CO43
  ATOMS44=CO44,NC44 LOCATION44=CO44
  ATOMS45=CO45,NC45 LOCATION45=CO45
  ATOMS46=CO46,NC46 LOCATION46=CO46
  ATOMS47=CO47,NC47 LOCATION47=CO47
  ATOMS48=CO48,NC48 LOCATION48=CO48
```

```
ATOMS49=CO49,NC49 LOCATION49=CO49
ATOMS50=CO50,NC50 LOCATION50=CO50
ATOMS51=CO51,NC51 LOCATION51=CO51
ATOMS52=CO52,NC52 LOCATION52=CO52
ATOMS53=CO53,NC53 LOCATION53=CO53
ATOMS54=CO54,NC54 LOCATION54=CO54
ATOMS55=CO55,NC55 LOCATION55=CO55
ATOMS56=CO56,NC56 LOCATION56=CO56
ATOMS57=CO57,NC57 LOCATION57=CO57
ATOMS58=CO58,NC58 LOCATION58=CO58
ATOMS59=CO59,NC59 LOCATION59=CO59
ATOMS60=CO60,NC60 LOCATION60=CO60
ATOMS61=CO61,NC61 LOCATION61=CO61
ATOMS62=CO62,NC62 LOCATION62=CO62
ATOMS63=CO63,NC63 LOCATION63=CO63
ATOMS64=CO64,NC64 LOCATION64=CO64
ATOMS65=CO65,NC65 LOCATION65=CO65
ATOMS66=CO66,NC66 LOCATION66=CO66
ATOMS67=CO67,NC67 LOCATION67=CO67
ATOMS68=CO68,NC68 LOCATION68=CO68
ATOMS69=CO69,NC69 LOCATION69=CO69
ATOMS70=CO70,NC70 LOCATION70=CO70
ATOMS71=CO71,NC71 LOCATION71=CO71
ATOMS72=CO72,NC72 LOCATION72=CO72
ATOMS73=CO73,NC73 LOCATION73=CO73
ATOMS74=CO74,NC74 LOCATION74=CO74
ATOMS75=CO75,NC75 LOCATION75=CO75
ATOMS76=CO76,NC76 LOCATION76=CO76
ATOMS77=CO77,NC77 LOCATION77=CO77
ATOMS78=CO78,NC78 LOCATION78=CO78
ATOMS79=CO79,NC79 LOCATION79=CO79
ATOMS80=CO80,NC80 LOCATION80=CO80
ATOMS81=CO81,NC81 LOCATION81=CO81
ATOMS82=CO82,NC82 LOCATION82=CO82
ATOMS83=CO83,NC83 LOCATION83=CO83
ATOMS84=CO84,NC84 LOCATION84=CO84
ATOMS85=CO85,NC85 LOCATION85=CO85
ATOMS86=CO86,NC86 LOCATION86=CO86
ATOMS87=CO87,NC87 LOCATION87=CO87
ATOMS88=CO88,NC88 LOCATION88=CO88
ATOMS89=CO89,NC89 LOCATION89=CO89
ATOMS90=CO90,NC90 LOCATION90=CO90
ATOMS91=CO91,NC91 LOCATION91=CO91
ATOMS92=CO92,NC92 LOCATION92=CO92
ATOMS93=CO93,NC93 LOCATION93=CO93
ATOMS94=CO94,NC94 LOCATION94=CO94
ATOMS95=CO95,NC95 LOCATION95=CO95
ATOMS96=CO96,NC96 LOCATION96=CO96
ATOMS97=CO97,NC97 LOCATION97=CO97
ATOMS98=CO98,NC98 LOCATION98=CO98
ATOMS99=CO99,NC99 LOCATION99=CO99
ATOMS100=CO100,NC100 LOCATION100=CO100
ATOMS101=CO101,NC101 LOCATION101=CO101
ATOMS102=CO102,NC102 LOCATION102=CO102
ATOMS103=CO103,NC103 LOCATION103=CO103
ATOMS104=CO104,NC104 LOCATION104=CO104
ATOMS105=CO105,NC105 LOCATION105=CO105
ATOMS106=CO106,NC106 LOCATION106=CO106
ATOMS107=CO107,NC107 LOCATION107=CO107
ATOMS108=CO108,NC108 LOCATION108=CO108
ATOMS109=CO109,NC109 LOCATION109=CO109
ATOMS110=CO110,NC110 LOCATION110=CO110
ATOMS111=CO111,NC111 LOCATION111=CO111
ATOMS112=CO112,NC112 LOCATION112=CO112
ATOMS113=CO113,NC113 LOCATION113=CO113
ATOMS114=CO114,NC114 LOCATION114=CO114
```

```
  ATOMS115=CO115,NC115 LOCATION115=CO115
  ATOMS116=CO116,NC116 LOCATION116=CO116
  ATOMS117=CO117,NC117 LOCATION117=CO117
  ATOMS118=CO118,NC118 LOCATION118=CO118
  ATOMS119=CO119,NC119 LOCATION119=CO119
  ATOMS120=CO120,NC120 LOCATION120=CO120
  ATOMS121=CO121,NC121 LOCATION121=CO121
  ATOMS122=CO122,NC122 LOCATION122=CO122
  ATOMS123=CO123,NC123 LOCATION123=CO123
  ATOMS124=CO124,NC124 LOCATION124=CO124
  ATOMS125=CO125,NC125 LOCATION125=CO125
  ATOMS126=CO126,NC126 LOCATION126=CO126
  ATOMS127=CO127,NC127 LOCATION127=CO127
  ATOMS128=CO128,NC128 LOCATION128=CO128
  COMPONENTS
...


smac1: SMAC ...
  SPECIES=m1
  SWITCH={RATIONAL R_0=0.6}
  SWITCH_COORD={EXP R_0=4}
  KERNEL1={GAUSSIAN CENTER=0 SIGMA=0.8}
  KERNEL2={GAUSSIAN CENTER=pi SIGMA=0.7}
...

# basis functions and the target distribution
bf1: BF_LEGENDRE ORDER=20 MINIMUM=1 MAXIMUM=14
td1: TD_WELLTEMPERED BIASFACTOR=2

VES_LINEAR_EXPANSION ...
  LABEL=ves1
  ARG=smac1
  BASIS_FUNCTIONS=bf1
  TARGET_DISTRIBUTION=td1
  TEMP=450
  GRID_BINS=100
... VES_LINEAR_EXPANSION

OPT_AVERAGED_SGD ...
  BIAS=ves1
  STRIDE=1000
  LABEL=o1
  STEPSIZE=0.01
  FES_OUTPUT=200
  BIAS_OUTPUT=200
  COEFFS_FILE=coeffs.data
  COEFFS_OUTPUT=50
  TARGETDIST_STRIDE=100
... OPT_AVERAGED_SGD

PRINT ARG=smac1 FILE=smac1.dat STRIDE=500
PRINT ARG=vol,ene,ves1.* FILE=colvar-ves.dat STRIDE=500
```