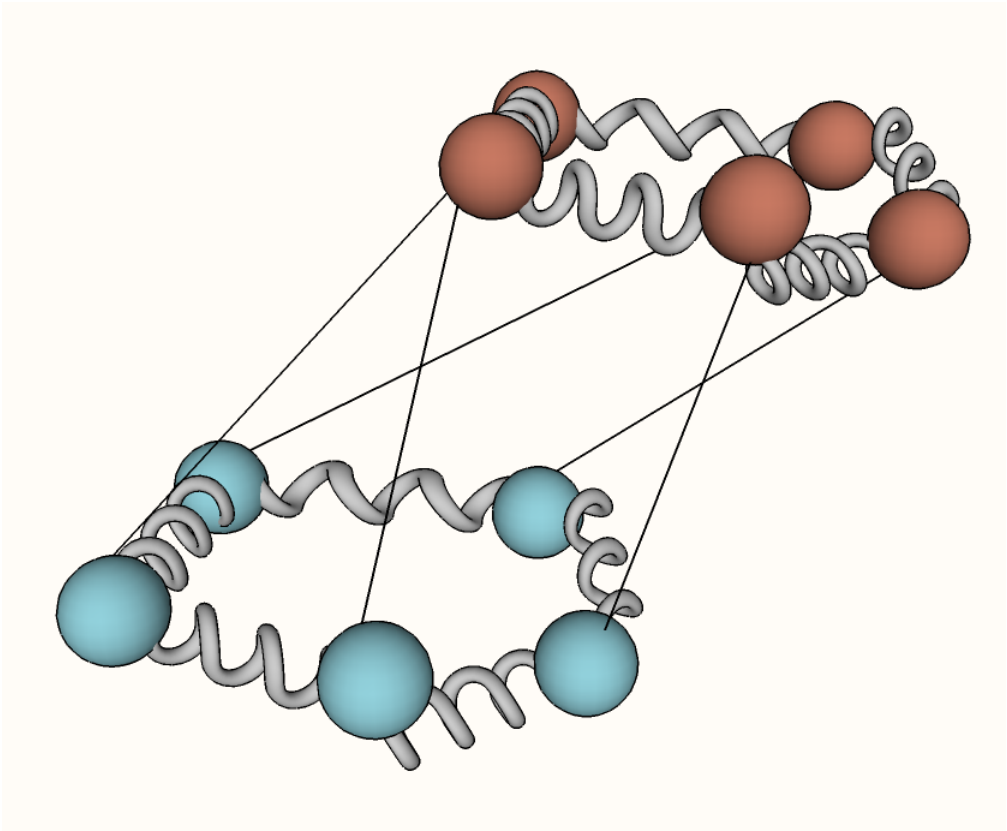
CHALMERS



The Monte Carlo Method and Quantum Path Integrals

Master's Thesis in Applied Physics

JOAKIM LÖFGREN

Department of Applied Physics
Division of Materials and Surface Theory
Chalmers University of Technology
Gothenburg, Sweden 2014

Thesis for the degree of Master of Science in Engineering Physics

The Monte Carlo Method and Quantum Path Integrals

JOAKIM LÖFGREN

CHALMERS

Department of Applied Physics
Division of Materials and Surface Theory
Chalmers University of Technology
Gothenburg, Sweden 2014

The Monte Carlo Method and Quantum Path Integrals

JOAKIM LÖFGREN

Department of Applied Physics
Division of Materials and Surface Theory
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone: +46 (0)31-772 1000

Cover:
Using the path integral formulation a quantum system can be mapped to a classical system of polymer-like entities.

Chalmers Reproservice
Gothenburg, Sweden 2014

Abstract

Cubic structured perovskites is a family of solids displaying a wide range of interesting physical properties, including several types of structural phase transitions. Frequently, these transitions give rise to competing crystal strucutres separated by small energy barriers in which case it is not obvious that zero-point fluctuations and other nuclear quantum effects can be neglected. One such perovskite is barium zirconate where the possible prescene of antiferrodistortive (AFD) phases related to tilting of the oxygen octahedra have been a matter of some debate. The interest in this material is motived by its application as a proton conducting electrolyte in solid oxide fuel cells. To take quantum effects into account when calculating properties a path integral formulation may be used. This approach leads to a multi-dimensional integral which can be calculated using Metropolis Monte Carlo, resulting in the path integral Monte Carlo method (PIMC).

In this study PIMC simulations are used to calculate various properties of barium zirconate, most notably the momentum distribution which is sensitive to nuclear quantum effects. To describe the inter-atomic interactions a rigid ion model is adopted with a simple pair potential consisting of a short-range Buckingham potential and the long-range Coulomb potential. Tuning of the parameters in the Buckingham potential allows for the introduction of AFD instabilities to the system. In this way two different model systems are established, one stable cubic system and one with AFD phases. Comparing the momentum distributions for these model system excludes the possibility that an oxygen atom can simultaneously occupy two different sites in the unstable system.

The thesis also serves as an introduction to PIMC simulations in general. Different algorithms for sampling new paths and calculating the momentum distribution are investigated and compared. All algorithms in the project have been implemented from scratch and the source code is made available in an appendix.

Keywords: path integral monte carlo, perovskites, octahedral tilting, momentum distribution

Acknowledgements

I would like to sincerely thank professor Göran Wahnström for his counsel and patience in supervising this thesis. It is rare event when a thesis combines so many of your interests and provides insights into so many different fields. Addional thanks go out to Erik Fransson and Johannes Laurell Håkansson for their many insights and a fruitful collaboration on modelling the system interactions, and to Erik Jedvik for the entertaining discussions in the late afternoons. I am also immensely grateful to all my family and friends who have kept me company and supported me over the years. I would never have come this far without you.

Joakim Löfgren, Gothenburg, early summer 2014

CONTENTS

1 Introduction **1**
- 1.1 Purpose and scope of the thesis . . . 2
- 1.2 Reading guide . . . 3

2 Barium zirconate **4**
- 2.1 The cubic perovskite structure . . . 4
- 2.2 Antiferrodistortive instabilities and octahedral tilting . . . 5
- 2.3 A quick note on unit systems . . . 7

3 Inter-atomic interactions **9**
- 3.1 Potentials . . . 9
- 3.2 The model potential part I: Short-range interactions . . . 10
- 3.3 Computer simulations and periodic boundary conditions . . . 11
- 3.4 An algorithm for calculating the potential . . . 13
- 3.5 The model potential part II: Long-range interactions and the Ewald summation . . . 14

4 The Monte-Carlo method **16**
- 4.1 Ensemble averages . . . 16
- 4.2 Monte-Carlo integration . . . 17
- 4.3 The Metropolis-Hastings algorithm . . . 21
- 4.4 The complete Metropolis Monte Carlo algorithm . . . 23
- 4.5 The pair correlation function . . . 25

5 Quantum statistical mechanics and path integrals **26**
- 5.1 The density operator . . . 26
- 5.2 The thermal density matrix . . . 27
- 5.3 Path integrals . . . 30
- 5.4 The classical isomorphism . . . 33

5.5 The path integral Monte Carlo method . . . 35
5.6 Estimators . . . 37
5.7 Improving the sampling . . . 38
5.7.1 Centre of mass displacements . . . 39
5.7.2 The Bisection Algorithm . . . 39
5.8 The momentum distribution . . . 42
5.9 Algorithms for computing for the momentum distribution . . . 43
5.9.1 The open chain method . . . 44
5.9.2 The trail method . . . 45

6 Results **48**
6.1 Verification part I: The potential . . . 49
6.2 Model potentials and antiferrodistortive instabilities . . . 51
6.3 Verification part II: The Path Integral Monte-Carlo method . . . 55
6.4 Barium Zirconate . . . 57
6.4.1 Internal energy . . . 58
6.4.2 The pair correlation function . . . 61
6.4.3 The momentum distribution . . . 63

7 Discussion **68**
7.1 Inter-atomic interactions and the Ewald summation . . . 68
7.2 Simulation algorithms . . . 69
7.2.1 The sampling of new paths . . . 69
7.2.2 Calculating the momentum distribution . . . 70
7.3 Properties of barium zirconate . . . 71
7.3.1 The momentum distribution . . . 71
7.4 Outlook and future prospects . . . 72

8 Conclusions **74**

References **77**

A The Ewald Summation **81**

B Notes on programming **87**
B.1 Programming in Fortran 90 . . . 87
B.2 Random number generation . . . 89
B.3 Source code . . . 89

CHAPTER 1

INTRODUCTION

Energy related materials is an active and important area of research where increasingly advanced applications require an understanding of the materials on an atomic level. One important example is fuel cells. In a fuel cell hydrogen is oxidised at the catode and the protons are then transported through an *electrolyte* to the anode, to which electrons are also led through an external circuit. The protons and electrons then combine with oxygen at the anode to form water and electrical power can be extracted from the current produced by the electrons. Lately there has been considerable interest in *solid oxide fuel cells* (SOFC) where the electrolyte is a ceramic, typically yttria/scandia stabilized zirconia conducting oxygen ions. The SOFC operates at high temperatures in the range $700 - 1000\,\mathrm{K}$ which can lead to complications and efforts have thus been made to reduce the operating temperature. One promising alternative is to use a proton-conducting solid oxide as electrolyte instead. Here, one of the main contenders is (yttrium-doped) barium zirconate, a perovskite structured oxide with remarkably good proton conductivity. Clearly, to construct an effective fuel cell one must have a good understanding of the structure and dynamics of the electrolyte and during the last decade perovskite oxides such as barium zirconate have been the subject of intense research.

Alongside the more traditional theoretical and experimental methods, computer simulations have become an everyday tool in materials science. Some of the benefits of simulations are, in no particular order:

- An excellent tool for testing models and theoretical ideas, scanning for specific properties or predicting new materials.
- One has complete control over the system and it is possible to probe extreme conditions which can be difficult to produce in a laboratory such as low temperature and high pressure environments.
- It is typically cheaper to perform a simulation rather than an experiment.
- Simulation codes and settings can be distributed and shared, allowing for easy reproduction and verification of results.

To study the dynamics and structure of a material, two of the most common simulation methods are molecular dynamics (MD) and Monte Carlo (MC). Both MD and MC require that one can describe the inter-atomic interactions in the system with sufficient accuracy. For heavier elements, a classical description of the nuclear motion is a valid approximation, the impact of quantum mechanical effects on the structure and thermodynamics is usually limited to systems consisting of lighter elements e.g. hydrogen or helium. A dramatic example is liquid helium-4 where the liquid undergoes a transition to a superfluid at very low temperatures, a purely quantum mechanical effect. Another situation where quantum corrections can be important arises when there are two or more competing structures in a material with small energy differences. This type of effect occurs in several perovskites where the oxygen atoms form octahedra around the cations. Distortions of these octahedra such as out-of-phase tilting can lead to new stable configurations that are relatively close in energy to the undistorted phase. In such a case it is not evident beforehand that quantum fluctuations can be neglected and a quantum mechanical treatment of the nuclear motion is necessary. One way to accomplish this is the path integral method which is also the main topic of this thesis. The path-integral approach to quantum mechanics was originally conceived by Feynmann although he drew heavily on ideas put forward by Dirac. Following this approach one can map the quantum mechanical problem to a classical system which leads to a multi-dimensional integral that can be sampled using MD or MC. This thesis focuses on MC techniques and the resulting method is known as path integral Monte Carlo (PIMC).

1.1 Purpose and scope of the thesis

The aim of this thesis is to show how PIMC simulations can be used to calculate quantum corrected properties for a system. More precisely, we will study a perovskite structured oxide, namely barium zirconate with a simple pair potential describing the inter-atmoic interactions in this system. This model potential consists of a Buckingham potential describing the short-range interactions and a Coulomb potential describing the electrostatic interaction between the ions. We are also interested in describing more generic behaviour found in perovskite structured oxides such as antiferrodistortive (AFD) phase transistions related to tilting of the oxygen octahedra. Here, a first goal is to show that it is possible to capture this sort of behaviour by tuning the parameters in the potential. The result is a simple model consisting of two potentials, one describing a system with only a cubic phase and the other one describing a similar system that also includes AFD phases. The potentials can then be used in conjunction with PIMC simulations in order to calculate properties with quantum mechanics taken fully into account. A main aim is to calculate the momentum distribution of the system, the shape of which is influenced by nuclear quantum effects e.g. tunneling and zero-point energy fluctuations. An important point here is to compare the results for the two different potentials. Since there are very few codes available which offer the right amount of control for these type of calculations we have elected to implement all algorithms from scratch. Thus a large part of the work consists of describing these algorithms and how they work in some detail.

1.2 Reading guide

In Chapter 2 the perovskite structure is introduced and the barium zirconate unit cell is described. Furthermore, the oxygen octahedra surrounding the zirconium cations are illustrated and the chapter concludes with a discussion of antiferrodistortive phase transitions relating to tilting of these octahedra. Chapter 3 treats inter-atomic interactions and describes in detail the different components of the pair potential. The long-range coulomb interaction is given a special treatment in Section 3.5 where the Ewald summation is introduced. The chapter also contains general information on how to set up a simulation and an outline of the actual algorithm for calculating the potential on a computer. Chapter 4 is essentially a review of some basic results from statistical mechanics, Monte Carlo integration and the Metropolis algorithm. The experienced reader should feel free to skip directly ahead to Chapter 5. Here we begin with a brief review of the density matrix formalism and quickly proceed to show how the thermal density matrix of a system can be expanded into a path integral. Following that we introduce the powerful PIMC method and the last part of Chapter 5 explore various improvements of the basic algorithm and also covers how to calculate the internal energy and momentum distribution. In Chapter 6, the first sections aim to verify that the programs are working correctly and also discusses modifications to the model potential. In the second part of the chapter the results from the main simulations are presented. Finally, in Chapter 7 the implications of our results are discussed and there is also an outlook discussing future prospects and possible ways to extend the study.

There are also two appendices in this report. Appendix A contains a derivation of the Ewald summation while Appendix B includes a brief discussion on the implementation of the algorithms in Fortran 90 as well as source code for many of the more important subroutines used. On a related note, throughout this thesis algorithms are described by general step-by-step instructions outlined in gray. The intent is that these instructions should be clear enough that any reader can confidently proceed to implement the algorithms in his or her favourite programming language. As mentioned above, Fortran 90 specific implementations can be found in Appendix B.

CHAPTER 2

BARIUM ZIRCONATE

Barium zirconate is a solid oxide belonging to the family of perovskite structured materials which have been studied extensively during the last hundred years due to a wide range of interesting properties including but not limited to ionic conductivity, magnetic properties, superconductivity and various other phase transitions such as ferroelectric and antiferrodistortive (AFD) transistions. In this chapter we will begin by describing the geometry of a material with perovskite structure and conclude with a more specific discussion of barium zirconate and AFD instabilites.

2.1 The cubic perovskite structure

A solid with a *perovskite structure* has the general chemical formula ABX_3. Here A and B are cations while X denotes of anions. In the simplest case we have a cubic perovskite which can be described geometrically as a bravais lattice with a five atom basis. The primitve lattice vectors are thus of the form $\mathbf{a}_i = a\hat{\mathbf{x}}_i$ where a is the lattice parameter and $i = 1,2,3$. An arbitrary lattice point can then be specified by a translation vector $\mathbf{T} = n_1\mathbf{a}_1 + n_2\mathbf{a}_2 + n_3\mathbf{a}_3$ for some integer combination (n_1, n_2, n_3). To each lattice point we then attach a basis consisting of five ions, the coordinates of each ion relative to the lattice point is given in Table 2.1.

Atom	Coordinate relative to lattice point
A	$\mathbf{x}_1 = a\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$
B	$\mathbf{x}_2 = a\,(0,0,0)$
X	$\mathbf{x}_3 = a\left(\frac{1}{2},0,0\right)$
X	$\mathbf{x}_4 = a\left(0,\frac{1}{2},0\right)$
X	$\mathbf{x}_5 = a\left(0,0,\frac{1}{2}\right)$

Table 2.1. *Coordinate vectors relative to a lattice point for the five atoms in the perovskite basis.*

For the specific case of barium zirconate which has the formula $BaZrO_3$, the cubic unit cell is illustrated in Fig. 2.1. Here the Ba^{2+} ions (green) are located in the middle of cell while the Zr^{4+} ions (blue) can be found in the corners and the O^{2-} ions (red) are located midway along the edges of the cell. Note that the radius of the spheres representing the ions in Fig. 2.1 are scaled in order to represent the true relative ionic radii.

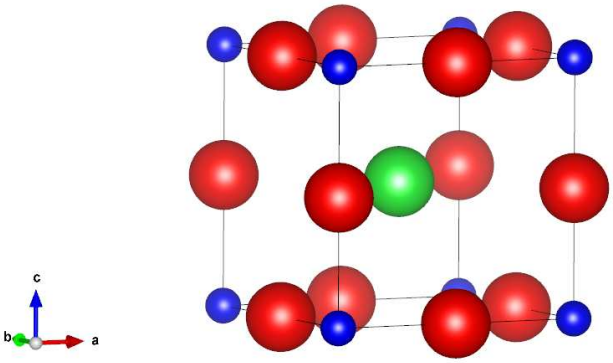


Figure 2.1. *The perovskite unit cell structure. The unit cell is cubic with a barium ion located in the centre (green), zirconium atoms in the corners (blue) and finally the oxygen atoms located half-way along the edges (red). This image was produced using VESTA [12].*

2.2 Antiferrodistortive instabilities and octahedral tilting

If we imagine a bulk sample of a cubic perovskite we can see from the unit cell Fig. 2.1 that the X anions will form octahedra around the B cations. Note that one octahedron around a B cation only involves the six closest X anions. Thus, in the case of barium zirconate the oxygen ions will form octahedra around the zirconium atoms as depicted in Fig. 2.2. It turns out that several perovskite structured oxides e.g. $CaTiO_3$, $PbZrO_3$ exhibit so called *antiferrodistortive* structural phase transitions related to tilting of oxygen octhadera seen in Fig. 2.2. These phase transitions are temperature dependent and typically only observed below a certain transistion temperature which is not necessarily easy to determine.

Geometrically we describe a tilt as a rigid rotation about an axis through the B cation in the centre of the octahedra although, stictly speaking, this is not true since two neighbouring octahedra share an oxygen atom meaning there will be some distortion when there is a tilt. Often the rotation is completely out-of-phase in one or more directions and it is customary to describe a tilt system using *glazer notation*. According to this notation a tilt is specified by $a^{\pm,0}b^{\pm,0}c^{\pm,0}$ where the letters specify the relative

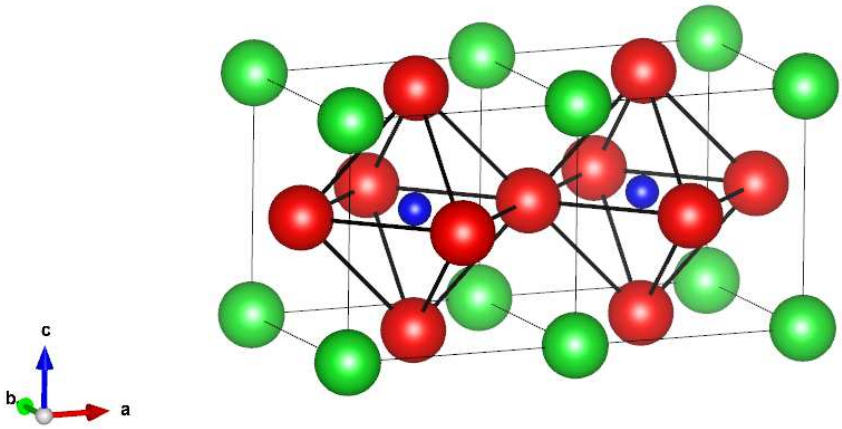


Figure 2.2. *An illustration of two ZrO_6 octahedra in barium zirconate. In perovskite systems with AFD instabilities there are phases where these octahedra exhibit out-of-phase or in-phase tilting in one or more directions. This image was produced using VESTA [12].*

magnitude of the rotations about the cartesian coordinate axes. As an example consider $a = b \neq c$ which means that we have an equal rotation angles about x and y while the rotation about z is either larger or smaller. The superscripts gives information about the phase of the tilt where a minus sign signifies out-of-phase rotations between neighbouring octahdera and vice versa. For instance, $a^+a^+c^-$ would mean in phase rotations with equal magnitude about the x- and y-axes while we have an out-of-phase rotation with a different magnitude about the z axis. A zero superscript indicates that there is no rotation about that particular axis.

In perovskites with AFD instabilites there are several competing structures where in addition to an untilted cubic structure there can exist stable or metastable phases with tilted octahedra. In these AFD phases the unit cell becomes distorted and no longer cubic, implying that the crystal symmetry has been lowered. Usually the energy difference between the different phases is very small which further complicates the situation since even small approximations in a calculation can have a bearing on the results.

The origin of the octahedral tilting is still not completely understood on a microscopic level but several explanations and models of varying complexity have been proposed. A particularily simple model that still has at least some predicative power comes from steric considerations. In this model the ions are considered hard spheres and one defines the *Goldschmidt ratio*

$$\tau = \frac{1}{\sqrt{2}} \left(\frac{R_A + R_X}{R_B + R_X} \right) \tag{2.1}$$

where $R_{A,B,X}$ are the ionic radii. For a perfect cubic structure the cell length can be expressed either as $\sqrt{2}(R_A + R_X)$ or $2(R_B + R_X)$ and the Goldschmidt ratio is simply the ratio of these two. It follows that a $\tau \approx 1$ indicates that all ions can approximately fit in the cubic lattice and we expect the system to be cubic. When $\tau < 1$ however there is a mismatch in size between the A and B cations and it is energetically favourable for the octahedra to tilt and the cell is no longer cubic. As an example consider $CaTiO_3$ which is known to have AFD instabilities and has a Goldschmidt ratio $\tau = 0.97$. This is a very crude model however, and as such it should only be relied on for prediciting tendencies.

A convenient way of determining if a system has instabilites is to calculate the phonon spectrum. Here, phase transitions which lower the crystal symmetry appear as soft phonon modes with imaginary frequencies. By analysing the polarisation vectors of these soft phonon modes one can then determine the corresponding displacements of the atoms in the new phase. Phonon spectra are typically calculated using density functional theory (DFT) or obtained from direct diagonalization of the dynamic matrix. Interestingly, analysis of phonon spectrum calculated for $BaZrO_3$ using DFT give different predictions based on the type of approximation used for the exchange-correlation functional. The details are not important here but using the local density approximation (LDA) one finds that there are indeed soft phonon modes corresponding to AFD phases while using the generalized gradient approximation (GGA) one instead finds that the system remains cubic [17]. Thus the result depends heavily on what apporoximations we make and it is not known whether the instabilites are really there. Experimental evidence suggests that the cubic strucutre is retained down to at least $T = 2\,\mathrm{K}$ [10]. It has been hypothesised that there is only a weak AFD instability and that long-range ordering of tilted octahedra is subsequently suppressed by quantum zero-point lattice vibrations, although a recent study claims to have refuted this idea [9]. While solving this issue is beyond the scope of this thesis it can be viewed as a step in the right direction. Instead of doing ab initio calculations, a simple model based on a pair potential is used and by modifying the parameters in this potential we can go from a cubic system to one with AFD instabilities as will be shown in Chapter 6.

2.3 A quick note on unit systems

In litterature on computational solid state physics and adjacent fields, results are often reported in what we shall refer to as *metallic units* [1]. In this system the basic units of some common physical quantities are:

In this thesis we will strictly adhere to these units when reporting results (unless stated otherwise) so as to not cause any confusion. Since one of our main goals is to take into account quantum mechanical corrections it is however more convenient to work in

[1] This name is, altough convenient, not widely in use.

Quantity	Unit	Abbreviation
Distance	ångström	Å
Energy	electronvolt	eV
Temperature	kelvin	K
Time	picosecond	ps
Charge	multiple of the electron charge	
Mass	grams/mole	g/mol

Table 2.2. *A table displaying the basic units of measurements for various common physical quantities according to the metallic unit system.*

atomic units when deriving our equations and implementing algorithms on the computer. In the atomic unit system Planck's constant, the electron charge and mass as well as the coulombic force constant are all unity by definition i.e. $\hbar = 4\pi\epsilon_0 = m_e = e = 1$. The atomic units of some common physical quantites are listed in Table 2.3 and compared with the correspoding metallic units.

Quantity	Unit	Conversion factors
Distance	bohr	$0.5292\,\text{Å}$
Energy	hartree energy (E_h)	$27.2114\,\text{eV}$
Temperature	kelvin	unchanged
Time	$\hbar/E_h$	$2.4189 \times 10^{-5}\,\text{ps}$
Charge	multiple of the electron charge	unchanged
Mass	multiple of the electron mass	$5.4858 \times 10^{-4}\,\text{g/mol}$

Table 2.3. *A table displaying the basic units of measurements for various common physical quantities according to the atomic unit system. For comparison with the metallic unit system the relevant conversion factors are listed in the rightmost column.*

CHAPTER 3

INTER-ATOMIC INTERACTIONS

The first section in this chapter covers the basics of of inter-atomic interactions and potentials. We then move on to describe a model for the short-range interactions in our barium zirconate system, how to calculate the total energy on a computer and what boundary conditions to use. The chapter concludes with a summary of the Ewald summation, an advanced method for calculating the electrostatic interaction in an efficient way.

3.1 Potentials

Consider a system of N interacting atoms[1]. Classically, this system can be described by a Hamiltonian

$$\mathcal{H} = \sum_{i=1}^{N} \frac{\mathbf{p}_i^2}{2m_i} + \mathcal{V}(\mathbf{q}_i) \tag{3.1}$$

where $\{\mathbf{q}_i\}_{i=1}^N$ and $\{\mathbf{p}_i\}_{i=1}^N$ denote sets of generalised coordinates and conjugate momenta. With complete knowledgde of the Hamiltonian and the relevant initial conditions one may solve for the motion of the system using Hamilton's equations. The potential $\mathcal{V}$ can be represented as a sum over N-body-interactions. In cartesian coordinates we have

$$\mathcal{V} = \sum_i \mathcal{V}_1(\mathbf{x}_i) + \sum_i \sum_{j>i} \mathcal{V}_2(\mathbf{x}_i,\mathbf{x}_j) + \sum_i \sum_{j>i} \sum_{k>i,j} \mathcal{V}_3(\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k) + \dots \tag{3.2}$$

In the above summation the first term does not represent an interaction between the atoms since it only depends on the coordinates of individual atoms, hence this

[1] Although we shall be mainly concerned with ionic systems, the basic theory outlined here applies equally well to a system of atoms or molecules.

term is only important in the prescence of an externally applied field (e.g. an electric field). The second term represents pairwise interactions between atoms and is, for many applications, the only relevant term. If important, the effects of three-body, i.e. the $\mathcal{V}_3$ term, or higher-order interactions can sometimes be modelled and included in the pair potential to yield an effective pair potential. In practice one often uses a model for the pair potential containing several free parameters which are then fitted to experimental data to yield correct values for a small set of properties. For a solid such a set may include e.g. the equilibrium lattice parameter, bulk modulus and so on. In this work we shall be exlusively concerned with just such an empirical pair potential for $BaZrO_3$, where the parameters are adjusted in order to create a set of two differnet potentials in order to allow for anti-ferrodistorive instabilities. This topic will be discussed in some detail in Chapter 6 Section 6.2.

3.2 The model potential part I: Short-range interactions

For ionic systems we can describe the inter-atomic interactions using a rigid-ion model where the electrons are effectively replaced by atomic charges and potential terms modelling van der Waals interaction and pauli repulsion. Recall that the van der Waals interaction is a quantum mechanical effect arising from fluctuating dipole moments and manifests as a weakly attractive force between two atoms. Pauli repulsion on the hand is as the name suggests a repulsive force, arising from overlapping electronical orbitals. When modelling these interactions they are usually grouped together in a single short-range pair potential. The two most popular forms are the *Lennard-Jones potential* and the *Buckingham potential*. We shall use the Buckingham potential which has the form

$$\mathcal{V}(r) = Ae^{-r/\rho} - \frac{C}{r^6}. \tag{3.3}$$

The first term on the right side of Eq. (3.3) represents the Pauli repulsion and is modelled using a simple exponential function giving a strong repulsion at short distances. The second term represents the van der Waals interaction and is attractive as noted above and has a characteristic $1/r^6$-dependence. We have yet to include a coulombic term in this potential since, as it turns out, calculating the Coulomb interaction is non-trivial matter and the discussion of this topic will be suspended until section 3.5. The three Buckingham parameters (A, ρ, C) are usually fitted to experimental data so that the potential reproduces known values of e.g. the lattice parameter.

For systems that are not monatomic the strength of the interactions depends on the type of the interacting atoms e.g. in the case of $BaZrO_3$ the Pauli repulsion between Ba- and O atoms will be different from the repulsion between Zr- and O atoms and so forth. Thus we end up several sets of the parameters (A, ρ, C) for the different interacting atomic species. For $BaZrO_3$, an appropriate set of fitted parameters have been determined by Stokes and Islam [4] and are listed in the table below.

Interaction	Parameter	Value
Ba-O	$A(eV)$	931.700
	$\rho(\mathring{\text{A}}^{-1})$	0.3949
	$C(eV\mathring{\text{A}}^{6})$	0.0000
Zr-O	A	985.869
	ρ	0.3760
	C	0.0000
O-O	A	22764.300
	ρ	0.1490
	C	27.890

Table 3.1. *A table displaying fitted parameters for the Buckingham potential, describing the short-range interactions in $BaZrO_3$. The parameter values were originally determined by Stokes and Islam [4].*

3.3 Computer simulations and periodic boundary conditions

If one wishes to perform atomic scale simulations on a real system such as a solid or a liquid on a computer, the first difficulty one encouters is that of finite memory. The sheer number of atoms in a piece of material is so large that we cannot possibly store positions and solve equations of motion et cetera for them all. Instead, one has to make due with a relatively small number of particles occupying a small volume. In this thesis we shall be exlusively concerned with the calculation of bulk properties which, in theory, requires an infinite system in order to eliminate surface effects. It turns out one can model the behaviour of such a system by a construction known as *periodic boundary conditions* (or PBC for short) which we will now describe in some detail.

First, let us assume that we have an N-particle system located in a cubic box with side L and volume V. Now, imagine making an infinite number of replicas of the system and placing them around the original box in a manner such that all of space in filled without any overlap. Consequently, each particle in the orginal box now has an inifinite number of replicas or *periodic images* in the surrounding boxes. If we turn on some interaction, the motion of the original particle is followed by all its periodic images. Thus, as a particle crosses the face of one box one if its images will enter the box on the opposite side. This scenario is depicted in figure Fig. 3.1.

The point is that while what we have described above is an infinite system, we only need to store on a computer the coordinates of each of the original particles since the motion of the images are identical. During the simulation, as a particle crosses one of the main boxes boundaries its coordinates are simply translated so that the particle reenters the box on the other side.

If a particle has the cartesian coordinate $\mathbf{x}_i$ then the coordinates of an arbitrary periodic image of the particle clearly has the form $\mathbf{x}_i + \mathbf{n}$ where $\mathbf{n} = L(n_x,n_y,n_z)$ and (n_x,n_y,n_z) are integers. It follows that we can write the total interaction energy for the

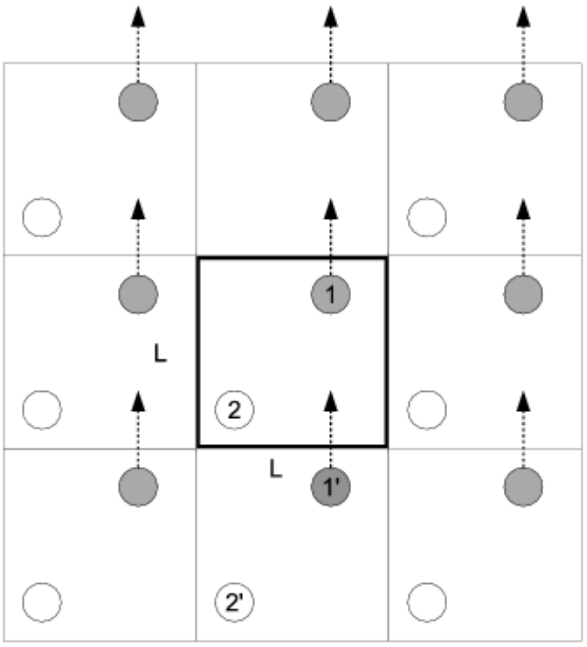


Figure 3.1. *An illustration of the periodic boundary conditions setup. The original simulation box is displayed in the centre with thick black borders, surrounded by its six closest periodic replicas. Note that the atoms labled* 1 *and* 2 *in the original box has a periodic image in each of the replicate boxes which follow the same motion as the original atom. Hence, when atom* 1 *leaves the box, one of its periodic images* $1'$ *will enter the box on the opposite side as depicted in the figure.*

system as a sum over particles and their images, assuming a central pair potential so that $\mathcal{V}_2(\mathbf{x}_i,\mathbf{x}_j) = \mathcal{V}_2(\mathbf{x}_i - \mathbf{x}_j)$ in Eq. (3.2) we can write

$$\mathcal{V} = \frac{1}{2}\sum_{\mathbf{n}}\sum_{i=1}^{N}\sum_{j=1}^{N}\mathcal{V}_2(\mathbf{x}_{ij} + \mathbf{n}). \tag{3.4}$$

where $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ and we implicitly assume that when $\mathbf{n} = 0$, the $i = j$ terms are exluded[2]. While equation Eq. (3.4) is exact and completely general for pair potentials in a periodic system the sum is infinite and hence needs to be truncated in order to be calculated on a computer. The most common way of limiting the number of interacting particles is known as the *minimum image convention* which states that a particle only interacts which the closest periodic image of each of the other particles. For a given particle pair, the distance bewteen one particle and the closest periodic image of the other is known as the *minimum image distance*. An equivalent way of visualising the minimum image convention is that a particle with cartesian coordinate $\mathbf{x}_i$ interacts only with those particles that can be found in a cubic box with side L centered on $\mathbf{x}_i$. As a

[2]This means that we allow an ion to interact with its periodic images but not with itself.

further restriction, one typically only includes interactions with particles within a sphere (rather than a box) of radius $r_{\text{cut}} < L/2$ centered on $\mathbf{x}_i$. The parameter r_{cut} is known as the cut-off radius for the potential and must be chosen large enough that the total interaction energy converges. In the case of the Buckingham potential Eq. (3.2) this is achievable for relatively small values of r_{cut} since the two potential terms decay as e^{-r} and $1/r^6$ respectively.

Note that the PBC are an artificial construction and as such can introduce unwanted correlations and distorsions in a simulation, especially when calculating forces in e.g. a molecular dynamics program. Therefore, one must avoid working with systems that are too small. In particular for a solid such as $BaZrO_3$ working with a single unit cell is not enough, instead we work with *supercell* consisting of several primitive cells (i.e. containing only a single lattice point each) packed next to each other. The size of a supercell is specified according to $N_{c,x}\times N_{c,y}\times N_{c,z}$ where N_x is the number of primitive cells in the x-direction and so on. In this thesis we will only deal with cubic supercells and for computational reasons they will never be larger than 4×4×4. A more detailed account of the concepts explained in this section can be found in [1].

3.4 An algorithm for calculating the potential

We will now make use the ideas developed in the last section and describe an algorithm for calculating the total short-range interaction energy of the system as defined by the Bucking potential. To simplify things we will consider a monatomic system with N atoms total so that the interaction can be described by a single set of Buckingham parameters A, ρ, C. As discussed in the last section a finite system with periodic boundary conditions must be used and to truncate the sum Eq. (3.4) a spherical cut-off r_{cut} is introduced. The algorithm then basically consists of looping over atom pairs and add up pairwise interactions provided that the minimum image distance is less than r_{cout}. How the minimum image distance is calculated depends on where the coordinate system is located relative to the simulation box. Here, there are two standard choices: a) let the origin coincide with the lower left corner of the box or b) with the centre of the box. In either case the minimum image distance r between two atoms with coordinate $\mathbf{x}_i$ and $\mathbf{x}_j$ is calculated according to

$$r = \left| \mathbf{x}_{ij} - L f_{\text{round}} \left(\frac{\mathbf{x}_{ij}}{L} \right) \right| \tag{3.5}$$

where f_{round} denotes a function that rounds each element of a vector to the nearest integer. The algorithm for calculating the interaction energy is summarised step by step below.

Calculating the Buckingham potential

1. Choose a radial cut-off which satifies the minimum image convention i.e. $r_{\text{cut}} < L/2$.
2. Loop over all possible atom pairs (i,j) and for each pair do the following:

2.1. Calculate the minimum image distance r:

$$r = \left| \mathbf{x}_{ij} - L f_{\text{round}} \left(\frac{\mathbf{x}_{ij}}{L} \right) \right|$$

where $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$

2.2. if $r < r_{\text{cut}}$, add to the total potential energy $\mathcal{V}$ the contribution from the interaction between atom i and j given by

$$\delta\mathcal{V} = Ae^{-r/\rho} - \frac{C}{r^6} \tag{3.6}$$

The most expensive element of this algorithm is clearly the evaluation of the transcendental exponential function. In general, much of the appeal of using a rigid-ion model with a pair potential is that the calculations become very cheap.

3.5 The model potential part II: Long-range interactions and the Ewald summation

Up until this point we have only accounted for the short-range interactions in our system. In a realistic ionic system one of course also has coulombic interactions between the charged ions. The electrostatic potential has the basic form

$$\Phi(r) = \frac{q}{r}. \tag{3.7}$$

For a periodic system consisting of N ions with charges $\{q_i\}_{i=1}^N$ we can write the total electrostatic interaction energy on the form Eq. (3.4):

$$\mathcal{V}_{\text{coul}} = \frac{1}{2} \sum_{\mathbf{n}} \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{q_i q_j}{|\mathbf{x}_{ij} + \mathbf{n}|}. \tag{3.8}$$

The crucial difference is that while the terms contributing to the Buckingham potential decay as e^{-r} and $1/r^6$ respectively, the coulomb potential only decays as $1/r$. Thus if we were to adopt an identical approach to computing this electrostatic energy as we did the short-range interactions (i.e. simply truncate the sum by introducting a radial cut-off) we would have to include far too many terms in Eq. (3.8). There is also a more subtle problem with the $1/r$ decay, in mathematical terms the summation in Eq. (3.8) is conditionally convergent, meaning that the result depends on the order of the summation. Thankfully, there is way to overcome these difficulties, namely a powerful technique for computing long-range interactions known as the *Ewald summation*. Unfortunately, the derivation is quite lengthy and as such we have chosen to place it in appendix A to which the curious reader is referred. We shall instead be content with a short summary of the results: if the system is periodic, one can replace the summation in Eq. (3.8) with two rapidly converging sums, one in real-space and one in reciprocal space. The final expression for the total electrostatic energy of the system is

$$\begin{aligned}\mathcal{V}_{\text{coul}} = &\frac{1}{2}\sum_{\mathbf{n}}\sum_{i=1}^{N}\sum_{j=1}^{N} q_i q_j \frac{\operatorname{erfc}(\alpha\,|\mathbf{x}_{ij}+\mathbf{n}|)}{|\mathbf{x}_{ij}+\mathbf{n}|} \\ &+ \frac{2\pi}{V}\sum_{\mathbf{k}\neq 0}\sum_{i=1}^{N}\sum_{j=1}^{N} q_i q_j \exp\left(i\mathbf{k}\cdot\mathbf{x}_{ij}\right)\frac{\exp\left(\frac{-k^2}{4\alpha}\right)}{k^2} \\ &- \frac{\alpha}{\sqrt{\pi}}\sum_{i} q_i^2\end{aligned} \tag{3.9}$$

Here $\mathbf{n} = L(n_x,n_y,n_z)$ as usual and similarily the reciprocal space vectors $\mathbf{k}$ are given by $\mathbf{k} = \frac{2\pi}{L}(n_{k_x},n_{k_y},n_{k_z})$ with $(n_{k_x},n_{k_y},n_{k_z})$ an integer vector. Since the two sums in Eq. (3.9) (the real-space sum over $\mathbf{n}$ and the reciprocal sum over $\mathbf{k}$) both decay rapidly we can proceed to truncate using two radial cut-offs, one in real-space and one in reciprocal space. The parameter α found in Eq. (3.9) is known as the splitting parameter and controls the relative convergence speed between the two sums. For a detailed derivation and discussion of these results the reader is referred to appendix A. An implementation of the Ewald summation in Fortran 90 can be found in appendix B.

CHAPTER 4

THE MONTE-CARLO METHOD

The previous chapter was dedicated to calculating the short- and long-range interactions in our system. Combining the Buckingham potential with Coulomb potential expanded using the Ewald summation resulted in a simple model allowing us to compute the total interaction energy of the system or for any one particle in the system. We would now like to put this knowledge to use and calculate properties, these could be e.g. thermodynamical quantities or properties related to the structure such as a pair correlation functions. Using tools from statistical mechanics, these properties can be obtained as ensemble averages defined by integrals involing a probability distribution function. The result is a non-trivial multi-dimensional integral over phase space. In order to calculate such integrals efficiently on a computer, we will introduce the powerful Metropolis Monte Carlo method (MMC). The following section provides a very brief review of some basic results from equilibrium statistical mechanics that we will need, readers that wish to refresh their knowledge on the subject are encouraged to consult one of the many excellent texts on the subject e.g. [5].

4.1 Ensemble averages

In classical statistical mechanics the *ensemble average* (or statistical average) of a quantity $\mathcal{O}$ is given by an integral

$$\langle \mathcal{O} \rangle = \int d\mathbf{\Gamma} \rho(\mathbf{\Gamma}) \mathcal{O}(\mathbf{\Gamma}) \tag{4.1}$$

where the variable $\mathbf{\Gamma}$ denotes a point in phase space and ρ is the probability distribution function. Note that for a classical N particle system a point in phase space is defined by the positions and momenta for all the N particles i.e. $\mathbf{\Gamma} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N, \mathbf{p}_1, \ldots, \mathbf{p}_N\}$. In this thesis we will work exclusively in the canonical ensemble where the number of particles N, the volume V and the temperature T are all constant and consequently

we sometimes refer to this as the NVT-ensemble. In this ensemble the equilibrium probability distribution function is the Boltzmann or *canonical distribution*

$$\rho_{\mathrm{NVT}}(\mathbf{\Gamma}) = \frac{1}{Z} e^{-\beta \mathcal{H}(\mathbf{\Gamma})} \tag{4.2}$$

where $\beta = \frac{1}{k_B T}$ and

$$Z = \int d\mathbf{\Gamma} e^{-\beta \mathcal{H}(\mathbf{\Gamma})} \tag{4.3}$$

is the partition function. Hence we can write the equilibrium average for a quantity $\mathcal{O}$ in canonical ensemble as

$$\langle \mathcal{O} \rangle = \int d\mathbf{\Gamma} \mathcal{O}(\mathbf{\Gamma}) \frac{e^{-\beta \mathcal{H}(\mathbf{\Gamma})}}{Z}. \tag{4.4}$$

The classical Hamiltonian has the form $\mathcal{H} = \mathcal{T} + \mathcal{V}$ where $\mathcal{T} = \sum_{i=1}^{N} \frac{\mathbf{p}_i}{2m_i}$ only depends on the momenta $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$ and the potential $\mathcal{V} = \mathcal{V}(\mathbf{X})$ only depends on the positions of the particles $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. Thus the integral over $e^{-\beta \mathcal{H}(\mathbf{\Gamma})}$ in the partition function Eq. (4.3) can be separated into one momentum integral and one position (configurational) integral. If $\mathcal{O}$ is independent of the momenta, the integrals over $\mathbf{P}$ in the nominator and denominator of Eq. (4.4) will cancel and one is left with an average

$$\langle \mathcal{O} \rangle = \frac{\int d\mathbf{X} \mathcal{O}(\mathbf{X}) e^{-\beta \mathcal{V}(\mathbf{X})}}{\int d\mathbf{X} e^{-\beta \mathcal{V}(\mathbf{X})}}. \tag{4.5}$$

Thus, fom here on we denote the position distribution $\rho_{\mathrm{NVT}}(\mathbf{X}) = \exp\left(-\beta \mathcal{V}(\mathbf{X})\right)/Z$ with $Z = \int d\mathbf{X} e^{-\beta \mathcal{V}(\mathbf{X})}$ and write our ensemble averages

$$\langle \mathcal{O} \rangle = \int d\mathbf{X} \rho_{\mathrm{NVT}}(\mathbf{X}) \mathcal{O}(\mathbf{X}). \tag{4.6}$$

Multi-dimensional integrals such as Eq. (4.6) cannot be evaluated using direct numerical integration methods e.g. Simpson's rule due to the sheer number of operations that would be required .Instead we will use a famous method based on results from statistics, namely the *Monte Carlo method* which is the topic of the next section.

4.2 Monte-Carlo integration

Monte-Carlo integration is different from most other numerical integration methods in that it draws on results from mathematical statistics and therefore has probabilistic elements. To illustrate how the method works, consider integrating a function of one variable f over the interval $[a,b]$:

$$I = \int_a^b dx f(x). \tag{4.7}$$

Let ξ be a continuous random variable with a probability density function ρ defined on [a,b] such that $\rho(x) \neq 0, \forall x \in [a,b]$ and $\int_a^b dx\rho(x) = 1$. Multiplying and the dividing by ρ we can rewrite Eq. (4.7)

$$\int_a^b dx f(x) = \int_a^b dx \rho(x) \frac{f(x)}{\rho(x)}. \tag{4.8}$$

Recall from statistics that if g is a arbitray function the expected value of $g(\xi)$ is given by

$$\langle g(\xi) \rangle = \int_a^b dx \rho(x) g(x). \tag{4.9}$$

Now let g be defined by $g(x) = f(x)/\rho(x)$, from Eqs. (4.8) and (4.9) it follows that

$$\langle g(\xi) \rangle = \left\langle \frac{f(\xi)}{\rho(\xi)} \right\rangle = \int_a^b dx \rho(x) \frac{f(x)}{\rho(x)} = \int_a^b dx f(x). \tag{4.10}$$

Hence, if we can find a way to approximate the expected value $\langle g(\xi) \rangle$ we have also found an approximation to our original integral Eq. (4.5). The obvious approach is to randomly draw a collection of samples $\{\xi_i\}_{i=1}^{N_s}$ from the distribution ρ and then approximate the expected value $\langle g(\xi) \rangle$ using the sample mean of g:

$$\langle g(\xi) \rangle \approx \bar{g}_{N_s} = \frac{1}{N_s} \sum_{i=1}^{N_s} g(\xi_i). \tag{4.11}$$

The law of large numbers, which is a fundamental result of statistics, guarantees that as N_s tends towards infinity the above approximation become exact i.e.

$$\lim_{N_s \to \infty} \bar{g}_{N_s} = \langle g(\xi) \rangle \tag{4.12}$$

We call Eq. (4.11) the Monte-Carlo estimate for I. In conclusion, we have the following recipe for estimating the integral Eq. (4.7):

Monte Carlo integration

1. Choose an appropriate probability density function ρ.
2. Generate a large collection of samples $\{\xi_i\}_{i=1}^{N_s}$ randomly drawn from ρ.
3. Compute the sample mean: $\frac{1}{N_s} \sum_{i=1}^{N_s} \frac{f(\xi_i)}{\rho(\xi_i)}$.
4. Approximate the integral using the sample mean: $\int_a^b dx f(x) \approx \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{f(\xi_i)}{\rho(\xi_i)}$.

The simplest choice one can make is to let ξ be uniformly distributed over [a,b] so that $\rho(x) = 1/(b-a)$ and

$$\int_a^b dx f(x) \approx \frac{b-a}{N_s} \sum_{i=1}^{N_s} f(\xi_i). \tag{4.13}$$

For many application this choice of ρ is far from optimal however, since every point in the domain of integration is sampled, on average, an equal number of times. This can be disadvantageous when the integrand f is only appreciable in certain regions of the domain and for any outlying points the integrand is small enough that only a few samples are actually required. The solution is to choose ρ in a way that it, to some extent, captures the behaviour of f. Such a choice of a non-uniform distribution ρ is referred to as *importance sampling* and will be a crucial component in our calculations.

It is completely straightforward to generalise the results derived above to multi-dimensional integrals. If f is a function of N_d variables $\{x_1, x_2, \ldots, x_{N_d}\}$ to be integrated over $D \subset \mathbb{R}^{N_d}$ and $\rho = \rho(x_1, x_2, \ldots, x_{N_d})$ a suitable (joint) probability density we can proceed in the exact same manner as above. The resulting estimate of the integral is

$$\int_D dx_1 \ldots dx_{N_d} f(x_1, x_2, \ldots, x_{N_d}) \approx \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{f(\boldsymbol{\xi}_i)}{\rho(\boldsymbol{\xi}_i)} \tag{4.14}$$

where $\{\boldsymbol{\xi}_i\}_{i=1}^{N_s}$ are drawn from the multi-dimensional probability density ρ. It is when the number of variables becomes large and the integrand is appreciable only over certain regions of the domain D that Monte-Carlo integration really shines. Taking a look at the ensemble average

$$\langle \mathcal{O} \rangle = \int d\mathbf{X} \rho_{\mathrm{NVT}}(\mathbf{X}) \mathcal{O}(\mathbf{X}) \tag{4.15}$$

from the previous section Section 4.1 we can see that it fits the profile for an ideal Monte-Carlo case perfectly: with N particles there are $3N$ coordinates that we need to integrate over. Furthermore, if the energy happens to be large for a particular configuration the integrand will be vanishingly small due to the inverse exponential in ρ_{NVT}. We can now write the Monte-Carlo estimate for Eq. (4.15). The integrand is $\rho_{\mathrm{NVT}}(\mathbf{X})\mathcal{O}(\mathbf{X})$ and if ρ is an arbitrary probability density, we have according to Eq. (4.10)

$$\langle \mathcal{O} \rangle = \left\langle \frac{\rho_{\mathrm{NVT}} \mathcal{O}}{\rho} \right\rangle \tag{4.16}$$

A suitable choice for ρ is to simply let $\rho = \rho_{\mathrm{NVT}}$. Since the canonical distribution varies exponentially with the energy it ought to give a good indication of where the integrand $\rho_{\mathrm{NVT}}\mathcal{O}$ is significant for most forms of $\mathcal{O}$. From Eq. (4.11) the Monte-Carlo estimate then reduces to

$$\langle \mathcal{O} \rangle \approx \frac{1}{N_s} \sum_{i=1}^{N_s} \mathcal{O}(\boldsymbol{\xi}_i) \tag{4.17}$$

where $\{\boldsymbol{\xi}_i\}_{i=1}^{N_s}$ are now randomly drawn from the canonical distribution ρ_{NVT}. Let us take a step back and look at what we have accomplished. When we set out at the beginning of Section 4 the goal was to be able to calculate properties of the $BaZrO_3$ system, such as thermodynamical quantities or correlation functions. These properties were then expressed as ensemble averages in Section 4.1, which subsequently forced us

to consider efficient ways to computing multi-dimensional integrals, finally leading us to this section and Monte-Carlo integration. The techniques developed here have led us to Eq. (4.17) and hence the only problem left is how to generate samples ξ_i from the canoncial distribution.

Before moving on to solving the problem of how to sample the canonical distribution, we will briefly discuss the error associated with a Monte Carlo estimate since the method does us no good unless it can produce approximations to the integral I with small errors. Going back to our one-dimensional example, the variance of $g(\xi) = f(\xi)/\rho(\xi)$ is defined as

$$\mathrm{Var}\left[g(\xi)\right] = \left\langle g(\xi)^2 \right\rangle - \left\langle g(\xi) \right\rangle^2 \tag{4.18}$$

and the standard deviation is $\sigma(g(\xi)) = \sqrt{\mathrm{Var}\left[g(\xi)\right]}$. It follows from the central limit theorem that if the samples $\{\xi_i\}_{i=1}^{N_s}$ are statistically independent then the sample mean $\bar{g}_{N_s}$ in Eq. (4.11) is approximately gaussian distributed with variance

$$\mathrm{Var}\left[\bar{g}_{N_s}\right] = \frac{\mathrm{Var}\left[g(\xi)\right]}{N_s}. \tag{4.19}$$

The error incurred by replacing the exact integral I with the Monte Carlo estimate Eq. (4.11) is then given by the standard deviation

$$\sigma\left[\bar{g}_{N_s}\right] = \frac{\sigma\left[g(\xi)\right]}{\sqrt{N_s}}. \tag{4.20}$$

Eq. (4.20) tells us that the error decreases as the square root of the number of samples wherein lies the power of Monte Carlo integration. We note further that in the case of no importance sampling i.e. $\rho \equiv 1$ then $g(\xi) = f(\xi)$ and the error is $\sigma\left[f(\xi)\right]/\sqrt{N_s}$, but if we choose ρ so that it captures the behaviour f then $\sigma\left[g(\xi)\right] < \sigma\left[f(\xi)\right]$ and we decrease the error. Eq. (4.19) relies on the assumption that the samples are statistically independent which is not true in a MMC simulation where new configurations are generated from old ones inducing a high amount of correlation. To account for this we introduce the statistical inefficieny s which can be interpreted as the number of MC steps between truly independent configurations i.e. N_s/s is the number of statistically independent samples. We can estimate s using block averaging. Let N_b be the block size and define the j:th block average of g:

$$G_j = \frac{1}{N_b}\sum_{i=1}^{N_b} g_{i+(j-1)N_b} \tag{4.21}$$

where $g_i \equiv g(\xi_i)$. An estimation of s is then given by

$$s = \lim_{N_b \to \text{large}} \frac{N_b \mathrm{Var}[G]}{\mathrm{Var}[g]} \tag{4.22}$$

We then replace Eq. (4.19) with

$$\mathrm{Var}\left[\bar{g}_{N_s}\right] = s\frac{\mathrm{Var}\left[g(\xi)\right]}{N_s}. \tag{4.23}$$

and the MC error estimate is given by the corresponding standard deviation.

4.3 The Metropolis-Hastings algorithm

In the previous section we used Monte Carlo integration and importance sampling to show that ensemble averages can be computed in an efficient way provided that we can find a method for sampling the canonical distribution. To accomplish this we will now introduce the *Metropolis-Hastings algorithm*, which is essentially a biased random walk through phase space that after an initial equilibration period will start to generate samples distributed according to ρ_{NVT}. Starting from an initial state $\mathbf{\Gamma}_0$, new states are chosen with a probability given by a transition rule

$$\mathcal{P}(\mathbf{\Gamma}_m \to \mathbf{\Gamma}_n) \tag{4.24}$$

i.e. Eq. (4.24) is interpreted as the probability of transitioning from the state $\mathbf{\Gamma}_m$ to $\mathbf{\Gamma}_n$. Note that the transition probability only depends on current state and not any of the previous states, in other words the random walk is memoryless and we say that it constitutes a *Markov chain*. For a finite state space $\{\mathbf{\Gamma}_0, \mathbf{\Gamma}_1, \ldots\}$ an arbitrary distribution can then be defined by a vector $\boldsymbol{\rho}$ where the m:th element ρ_m gives the probability of finding the system in state $\mathbf{\Gamma}_m$. Similarily, we can regard the transistion rule Eq. (4.24) as a matrix $\boldsymbol{\mathcal{P}}$ with elements $\mathcal{P}_{nm} \equiv \mathcal{P}(\mathbf{\Gamma}_m \to \mathbf{\Gamma}_n)$ (note the intentional reversal of the order of the labels m and n in this definition). Each individual column of $\boldsymbol{\mathcal{P}}$ sum up to unity and we call a matrix with this property a stochastic matrix. If the distribution at step k of the walk is $\boldsymbol{\rho}^{(k)}$, making a transition according to $\boldsymbol{\mathcal{P}}$ will thus alter the distribution and the new distribution is given by a matrix multiplication

$$\boldsymbol{\rho}^{(k+1)} = \boldsymbol{\mathcal{P}}\boldsymbol{\rho}^{(k)}. \tag{4.25}$$

In this notation the limiting distribution $\boldsymbol{\rho}^{(\infty)}$ is

$$\boldsymbol{\rho}^{(\infty)} = \lim_{k\to\infty} \boldsymbol{\mathcal{P}}^k \boldsymbol{\rho}^{(0)} \tag{4.26}$$

where $\boldsymbol{\rho}^{(0)}$ denotes the initial distribution. From Eq. (4.26) it is apparent that $\boldsymbol{\rho}^{(\infty)}$ is a solution to the eigenvalue equation

$$\boldsymbol{\mathcal{P}}\boldsymbol{\rho}^{(\infty)} = \boldsymbol{\rho}^{(\infty)} \tag{4.27}$$

One can prove [1] that if the transition rule is defined in a way such that the resulting Markov chain is ergodic, meaning that any one state of the system can be reached in a finite number of transitions regardless of the intial state, then Eq. (4.27) has a unique solution. In the Metropolis-Hastings algorithm, one chooses the transition rule $\boldsymbol{\mathcal{P}}$ so

that the limiting distribution is given by canonical distribution $\boldsymbol{\rho}^{(\infty)} = \boldsymbol{\rho}_{\mathrm{NVT}}$. More precisely, $\boldsymbol{\mathcal{P}}$ is constructed to statisfy the condition of *detailed balance*

$$\rho_m \mathcal{P}_{nm} = \rho_n \mathcal{P}_{mn}. \tag{4.28}$$

Summing over n in Eq. (4.28) we find that the left-hand side $\sum_n \rho_m \mathcal{P}_{nm} = \rho_m$ since the sum over a column of $\mathcal{P}$ is unity as noted above and hence

$$\sum_n \mathcal{P}_{mn} \rho_n = \rho_m. \tag{4.29}$$

But this is just Eq. (4.27) again and we can conclude that if $\boldsymbol{\mathcal{P}}$ satisfies detailed balance (and has columns which all sum to unity) the Markov chain will converge to a unique distribution. The next step is to write $\mathcal{P}(\boldsymbol{\Gamma}_m \to \boldsymbol{\Gamma}_n)$ as the product of a trial transition $\mathcal{T}(\boldsymbol{\Gamma}_m \to \boldsymbol{\Gamma}_n)$ and an acceptance probability $\mathcal{A}(\boldsymbol{\Gamma}_m \to \boldsymbol{\Gamma}_n)$:

$$\mathcal{P}(\boldsymbol{\Gamma}_m \to \boldsymbol{\Gamma}_n) = \mathcal{T}(\boldsymbol{\Gamma}_m \to \boldsymbol{\Gamma}_n)\mathcal{A}(\boldsymbol{\Gamma}_m \to \boldsymbol{\Gamma}_n) \tag{4.30}$$

or in matrix form

$$\mathcal{P}_{nm} = \mathcal{T}_{nm}\mathcal{A}_{nm}. \tag{4.31}$$

In the traditional Metropolis-Hastings algorithm the trial transition has the form of a randomly proposed movement for a single atom and whether the proposed movement is accepted or not is subsequently determined by the acceptance probability. Thus, consider an atom labled i with a coordinate $\mathbf{x}_i^m$ and a box B centered around $\mathbf{x}_i^m$ with side δl. A new position $\mathbf{x}_i^n$ for this atom is then proposed with probability

$$T_{nm} = \begin{cases} 1/N_{\mathrm{cube}}, & \mathbf{x}_i^n \in B \\ 0, & \text{otherwise} \end{cases} \tag{4.32}$$

where N_{cube} is the total number of states insibe B which is, of course, finite on a computer. Eq. (4.32) means that we propose a new position for an atom with uniform probability inside a box with side δl centered around the atoms current location. Whether to accept this movement or not is then determined by the acceptance probability $\mathcal{A}$ defined for $m \neq n$ as

$$A_{nm} = \begin{cases} 1, & \frac{\rho_{\mathrm{NVT}}(\mathbf{X} \leftarrow \mathbf{x}_i^n)}{\rho_{\mathrm{NVT}}(\mathbf{X} \leftarrow \mathbf{x}_i^m)} \geq 1 \\ \frac{\rho_{\mathrm{NVT}}(\mathbf{X} \leftarrow \mathbf{x}_i^n)}{\rho_{\mathrm{NVT}}(\mathbf{X} \leftarrow \mathbf{x}_i^m)}, & \text{otherwise} \end{cases} \tag{4.33}$$

where $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ are the current positions of all the atoms as usual. The notation $\mathbf{X} \leftarrow \mathbf{x}_i^n$ indicates that we replace the coordinate of atom i with $\mathbf{x}_i^n$ in the configuration $\mathbf{X}$, but keep the configuration otherwise unchanged. When defined in this way the acceptance probability is symmetric and it follows that the complete transition rule will satisfy detailed balance [1]. From section Section 4.1 $\rho_{\mathrm{NVT}}(\mathbf{X}) = \exp\left[-\beta \mathcal{V}(\mathbf{X})\right]/Z$ so it follows that

$$\frac{\rho_{\text{NVT}}(\mathbf{X}')}{\rho_{\text{NVT}}(\mathbf{X})} = \frac{\exp\left[-\beta\mathcal{V}(\mathbf{X}')\right]/Z}{\exp\left[-\beta\mathcal{V}(\mathbf{X})\right]/Z} = \exp\left[-\beta\left(\mathcal{V}(\mathbf{X}') - \mathcal{V}(\mathbf{X})\right)\right]. \tag{4.34}$$

If we define the difference in potential energy $\Delta\mathcal{V} = \mathcal{V}(\mathbf{X} \leftarrow \mathbf{x}_i^n) - \mathcal{V}(\mathbf{X} \leftarrow \mathbf{x}_i^m)$ we can rewrite the acceptance probability Eq. (4.33) using Eq. (4.34):

$$A_{nm} = \begin{cases} 1, & \Delta\mathcal{V} \leq 0 \\ \exp\left[-\beta\Delta\mathcal{V}\right], & \text{otherwise} \end{cases} \tag{4.35}$$

We have now arrived at the final form of the acceptance probability, combining this with the trial transition Eq. (4.32) yields a complete transition rule which can be used to generate a random walk in phase space that will eventually converge in the sense that the generated states are sampled from the canonical distribution. One of the main reasons that the Metropolis-Hastings algorithm is so effective is that there is no need to compute the normalisation factor $1/Z$ for the canonical distribtuion since this factor is cancelled in Eq. (4.34) and we only have to worry about computing the difference in potential energy. Furthermore, since only a single atom is moved in each step, if we have a pair potential $\Delta\mathcal{V}$ can be obtained by subtracting the potential energy of the atom being moved in old configuration from the potential energy of the same atom in the new configuration:

$$\Delta\mathcal{V} = \mathcal{V}(\mathbf{X} \leftarrow \mathbf{x}_i^n) - \mathcal{V}(\mathbf{X} \leftarrow \mathbf{x}_i^m) = \sum_{j\neq i}^{N}\left[\mathcal{V}_2(\mathbf{x}_i^n - \mathbf{x}_j) - \mathcal{V}_2(\mathbf{x}_i^m - \mathbf{x}_j)\right] \tag{4.36}$$

Here we have used again the notation introduced in Section 3.1 Eq. (3.2) where $\mathcal{V}_2(\mathbf{x}_i - \mathbf{x}_j)$ represents the pairwise interaction (short-range as well as coulombic) between atoms i and j. Eq. (4.36) means that at no point in our algorithm are we forced to compute the total interaction energy for the entire system.

4.4 The complete Metropolis Monte Carlo algorithm

We will now combine the results from Sections 4.1, 4.2 and 4.3 into a single powerful algorithm. The algorithm is quite general but to make things more concrete we will comment on what some of the steps entail for our $BaZrO_3$ system. When combining the Metropolis algorithm with Monte Carlo integration one speaks of a *Metropolis Monte Carlo* (MMC) or a *Markov-Chain Monte Carlo* (MCMC) method. For applications in physics when the limiting distribution is the Boltzmann distribution the name canonical Monte Carlo is also used sometimes, in this thesis we shall stick with the name Metropolis Monte Carlo however. The full algorithm is summarised below, note that we have dispensed with the superscript indices used in the previous section and now write the old position of an atom i as $\mathbf{x}_i^{\text{old}}$ and similarily the new position after a trial displacement is denoted $\mathbf{x}_i^{\text{new}}$.

The Metropolis Monte Carlo method

1. Initialise the postions of all the N atoms. For a liquid or a solid such as $BaZrO_3$ an appropriate choice would be to use the $T = 0$ equilibrium configuration.
2. Choose at random an atom i with coordinate $\mathbf{x}_i^{\text{old}}$ and generate a trial state by displacing it symmetrically according to $\mathbf{x}_i^{\text{new}} = \mathbf{x}_i^{\text{old}} + \delta l(2\boldsymbol{\eta} - 1)$ where $\boldsymbol{\eta}$ is random vector drawn from the uniform distribution on [0,1]. Note that simultaneous storage of the both new and old coordinate is required at this point.
3. Compute the difference in potential energy: $\Delta\mathcal{V} = \mathcal{V}(\mathbf{X} \leftarrow \mathbf{x}_i^{\text{new}}) - \mathcal{V}(\mathbf{X} \leftarrow \mathbf{x}_i^{\text{old}})$. If the potential consists of pairwise interactions use Eq. (4.36).
4. If $\Delta\mathcal{V} \leq 0$ the trial state is immediately accepted and the old coordinate can be deleted. If $\Delta\mathcal{V} > 0$ generate a uniform random number q in [0,1]:
   - 4.1. If $q \leq \exp(-\beta\Delta\mathcal{V})$ the trial state is accepted and the old position can be deleted.
   - 4.2. Else if $q > \exp(-\beta\Delta\mathcal{V})$ the trial state is rejected and the atom's old position must be restored.
5. After one or several atoms have been moved, update the average value of any property of interest $\mathcal{O}(\mathbf{X})$, note that all moves contribute equally to the average i.e. regardless of whether the old position was restored or the trial state was accepted. Also update the error.
6. Repeat steps 2 to 5 a large number of times.
7. Compute the ensemble average of $\mathcal{O}$ by normalising with the number of times the code has passed through step 5. Also compute the errorbars.

A few comments are appropriate here. Firstly, in step 1 we could instead give each atom a random starting position in the simulation box but this would slow down the convergence since we would begin from an extremely unlikely starting configuration. For a solid going to a non-zero temperature below the melting point means that the atoms will vibrate around their equilibrium position and hence intialising the system using this configuration is a better approach. In steps 2 through 4 we make a transition according to the rule defined by Eqs. (4.32) and (4.35), and it is important to note that if the trial state is rejected we keep the old configuration which still contributes to the average in step 5.

If we are doing a simulation at non-zero T the starting configuration, no matter how well we chose it, will still be in an unlikely state and hence there is a certain *thermalisation* period before we start to sample the correct distribution. To adjust for this we can start by running the simulation without actually updating the averages. We call this the equilibration period and it usually extends over a number of steps corresponding to a few multiples of the thermalisation period. After equlibration when we are sure that the correct distribution is being sampled we start to record $\mathcal{O}(\mathbf{X})$. Sometimes when updating $\mathcal{O}(\mathbf{X})$ is an expensive operation in itself one typically waits until several moves have been accepted before updating the average, see step 5 above. It is convenient to divide an MC simulation into *cycles*, where one cycle constitutes N attempted one atom moves meaning that a trial movement has been proposed for each

atom in the system on average once. Properties $\mathcal{O}$ which are expensive to calculate are usually only updated once each MC cycle.

There is also the question of the free parameter δl, which controls the maximum length of the trial displacement in any direction. If we choose a very small δl we will find that more moves tend to be accepted since the change in potential energy will not be as big. This might lead to slow convergence since the atoms are only allowed to take very small steps toward the optimal configuration. We might also worry that we are not sampling the correct distribution since unlikely moves will be accepted more frequently as well. Conversely, if we choose δl very large, we will find that only a small number of moves are accepted and in this case we might not be exploring a large enough region of phase space in order to get a reliable estimate of the ensemble average. To be on the safe side it is therefore common practice to tune the δl parameter so that roughy half of the trial moves are accepted.

4.5 The pair correlation function

As an important example of a property that can be calculated using the canonical Monte Carlo Method we will consider the *pair correlation function* g, which describes the distribution of atomic pairs in the system. More precisely, in the canonical ensemble we define the pair correlation as the integral

$$g(\mathbf{x}_1,\mathbf{x}_2) = \frac{V^2(N-1)}{NZ} \int d\mathbf{x}_3 d\mathbf{x}_3 \dots d\mathbf{x}_N e^{-\beta \mathcal{V}(\mathbf{X})} \tag{4.37}$$

This equation can be cast in an equivalent form that shows clearly how to calculate g in a MC simulation. Integrating over the remaining two coordinates to get an ensemble average and introducing δ-functions to compensate we see that, since the choice of $\mathbf{x}_1$ and $\mathbf{x}_2$ is arbitrary provided the system in monatomic, Eq. (4.37) can be rewritten

$$g(r) = \frac{1}{\rho^2} \left\langle \sum_i \sum_{j \neq i} \delta(\mathbf{x}_i)\delta(\mathbf{x}_j - \mathbf{x}) \right\rangle = \frac{V}{N^2} \left\langle \sum_i \sum_{j \neq i} \delta(\mathbf{x} - \mathbf{x}_{ij}) \right\rangle \tag{4.38}$$

with $r = |\mathbf{x}|$. From Eq. (4.38) it is clear that $g(r)$ can easily be calculated using the canonical MMC method, all we have to do is to discretise the δ-function and then average the sum $\frac{V}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \delta(r - |\mathbf{x}_i - \mathbf{x}_j|)$ over a large number of configurations. To do this we loop over all atomic pairs and calculate the corresponding distances while collecting all the occurences in a histogram. A Fortran 90 program for calculating $g(r)$ has been implemented and can be found in appendix B. Section 6 contains some examples of $g(r)$ for our $BaZrO_3$ system.

CHAPTER 5

QUANTUM STATISTICAL MECHANICS AND PATH INTEGRALS

Thus far we have studied atomic systems from a classical viewpoint using the language statistical mechanics. As mentioned in the introduction, a classical treatment of the nuclear motion is not always a valid assumption. In general, quantum effects are mostly important when the temperautre is low and the system consists of lighter atoms such as hydrogen or helium. However, when there are competing structures with small energy differences such as in e.g. barium zirconate which has AFD phases it is not obvious that quantum effects can be neglected. In this section we will show how to treat the nuclear motion using quantum statistical mechanics, the first point of order to develop the density operator formalism. The density operator contains in principle all information we need about the system (it is analogous to the propagator) and by expanding this into a so called *path integral* we will be able to compute it numerically.

5.1 The density operator

In Section 4.1 we saw that in classical statistical mechanics the fundamental quantitity is the probability distribution which can be integrated to yield the probability of finding the system in a certain region of phase space. With knowledge of the probability density one can proceed to define e.g. thermodynamical quantities or transport properties as averages over phase space. In quantum statistical mechanics the density operator plays a similar role and generalises the concept of a wave function. More precisely, the density operator describes statistical mixtures of states. As an easy example consider a spin-$\frac{1}{2}$ system and suppose that we know that with probability P_1 the state is $|\uparrow\rangle$ and with probability $P_2 = 1 - P_1$ the state is $|\downarrow\rangle$. We call this a *statistical mixture* of $|\uparrow\rangle$ and $|\downarrow\rangle$ states or simply a *mixed state*, while e.g. $|\uparrow\rangle$ is referred to as a *pure state*. Note that a statistical mixture is fundamentally different from a superposition of $|\uparrow\rangle$ and $|\downarrow\rangle$ inasmuch as any such linear combination will have a definite spin direction. To

be able to describe statistical mixtures of states we will introduce the density operator formalism.

A statistical mixture of states is specified by a set of state kets $\{|\psi_i\rangle\}$ with corresponding probabilities P_i. The density matrix is defined as

$$\rho = \sum_i P_i |\psi_i\rangle\langle\psi_i|. \tag{5.1}$$

If the weights P_i in the above equation are to be interpreted as probabilities they must satisfy the normalisation condition $\sum_i P_i = 1$ which gives a corresponding normalisation condition for the density operator itself

$$\mathrm{tr}(\rho) = 1. \tag{5.2}$$

We are often interested in determining the average of an observable $\mathcal{O}$ since these are the values one can measure. For a pure state this average would simply be the expectation value of $\mathcal{O}$ but for a mixed state the natural definition is to weigh the expectation values of $\mathcal{O}$ taken relative to states $|\phi_i\rangle$ with the corresponding probabilities P_i and then sum over all such terms:

$$\langle\mathcal{O}\rangle = \sum_i P_i \langle\psi_i|\mathcal{O}|\psi_i\rangle. \tag{5.3}$$

Eq. (5.3) is the quantum mechanical equivalent to the classical ensemble average Eq. (4.1). To illustrate the usefulness of the density operator we will now show that it can be used to calculate the ensemble average Eq. (5.3). To see how this is done we will rewrite Eq. (5.3) in an arbitrary basis $\{|b\rangle\}$ using the completeness relation $\sum_b |b\rangle\langle b| = 1$:

$$\begin{aligned}\langle\mathcal{O}\rangle &= \sum_b \sum_{b'} \left(\sum_i P_i \langle b|\psi_i\rangle\langle\psi_i|b'\rangle \right) \langle b'|\mathcal{O}|b\rangle \\ &= \sum_b \sum_{b'} \langle b|\rho|b'\rangle\langle b'|\mathcal{O}|b\rangle = \mathrm{tr}(\rho\mathcal{O})\end{aligned} \tag{5.4}$$

where we recognise the expression enclosed by paratheses in the first line as the matrix element of the density operator written in the basis $\{|b\rangle\}$. Often the use of a basis is implied and one then simply refers to ρ as the density matrix. Equation Eq. (5.4) states that in order to calculate the ensemble average of $\mathcal{O}$ we simply need to take the trace of the matrix product of ρ and $\mathcal{O}$ and since the trace is independent of the basis we can work in whatever basis happens to be convenient.

5.2 The thermal density matrix

Consider a system in thermal equilibrium at temperature T where the number of particles N and volume V are constant. In what follows all the particles in our system are

assumed to be distinguishable so we can disregard bose and fermi statistics. Furthermore, suppose the exact energy eigenstates and eigenvalues are $|\phi_n\rangle$ and E_n i.e.

$$\mathcal{H}|n\rangle = E_n|n\rangle. \tag{5.5}$$

From statistical mechanics we know that the appropriate probability distribution for this type of system is the canonical distribution which gives the probability of finding the system in a state with energy E_n as

$$P_n = \frac{e^{-\beta E_n}}{Z} \tag{5.6}$$

where the normalising constant $Z = \sum_n e^{-\beta E_n}$ is the partition function and $\beta = \frac{1}{k_B T}$ as usual. The density operator can then be written

$$\rho = \sum_n \frac{e^{-\beta E_n}}{Z}|n\rangle\langle n| = \frac{e^{-\beta\mathcal{H}}}{Z}. \tag{5.7}$$

In literature on quantum statistical mechanis is somewhat of a convention to omit the partition function Z in the above equation and refer to

$$\rho = e^{-\beta\mathcal{H}} \tag{5.8}$$

as the thermal density operator. This convention will be followed for the rest of this report. As an imidiate consequence of Eq. (5.8), note that the partition function is related to the thermal density matrix [1] through $Z = \mathrm{tr}(\rho)$ and the average $\langle\mathcal{O}\rangle = \frac{1}{Z}\mathrm{tr}(\rho\mathcal{O})$. With these results one can proceed to identify familiar thermodynamic quantitites with operator averages. To illustrate this consider the internal energy E of the system which is identified with $\langle\mathcal{H}\rangle$:

$$\begin{aligned} E = \langle\mathcal{H}\rangle = \frac{1}{Z}\mathrm{tr}(\rho\mathcal{H}) &= \sum_n \left(\frac{E_n e^{-\beta E_n}}{e^{-\beta E_n}}\right) \\ &= \sum_n \left(-\frac{\partial}{\partial\beta}\ln\left(e^{-\beta E_n}\right)\right) \\ &= -\frac{\partial}{\partial\beta}\ln(Z). \end{aligned} \tag{5.9}$$

This should be a familiar formula to any student of thermodynamics. When performing calculations, especially on a computer, one often works in the position-space representation. The thermal density matrix matrix element at temperature T is then a continuous function

$$\rho(\mathbf{X}, \mathbf{X}'; \beta) = \langle\mathbf{X}|\rho|\mathbf{X}'\rangle \tag{5.10}$$

[1] Since unnormalised weights are being used for the density matrix the trace is no longer unity.

where $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ are the coordinates of all the N particles in the system. It is sometimes useful to express the thermal density matrix using the eigenenergy wavefunctions $\phi_n(\mathbf{x}) = \langle \mathbf{x}|n\rangle$, Eq. (5.10) then reads

$$\rho(\mathbf{X}, \mathbf{X}'; \beta) = \sum_n \phi(\mathbf{X})^* \phi(\mathbf{X}') e^{-\beta E_n}. \tag{5.11}$$

Using the completeness relation for the position base kets

$$\int d\mathbf{X} |\mathbf{X}\rangle\langle\mathbf{X}| = 1 \tag{5.12}$$

the average of an observable $\mathcal{O}$ in position-space becomes

$$\langle \mathcal{O} \rangle = \mathrm{tr}(\rho\mathcal{O}) = \frac{1}{Z} \int d\mathbf{X} d\mathbf{X}' \rho(\mathbf{X}, \mathbf{X}'; \beta) \langle \mathbf{X}'|\mathcal{O}|\mathbf{X}\rangle. \tag{5.13}$$

Similarily, the parition function is written

$$Z = \mathrm{tr}(\rho) = \int d\mathbf{X} \langle \mathbf{X}|\rho|\mathbf{X}\rangle. \tag{5.14}$$

Before moving on to the much anticipated path integrals we will consider a simple, yet important, example of a thermal density matrix, namely that of a *free particle* in a cubic box with side L and periodic boundary conditions. From elementary quantum mechanics we know that the solutions to the Hamiltonian eigenvalue problem are plane waves $\phi_{\mathbf{k}}(\mathbf{x}) = \frac{1}{\sqrt{V}} e^{i\mathbf{k}\cdot\mathbf{r}}$ where the wave vectors are given by $\mathbf{k} = \frac{2\pi}{L}\mathbf{n}$ with $\mathbf{n}$ an integer vector. The associated eigenenergy values are $E_{\mathbf{k}} = \frac{\hbar^2 \mathbf{k}^2}{2m}$. Together with Eq. (5.11) these results allow us to write the free particle density matrix

$$\rho(\mathbf{x},\mathbf{x}'; \beta) = \frac{1}{V} \sum_{\mathbf{k}} e^{-\beta\lambda\mathbf{k}^2} e^{-i\mathbf{k}\cdot(\mathbf{x}-\mathbf{x}')} \tag{5.15}$$

where $\lambda = \frac{\hbar^2}{2m}$ if we write the $\hbar$ explicitly[2]. This expression can be simplified somewhat if we assume that the thermal de Broglie wavelength $\Lambda \equiv \sqrt{\beta\lambda} \ll L$. Λ is approximately the de Broglie wavelength of the particle at temperature T and we are thus assuming that this wavelength is much smaller than the size of the box. This means that the $\mathbf{k}$-values are densely packed in the $\mathbf{k}$-space volume $(2\pi)^3/V$ and the sum in Eq. (5.15) can be replaced by an integral $\frac{1}{V}\sum_{\mathbf{k}} \longrightarrow \int \frac{d\mathbf{k}}{(2\pi)^3}$. Note that this approximation only becomes exact in the limit of an inifinite box and thus fails to account for the periodic boundary conditions. Carrying out the integration the result is

$$\rho(\mathbf{x},\mathbf{x}'; \beta) = \frac{1}{(4\pi\lambda\beta)^{3/2}} \exp\left(\frac{(\mathbf{x}-\mathbf{x}')^2}{4\lambda\beta}\right). \tag{5.16}$$

The generalisation to N free particles is trivial:

[2]Here we briefly deviate from the use of atomic units so the reader can get some sense of where the $\hbar$ factor will appear in the final equation

$$\begin{aligned}\rho(\mathbf{X},\mathbf{X}';\beta) &= \frac{1}{(4\pi\lambda\beta)^{3N/2}}\exp\left(\frac{(\mathbf{X}-\mathbf{X}')^2}{4\lambda\beta}\right)\\ &= \frac{1}{(4\pi\lambda\beta)^{3N/2}}\exp\left(\sum_{i=1}^{N}\frac{(\mathbf{x}_i-\mathbf{x}_i')^2}{4\lambda\beta}\right)\end{aligned} \tag{5.17}$$

5.3 Path integrals

Before doing anything, let us assess our current situation. In order to incorporate quantum effects into our simulations we have been forced to abandon the region of classical statistical mechanics and venture into the quantum regime. The thermal density matrix in principle contains all the information necessary in order to calculate any desired property of the system. We were also able to calculate the thermal density matrix in the simple case of a system of N free particles confined to a cubic box but it is clear (since solving for the density matrix of an N-particle system is at least as complicated as solving the N-particle Schrödinger equation) that we, in general, cannot hope to find analytical solutions. As so often happens, one is thus forced to resort to numerical methods. There are plenty available but we shall focus on a particularily powerful method known as the *Path Integral Monte-Carlo* method, or PIMC for short. The starting point of the PIMC method is the expansion of the density matrix in a path integral. The result will be a seemingly much more complicated expression than what we have encountered so thus far. By serendipity however, it turns out that the path integral expression is very amendable to numerical calculations. The brief presentation of the subject found here is based on a more detailed account by Ceperley [2] to which the curious reader is referred. Before getting into the details let us record one essential property of the density matrix:

$$e^{-(\beta_1+\beta_2)\mathcal{H}} = e^{-\beta_1\mathcal{H}}e^{-\beta_2\mathcal{H}} \tag{5.18}$$

i.e. the taking the product of two density matrices we obtain a new density matrix at a lower temperature. To write Eq. (5.18) in the position-space representation, apply the completeness relation Eq. (5.12) between the two density matrices in the right-hand side of Eq. (5.18):

$$\begin{aligned}\rho(\mathbf{X}_1,\mathbf{X}_3;\beta_1+\beta_2) &= \langle\mathbf{X}|\rho(\beta_1)\rho(\beta_1)|\mathbf{X}''\rangle\\ &= \int d\mathbf{X}_2\rho(\mathbf{X}_1,\mathbf{X}_2;\beta_1)\rho(\mathbf{X}_2,\mathbf{X}_3;\beta_2)\end{aligned} \tag{5.19}$$

Now, imagine we would like an expression for the density matrix at temperature T, by making repeated use of the property Eq. (5.18) this density matrix can be written as a product of M density matrices at a higher temperature $M\times T$ i.e.

$$e^{-\beta\mathcal{H}} = \left(e^{-\tau\mathcal{H}}\right)^M \tag{5.20}$$

where $\tau = \frac{\beta}{M}$. To write this in position-space one simply repeats the process used to reach Eq. (5.19): connect all the density operators on the right-hand side of Eq. (5.20) by integrals using the completeness relation and then squeeze the result in between a bra and a ket to obtain a matrix element. The resulting expression for the thermal density matrix at temperature T is

$$\rho(\mathbf{X}_0,\mathbf{X}_M;\beta) = \int d\mathbf{X}_1 d\mathbf{X}_2 \cdots d\mathbf{X}_{M-1} \rho(\mathbf{X}_1,\mathbf{X}_2;\tau)\rho(\mathbf{X}_2,\mathbf{X}_3;\tau)\cdots\rho(\mathbf{X}_{M-1},\mathbf{X}_M;\tau) \tag{5.21}$$

This is our basic path integral. It is customary to refer to $\tau = \frac{\beta}{M}$ as the time step and to $\mathbf{X}_k = \{\mathbf{x}_{1k},\mathbf{x}_{2k},\ldots,\mathbf{x}_{Nk}\}$ as the k:th time slice. Two neighbouring slices are separated by a time step τ and hence we can assign a time $t_k = k\tau$ to slice $\mathbf{X}_k$. From this viewpoint, Eq. (5.21) is a "sum" over discrete paths where a path is essentially a trajectory of the system: $\{\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_M\}$. Note that referring to τ as the time step can be somewhat misleading since we are not actually talking about the real time in the quantum system and has the units of inverse energy. The terminology derives from the pure state QM path integrals where one derives an expression identical to Eq. (5.21) for the propagator of the system, in which case one would really be dealing with a time step.

The expression Eq. (5.21) looks very complicated but, as it turns out, it is possible to find a reasonably accurate and cheap-to-compute approximation to the density matrix if the temperature is high. This means that if we choose M so that MT is large enough we can replace all the density matrices on the right-hand side of Eq. (5.21) using this approximate expression and (hopefully) still obtain a reasonable approximation to the low temperature density matrix on the left-hand side. Thus, while all results have been exact up to this point we will now introduce some approximations. The Hamilton operator is the sum of the kinetic energy operator and a potential energy operator $\mathcal{H} = \mathcal{T} + \mathcal{V}$. From the Baker-Campbell-Hausdorff lemma it follows that for two operator A and B:

$$e^A e^B = e^{A+B+\frac{1}{2}[A,B]+\text{H.O.T}}. \tag{5.22}$$

Letting $A = -\tau\mathcal{T}$ and $B = -\tau\mathcal{V}$ in Eq. (5.22) yields

$$e^{-\tau\mathcal{T}} e^{-\tau\mathcal{V}} = e^{-\tau(\mathcal{T}+\mathcal{V})+\frac{\tau^2}{2}[\mathcal{T},\mathcal{V}]+\text{H.O.T}}. \tag{5.23}$$

Thus, neglecting all commutators between $\mathcal{T}$ and $\mathcal{V}$ we have the following approximation

$$e^{-\tau(\mathcal{T}+\mathcal{V})} \approx e^{-\tau\mathcal{T}} e^{-\tau\mathcal{V}} \tag{5.24}$$

Since this approximation means throwing away a term that is second order in τ we can improve the accuracy by increasing M .In path integral litterature Eq. (5.24) is referred to as the *primitive approximation*. We can now approximate the density matrix at temperature MT:

$$\rho(\mathbf{X},\mathbf{X}'';\tau) \approx \langle \mathbf{X}|e^{-\tau\mathcal{T}}e^{-\tau\mathcal{V}}|\mathbf{X}''\rangle = \int d\mathbf{X}' \langle \mathbf{X}|e^{-\tau\mathcal{T}}|\mathbf{X}'\rangle\langle\mathbf{X}'|e^{-\tau\mathcal{V}}|\mathbf{X}''\rangle \tag{5.25}$$

As usual, we are working in the canonical ensemble with a system of distinguishable particles confined to a cubic box with periodic boundary conditions, it follows that the first matrix element on the right-hand side of Eq. (5.25) is the free particle density matrix given by Eq. (5.17):

$$\langle \mathbf{X}|e^{-\tau\mathcal{T}}|\mathbf{X}'\rangle \approx \frac{1}{(4\pi\lambda\tau)^{3N/2}} \exp\left(\frac{(\mathbf{X}-\mathbf{X}')^2}{4\lambda\tau}\right). \tag{5.26}$$

The "$\approx$" sign serves to remind us that Eq. (5.26) holds if the thermal de Broglie wavelength $\Lambda_\tau \equiv \sqrt{\tau\lambda} \ll L$. Now assume that $\mathcal{V}$ and $\mathcal{H}$ commute (this is usually the case), and can thus be simultaneously diagonalised, so that $\mathcal{V}|\mathbf{X}\rangle = \mathcal{V}(\mathbf{X})|\mathbf{X}\rangle$. For the second matrix element on the right-hand side of Eq. (5.26) this means

$$\langle \mathbf{X}'|e^{-\tau\mathcal{V}}|\mathbf{X}''\rangle = e^{-\tau\mathcal{V}(\mathbf{X}'')}\delta(\mathbf{X}'-\mathbf{X}''). \tag{5.27}$$

Upon substituting Eqs. (5.27) and 5.26 into Eq. (5.25) one finds that

$$\rho(\mathbf{X},\mathbf{X}'';\tau) \approx \frac{1}{(4\pi\lambda\tau)^{3N/2}} e^{\frac{(\mathbf{X}-\mathbf{X}'')^2}{4\lambda\tau}} e^{-\tau\mathcal{V}(\mathbf{X}'')}. \tag{5.28}$$

This is the desired approximation to the high temperature density matrix, valid in the primitive approximation Eq. (5.24). Returning to Eq. (5.21) we can now write this path integral in the primitive approximation using Eq. (5.28) to replace all the matrix elements on the right-hand side of Eq. (5.21), the result is

$$\begin{aligned}\rho(\mathbf{X}_0,\mathbf{X}_M;\beta) \approx & \frac{1}{(4\pi\lambda\tau)^{3NM/2}} \int d\mathbf{X}_1 d\mathbf{X}_2 \cdots d\mathbf{X}_{M-1} \cdots \\ & \exp\left(-\tau\left(\sum_{k=1}^{M} \frac{(\mathbf{X}_k-\mathbf{X}_{k-1})^2}{4\lambda\tau^2} + \mathcal{V}(\mathbf{X}_k)\right)\right)\end{aligned} \tag{5.29}$$

Since we have now made use of the approximation Eq. (5.28) a total of M times there seems to be a possibility of accumulating an error when M becomes large. This is not the case however and in fact taking the limit $M \to \infty$ the error goes to zero. This is expressed in the Trotter formula [15]

$$e^{-\beta(\mathcal{T}+\mathcal{V})} = \lim_{M\to\infty} \left(e^{-\frac{\beta}{M}\mathcal{T}} e^{-\frac{\beta}{M}\mathcal{V}}\right)^M. \tag{5.30}$$

To simplify our notation we will refer to the two neighbouring time slices $(\mathbf{X}_{k-1},\mathbf{X}_k)$ as the k:th link and then define the *action* of this link as

$$\mathcal{S}_{k-1,k} = \left(\frac{(\mathbf{X}_k-\mathbf{X}_{k-1})^2}{4\lambda\tau} + \tau\mathcal{V}(\mathbf{X}_k)\right). \tag{5.31}$$

The total action is then obtained by summing over all the links: $S = \sum_{k=1}^{M} \mathcal{S}_{k-1,k}$. Note that the specific form of the action given in Eq. (5.31) is dependent on the approximation made in Eq. (5.24). The general definition of the action is $\mathcal{S}_{k-1,k} =$

$-\ln\left(\rho(\mathbf{X}_{k-1},\mathbf{X}_k)\right)$ and one then refers to the expression Eq. (5.31) as the *primitive action*.

With Eq. (5.31), we can write Eq. (5.29) in the shorthand notation

$$\rho(\mathbf{X}_0,\mathbf{X}_M;\beta) \approx \frac{1}{(4\pi\lambda\tau)^{3NM/2}} \int d\mathbf{X}_1 d\mathbf{X}_2 \cdots d\mathbf{X}_{M-1} \exp\left(-\sum_{k=1}^{M} .\mathcal{S}_{k-1,k}\right) \tag{5.32}$$

In particular, the partition function becomes

$$Z \approx \frac{1}{(4\pi\lambda\tau)^{3NM/2}} \int d\mathbf{X}_1 d\mathbf{X}_2 \cdots d\mathbf{X}_{M-1} d\mathbf{X}_M \exp\left(-\sum_{k=1}^{M} \mathcal{S}_{k-1,k}\right)\Bigg|_{\mathbf{X}_0=\mathbf{X}_M} \tag{5.33}$$

The point here is that looking at the integrands in Eqs. (5.32) and (5.33), we see that they remain positive at all temperatures and are very reminiscent of a configurational intergral for a classical system, see Section 4.1. This suggests the use of Monte-Carlo simulations in order to compute the integrals.

5.4 The classical isomorphism

By careful interpretation of Eqs. (5.32) and (5.33) it turns out that we can map our quantum system to a classical polymer system. Burrowing some terminology from the mathematical theory of abstract spaces this map is referred to as an *isomorphism*. To see how such a connection can arise, we need to take a closer look at the different interactions in the quantum system. The kinetic part of the interaction found in the exponential in Eq. (5.29) is

$$\sum_{k=1}^{M} \frac{(\mathbf{X}_k - \mathbf{X}_{k-1})^2}{4\lambda\tau^2} = \sum_{k=1}^{M}\sum_{i=1}^{N} \frac{(\mathbf{x}_{ik} - \mathbf{x}_{ik-1})^2}{4\lambda\tau^2} = \sum_{i=1}^{N}\left(\sum_{k=1}^{M} \frac{(\mathbf{x}_{ik} - \mathbf{x}_{ik-1})^2}{4\lambda\tau^2}\right). \tag{5.34}$$

Looking closely at the expression enclosed by parantheses in Eq. (5.34) we can see that it has the same form as the potential for a set of particles $\{\mathbf{x}_{ik}\}_{k=1}^{M}$ where each particle $\mathbf{x}_{ik}$ is coupled to its closest neighbours $\mathbf{x}_{ik-1}$ and $\mathbf{x}_{ik+1}$ by springs. Hence we can think of each atom i as a sort of polymer consisting of M particles which we refer to as *beads*. We should keep in mind that a set of beads $\{\mathbf{x}_{ik}\}_{k=1}^{M}$ really represents the coordinates of a single atom i in the different time slices. As we saw an example of in Section 5.2, many thermodynamical quantities are directly related to the partition function. In this case we need to take the trace of the density matrix and only diagonal elements are involved. The same is true for quantities that are diagonal in position-space like the pair correlation function. We must thus take the trace in Eq. (5.29) i.e. integrate over $\mathbf{X}_M$ and let $\mathbf{X}_0 = \mathbf{X}_M$ in the integrand. Correspondingly in the polymer system, the first and last bead belonging to a particular chain must be identical, resulting in

a *ring polymer*. The words polymer and chain will be used interchangeably and a ring polymer is thus sometimes referred to as a closed chain. Later on we will also encouter open polymers which will also be called open chains.

There is also a potential interaction in the polymer system, from Eq. (5.29) we see that this interaction has the form

$$\sum_{k=1}^{M} \mathcal{V}(\mathbf{X}_k) = \frac{1}{2} \sum_{k=1}^{M} \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \mathcal{V}_2(\mathbf{x}_{ik} - \mathbf{x}_{jk}) \right). \tag{5.35}$$

In the ring polymer system we interpret this as a potential interaction between equally labeled beads belongning to different atoms. Hence the potential interaction in the quantum system translates to a rather peculiar potential interaction in the polymer system, where only beads in same time slice as allowed to interact. The different interactions in the polymer system are represented pictorially in Fig. 5.1.

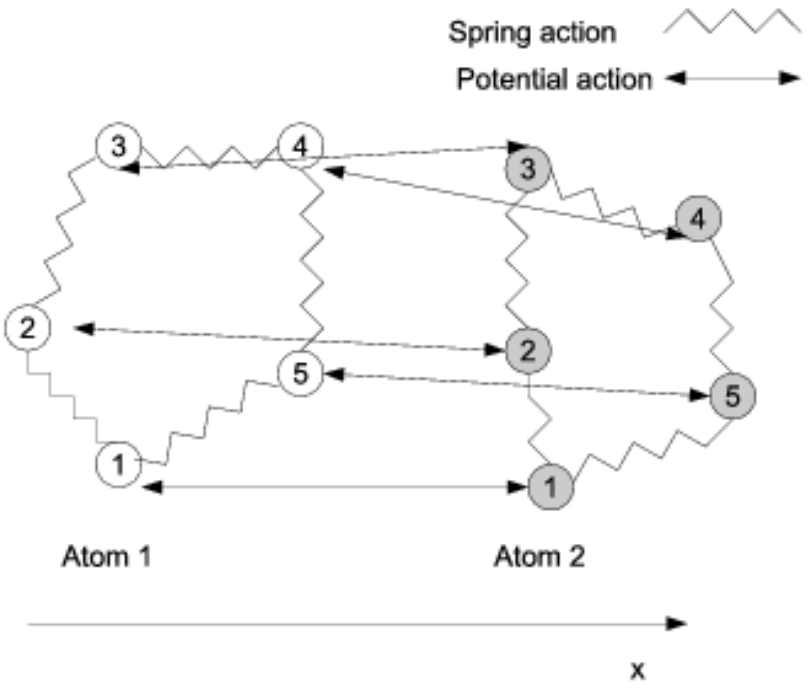


Figure 5.1. *An illustration of the classical isomorphism in one dimension. The figure depicts two atoms, each represented as a polymer consisting of five beads. The kinetic interaction between neighbouring beads belonging to the same atom is represented by springs. The potential interaction between beads which are in the same time slice (as indicated by the numbers) but belong to different atoms is represented by double arrows. Also note that according to Eq.* (5.33) *the first and last bead are connected so that each atom is represented by a ring polymer.*

With the classical isomorphism we realise that in order to compute the path integral

we can just do a classical Metropolis Monte Carlo simulation for a ring polymer system with interactions described by Eqs. (5.34) and (5.35). A more thorough explanation of the algorithm will be given in following section. As a final remark, the assumption of distinguishable particles that we have made makes the situation considerably easier. Including bose statistics would further complicate the picture of the paths given above while taking fermi statistics into account comes with an infamous sign problem, see [2].

5.5 The path integral Monte Carlo method

We will now describe in detail how to perform Monte-Carlo simulations to compute configurational integrals in the path integral formalism. As we saw in Section 5.4 the quanutm system can be mapped to classical ring polymer system with interactions defined by Eqs. (5.34) and (5.35). It therefore suffices to do a classical canonical Monte-Carlo simulation of this polymer system, which means that only a few modifications to the algorithm presented in Section 4.4 are necessary. Firstly, since each atom is now represented by M beads, there are NM beads total and $3NM$ coordinates that we need to store. Note that since we have a ring polymer system the first and last slice are identical i.e. $\mathbf{X}_0 = \mathbf{X}_M$ and hence we only store one of them which explains why we use M beads rather than $M+1$. For the trial moves, the simplest approach is to propose a displacement for one bead at a time in exactly the same way as we did for the atoms in the classical simulation. The major difference instead comes during the rejection step since there is now a new harmonic interaction given by Eq. (5.34) that we must calculate and take into account. Consider then moving a single bead located at $\mathbf{x}_{ik}^{\text{old}}$ to a new position $\mathbf{x}_{ik}^{\text{new}}$. In PIMC simulation it is customary to refer to the action or difference in action rather than the interaction energy, the distinction being that one multiplies the interaction energy by τ to obtain the corresponding action [3]. To compute the change in potential action we need only look at slice k since the potential action is local to each slice according to Eq. (5.35), thus we get

$$\Delta S_{\text{pot}} = \tau \left(\mathcal{V}(\mathbf{X}_k \leftarrow \mathbf{x}_{ik}^{\text{new}}) - \mathcal{V}(\mathbf{X}_k \leftarrow \mathbf{x}_{ik}^{\text{old}}) \right). \tag{5.36}$$

Note that much like in Section 4.4, if we have a pair potential there is no need to compute the potential action for the entire configuration in the current slice since not all beads are moving. We can thus save a lot of effort by computing the potential action difference as

$$\Delta S_{\text{pot}} = \tau \sum_{j \neq i}^{N} \left[\mathcal{V}_2(\mathbf{x}_{ik}^{\text{new}} - \mathbf{x}_{jk}) - \mathcal{V}_2(\mathbf{x}_{ik}^{\text{old}} - \mathbf{x}_{jk}) \right]. \tag{5.37}$$

For the kinetic part of the action we need only look at the current atom i to which the bead with label ik belongs, since according to Eq. (5.34) only beads belonging to the

[3] Note that the action quantity as defined here is dimensionless since τ has the unit of inverse energy (rather than having units of energy $\times$ time).

same atom are connected by springs. The bead ik is then a part of two links, namely $(\mathbf{x}_{ik-1},\mathbf{x}_{ik})$ and $(\mathbf{x}_{ik},\mathbf{x}_{ik+1})$ and the change in potential action is therefore

$$\Delta S_{\text{kin}} = \frac{1}{4\lambda\tau}\Big[(\mathbf{x}_{ik}^{\text{new}} - \mathbf{x}_{ik-1})^2 + (\mathbf{x}_{ik}^{\text{new}} - \mathbf{x}_{ik+1})^2 - \Big(\mathbf{x}_{ik}^{\text{old}} - \mathbf{x}_{ik-1}\Big)^2 - \Big(\mathbf{x}_{ik}^{\text{old}} - \mathbf{x}_{ik+1}\Big)^2\Big]. \tag{5.38}$$

Whenever $k = 1$ in Eq. (5.49) we simply set $k - 1 = M$ in agreement with the ring polymer condition and conversely if $k = M$ we set $k + 1 = 1$. Now, the total change in action is $\Delta S = \Delta S_{\text{pot}} + \Delta S_{\text{kin}}$ and hence the correct Boltzmann factor to use when testing whether to accept or reject the proposed move is $\exp(-\Delta S)$. For clairty's sake all the steps in the Path integral Monte-Carlo method are summarised below.

The Path integral Monte Carlo method

1. Initialise the postions of all the NM beads. For a liquid or a solid such as $BaZrO_3$ an appropriate choice would be to place all the beads belonging to the same atom at the coordinate that atom would have in the $T = 0$ equilibrium configuration.
2. Choose at random a slice k and an an atom i. Propose a trial movement for the bead $\mathbf{x}_{ik}$ by displacing it symmetrically according to $\mathbf{x}_{ik}^{\text{new}} = \mathbf{x}_{ik}^{\text{old}} + \delta l(2\boldsymbol{\eta} - 1)$ where $\boldsymbol{\eta}$ is random vector drawn from the uniform distribution on [0,1].
3. Compute the diffence in potential action between the new and old configuration: $\Delta S_{\text{pot}} = \tau\left(\mathcal{V}(\mathbf{X}_k \leftarrow \mathbf{x}_{ik}^{\text{new}}) - \mathcal{V}(\mathbf{X}_k \leftarrow \mathbf{x}_{ik}^{\text{old}})\right)$
4. Compute the diffence in kinetic action between the new and old configuration:

$$\Delta S_{\text{kin}} = \frac{1}{4\lambda\tau}\Big[(\mathbf{x}_{ik}^{\text{new}} - \mathbf{x}_{ik-1})^2 + (\mathbf{x}_{ik}^{\text{new}} - \mathbf{x}_{ik+1})^2 - \Big(\mathbf{x}_{ik}^{\text{old}} - \mathbf{x}_{ik-1}\Big)^2 - \Big(\mathbf{x}_{ik}^{\text{old}} - \mathbf{x}_{ik+1}\Big)^2\Big]$$

5. If $\Delta\mathcal{S} = \Delta S_{\text{pot}} + \Delta S_{\text{kin}} \leq 0$ the trial state is immediately accepted and the old coordinate can be deleted. If $\Delta\mathcal{S} > 0$ generate a uniform random number q in [0,1]:
   - 5.1. If $q \leq \exp(-\Delta\mathcal{S})$ the trial state is accepted and the old coordinate can be deleted.
   - 5.2. Else if $q > \exp(-\Delta\mathcal{S})$ the trial state is rejected and the atom's original position must be restored.
6. After one or several beads have been moved, update the average value of any property of interest $\mathcal{O}$ using the newly generated configuration. Note that all moves contribute equally to the average i.e. regardless of whether the old position was restored or the trial state was accepted. Also update the error.
7. Repeat steps 2 to 6 a large number of times.
8. Compute the ensemble average of $\mathcal{O}$ by normalising with the number of times the code has passed through step 6. Also compute the errorbars.

A few comments are appropriate here. In the first step we must choose initial positions for all the beads in the path. In Section 4.4 we saw that in the canonical Monte-Carlo Method one usually initialises a solid in the $T = 0$ equilibrium configuration. Extending this idea to a path integral Monte-Carlo simulation we simply put all the beads representing the same atom in the $T = 0$ equilibrium position for that atom. One could conceivably displace the beads randomly around these equilibrium positions but this will in general only slow down the convergence. The reason is that the kinetic spring action Eq. (5.34) limits the distance between neighbouring beads to within the order of the thermal wavelength, larger separations quickly become unfavourable energetically. Regarding the choice of the number of beads M to use in the simulation we can conclude from Eq. (5.30) that using a larger number of beads gives a better approximation of the quantum mechanical effects. Comparing the PIMC method above and the classical MMC method it is also clear that for $M = 1$ the PIMC method reduces to the MMC method. Typically, the number of beads actually used in a PIMC simulation must be determined from a convergence study where one tries to determine what value M yields an asymptotic value for a quantity of interest, see Section 6.4. For a classical MMC simulation we defined a MC cycle as N attempted moves and analogously we will define a PIMC cycle as MN attempted moves meaning that a trial movement has been proposed for each bead in system on average once. Just as in the classical MMC we have an adjustable parameter δl controlling the maximum distance in any direction a bead can be displaced. For the same reasons mentioned in Section 4.4 this parameter is tuned so that roughly half the proposed movements are accepted.

5.6 Estimators

In the path integral formalism there are frequently several different ways of calculating the same property and we refer to these as different *estimators*. In theory, each estimator naturally gives the same result, but in practice they may not be equally well suited for numerical calculations e.g. one estimator might give a significantly higher variance than another. In this thesis we shall mainly be concerned with computing pair correlation functions as well as position and momentum distributions, the later of which are treated separately in Section 5.8. To illustrate the concept of estimators we will briefly discuss how to compute the total internal energy of the system. This will be useful when studying the convergence of the Monte-Carlo simulations, see Chapter 6. The simplest estimator for the energy is the *thermodynamic estimator* which is based on the relation given in Section 5.2 Eq. (5.9):

$$E = \langle \mathcal{H} \rangle = \frac{1}{Z} \mathrm{tr}(\rho \mathcal{H}) \tag{5.39}$$

In the path integral formalism with the primitive approximation one can show [2] that this leads to

$$E = \left\langle \frac{3}{2\tau} - \sum_{k=1}^{M} \frac{(\mathbf{X}_k - \mathbf{X}_{k-1})^2}{4M\lambda\tau^2} - \frac{1}{M}\sum_{k=1}^{M} \mathcal{V}(\mathbf{X}_k) \right\rangle. \tag{5.40}$$

In a PIMC simulation we would thus compute the sums inside the mean in Eq. (5.40) and then average over a large number of generated configurations in order to obtain the total internal energy E. It is also possible to derive other estimators for the energy, one way is to use the quantum mechanical version of the virial therorem leading to the virial estimator.

For the pair correlation function $g(r)$, no major modifications to the method described in Section 4.5 are necessary, we simply have to average over each time slice as well, leading to the following estimator

$$g(r) = \left\langle \frac{V}{MN^2} \sum_{k=1}^{M} \sum_{i=1}^{N} \sum_{j \neq i} \delta(\mathbf{x}_{ik} - \mathbf{x}_{jk} - \mathbf{x}) \right\rangle. \tag{5.41}$$

As a final remark, note that the form of any particular estimator is dependent on the choice of approximation for the action which is in turn decided by the approximation in Eq. (5.24). There are other approximations to the action with a higher order of accuracy such as the *Li-Broughton action* which we will not consider in this report. The reader should be aware however that the form Eq. (5.40) is valid only in the primitive approximation.

5.7 Improving the sampling

In the simplest form of the PIMC method described in the previous section new paths are sampled by moving each bead individually with a displacement proposed the same way as in the canonical Monte-Carlo Method. We call this type of move a *single slice* move. Unfortunetaly, there is a fundamental problem with only using single slice moves for sampling the path. To get accurate results we want to use as many beads as possible i.e. M should be a large integer in which case the time step $\tau = \beta/M$ would be very small. Now, since two neighbouring bead are effetively coupled by a spring with a certain stiffness, we find that their relative motition is restricted to a scale set by the thermal de Broglie wavelength $\Lambda = \sqrt{\lambda\tau}$. This wavelength decreases with the number of beads and consequently the rate of convergence decreases as we increase M in order to make the calculations more accurate. Another situation where using only single slice is not satisfactory is when there exist alternate stable configurations involving specific movements of entire atoms, which is the certainly the case for perovskite oxides with AFD phases.

To remedy these inadequacies of the single slice moves we will introduce two new types of moves in this section. *Centre of mass displacements* are useful for converging the potential action and exploring AFD configurations et cetera. *Bisection moves* can be used to replace the single slice moves and improves the convergence considerably by

samlping the kinetic action exactly and using the potential action to pre-reject unlikely configurations early on.

5.7.1 Centre of mass displacements

The most straightforward way to improve the sampling is to introduce centre of mass displacements (COM), which are the PIMC equivalent to a one-atom move in a classical MC simulation. One atom is chosen randomly and all beads belonging to that atom are subsequently displaced using the same uniform random displacement for each bead. Note that the inter-bead distances are preserved by this type of move and hence it can be visualised as displacing an entire polymer ring by a fixed distance through its centre of mass. COM moves are particularly useful for converging the potential part of the action Eq. (5.35) and in addtion are very easy to implement. As with the single slice moves there is a free parameter that is tuned to maintain a fifty-percent acceptance ratio for the COM moves.

5.7.2 The Bisection Algorithm

Another problem with single slice moves not mentioned in the introduction to this section is the question of sampling the kinetic part of the action. This part of the action converges very slowly and typically sets the rate of convergence for systems with a weak potential interaction, e.g. systems described by a Lennard-Jones potential. In a bisection move, the kinetic action is sampled exactly, thus removing entirely the need to calculate and Metropolis test the kinetic action. In addition, each move involves several beads and is constructed in such a way that it has a very high probability of being accepted.

More precisely, in a bisection move we randomly choose an atom i for which we wish to sample a new path. A part of the path is then cut out to be sampled and we refer this partial path as a *clip*. The sampling procudure is then partitioned into bisection *levels* where the number of beads included in the clip N_c is determined by the maximum level $l_{\max}$ according to $N_c = 2^{l_{\max}} + 1$. To make things concrete let us consider the case when $l_{\max} = 3$ so that $N_c = 9$, if we assume the clip begins at bead 1 the following beads are included in the clip: $\{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{i9}\}$. Beginning at the *coarsest* level $l = 3$ only the midpoint $\mathbf{x}_{i5}$ of the clip is sampled and the new coordinate must be chosen in a way such that the kinetic action (i.e. the free particle density matrix) is sampled exactly. One can show analytically [2] that this condition is satisfied if we choose

$$\mathbf{x}_{i5}^{\text{new}} = \frac{\mathbf{x}_{i1} + \mathbf{x}_{i9}}{2} + \boldsymbol{\zeta}\sqrt{4\tau\lambda} \tag{5.42}$$

where $\boldsymbol{\zeta}$ is a normally distributed random vector with mean zero and unity variance. In this level one thus "pretends" that the clip only consists of the three beads $\{\mathbf{x}_{i1}, \mathbf{x}_{i5}, \mathbf{x}_{i9}\}$ separated by a time step 4τ. We then proceed recursively to sample the remaining beads except for the endpoints $\{\mathbf{x}_{i1}, \mathbf{x}_{i9}\}$ which remain fixed. In level $l = 2$ we sample new midpoints $\mathbf{x}_{i3}$,$\mathbf{x}_{i7}$ for the two subclips $\{\mathbf{x}_{i1}, \mathbf{x}_{i3}, \mathbf{x}_{i5}\}$ and $\{\mathbf{x}_{i5}, \mathbf{x}_{i7}, \mathbf{x}_{i9}\}$ according to

$$\mathbf{x}_{i3}^{\text{new}} = \frac{\mathbf{x}_{i1} + \mathbf{x}_{i5}}{2} + \boldsymbol{\zeta}\sqrt{2\tau\lambda} \tag{5.43}$$

and

$$\mathbf{x}_{i7}^{\text{new}} = \frac{\mathbf{x}_{i5} + \mathbf{x}_{i9}}{2} + \boldsymbol{\zeta}\sqrt{2\tau\lambda} \tag{5.44}$$

Note that in Eqs. (5.42) and (5.43) the last term on the right-hand side now contains a factor 2τ instead of 4τ since the time step separating the beads $\{\mathbf{x}_{i1},\mathbf{x}_{i3},\mathbf{x}_{i5},\mathbf{x}_{i7},\mathbf{x}_{i9}\}$ in this level is only 2τ. The bottom of the recursion is reached at the *finest* level $l = 3$ where new midpoints $\mathbf{x}_{i2},\mathbf{x}_{i4},\mathbf{x}_{i6},\mathbf{x}_{i8}$ must be sampled for the 4 new subclips obtained after the last completed bisection level. The time step separating two neighbouring beads is now τ and the sampling of the 4 midpoints is otherwise completely analogous to Eqs. (5.43) and (5.44). The entire recursive sampling process is illustrated in Fig. 5.2, here beads that do not move in the current level are displayed in gray while the displacement for the beads that are sampled is indicated by arrows.

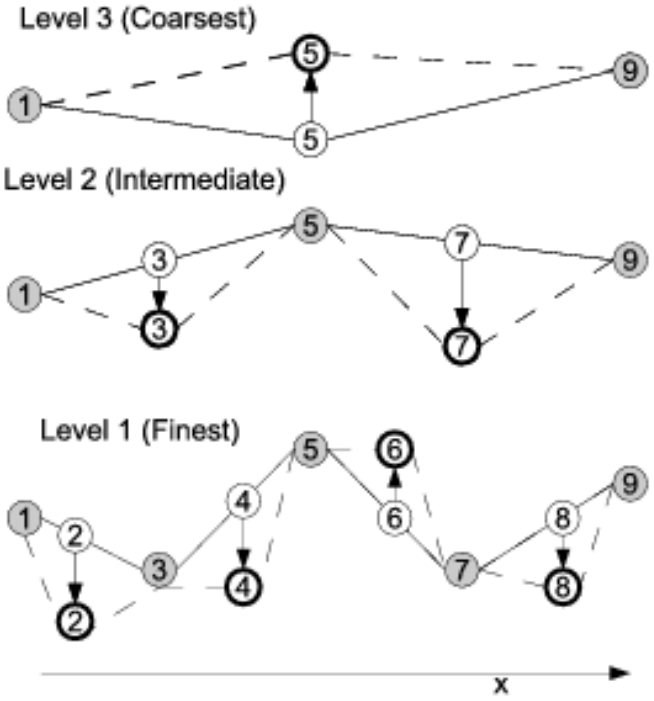


Figure 5.2. *An illustration of the bisection process. Beads that are not sampled in the current level are displayed in gray while moving beads are displayed in white with arrows indicating the new positions. Note the recursive nature of the procedure where midpoints are succesively sampled as new subintervals are created.*

In this way, the free-particle density matrix is exactly sampled and we do not have to worry about the kinetic action. Rather, a bisection move must be either accepted or rejected based on the potential action. The obvious way to go about this would be to recursively sample the whole clip as described above and then proceed to do a Metropolis test of the entire newly generated clip. While there is no fault in this method there is a considerable increase in efficiency to be gained by rejecting unlikely configurations already at the coarser levels $l = 3{,}2$ based on the potential action. It is easy to see why this must be true since beads closer to the endpoints, which are already themselves in a favourable position, are only moved a little bit while the midpoint could potentially move a great deal more and hence it is likely that the change in potential action is unfavourable here. Consequently, at level $l = 3$ we calculate

$$\Delta\mathcal{V}(l=3) = \mathcal{V}(\mathbf{X}_5 \leftarrow \mathbf{x}_{i5}^{\text{new}}) - \mathcal{V}(\mathbf{X}_5) \tag{5.45}$$

and do a Metropolis test with the Boltzmann factor: $\exp\left[-4\tau\Delta\mathcal{V}(l=3)\right]$. Similarly for the finer levels, noting that the time step in level l is $2^{l-1}\tau$ and the beads can be indexed $k = 1 + ns$ with the step $s = 2^{l-1}$ and $n = 1, \ldots, 2^{(l_{\max}-l-1)} - 1$, we compute

$$\Delta\mathcal{V}(l) = \sum_{n=1}^{2^{(l_{\max}-l-1)}-1} \Big[\mathcal{V}(\mathbf{X}_{1+ns} \leftarrow \mathbf{x}_{i1+ns}) - \mathcal{V}(\mathbf{X}_{1+ns})\Big] \tag{5.46}$$

and accept or reject based on the Boltzmann factor: $\exp\left[-2^{l-1}\tau\Delta\mathcal{V}(l) + 2^{l}\tau\Delta\mathcal{V}(l+1)\right]$. Note that Eq. (5.46) is completely general and holds regardless of the maximum number of levels $l_{\max}$. The bisection move will only be accepted if all levels are accepted meaning that should one of the coarser bisections be rejected before we reach $l = 1$ the whole bisection process is aborted. Hence, no time is wasted considering improbable moves that will almost certainly be rejected at the finer levels. Also note that in the finest level the exact action is computed while in the coarser levels the sampling and Metropolis testing is effectively done at a different temperature since the time step is larger than τ. An implementation of the bisection algorithm in Fortran 90 can be found in appendix B. We also provide a more general step-by-step description of the algorithm in which step 2 to 5 in the original PIMC algorithm described in Section 5.5 are to be replaced by the following:

The bisection algorithm

1. Choose an atom i at random and a random starting bead k_1 for the clip. Define the clip by $\left\{\mathbf{x}_{ik_1}, \mathbf{x}_{ik_2}, \ldots, \mathbf{x}_{ik_{N_c}}\right\}$ where $N_c = 2^{l_{\max}} + 1$ and taking the ring polymer condition into account. The maximum level $l_{\max}$ is set before the simulation is performed and must not give a clip size that exceeds the number of beads i.e. it should satisfy $N_c < M$.
2. Loop over levels starting at the coarsest level $l = l_{\max}$ and proceeding down to $l = 1$. In each level l do the following:

2.1. Sample $2^{l_{\max}-l}$ new midpoints:

$$\mathbf{x}_{ik}^{\text{new}} = \frac{\mathbf{x}_{ik-s} + \mathbf{x}_{ik+s}}{2} + \boldsymbol{\zeta}\sqrt{2^{l-1}\tau\lambda}$$

for $k = 1 + 2ms$ with the step $s = 2^{l-1}$ and $m = 1, \ldots, 2^{l_{\max}} - l$. If $l = l_{\max}$ save the old positions in case the bisection is later rejected.

2.2. Calculate the difference in potential action:

$$\Delta\mathcal{S}_{\text{pot}} = 2^{l-1}\tau\Delta\mathcal{V}(l) = 2^{l-1}\tau \sum_{n=1}^{2^{(l_{\max}-l-1)}-1} \Big[\mathcal{V}(\mathbf{X}_{1+ns} \leftarrow \mathbf{x}_{i1+ns}) - \mathcal{V}(\mathbf{X}_{1+ns})\Big]$$

with $s = 2^{l-1}$ and $n = 1, \ldots, 2^{(l_{\max}-l-1)} - 1$.

2.3. If $\Delta\mathcal{S}_{\text{pot}} \leq 0$ the level is accepted and we proceed to the next level. If $\Delta\mathcal{S}_{\text{pot}} > 0$ generate a uniform random number q in [0,1]. If $q \leq \exp(-\Delta\mathcal{S}_{\text{pot}})$ the level is accepted, else if $q > \exp(-\Delta\mathcal{S}_{\text{pot}})$ the entire bisection move is rejected and the old positions must be restored.

3. If all levels were accepted, the entire bisection move is accepted and old coordinates the in path should be replaced with the newly sampled coordinates in the clip.

5.8 The momentum distribution

An important property of a solid or a liquid is the *momentum distribution*, classically this is given by the Maxwell distribution which is independent of the potential. In the quantum case there is no such uncoupling however, and quantum effects related to the nuclear motion e.g. zero-point energy fluctuations and tunneling can significantly alter the momentum distribution, thus making it a desireable object to calculate. One can also measure the momentum distribution experimentally through neutron Compton scattering experiements [11] allowing for comparison between experimental and simulation results.

Before defining the momentum distribution, consider the *single particle density matrix* $\rho(\mathbf{x}_1,\mathbf{x}_1';\beta)$, which can be obtained from the many-body density matrix by integrating out the dependence on the coordinates of all atoms but one:

$$\rho(\mathbf{x}_1,\mathbf{x}_1';\beta) = \int d\mathbf{x}_2 \ldots d\mathbf{x}_N \langle \mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N|\rho|\mathbf{x}_1',\mathbf{x}_2,\ldots,\mathbf{x}_N\rangle. \tag{5.47}$$

In Section 5.3 an arbitrary matrix element of the density matrix in position-space was expanded into a path integral, see Eq. (5.29). Applying this expansion to the matrix element on the right-hand side of Eq. (5.47) and then interpreting the result through the semi-classical isomorphism we find that this corresponds to a polymer system where with $N-1$ ring polymer chains and one open chain representing atom 1. We then define the end-to-end distance distribution of the open chain according to

$$n(\mathbf{R}) = \int d\mathbf{x}_1 d\mathbf{x}_1' \delta\left(\mathbf{x}_1 - \mathbf{x}_1' - \mathbf{R}\right) \rho(\mathbf{x}_1,\mathbf{x}_1';\beta) \tag{5.48}$$

which can be interprested as a measure of the uncertainity of the atom's position. Note that in order to calculate the single-particle density matrix $n(\mathbf{x}_1,\mathbf{x}_1')$ one needs access to off-diagonal density matrix elements as is evident from Eq. (5.47). The momentum distribution is now defined as the fourier transform of the single particle density matrix

$$n(\mathbf{k}) = \frac{1}{(2\pi)^3} \int d\mathbf{x}_1 d\mathbf{x}_1' n(\mathbf{x}_1,\mathbf{x}_1').e^{-i\mathbf{k}\cdot(\mathbf{x}_1-\mathbf{x}_1')} \tag{5.49}$$

Using Eq. (5.48) an equivalent form is

$$n(\mathbf{k}) = \frac{1}{(2\pi)^3} \int d\mathbf{R} n(\mathbf{R}) e^{-i\mathbf{k}\cdot\mathbf{R}}. \tag{5.50}$$

A dramatic example of how quantum effects can change the momentum distribution occurs in superfluid helium where the momentum distribution does not go to zero with increasing k as expected classically. Instead the distribution tends towards a finite value which is related to the condensate fraction of the superfluid phase [2]. A more relevant example for this thesis is the symmetric double-well potential which has two equivalent minima separated by a potential barrier. Suppose that the temperature is low so that the ground state is the most probable and the barrier height is such that the lowest energy level is located below the barrier, depending on the barrier height a particle may not have enough energy to surmount the barrier but can still tunnel through it with a certain probability. The spatial wave function is then bimodal and extends over a distance corresponding to the separation of the equivalent minima. As the temperature increases the particle's energy will increase until it can surmount the barrier (in a classical sense) and this effect will soon come to dominate the tunneling effect if the temperature continues to increase. Another possibility is that the barrier is so low that the lowest energy level is located above the barrier meaning that zero-point motion has effectively erased the barrier. The resulting wavefunction would then be unimodal and the particle is simultaneously occupying the two equivalent sites. By the Heisenberg uncertainity principle this would show as a narrowing of the momentum distribution compared to the case where either one or the other site is occupied. Another feature which is indicative of tunneling but can also be observed when the barrier is erased by zero-point motion is a node and subsequent tail at the end of the momentum distribution [7].

5.9 Algorithms for computing for the momentum distribution

In a simulation one typically calculates $n(R) = n(|\mathbf{R}|)$ in Eq. (5.48) or just the radial part $n(r)$ and then takes the fourier transform in order to obtain $n(k)$. As we mentioned in the

previous section, calculating the single particle density matrix by definition requires off-diagonal density matrix elements. These cannot be obtained from an ordinary PIMC simulation such as was described in Section 5.5, where only the diagonal part of the density matrix was required, leading to the ring polymer condition. An off-diagonal element of the density matrix, on the other hand, corresponds to an open polymer chain which makes the situation much more complicated. Indeed, if we wish to allow one or more chains to be open, one cannot calculate diagonal properties in the same simulation and additional code is required to account for the open chains. This is entirely the case in the *open chain method*, which is the most common method for calculating $n(R)$. While the implementation is relatively simple this method has a few shortcomings to be discussed later. Another more recent method is the *trail method* introduced in [3] which allows for the momentum distribution to be calculated in an all-closed (i.e. ring) polymer simulation. Both methods are covered below.

5.9.1 The open chain method

The most straightforward way to calculate $n(R)$ is the *open chain method*. From Eq. (5.47) and *Eq.* (5.48) it follows directly that $n(R)$ can be calculated as an ensemble average in a polymer system consisting of $N-1$ ring polymer chains and one open chain. The estimator for $n(R)$ is then

$$n(R) = \left\langle \delta\left(\left|\mathbf{x}_1 - \mathbf{x}_1'\right| - R\right) \right\rangle \tag{5.51}$$

During the simulation, the ring polymers are treated in exactly the same way as before, see Section 5.5. Assuming that the polymer representing atom 1 is to be open, we introduce a new bead $\mathbf{x}_{1M+1}$ representing the endpoint of this chain which now consists of the beads $(\mathbf{x}_{11}, \mathbf{x}_{12} \ldots, \mathbf{x}_{1M}, \mathbf{x}_{1M+1})$ and move this new bead in usual way during the Monte Carlo simulation, only taking into account that the chain is open when calculating the kinetic action since the two endpoints only have one neighbour. For each newly generated configuration, the end-to-end distance $|\mathbf{x}_{11} - \mathbf{x}_{1M+1}|$ is recorded and the resulting histogram of these distances must then be directly proportional to $n(R)$. One of the major drawbacks of this method is that no diagonal properties can be calculated in the same simulation since these require all chains to be closed as described in Section 5.4. Furthermore, it turns out that the endpoints of the open chain tend to favour larger separations meaning that the single-particle density matrix is undersampled for smaller separations. To remedy this it is possible to artificially introduce a bias to the sampling but we shall not pursue the matter further, more details can be found in [2]. Below we include a modified step-by-step adaption of the PIMC method described in Section 5.5 to the new polymer system with one open chain:

The open chain method

1. Choose one atom i' to be open and introduce an extra bead $M+1$ with coordinate $\mathbf{x}_{i'M+1}$ desribing the endpoint of the open polymer chain. Perform steps 1 through 3 of regular PIMC algorithm described in Section 5.5 where a random atom i is

symmetrically displaced and the difference in potential action is calculated. The new bead $M+1$ is treated exactly the same as all other beads.

2. Compute the diffence in kinetic action between the new and old configuration, if $i \neq i'$

$$\Delta S_{\text{kin}} = \frac{1}{4\lambda\tau}\left[(\mathbf{x}_{ik}^{\text{new}} - \mathbf{x}_{ik-1})^2 + (\mathbf{x}_{ik}^{\text{new}} - \mathbf{x}_{ik+1})^2 - \left(\mathbf{x}_{ik}^{\text{old}} - \mathbf{x}_{ik-1}\right)^2 - \left(\mathbf{x}_{ik}^{\text{old}} - \mathbf{x}_{ik+1}\right)^2\right].$$

However if $i = i'$ there are three different cases depending on which bead k is displaced. If $k \neq 1$ and $k \neq M+1$ we use the equation above for ΔS_{kin}. However, if $k = M+1$ then

$$\Delta S_{\text{kin}} = \frac{1}{4\lambda\tau}\left[\left(\mathbf{x}_{i'M+1}^{\text{new}} - \mathbf{x}_{i'M}\right)^2 - \left(\mathbf{x}_{i'M+1}^{\text{old}} - \mathbf{x}_{i'M}\right)^2\right]. \tag{5.52}$$

and similarily if $k = 1$ then

$$\Delta S_{\text{kin}} = \frac{1}{4\lambda\tau}\left[(\mathbf{x}_{i'1}^{\text{new}} - \mathbf{x}_{i'2})^2 - \left(\mathbf{x}_{i'1}^{\text{old}} - \mathbf{x}_{i'2}\right)^2\right]. \tag{5.53}$$

3. If $\Delta\mathcal{S} = \Delta S_{\text{pot}} + \Delta S_{\text{kin}} \leq 0$ the trial state is immediately accepted and the old coordinate can be deleted. If $\Delta\mathcal{S} > 0$ generate a uniform random number q in [0,1]:
   - 3.1. If $q \leq \exp(-\Delta\mathcal{S})$ the trial state is accepted and the old coordinate can be deleted.
   - 3.2. Else if $q > \exp(-\Delta\mathcal{S})$ the trial state is rejected and the atom's original position must be restored.
4. After each bead has been moved on average once, redcord the end-to-end distance $|\mathbf{x}_{i'1} - \mathbf{x}_{i'M+1}|$ in a histrogram.
5. Repeat steps 2 to 6 a large number of times.
6. Normalise the histogram of the end-to-end distances to obtain $n(R)$ and the perform a fourier transform to obtain $n(k)$ as well.

5.9.2 The trail method

In the trail method one performs a standard PIMC simulation with N closed polymers, just like we described in Section 5.5. To calculate the single-particle density matrix the simulation is then halted after each MC cycle and during this time a polymer chain is cut open and each bead is displaced. The ratio of the density matrices of the displaced chain to the undisplaced one is then added to a histogram which is normalised at the end of the simulation, yielding directly the singe-particle density matrix. The trail method is inspired by a similar algorithm known as the *MacMillan method* [2] [3], the

difference lies in how a chain is displaced after it is cut open during a halted simulation. Assume that we choose the path representing atom 1 to become momentarily open and cut this path at an arbitrary bead $\mathbf{x}_{1k}$, which thus makes up one end of the now open chain. In the MacMillan method an extra coordinate representing the other end of the chain is then introduced and sampled from the free-particle distribution, while the other beads remain at their original positions. Note that already here we restrict ourselves to sampling distances on the order of the thermal wavelength because of the effective spring constant imposed by the kinetic action. In the trail method this problem is circumvented by instead displacing each bead an amount proportional to the distance from bead to the point where the chain was cut. The two endpoints of the chain thus always have the largest relative displacement while two neighbouring beads have a small relative displacement so that the springs connecting them are not stretched in an improbable manner. To be more precise, assume we cut the chain at an arbitray point which now makes up one end of the open chain and we label it k_1 as a reference. The rest of the beads are then labled starting from this incision point and we introduce a new bead k_{M+1} to represent the other end of the chain. The *trail displacement* of the chain now proceeds according to

$$\mathbf{x}_{ik_\alpha}^{\text{new}} = \mathbf{x}_{ik_\alpha}^{\text{old}} + \frac{\delta_t(\alpha-1)}{M}(2\boldsymbol{\eta}-1) \tag{5.54}$$

for $\alpha = 1, \ldots, M+1$ with $\boldsymbol{\eta}$ a uniform random vector on [0,1]. Note that the bead where we cut the chain is not displaced since we assumed this to be the $k_{\alpha=1}$ bead meanwhile the displacement has a maximum value at the other end of the chain where $k_{\alpha=M+1}$. In Eq. (5.54) the length scale of the displacement is controlled by the trail displacement parameter δ_t. Clearly this parameter is at least of the same order as the thermal wavelength, but depending on the system it can be significantly larger. Now, let $\Delta\mathcal{S}_{\text{kin}}$ denote the difference in kinetic action between the displaced and undisplaced chain and let $\Delta\mathcal{S}_{\text{pot}}$ be the difference in potential action. The ratio is the density matrices is thus given by $\exp\left[-(\Delta\mathcal{S}_{\text{kin}} + \Delta\mathcal{S}_{\text{pot}})\right]$ and the estimator for the single-particle matrix is simply the average:

$$n(\mathbf{x}_{1k_1}, \mathbf{x}_{1k_{M+1}}) = \left\langle e^{(-(\Delta\mathcal{S}_{\text{kin}}+\Delta\mathcal{S}_{\text{pot}}))} \right\rangle. \tag{5.55}$$

We also provide a general step-by-step description of the algorithm in which the following operations are to be perfomed at step 6 of the PIMC simulation described in Section 5.5. Note that we use the superscripts "old" and "new" to indicate whether a coordinate refers the the original (old) position of the bead or the new position in the displaced chain.

The Trail method

1. Choose a random atom i and a random bead k_1 in the ring polymer chain representing i (which consists of M beads) as the point of inciscion. The chain is now considered to be open with one end defined by bead k_1 and the other end by a new bead k_{M+1} which is introduced and intialised so that $\mathbf{x}_{ik_1}^{\text{old}} = \mathbf{x}_{ik_{M+1}}^{\text{old}}$.

2. Compute the old kinetic action of the open chain

$$\mathcal{S}_{\text{kin}}^{\text{old}} = \sum_{\alpha=1}^{M} \frac{(\mathbf{x}_{ik_{\alpha+1}}^{\text{old}} - \mathbf{x}_{ik_\alpha}^{\text{old}})^2}{4\lambda\tau}$$

and the old potential action:

$$\mathcal{S}_{\text{pot}}^{\text{old}} = \sum_{\alpha=1}^{M+1} \mathcal{V}(\mathbf{X}_{k_\alpha})$$

3. Perform a trail displacement of the open chain according to

$$\mathbf{x}_{ik_\alpha}^{\text{new}} = \mathbf{x}_{ik_\alpha}^{\text{old}} + \frac{\delta_t(\alpha-1)}{M}(2\boldsymbol{\eta} - 1)$$

with $\boldsymbol{\eta}$ a uniform random vector on [0,1]. An apporiate value of the trail displacement parameter δ_t must be chosen beforehand.

4. Compute the new kinetic action, i.e. that of the displaced open chain

$$\mathcal{S}_{\text{kin}}^{\text{new}} = \sum_{\alpha=1}^{M} \frac{(\mathbf{x}_{ik_{\alpha+1}}^{\text{new}} - \mathbf{x}_{ik_\alpha}^{\text{new}})^2}{4\lambda\tau}$$

and the new potential action:

$$\mathcal{S}_{\text{pot}}^{\text{old}} = \tau \sum_{\alpha=1}^{M+1} \mathcal{V}(\mathbf{X}_{k_\alpha} \leftarrow \mathbf{x}_{ik_\alpha}^{\text{new}}) \tag{5.56}$$

5. Calculate the ratio of the displaced to the undisplaced density matrix: $\exp\left[-\left(\Delta\mathcal{S}_{\text{pot}} + \Delta\mathcal{S}_{\text{kin}}\right)\right]$ where $\Delta\mathcal{S}_{\text{kin}} = \mathcal{S}_{\text{kin}}^{\text{new}} - \mathcal{S}_{\text{kin}}^{\text{old}}$ and $\Delta\mathcal{S}_{\text{pot}} = \mathcal{S}_{\text{pot}}^{\text{new}} - \mathcal{S}_{\text{pot}}^{\text{old}}$ and calculate the distance $\left|\mathbf{x}_{ik_1}^{\text{new}} - \mathbf{x}_{ik_{M+1}}^{\text{new}}\right|$. Add these values to two separate histograms: the action histogram and the distance histogram.
6. Repeat steps 1 through 5 a total N times so that each atom is, on average, open once before continuing the all-closed polymer simulation.

After a sufficient number of MC cycles have passed and the simulation is completed one then obtains the single-particle density matrix by normalising the action histogram using the distance histogram. To obtain the momentum distribution, it suffices to do a one-dimensional fourier transform of $n(r) = n(|\mathbf{x}_{11} - \mathbf{x}_{1M+1}|)$ as obtained from the normalised histogram. This concludes the first part of the thesis and we now have all the tools to analyse the structure of barium zirconate in the context of a full quantum mehcanical treatment of the nuclear motion.

CHAPTER 6

RESULTS

In this chapter we will present the results of our simulations, focusing on pair correlation functions and momentum distributions for the $BaZrO_3$ system. The first part of this chapter, however, will be dedicated to verifying that all the algorithms that have been implemented are working properly.

The source code for all our programs can be found in appendix B, here we will only provide a brief summary of the most important component programs needed to perform a PIMC simulation:

- `InitialiseLattice.f90`: Initialises the system. A $BaZrO_3$ supercell of a specified dimension is created and the atoms are placed in their equilibrium configuration. Arrays containing atomic type, charge and mass are also created.
- `Potential.f90`: Calculates the total interaction energy of a single atom in the field of all the other atoms. Note that this includes both the Buckingham potential which we described how to calculate thoroughly in Section 3.4 and the Ewald summation which was the topic of Section 3.5. For more details on how to implement the Ewald summation the reader is referred to appendices A and B.
- `Metropolis.f90`: Implements the PIMC method as described in Section 5.5. For the sampling of new paths we have implemented both single slice sampling as well as the bisection method (which is available in separate version of the Metropolis program). Note that both versions use COM moves to help find AFD configurations.
- `OpenChain.f90`: Implements the open chain algorithm as described in Section 5.9. Note that this program is very similar to the regular PIMC program, the exception being that one polymer chain is now open and diagonal properties cannot be calculated.
- `PairCorrelation.f90`: Used in conjunction with the `Metropolis.f90` subroutine to calculate the pair correlation function.
- `EnergyEstimator.f90`: Implements the thermodynamic estimator for the internal

energy, see Eq. (5.40). Used in conjunction with the `Metropolis.f90` subroutine.
- `TrailMethod.f90`: Implements the Trail method for calculating the single particle density matrix, see Section 5.9.2.

Some minor subroutines for writing data etc. have not been listed above. All post-processing of the simulation data has been carried out in MATLAB.

6.1 Verification part I: The potential

A good starting point is to verify the program responsible for calculating the potential i.e. both the short-range interaction given by the Bucking potential in Section 3.2 as well as the long-range coulomb interaction given by the Ewald summation in Section 3.5. As a first step to verifying that our program is calculating the potential correctly we will use it to calculate the equilibrium lattice constant for the $BaZrO_3$ system. For this purpose we will use the original set of parameters listed in Table 3.1, which have been reported to give an equilibrium lattice constant $a_0 = 4.188\,\text{Å}$ [4]. To test whether our program can reproduce this value we will use a simple method: Several $BaZrO_3$ supercells are initialised using different lattice parameters with values in a suitable interval. For each value of the lattice parameter we calculate the total interaction energy of the system, the correct equilibrium lattice paramter is then the value for which the total interaction energy is a minimum. The resulting curve is displayed in Fig. 6.1 where the interaction energy per unit cell is plotted as a function of the lattice parameter a_0.

Clearly from Fig. 6.1 the minimum in energy occurs somewhere in the vicinity of $a_0 \approx 4.2\,\text{Å}$ and by doing a careful scan with a small step around that value we find that the exact equilibrium lattice constant is $a_0 = 4.188\,\text{Å}$ in agreement with the value found in [4].

As a further check for the coulombic interaction and the implementation of the Ewald summation one can compute the Madelung constant which is a dimensionless value that depends only on the lattice structure and ionic charges and has been well documented for nearly all of the commonly occuring lattice types. Consider a lattice with N ions, in SI units the total electrostatic potential felt by an ion i in the field of the $N-1$ other ions is given by

$$\Phi_i = \frac{e}{4\pi\epsilon_0} \sum_{j \neq i} \frac{Z_j}{r_{ij}} \tag{6.1}$$

where $r_{ij} = |\mathbf{x}_i - \mathbf{x}_j|$ is the distance between atoms i and j and Z_j is the number of elementary charges on ion j. If we denote by d the nearest neighbour distance in the lattice the Madelung constant can be defined as

$$M_i = \sum_{j \neq i} \frac{Z_j}{r_{ij}/d}. \tag{6.2}$$

We can then write electrostatic potential $\Phi_i = \frac{e}{4\pi\epsilon_0}\frac{M_i}{d}$ and the corresponding electrostatic energy is

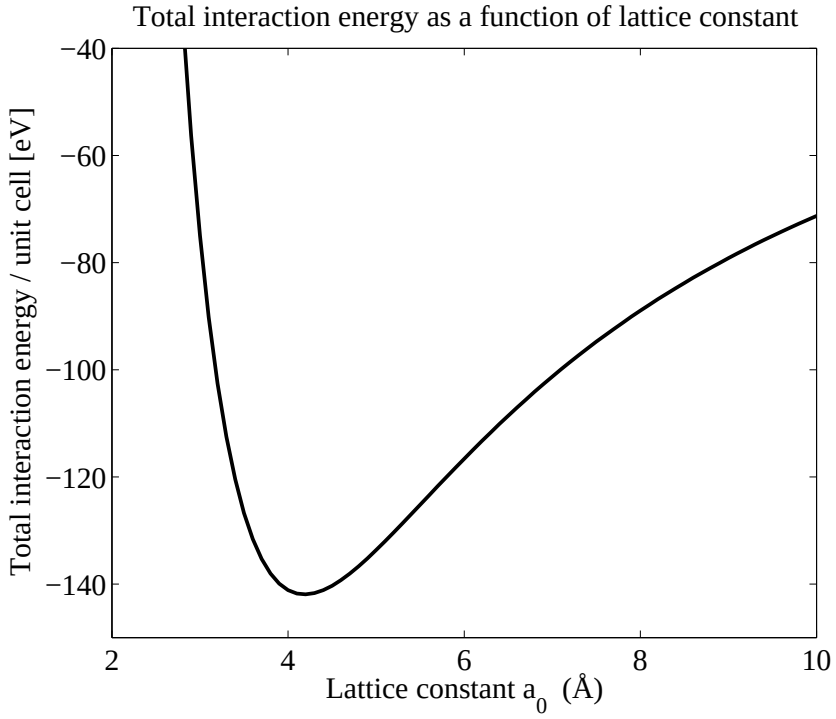


Figure 6.1. *The total interaction energy per unit cell as a function of lattice parameter a_0 for a 4×4×4 $BaZrO_3$ supercell.*

$$\mathcal{V}_i = \frac{e^2}{4\pi\epsilon_0}\frac{M_i}{d}. \tag{6.3}$$

The number of Madelung constants associated with a particular crystal structure depends on the number of different ions and the crystal symmetry. For $BaZrO_3$ we are looking at three different values: M_{Ba}, M_{Zr} and M_{O}. These three different constants can be combined to form a single, effective Madelung constant proportional to the total electrostatic energy of one $BaZrO_3$ formula unit. We define this effective Madelung constant [1] as

$$M_{\mathrm{eff}} = \frac{1}{2}\left(Z_{\mathrm{Ba}}M_{\mathrm{Ba}} + Z_{\mathrm{Zr}}M_{\mathrm{Zr}} + 3Z_{\mathrm{O}}M_{\mathrm{O}}\right) = \frac{1}{2}\left(2M_{\mathrm{Ba}} + 4M_{\mathrm{Zr}} - 6M_{\mathrm{O}}\right) \tag{6.4}$$

and consequently the total electrostatic energy per unit formula is $\mathcal{V}_{\mathrm{formula}} = M_{\mathrm{eff}}\frac{e^2}{d}$. Now, we can easily calculate M_{Ba}, M_{Zr} and M_{O} using our `Potential.f90` subroutine if we temporarily neglect all the contributions from the short-range Buckingham potential. The program can then output the pure electrostatic energy felt an arbitrary ion in the field of all the other ions and it is then a small matter to compute M_{eff} from Eq. (6.4) since e.g. $M_{\mathrm{Ba}} = \mathcal{V}_{\mathrm{Ba}}\frac{4\pi\epsilon_0 d}{e^2}$. We find that

[1] In litterature this is often just plainly referred to as the Madelung constant

$$M_{\text{eff}} = -24.7549 \tag{6.5}$$

which is in good agreement with e.g. the value $M_{\text{eff}} = -24.7550$ reported in [6]. All calculations presented in this section were performed on a 4×4×4 $BaZrO_3$ supercell containing a total of 320 ions. For the Buckingham potential a raidial cut-off $r_{\text{cut}} = 6\,\text{Å}$ was used while for the Ewald summation the tolerance was set to $\epsilon = 10^{-7}$. We note briefly that this tolerance factor is approximately an upper bound for the error and the values for the radial cut-offs in real and reciprocal space mentioned in Section 3.5 are then determined by ϵ. For a more detailed discussion of parameters used in the Ewald summation the reader is referred to appendix A.

In conclusion, our program for calculating potential accurately reproduces the known values of both the equlibrium lattice constant for $BaZrO_3$ as well as the Madelung constant for a perovskite structure system and proceed, reasonably condifident that the program is working as intended.

6.2 Model potentials and antiferrodistortive instabilities

We saw in Chapter 2 that in many perovskite structured oxides there are antiferrodistortive phases where the oxygen octahedra are tilted and where the tilting may be out-of-phase between neighbouring octahedron in one or more of the x,y and z-directions. The energy difference between these phases and the cubic strucutre is usually very small. In the case of $BaZrO_3$ it was also mentioned that calculations have given no conclusive indications as to whether such antiferrodistortive instabilites can be found or if the material remains cubic all the way down to $T = 0$. While we do not attempt to solve this issue in this thesis, what we have done is create two sets of Buckingham pair potential parameters for barium zirconate where one set describes a system with AFD instabilities while the other set given a cubic system and the purpose of this section is to show how this was done. In Chapter 2 Section 2.2 we briefly described a steric model which partly explains the origins of the octahedral tilting and saw that instabilities are related to the Goldschmidt ratio which in turn depends on the ionic radii. In this model tilting occurs because of a mismatch in size between the A and B cations in a perovskite. In the rigid-ion model with a Buckingham potential, the relative ionic radii are partly determined by the ρ parameters found in the exponential Pauli repulsion term:

$$\mathcal{V}(r) = Ae^{-\frac{r}{\rho}} - \frac{C}{r^6}. \tag{6.6}$$

This suggests that in order to go from a stable system to an unstable one or vice versa one can simply change the relative magnitudes of the ρ parameters for the Ba-O and the Zr-O interactions (while leaving the other parameters unchanged). A good starting point is to first investigate the original set of parameters values by Stokes and Islam given in Table 3.1 by looking at the phonon spectrum as mentioned in Section 2.2. The following analysis and results were kindly contributed by E. Fransson and J. Håkansson [16] who have calculated phonon spectra and analysed the resulting eigenmodes for

different sets of parameters ($\rho_{\text{Ba-O}}$,$\rho_{\text{Zr-O}}$). For the Stokes values $\rho_{\text{Ba-O}} = 0.3949\,\text{Å}^{-1}$ and $\rho_{\text{Zr-O}} = 0.3760\,\text{Å}^{-1}$ the resulting phonon spectrum is displayed in Fig. 6.2.

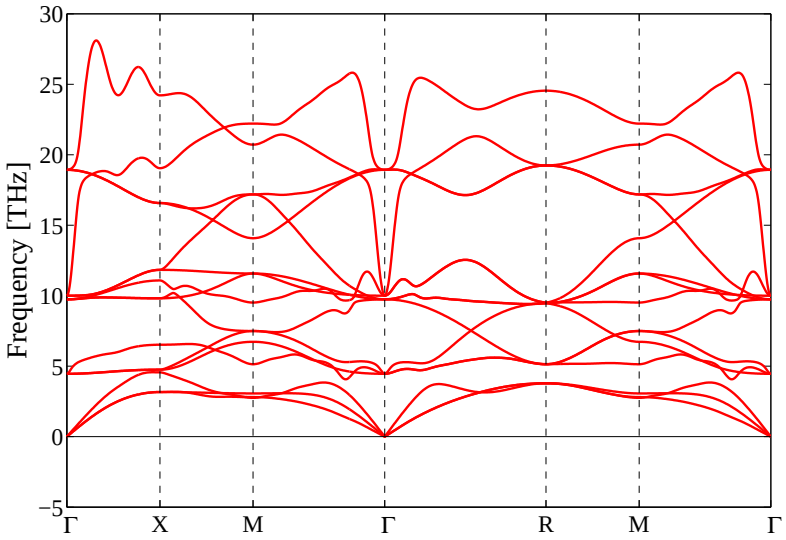


Figure 6.2. *Phonon spectrum for the pair potential described in Chapter 3. Phonon mode frequencies are displayed along lines of high symmetry in the brillouin zone. Note that there are no soft modes with imaginary frequency present indicating that the system is stable and will remain cubic.*

Here, the phonon mode frequencies are plotted along the usual high symmetry lines of the brillouin zone. Imaginary frequencies are represented as negative values in this figure but as we can see such soft phonon modes are absent, indicating that the crystal is stable. This leads us to conclude that the set of parameters determined for the Buckingham potential by Stokes and Islam yields a $BaZrO_3$ system without AFD instabilites. However, changing the size of the parameters to $\rho_{\text{Ba-O}} = 0.3820\,\text{Å}^{-1}$ and $\rho_{\text{Zr-O}} = 0.3885\,\text{Å}^{-1}$, which corresponds to a decrease in the Goldschmidt ratio, results in a quite different phonon spectrum shown in Fig. 6.3.

Observe the prescene of imaginary R- and M-point frequencies in Fig. 6.3 which are indicative of phase transistions that lowers the crystal symmetry. An analysis of the polarisation vectors revealed displacements corresponding to four different unstable modes: three R-point modes specified in glazer notation as $a^0b^-b^-$, $a^-b^-b^-$, $a^-a^-a^-$ and one M-point mode $a^0a^0c^+$. However, as it turns out, only AFD phases with $a^-a^-a^-$ tilts are actually observed during simulations and therefore we shall focus on this type of tilt. The absence of AFD phases with tilts specified by $a^0b^-b^-$, $a^-b^-b^-$ and $a^0a^0c^+$ is related to the depth and location of the energy minima as a function of rotation angle for the tilted structures, see Fig. 6.4 for an example of such a curve. Henceforth, we shall refer to the orginal set of parameters as the *cubic system* since in this case there

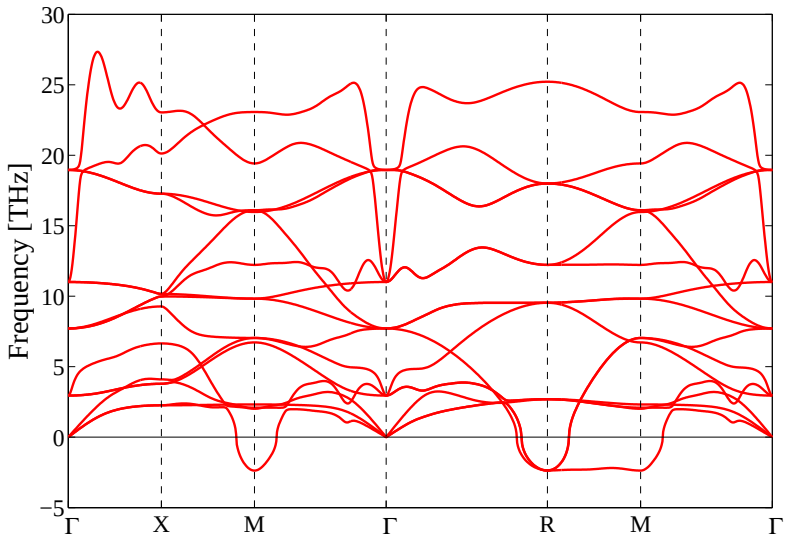


Figure 6.3. *Phonon spectrum for the pair potential with new values of $\rho_{Ba\text{-}O}$ and $\rho_{Zr\text{-}O}$ described above. This time there are soft phonon modes with imaginary frequency indicating a possible phase transition where the crystal symmetry is lowered.*

are no AFD instabilities according to Fig. 6.2 while the new set of parameters shall be referred to as the *tilt system*. For the sake of clarity we summarise these two model potentials below:

1. **The cubic system**: Uses the original set of Buckingham potential parameters found in Table 3.1 which yields a cubic system.
2. **The tilt system**: Uses the same parameters as the cubic system except for the values of $\rho_{\text{Ba-O}}$ and $\rho_{\text{Zr-O}}$ which are changed to $\rho_{\text{Ba-O}} = 0.3820\,\text{Å}^{-1}$ and $\rho_{\text{Zr-O}} = 0.3885\,\text{Å}^{-1}$. This system has AFD phases where the oxgen octahedra exhibit $a^-a^-a^-$ tilts.

Naturally, one also would like to know how close the unstable tilt system is to the original cubic structure in energy. This is easy to caculate since we already have a program for calculating the total energy of the system, meaning we only have to rotate the octhedra to establish a $a^-a^-a^-$ tilt and then record the difference in energy between the tilted and the original structure. These calculations have been carried out for different octahedral rotation angles in a small range and are displayed in Fig. 6.4. Since two neighbouring octahedra share a common oxygen atom, the tilting is not truly a rigid rotation but involves a small amount of distortion. Therefore, we have chosen a small displacement parameter Δ to characterise the magnitude of the tilt meaning each oxygen atom in an octahedron is displaced an amount Δ along a straight line in the direction necessary to establish an overall $a^-a^-a^-$-tilt. Keep in mind that changing the

parameters in the potential also changes the equilibrium lattice constant and for the new tilt system we find that $a_0 = 4.255\,\text{Å}$. Since the lattice constant and other properties have now changed slightly, one could think of this new model potential as describing some other cubic perovskite closely related to barium zirconate. The important point is that in this system, which still highly resembles the old one, there are now AFD phases.

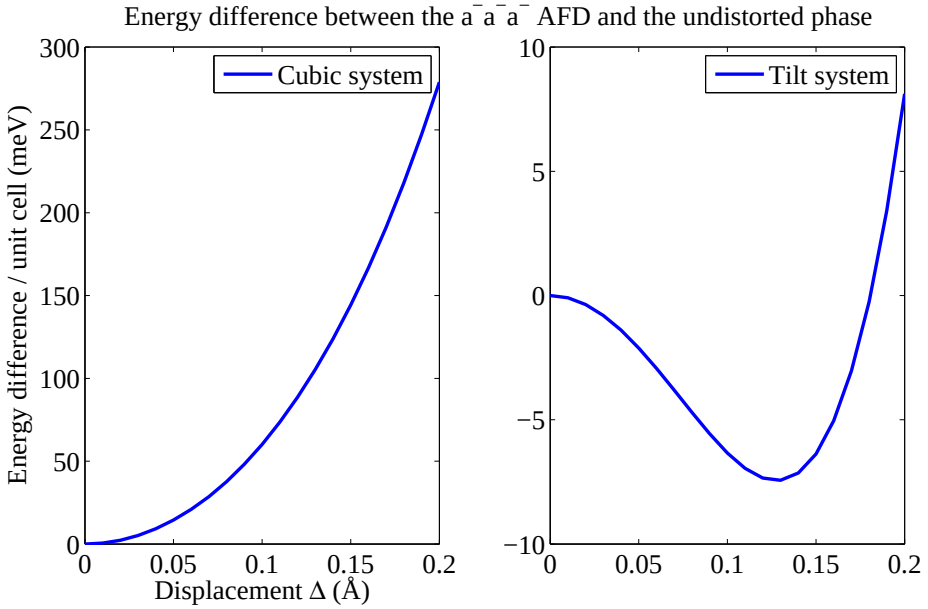


Figure 6.4. *The difference in energy between undistorted and AFD phases for the two model potentials as a function of the displacement parameter Δ which determines the magnitude of the tilt. Note that energy increases for the cubic potential as expected while for the unstable potential the system can decrease its energy by tilting the octahedra. In glazer notation this tilt is represented by $a^-a^-a^-$.*

From Fig. 6.4 it is apparent that tilting the octahedra increases the energy in the case of the cubic system while the energy is lowered for the tilt system as expected. By inspecting the rightmost figure we see that the maximum in energy difference between the AFD and undistorted phase for the tilt system occurs at approximately $\Delta = 0.13\,\text{Å}$ where energy for the AFD phase is lower by an amount $\Delta E \approx -7.4\,\text{meV}$ per unit cell. Clearly this is a relatively small energy difference and it is not evident that quantum fluctuations can be neglected.

Identical curves to those in Fig. 6.4 can be computed for the $a^0b^-b^-$, $a^-b^-b^-$ and $a^0a^0c^+$ tilt modes and it is found that the minima in energy difference between the cubic and distorted strucutures are more shallow for these types of tilt and occur at larger values of Δ. As a consequence, only $a^-a^-a^-$ AFD phases are actually observed. Note that geometrically, an $a^-a^-a^-$ mode corresponds to a rotation of the octahedra about an axis $\boldsymbol{\omega} = (1, 1, 1)$ through the zirconium atom. This rotation axis is only unique up to a sign however, and consequently there are eight different possibilities $\boldsymbol{\omega} = (\pm 1, \pm 1, \pm 1)$. For each oxygen ion there are thus several equivalent sites which

are separated by energy barriers. Thermal crossings of these barriers are possible at non-zero temperatures but as the temperature tends towards zero the oxygen ions will get stuck in one of the sites so that in the ground state an $a^-a^-a^-$ tilt with a definite axis of rotation $\boldsymbol{\omega}$ is established. Naturally, there is an equal probability to find each one of the eigth rotations $\boldsymbol{\omega} = (\pm 1, \pm 1, \pm 1)$ in the ground state. On the other hand at sufficiently high temperatures, the oxygen ions can move freely between the different sites and the resulting system looks on average cubic [16]. These equivalent sites for the oxygen atom in the tilt system will be important when we study the momentum distribution later on.

6.3 Verification part II: The Path Integral Monte-Carlo method

In Section 6.1 we verified that the potential is evaluated correctly by our program and similarily, we now verify the implementation of the PIMC method as described in Section 5.5. A convenient way to verify a large part of the program is to study a quantum harmonic oscillator with no external driving forces. In this particularily simple case one can derive an exact solution for the internal energy of the system as a function of temperature and we can then check to see whether our program reproduces this value. In particular, we compare the energies obtained by the different methods for sampling the paths introduced in Chapter 5 Sections 5.5 and 5.7. In addition, to validate the implementations of the open chain method and the trail method introduced in Section 5.9 we compare the distributions obtained to the analytic expression for the ground state position distribution.

Consider a single, one-dimensional quantum mechanical harmonic oscillator (QHO). The potential has the familiar quadratic form

$$\mathcal{V}(x) = \frac{1}{2} m\omega^2 x^2 \tag{6.7}$$

where m is the mass and ω is the angular frequency. Recalling Section 5.2 we can, with a bit of algebra, evaluate the partition function

$$Z = \mathrm{tr}\left(e^{-\beta\mathcal{H}}\right) = \int dx \langle x|e^{\beta\mathcal{H}}|x\rangle = \frac{e^{-\frac{\beta\hbar\omega}{2}}}{1 - e^{-\beta\hbar\omega}}. \tag{6.8}$$

The internal energy is given by Eq. (5.9):

$$E = \langle \mathcal{H} \rangle = -\frac{\partial}{\partial\beta} \ln(Z) = \frac{\hbar\omega}{2} \coth\left(\frac{\beta\hbar\omega}{2}\right). \tag{6.9}$$

From Eq. (6.9) we can evaluate the energy for a given angular frequency and temperarature. To make things simple we work in the natural units of the oscillator $\hbar = \lambda = k_B = 1$ and in addition we set the oscillator angular frequency $\omega = 1$. For $T = 0.1\,\mathrm{K}$ we find from Eq. (6.9) that $E = 0.5000\,\mathrm{K}$ and expect to be able to reproduce

this value in a PIMC simulation using the thermodynamic estimator Eq. (5.40) to compute the energy. A few minor modifications to the collection of subroutines described in the introduction to this chapter are necessary: one has to write a new subroutine for calculating the potential, but this is trivial given the simple harmonic form Eq. (6.7) and furthermore there is no need to use any periodic boundary conditions. To intialise the oscillator path we set $x = 0$ for each bead. This is also a good opportunity to compare the performance of the simple single slice sampling technique introduced in Section 5.5 to the more advanced bisection algorithm described in Section 5.7.2. As discussed in Sections 5.5, the number of beads M controls how well quantum mechanical effects are approximated in the simulation. A value $M = 1$ corresponds to a classical MMC simulation and higher values of M will yield increasingly better approximations. Thus, to reproduce the theoretical value $E = 0.5000\,\mathrm{K}$ it is essential that enough beads are included. To ensure this we calculate the energy for a wide range of M values and check how many beads are necessary to reach an asymptotic value. Fig. 6.5 shows a comparison of the energy at $T = 0.1\,\mathrm{K}$ as a function of the number of beads calculated using single slice and bisection sampling, respectively. In generating this figure all the PIMC simulations were allowed to run until the estimated MC error[2] was less than 1×10^{-4}.

From Fig. 6.5 it is immediately apparent that for values of roughly $M > 50$ an asymptotic value very close to the theoretical value $E = 0.5000\,\mathrm{K}$ can be obtained equally well using either sampling technique. Indeed, the figure suggests that in terms of the number of beads required for convergence, sampling new paths using the bisection algorithm offers no advantage over single slice sampling. This is to be expected since regardless of the sampling technique we are still working within the primitive approximations and reducing the number of beads needed can only be achieved using a higher-order action approximation. Instead, to reveal the difference in performance between the two sampling techniques we will look at the time required for an individual PIMC simulation to converge with an MC error less than 1×10^{-4} as a function of the number of beads, Fig. 6.6. Here it becomes clear that the bisection algorithm vastly outperforms single slice sampling when it comes to the rate with which phase space is explored, e.g. using $M = 100$ beads the PIMC simulation using bisection sampling only takes approximately one third of the time required using single slice sampling[3].

This is the desired result since the whole reason for introducing the bisection algorithm was to improve the sampling of new paths. Also keep in mind that this is just a model system for which evaluating the potential is relatively cheap, for a real system such as the barium zirconate system we will study in the upcoming sections, using bisection sampling can significantly reduce the simulation time. In conclusion, the implementation of the PIMC algorithm gives satisfactory results for a QHO model system and we have also shown that the sampling can be improved by a considerable amount using the bisection algorithm.

Our next task is to verify the algorithms used to calculate the momentum distri-

[2]Calculated using the relations introduced at the end of Section 4.2, including an estimate of the statistical inefficiency.

[3]These exact numbers depends, of course, on the system and implementation but the trend is quite clear.

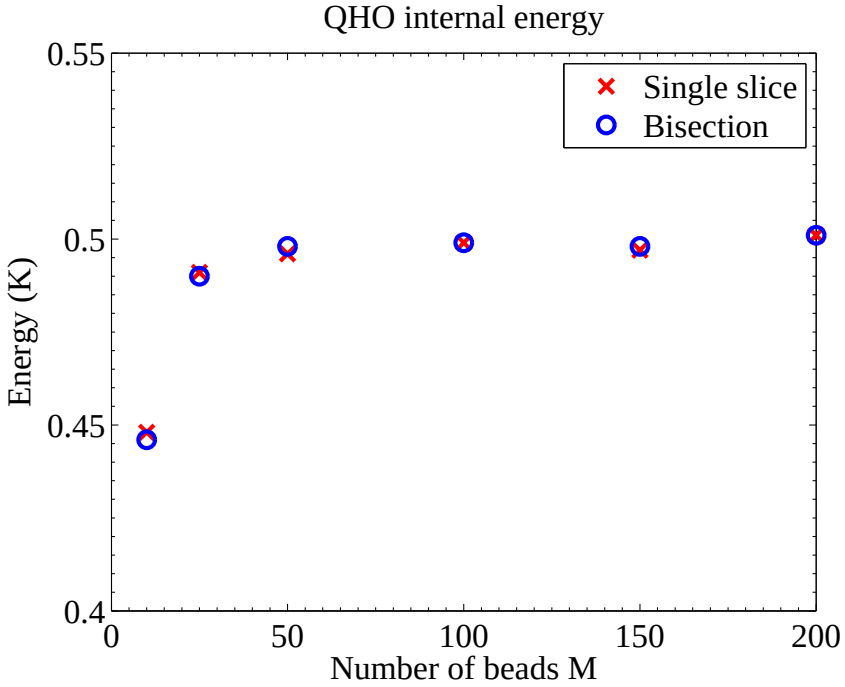


Figure 6.5. *The internal energy for a quantum harmonic oscillator system as a function of the number of beads M used in the PIMC simulation. Two sets of data are shown, the blue circles show the results obtained using the bisection algorithm to sample new paths while the red crosses show the corresponding results using single slice sampling. Observe that the performance of the two sampling methods in this case is identical and an symptotic value of the energy is attained for roughly $M > 50$.*

bution. Recall from Section 5.8 that the momentum distribution can be obtained as the fourier transform of the single particle density matrix or equivalently the end-to-end distance distribution. This can be calculated using either the open chain method or the trail method, both described in Section 5.9. A suitable test is to compare the distribution obtained using these two methods to the QHO ground state position distribution if we perform our calculations at a low temperature so that mainly the ground state is populated. Fig. 6.7 shows the position distributions obtained for $T = 0.1\,\mathrm{K}$ using the two different methods with $M = 100$ beads. The analytical predictions are displayed in the same figure and we can see that both methods produce distributions that agree farily well with analytical prediction. Comparing the performance of the two methods the open chain algorithm converged significantly faster, within the span of a few minutes on an average desktop computer compared to the trail method which took several hours.

6.4 Barium Zirconate

In Section 6.2 we established two model potentials for $BaZrO_3$, both based on a Buckingham potential for the short-range interaction and the Ewald summation for the long-

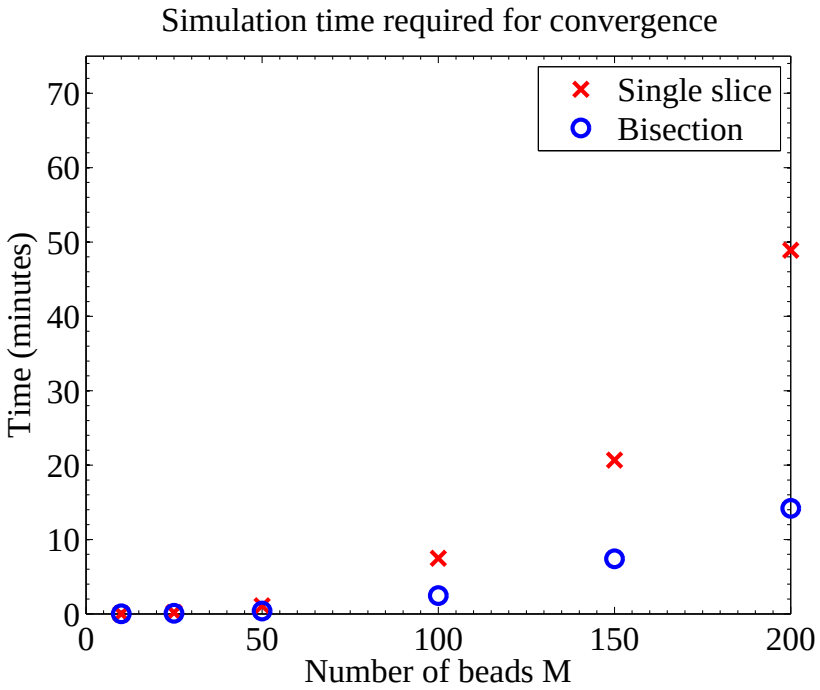


Figure 6.6. *The time required in order to converge the internal energy of a QHO with an error better than 1×10^{-4} as a function of the number of beads used. Blue circles show the times measured using the bisection algorithm to sample new path while the red crosses are the corresponding times obtained using single slice sampling. Here we note that bisection sampling yields a much better rate of convergence and that the discrepancy between the two methods grows with the number of beads.*

range electrostatic interaction. The first model potential describes a system without AFD instabilities i.e. is always cubic while the other potential describes a tilt system where there are AFD phases. The purpose of this section is to show the results from calculations of pair correlation functions, momentum distribution and the internal energy. These calculations have been performed for both model potentials and comparing the results will enable us to see the impact of the AFD instabilites when nuclear quantum effects are taken into account. In the first section, we look at the internal energy of the system which is an important property in itself but here we will mainly use it as a means of studying equilibration and convergence. The pair correlation function is treated in the following section providing a nice visualisation of the quantum fluctuations. The last part of this chapter deals with the single particle density matrix, momentum distribution and algortihms used for their calculation.

6.4.1 Internal energy

The internal energy of the system can be calculated in a PIMC simulation using the thermodynamic estimator described in Section 5.6. The difference from a classical MMC

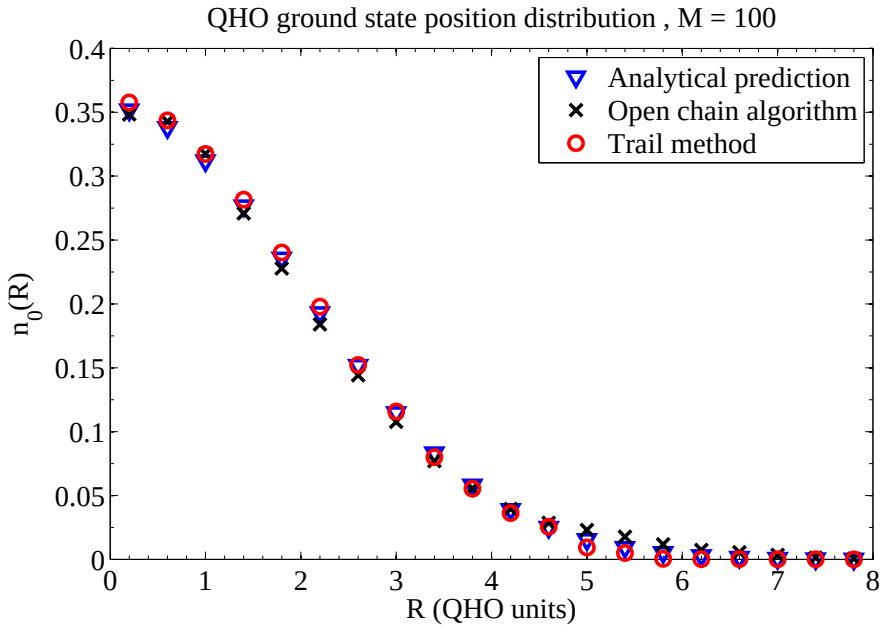


Figure 6.7. *Verification of the open chain algorithm and the trail method by comparing the end-to-end distance distributions obtained at $T = 0.1\,\mathrm{K}$ to the ground state probability distribution for the harmonic oscillator. From the figure it is observed that both algorithms yield distributions in good agreement with the analytical prediction. The convergence rate for the trail method turned out to be a lot slower.*

simulation is the addition of a kinetic contribution from the springs coupling neighbouring beads. While the internal energy is an important property of the system we shall be more interested in calculating the momentum distribution (see Section 6.4.3) and the energy calculations will primarily be used as a tool to study the equilibration and convergence. In an ideal scenario one would determine a set of quantities to be calculated and then perform a detailed convergence study for each of these properties, determining the number of PIMC cycles as well as the number of beads required to obtain converged results. In this work we are more interested in the qualitative behvaiour of the system however, and will adopt a less rigorous approach when checking for convergence.

To get an idea of how long the system needs to equilibrate we can look at the total energy for each generated configuration, i.e. the expression inside the angular brackets in Eq. (5.40), as a function of the number of PIMC cycles. Such a plot is displayed in Fig. 6.8 where the energy for each newly generated configuration of a $BaZrO_3$ system is recorded at $T = 30\,\mathrm{K}$.

From Fig. 6.8 a rapid initial increase in energy is observed during the first 50 PIMC cycles after which the curve becomes more even and the energy starts to fluctuate around a mean value. This behaviour makes sense since the system was initialised in the $T = 0\,\mathrm{K}$ equilibrium configuration. From Fig. 6.8 it can be discerned that the thermalisation period is somewhere around 50 PIMC cycles in this case and it is good practice to

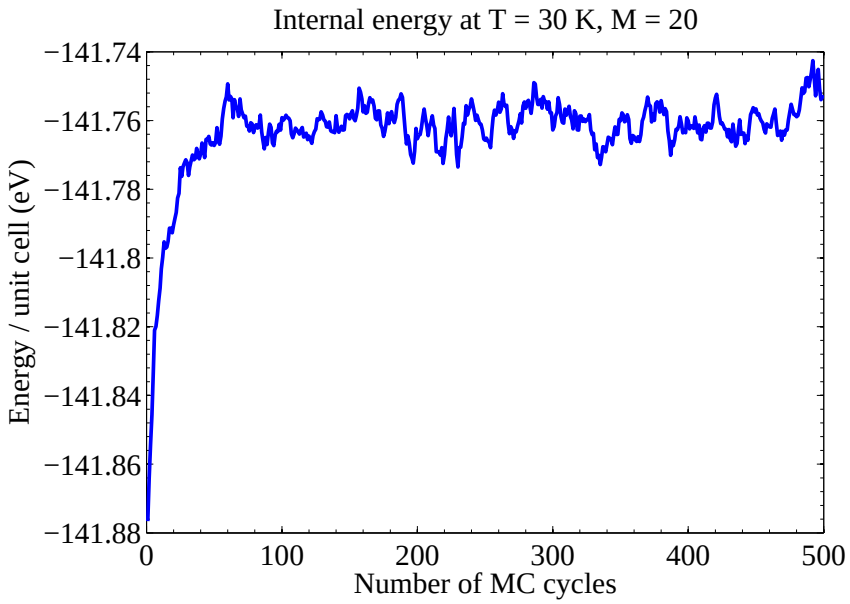


Figure 6.8. *Equilibration in a PIMC simulation of $BaZrO_3$. Initially all the beads are in the $T = 0\,\mathrm{K}$ equilibrium configuration but as the simulation proceeds the internal energy of the generated configurations starts to increase and after roughly 50 PIMC cycles the energy of the configurations stabilises and starts to fluctuate.*

let the actual equilibration period N_{eq} extend over a number of steps corresponding to a few multiples of this thermalisation time. Thus we will always use at least $N_{\mathrm{eq}} = 200$ equilibration cycles when attempting to calculate any property of the system. An exception occurs if we do a so called *warm start* where the initial distribution is no longer the $T = 0\,\mathrm{K}$ configuration but rather some configurations stored from a previous run. If the conditions were similar during this preliminary run we can afford to use a shorter equilibration period. This will be the case when calculating properties for the tilt system.

We will now look at the internal energy as a function of the number of beads M used. Since increasing the number of beads used in a simulation quickly becomes expensive one should do a convergence study to determine how many beads are required to reach an asymptotic value, much like we did in the previous section with the QHO. Fig. 6.9 shows the internal energy for a $2{\times}2{\times}2$ $BaZrO_3$ system for a few different values of M and at two different temperatures. In creating this figure, all the simulations were allowed to run until the estimated MC error was smaller than 1×10^{-4}.

We observe that at $T = 30\,\mathrm{K}$, increasing the number of beads yields a steadily increasing energy from $M = 1$ to somewhere around $M = 25$ where an asymptotic value has been obtained. Thus we can conclude that at this particular temperature, if we are interested in the internal energy of the system, choosing $20 < M < 30$ would be a suitable compromise between accuracy and computational efficiency. As the temperature increases the expectation is that fewer beads will be required since quantum effects will

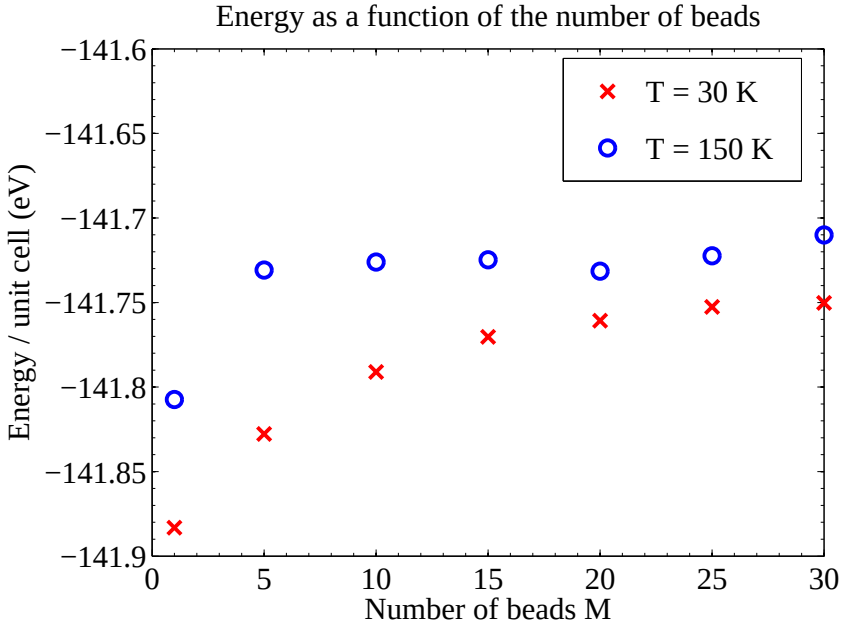


Figure 6.9. *Comparing the internal energy of a 2×2×2 $BaZrO_3$ for several different values of the number of beads M and at two different temperatures. For the lower temperature $T = 30\,\mathrm{K}$ at least 20 beads are required to get close to an asymptotic value while at a higher temperature $T = 150\,\mathrm{K}$ we could get away with using only $M \approx 5$ beads.*

not be as important. This is indeed the case looking at the $T = 150\,\mathrm{K}$ measurements where increasing the number of beads beyond $M = 5$ does not significantly change the value of the internal energy. Hence, in this case we could get away with using a smaller number of beads compared to the low temperature case. Since we are mostly interested in qualitative behaviour, we will restrict ourselves to $M = 20$ beads when calculating the pair correlation function and momentum distribution at $T = 30\,\mathrm{K}$ in the upcoming sections. It should also be noted that the convergence with respect to the number of beads might look slightly different for the momentum distribution compared to the energy. Ideally one would do a separate convergence study for the momentum distribution but we shall take the number $M = 20$ on faith in order to save time.

6.4.2 The pair correlation function

The pair correlation function was defined in Section 4.5 and describes the distribution of atomic pairs in the system. For systems that are not monatomic, rather than looking at $g(r)$ for the entire system one defines *fractional pair correlations* where each possible type of two-atom combination that can be formed is described by a separate correlation function. For a $BaZrO_3$ system there are four possibilites: *Ba-O*, *Zr-O*, *Ba-Zr* and *O-O*. Out of these four types of pairs we are only really interested in the distribution for pairs of oxygen atoms since these are the atoms which are displaced in an AFD phase, see Chapter 2. In this section we will look at pair correlation functions calculated for

the two different model systems described in Section 6.2 using the estimator Eq. (5.41) defined in Section 5.6 . All calculations presented here were performed on a 4×4×4 $BaZrO_3$ supercell using a radial cut-off $r_{\text{cut}} = 8\,\text{Å}$ for the Buckingham potential and the ewald error parameter $\epsilon = 1 \times 10^{-7}$. Fig. 6.10 shows how the pair correlation function for the cubic system using $M = 20$ and $M = 1$ beads respectively, included are also the expected δ-function peaks for the $T = 0\,\text{K}$ structure.

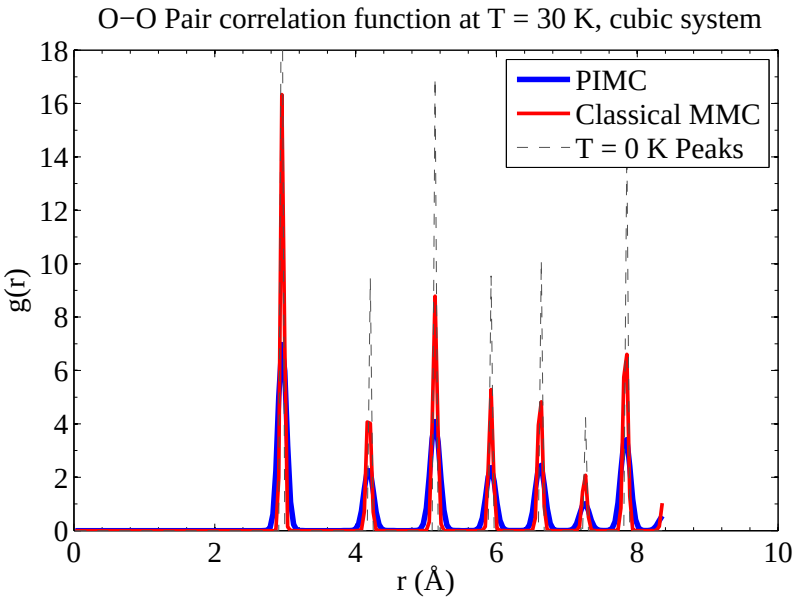


Figure 6.10. *Oxygen-Oxygen pair correlation function for a $BaZrO_3$ supercell system. Three different sets of data are shown, the red curve corresponds to a PIMC simulation with $M = 1$ which equivalent to a classical MMC simulation while the blue curve shows the results from a PIMC simulation with $M = 20$. Finally, the dotted grey lines shows the expected peaks for the $T = 0\,\text{K}$ configuration. When quantum effects are taken into account we see that the delocalisation in the system increases since the blue distribution is wider.*

Note that since using $M = 1$ beads reduces the PIMC method to the classical MMC method we can observe a purely thermal broadening of the peaks compared to the $T = 0\,\text{K}$ case. When $M \neq 1$ however, there is an addition to the width of the peaks due to quantum delocalisation. Note that in order for the quantum effects to appear the temperature needs to be low enough and we have found that $T = 30\,\text{K}$ is sufficiently low that quantum delocalisation can be observed as evidenced by Fig. 6.10. The situation becomes more interesting looking at the tilt system, here we expect the oxygen ions to find alternative configurations corresponding to tilted octahedra given that the temperature is low enough that the difference between alternate sites for the oxygen ions are not erased by thermal fluctuations. Fig. 6.11 shows the *O-O* pair correlation function calculated for the tilt system with $M = 1$ and $M = 20$ respectively.

Comparing the classical $M = 1$ case here to the corresponding curve in Fig. 6.10 it is clear that there exist additional peaks in the tilt system corresponding to the positions

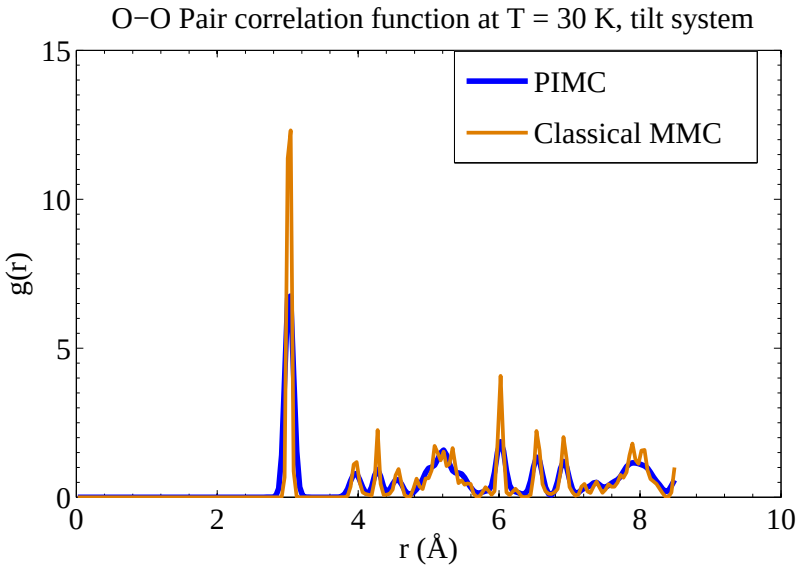


Figure 6.11. *Oxygen-Oxygen pair correlation function for a $BaZrO_3$ supercell system with AFD instabilities. We note the appearance of alternative peaks corresponding to rotated octahedra in this figure. When quantum effects are taken into account the alternative peaks are partly erased as seen by comparing the classical result (orange) to the PIMC result (blue).*

occupied by oxygen pairs in a system with rotated octahedra. When we take quantum fluctuations into account however, delocalisation partly erases some of these additional peaks or make them less distinguished as evident from the $M = 20$ correlation function in Fig. 6.11. A crucial component in obtaining this figure was to do a warm start where after running an initial simulation with $M = 1$ during which system was allowed to thermalise and then find the alternate configurations corresponding to tilted octahedra, the resulting configuration was saved and used to restart a simulation with $M = 20$ beads.

6.4.3 The momentum distribution

The pair correlation function gives information about the structure of the system but if one is interested in the impact of nuclear quantum effects one needs to study the single particle density matrix (or rather end-to-end distance distribution which is what one actually calculates) and the momentum distribution. These were defined in Section 5.8 and we also provided algorithms for their calculation in Section 5.9. We have already seen an example of a end-to-end distance distribution for the simple case of a harmonic oscillator in Section 6.3. For a real system such as a $BaZrO_3$ supercell, calculating the momentum distribution is computationally more demanding and the situation is further complicated by the long-range Coulomb interaction and the prescence of AFD phases. In particular, during our calculations we were unable to obtain converged results for $BaZrO_3$ using the trail method (see Section 7.2.2 for an expanded discussion of this result) and

hence all distributions displayed in this section have been calculated using the open chain algorithm. Since only the oxygen atoms partake in an octahedral tilt it is for these atoms we have calculated the momentum distribution. First we will look at the free particle momentum distribution for the oxygen atoms i.e. the distribution obtained when the potential interaction is ignored. But this is just the classical isothermal momentum (Maxwell) distribution $n(k) \propto \exp[-k^2/(2mk_BT)]$ and comparing the calculated free particle distribution to the analytical expression gives us an additional way to check that the algorithm is working properly. This comparison is displayed in Fig. 6.12 which shows that the calculated free particle distribution is in excellent agreement with the expected Maxwellian distribution.

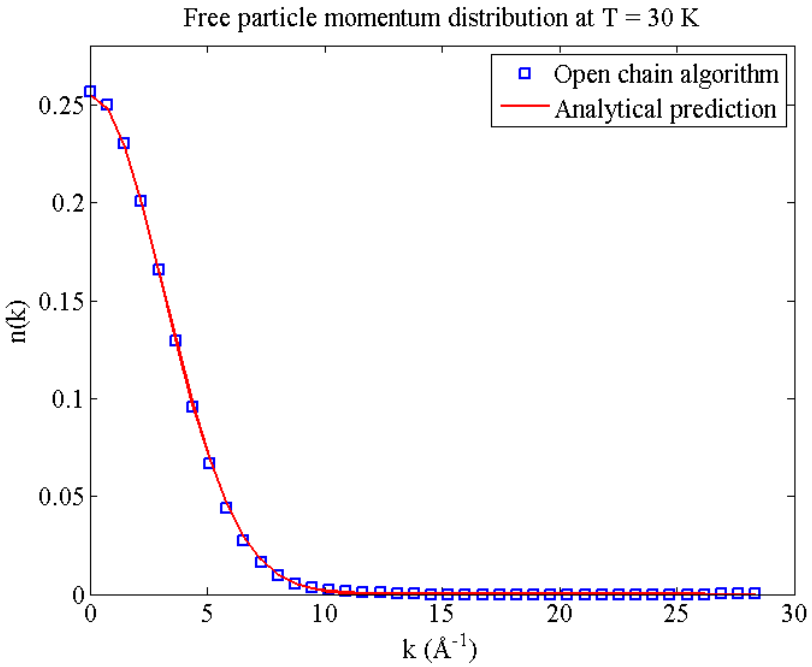


Figure 6.12. *The free particle momentum distribution for the oxygen atoms. The red curve shows the predicted Maxwellian distribution and the squares are the values calculated using the open chain algorithm. Clearly the two are in good agreement indicating that the open chain algorithm implementation is working.*

Turning to the actual $BaZrO_3$ system with the potential interaction tuned on we look at the end-to-end distance distribution for the two model potentials displayed in Fig. 6.13. As a reference point we have also included the free particle end-to-end distance distribution in this figure (this is the distribution we had to fourier transform in order to obtain Fig. 6.12). According to the classical isomorphism this is just the end-to-end distance distribution for a linear polmer without any potential interaction and hence we get a gaussian distribution as in Fig. 6.13, indeed one can show analytically that the fourier transform in this case is exactly the Maxwell distribution [2]. For the interacting system Fig. 6.13 shows that the distance distributions are much more narrow compared

to the free particle distribution as expected. Furthermore we see that the distribution sfor the tilt and cubic systems respectively are fairly similar although the tilt system distribution is slightly wider.

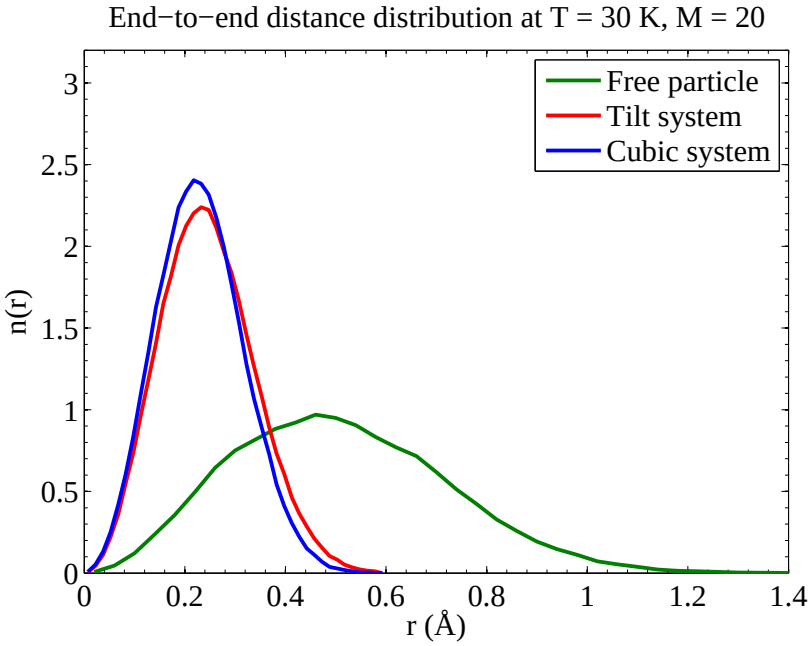


Figure 6.13. *The end-to-end distance distribution $n(R)$ for oxygen atoms in barium zirconate. The plot shows a comparison between the distribution obtained for the cubic system and the corresponding distribution for the tilt system. Evidently the two $n(R)$ are very similar, the tilt system distribution being slightly wider at larger separations. As a reference point the free particle end-to-end distribution is also dsiplayed, this distribution is gaussian as one would expect for a linear polymer.*

Unlike the pair correlation function, the convergence is very slow here, roughly 150000 MC cycles were used to obtain Fig. 6.13, and in order to save time all calculations were performed on a 2×2×2 $BaZrO_3$ supercell [4] but otherwise using the same settings, including $M = 20$ beads and $T = 30\,\mathrm{K}$.

A complementary picture is provided by the momentum distribution, given by the fourier transform of $n(R)$ as described in Section 5.8. Fig. 6.14 shows the momentum distributions corresponding to the end-to-end distance distributions displayed in Fig. 6.13 along with the free particle momentum distribution from Fig. 6.12. Compared to the free particle (classical) momentum distribution we can see that when quantum mechanics is taken into account the end result is a significantly wider distribution which, keeping the Heisenberg uncertainty principle in mind, we could already have guessed from Fig. 6.13.

From Fig. 6.14 we also note that the momentum distribution for the tilt system is

[4] Preliminary results indicate that the shape of the distribution is not overly sensitive to the system size but the purpose of this section is to provide a picture of the general behaviour rather than showing the result of a long simulation with very small errors.

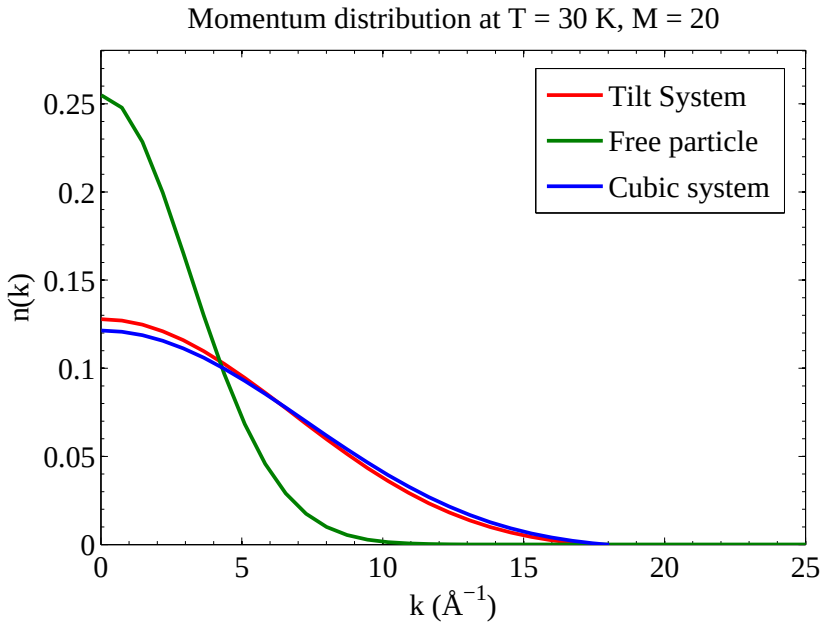


Figure 6.14. *The momentum distribution $n(k)$ for oxygen atoms in barium zirconate. The plot shows a comparison between the cubic and the tilt system. We note that the two distributions are almost identical. As a reference point the free particle momentum distribution is also included and comparing the different curves we find that including quantum effects significantly alters the momentum distribution.*

slightly more narrow than the corresponding distribution for the cubic system which also makes sense given Fig. 6.13 and the uncertainity principle. As discussed in Section 5.8 we should look at the tails of the momentum distribution for indicators of nuclear quantum effects. In particular, we are interested to see if there is a node in the tail of the tilt system momentum distribution. However, zooming in on the tails of the distributions in Fig. 6.14 we end up with Fig. 6.15 from which it is evident that there are no observable node in the tilt system distribution and indeed the two tails are almost identical.

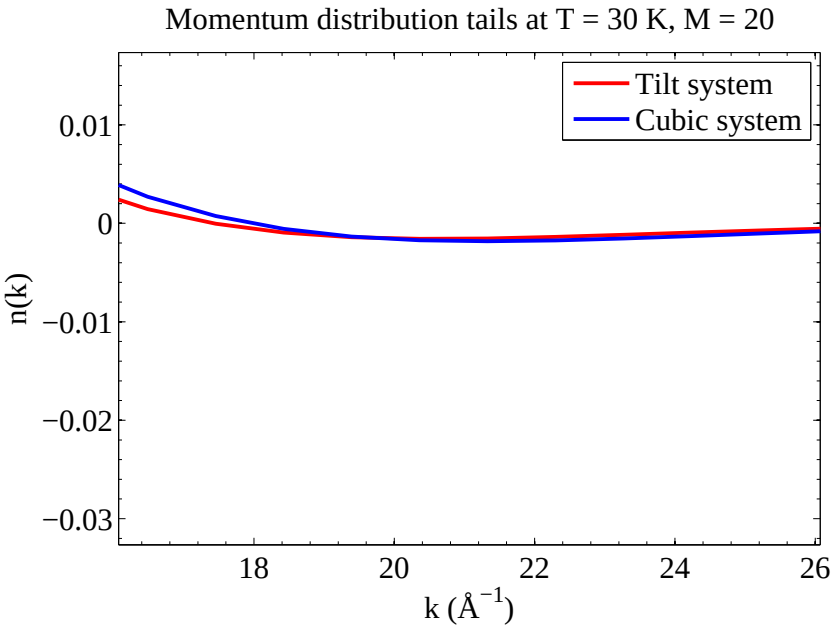


Figure 6.15. *Zooming in on the momentum distribution tails from Fig. 6.14. Observe that although there is a slight dip in the distributions there is no visible nodes and in particular the two tails are almost identical.*

CHAPTER 7

DISCUSSION

In this chapter we discuss the results presented in Chapter 6, the relative performance of the different algorithms used and various other topics which have been touched upon. As previously noted, the purpose of this thesis has been twofold. One part consisted of showing how PIMC simulations can be used to calculate properties such as the momentum distribution and what algorithms to use. The second part was to apply these simulation techniques to a $BaZrO_3$ system for which we have established a simple model. Here, an important point was to obtain results for the two different model systems, one describing a cubic system and one describing a system with AFD instabilities, so that the two can be compared in the light of simulations where nuclear quantum effects are taken into account. The reader should be aware that the main aim has been to obtain results indicating general behaviour rather than conducting precise measurements of quantities like the momentum distribution. As such, certain liberties have been taken in some cases, most notably the omission of proper error bars.

7.1 Inter-atomic interactions and the Ewald summation

The model used in this work combining the Buckingham potential and Ewald summation for the electrostatic potential (see Chapter 3) is a standard one and only a few remarks are necessary. In defining the model potential we have used a rigid ion description in which the electrons are replaced by atomic net charges, van der Waals interaction and Pauli repulsion. It is evident that this type of model cannot accurately describe e.g. polarisability. One way around this is to simulate the effects of having an electron cloud by using a *shell model*. Here, an atom is considered as consisting of both a core and a shell which are coupled by a harmonic spring [20]. Previous results have indicated however that extending the model potential to include this interaction has very little impact on the kind of stuctural properties we are interested in here and to keep the model simple we have chosen not to include such a shell model interaction.

Regarding the Ewald summation, summarised in Section 3.5 and investigated more

thorougly in appendix A, we note the existence of equivalent techniques for calculating the long-range interaction such as the particle-particle, particle mesh method (PPPM) [19]. Here one takes a quite different, more complicated approach to compute the electrostatic energy and is rewarded by a better overall scaling of $O(N \log N)$ rather than $O(N^{3/2})$ for the original Ewald summation [1] [19]. Generally speaking, there is a larger prefactor for the scaling in the case of PPPM however and one therefore only sees an increase in efficiency for systems with a large number of particles. This motivated us to stick with the original Ewald summation method.

7.2 Simulation algorithms

We will now discuss the algorithms used in PIMC simulations. In particular, we are interested in comparing the relative performance of the different options presented for sampling new paths and for calculating the momentum distribution.

7.2.1 The sampling of new paths

In Section 5.5 we described thoroughly the basic PIMC algorithm with single slice sampling i.e. where one bead is moved at a time akin to the classical MMC method. The shortcomings of this method of sampling new paths were discussed in Section 5.7 where we also introduced two new types of moves in order to improve the sampling. The centre of mass displacements (COM) described there have been used in practically all the simulations performed to obtain the results presented in Chapter 6. It is important to note that since this type of move consists of moving an atom as a whole, they have a limited impact on the convergence since quantum effects require relative displacements between beads. Thus, COM moves must be carefully managed in the simulation and our strategy has been to perform one COM move every PIMC cycle (where one PIMC cycle consists of sampling on average a new path for each atom in the system i.e. in the case of single slice sampling one MC cycle consists of one COM move plus randomly displacing NM beads). We have found that this gives a satisfactory convergence rate for the properties of interest. However, when performing calculations on the tilt system introduced in Section 6.2 it was also discovered that if we limit the number of COM moves to one every cycle, configurations with tilted octahedra are not found. This is not suprising since a tilted configurations involves a collective displacement of several oxygen atoms, incredibly unlikely to be achieved using single slice sampling. Rather than increasing the number of COM moves performed over the entire simulation, which would be ineffective in the long run, a better solution is to run a long simulation with $M = 1$ and then restarting the simulation with $M = 20$ from the resulting configuration. Using $M = 1$ the PIMC method reduces to the classical MMC method and finding configurations involving collective displacements of atoms is no longer a problem. This is a useful technique in general, especially for larger systems where the thermalisation can be accelerated considerably by restarting the main simulation from an $M = 1$ simulation.

[1] Note that this scaling is only achieved if the parameters are chosen properly, see appendix A

The second part of Section 5.7 treated bisection moves which can be used to replace the single slice moves as a means of sampling new paths. Several previous investigations have estblished the bisection method as superior [2] [3], a result that is echoed in Section 6.3 where we compared bisection sampling to single slice sampling by calculating the internal energy for a harmonic oscillator and looking at the errorbars. This is certainly not suprising, given the careful construction of the bisection move. One of the main factors responsible for the increased convergence rate are that beads in a path are sampled directly from the free particle distribution and for the coarser levels this sampling is done at a higher temperature. This allows the beads to move larger distances since the stiffness of the springs coupling neighbouring beads decreases with temperautre. Another important point is that unlikely moves are rejected at the coarser levels, removing the need to perform expensive calculations of the potential interaction at the finer levels and as a result we explore phase space faster. One notable drawback of the bisection method is that it is hard to combine with the open chain method. The origin of the problem is in a sense technical, having an open chain severly limits the number of ways in which we can bisect it. As a consequence, for a fixed chain length and a fixed maxiumum bisection level, biases can arise where some beads are much more likely to be sampled than others.

7.2.2 Calculating the momentum distribution

In Section 5.9 we described two different algorithms for calculating the momentum distribution (or rather the end-to-end distance distribution which then has to be fourier transformed in order to obtain the momentum distribution). The most straightforward algorithm is the open chain method which was derived by interpreting the defining equation of the single particle density matrix in the language of the classical isomorphism. While the implementation is simple, we noted that one of the major disadvantages is the fact that one cannot calculate diagonal properties in the same simulation. Another option the trail method in which one performs an ordinary simulation with ring polymers which is halted after each MC cycle. During this pause, paths (belonging to the atom type for which we wish to calculate the momentum distribution) are cut open and displaced one at a time. To obtain $n(r)$ one calculates the ratio of the density matrix of the now open chain to that of the previously closed chain. The occurences are added to a histogram which then gives $n(r)$ at the end of the simulation.

In Section 6.3 we verified both these methods by comparing the caculated $n(R)$ for a harmonic oscillator at a low temperature to the analytical expression for the ground state single particle density matrix. Here it was found that both methods indeed yield the correct result but that the convergence was significantly slower using the trail method. While the open chain algorithm yielded a converged result within minutes the trail method needed several hours. The situation deteriorated even further when we looked at the momentum distribution for the $BaZrO_3$ system in Section 6.4.3. In this case the open chain algorithm yielded acceptable results within the span of a dozen hours (corresponding to roughly 100000 MC cycles) while the trail method failed to converge completely under the time span for which it was allowed to run (3 days on a desktop

computer, corresponding to roughly 200000 MC cycles). In the case of the QHO we also observed that the convergence of $n(R)$ at large distances seemed to be the most troublesome. These result are in stark contrast to the performance reported by the inventors of the trail method [3] who reported an increase in the rate of convergence using the trail method with a reduced variance. Taking a closer look at their report it appears most of their testing was done on a liquid neon system which only has short-range interactions which are modelled by either a Lennard-Jones potential or the Aziz HFD-C2 potential. Using the latter potential we were able to reproduce in a preliminary study, coming to the same conclusion that the trail method converges significantly faster in this case. A possible explanation[2] lies in the magnitude of the interactions. Since the ratio of density matrices required in the trail method varies exponentially with the difference in action, large action differences may lead to an increased variance. This is certainly the case for the $BaZrO_3$ system where small displacements can lead to large differences in action due to the coulomb interaction. For the QHO it depends on the angular frequency ω, in our simulations we set $\omega = 1$ which gives relatively large actions. For either of the two systems, the difference is likely to be large at large R which would explain the convergence difficulties encountered during the calculation of $n(R)$ in Section 6.3.

The recommended approach is thus to use the open chain method for systems which include long-range interactions (ionic system) or strong external potential while the trail method is preferable for systems with short-range or weak interactions (noble gases, Lennard-Jones systems etc). The major drawback of using the open chain method is that we can no longer use bisection sampling as explained in the previous section. However, for ionic systems, the accelerated convergence compared to the trail method more than compensates the use of a more crude sampling technique.

7.3 Properties of barium zirconate

Having dealt with the technical aspects of the PIMC algorithms we will now turn to analysing the actual results obtained for the $BaZrO_3$ system and presented in Section 6.2 and Section 6.4.

The pair correlation functions displayed in Section 6.4.2 do not require much comment. The expected tendency of peaks to become broader as the temperature increases is observed as well as a broadening due to quantum delocalisation when $M \neq 1$. Note that these two different mechanisms for delocalisation are in a sense competeting, at lower temperatures quantum delocalisation dominates and the quantum corrected pair correlation is thus quite different from the classical one.

7.3.1 The momentum distribution

We explained in Section 6.2 the tilt system has AFD phases where the tilt can be described as $a^-a^-a^-$ in glazer notation. Geometrically this corresponds to a rotation of

[2]Barring any subtle mistakes in the implementation that the author is not aware of.

the octahedra about an axis $\boldsymbol{\omega} = (1, 1, 1)$ through the zirconium atom. This rotation axis is only unique up to a sign however, and consequently there are eight degenerated structures $\boldsymbol{\omega} = (\pm 1, \pm 1, \pm 1)$. There are thus different sites for the oxygen atoms separated by energy barriers, much like in the case of the double potential well that was discussed in Section 5.8. It is observed that the oxygen atoms can move between these configurations when they have enough energy to surmount the barrier but at lower temperatures they tend to get stuck in one of the sites. This begs the question of whether the oxygen atoms can simultaneously occupy two such sites which would be possible if the energy barrier is so low that zero-point motion effectively erases it. To answer this we compare the momentum distributions displayed in Fig. 6.14 since for the cubic system there can be no tilt and the oxygen atoms are confined to vibrate around their equilibrium position. Looking at the momentum distributions calculated in Section 6.4.3 there is indeed a slight narrowing of the momentum distribution in the tilt system which could be indicative of tunneling as has been suggested by many others, see e.g. [7]. This narrowing is typically followed by a node in the tail of the momentum distribution however, regardless of whether the lowest lying energy level is actually above or below the barrier. Evidently from Fig. 6.15 no such node appears for the tilt system and indeed the tails of the distributions for the two systems are more or less identical. A concern here is that the error of the calculations could be quite large but as a preliminary conclusion it seems unlikely that two sites can be occupied simultaneously by an oxygen atom. Given more time, an exhaustive investigation would preferably include a detailed convergence study, momentum distributions for several temperatures both above and below $T = 30\,\mathrm{K}$ and a larger system size.

7.4 Outlook and future prospects

There are many exciting prospects and possible ways to extend the work done in this thesis. Most pressing however is the need to do more thorough investigations of the momentum distribution for a wide range of temperatures below and around the AFD transition point. This would most likely require an MPI or OpenMP parallel implementation of the programs used here in order to speed up the calculations by moving them onto a cluster. This would also include carefully checking that the erros are small enough and that a sufficient number of beads are used. This would allow us to more conclusively rule out or confirm the role of nuclear quantum effects in regards to the transition to an antiferrodistortive state. Due to time limitations such a study could unfortunately not be presented here. There is also the question of trying to improve the open chain algorithm for calculating the momentum distribution since the results presented here indicate that the trail method might be unsuitable for strongly-interacting systems. Another possbility which has barely been mentioned in this thesis is to use path integral molecular dynamics (PIMD) instead of the Monte Carlo approach described here. It would be interesting to study the advantages and disadvantages of such an approach for a strongly interacting real system such as barium zirconate.

Other prospects include exploring high-order action approximations such as the Li-

Broughton action which could help make the PIMC simulations more efficient. Again, due to time limitations we did not explore the many options here, it should also be noted that many higher-order schemes require the forces to be known. This is not necessarily a problem since analytical derivatives can be obtained directly from the Ewald summation and Buckingham potential. Moving forward one could also proceed, using the PIMC toolbox, to look at different perovskites which exhibit not only AFD transitions but so-called ferrodistortive transitions as well. One would then have a large set of competing structures and the interplay between these could possibly be affected by nuclear quantum effects.

CHAPTER 8

CONCLUSIONS

Below we summarise briefly the topics covered in this thesis along with the most important conclusions. To study the structure and dynamics of barium zirconate a rigid ion model with a pair potential which is the sum of a Buckingham potential and the Coulomb potential can be employed. Analysing the phonon spectrum with respect to soft phonon modes appearing as imaginary frequencies provides a method of testing whether a certain set of values for the potential parameters give rise to a system with AFD instabilities. Using this strategy two model systems have been established one stable cubic system and one which exhibits AFD phases which can be described in glazer notation as $a^-a^-a^-$. The difference in energy between the configurations in the different phases is small, on the scale of milielectronvolts, allowing for the possbility that the structure is influenced by nuclear quantum effects.

Path integral Monte Carlo is an effective method for obtaining quantum corrected properties for a solid or a liquid. In particular, the momentum distribution which is of special interest since it works as a fingerprint for nuclear quantum effects can be calculated during a PIMC simulation. The degree of accuracy with which quantum effects are included in a PIMC simulation is determined by the number of beads M used. To determine a suitable value for M the internal energy of a $BaZrO_3$ system was calculated for different values of M. For temperatures around $T = 30\,\mathrm{K}$ a value $M = 20$ is sufficient to yield an approximately asymptotic value of the internal energy. If the temperature is increased fewer beads are needed, indeed at $T = 150\,\mathrm{K}$ five beads are sufficient. An illustration of the effect of quantum delocalisation in the system is given by the pair correlation function. Here our calculation reveal that a quantum treatment at $T = 30\,\mathrm{K}$ yields significantly wider peaks in the pair correlation function than expected from corresponding classical calculations. This broadening of peaks due to quantum delocalisation competes with thermal broadening where the latter dominates at higher temperatures.

The main property of interest for the $BaZrO_3$ system is the momentum distribution. Here two distributions were calculated at $T = 30\,\mathrm{K}$, one for the cubic system and

one for the tilt system which has AFD instabilities. By comparing the momentum distributions for these two systems we concluded that the distribution for the tilt system is slightly more narrow. However, seeing as the difference was quite small and the tails of the two distributions were otherwise identical i.e. no nodes were found in the tilt system distribution, we make the preliminary conclusion that there is no simultaneous occupation of several equivalent sites for the oxygen ions.

Regarding the algorithms used for calculating the momentum distribution during a PIMC simulation we conclude that either the open chain algorithm or the trail method may be used. The advantage of the trail method is that properties diagonal in positions space can be calculated in the same simulation as the momentum distribution which is not possible in the open chain algorithm where one polymer must be open during the simulation. It is also possible to combine the trail method with more advanced sampling methods such as the bisection algorithm which can dramatically increase the rate with which phase space is explored. On the other hand, the convergence rate for the trail method appears to be highly dependent on the strength of the interaction in the system, the perfomance being significantly worse for strongly interacting systems such as $BaZrO_3$. The open chain algorithm on the other hand can be used to calculate the momentum distribution for any system but converges slower than the trail metod for weakly interacting system.

As mentioned above the convergence rate of a PIMC simulation can be improved by introducing more advanced techniques for sampling new paths. For this purpose we also implemented an alternative version of the main PIMC program making use of the bisection algorithm. By studying a simple harmonic oscillator model system the superiority of the bisection algorithm over regular single slice sampling was established. In particular, when calculating the internal energy of the oscillator it was found that simulations using the bisection based sampling converged approximately three times faster than the corresponding single slice sampling simulations. The difference in performance was also found to increase with the number of beads used. Another measure which can be taken to explore phase space more efficiently is centre of mass displacements which turned out to be a necessary component to discover configurations with rotated octahedra in $BaZrO_3$.

REFERENCES

[1] Allen, M. P. and Tildesley D. J. (1991) *Computer Simulation of Liquids*, Oxford University Press.

[2] Ceperley, D. M. (1995) *Path integrals in the theory of condensed helium*, Rev. Mod. Phys. vol. 67, no 2, pp. 279-356.

[3] Brualla, L. M. (2002) *Path integral Monte Carlo: Algorithms and applications to quantum liquids.*, `http://www.tdx.cat/handle/10803/6577` (2014-05-03).

[4] Stokes, S. J. and Islam M. S. (2010) *Defect chemistry and proton-dopant association in $BaZrO_3$ and $BaPrO_3$* , Journal of Materials Chemistry, vol 20, no 30, pp. 6258-6264.

[5] Chandler, D. P (1987) *Introduction to modern statistical mechanics*, Oxford University Press.

[6] Q. C. Johnson and D. H. Templeton (1961) *Madelung Constants for Several Stuctures*, Journal of Chemical Physics, vol 34, no 6, pp. 2004-2007.

[7] Morrone, J. A. et al (2009) *Tunneling and delocalization effects in hydrogen bonded systems: A study in position and momentum space* The Journal of Chemical Physics, vol 130, no 20, 204511

[8] Engel, H. et al (2012) *Momentum Distribution as a Fingerprint of Quantum Delocalization in Enzymatic Reactions: Open-Chain Path-Integral Simulations of Model Systems and the Hydride Transfer in Dihydrofolate Reductase*, Journal of Chemical Theory and Computation, vol 8, no 4, pp. 1223-1234.

[9] Lebedev, A. I. and Sluchinskaya, I. A. (2013) *Combined first-principles and EXAFS study of structural instability in $BaZrO_3$*, `http://arxiv.org/abs/1304.6359` (2014-05-03)

[10] Akbarzadeh, A. R. et al (2005) *Combined theoretical and experimental study of the low-temperature properties of $BaZrO_3$*, Physical Review B, vol 72, no 20, 205104

[11] Reiter, G. F. et al (2002) *Direct Observation of Tunneling in KDP using Neutron Compton Scattering*, Physical Review Letters, vol 89, no 13, pp. 283-288.

[12] Momma, K. and Izumi, F. (2011) *VESTA 3 for three-dimensional visualization of crystal, volumetric and morphology data*, Journal of Applied Crystallography, vol 44, no 6, pp. 1272-1276.

[13] Matsumoto, M. and Nishimura, T. (1998) *Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator* ACM Trans. on Modeling and Computer Simulation, vol 8, no 1, pp. 3-30.

[14] Miller, A. (1999) *Allan Millers Fortran Software*, `http://jblevins.org/mirror/amiller/mt19937.f90` (2014-05-03).

[15] Trotter, H. F. (1959), *On the product of semi-groups of operators* Proceedings of the American Mathematical Society, vol 10, no 4, pp. 545-551

[16] Fransson, E. Laurell Håkansson, J. (2014) *Local and Global Ordering in Barium Zirconate, a Model Potential Study*

[17] Jedvik, E. (2014) *Chemical expansion in perovskite structured oxides*

[18] Perram, J. W. et al (1988) *An algorithm for the simulation of condensed matter which grows as the $N^{3/2}$ power of the number of particles* Molecular Physics, vol 65, no 4, pp. 875-893

[19] Gibbon, P. and Sutmann, G. (2002) *Long-Range Interactions in Many-Particle Simulation* NIC series, vol 10, pp. 467-506

[20] Kiang, C. and Goddard III, W. A. (1992) *Effective Hamiltonians for motions with disparate time scales: The quantum shell model and the classical statistical shell model* Journal of Chemical Physics, vol 98, no 2, pp. 1451-1457

Appendices

APPENDIX A

THE EWALD SUMMATION

In the following section we will present a formal derivation of the Ewald summation method for computing the total electrostatic (or Coulombic) interaction of a charge-neutral ionic system. We will also briefly discuss how to select appropriate values for the various parameters. To keep things simple we will consider a cubic simulation box with side L, volume V and containing N ions. If we further assume periodic boundary conditions, we saw in Chapter 3 that the potential can be written as a sum over the basic simulation cell and all its periodic images, defined by translation vectors of the form $\mathbf{n} = L(n_x,n_y,n_z)$ with $n_i \in \mathbb{Z}$. Thus, we may write (in atomic units) the electrostatic potential at a point of observation $\mathbf{x}$ due to all the ions located at $\mathbf{x}_j$ on the following form

$$\Phi(\mathbf{x}) = \sum_{\mathbf{n}} \sum_{j=1}^{N} \frac{q_j}{|\mathbf{x} - \mathbf{x}_j + \mathbf{n}|} \tag{A.1}$$

where we implicitly assume that if $\mathbf{x} = \mathbf{x}_j$ for any j this term is excluded for $\mathbf{n} = 0$. Evaluating this expression might look straightforward but since the terms only decay as $1/r$ we end up with a sum that is, in mathematical terms, conditionally convergent meaning that the result will depend on the order in which the terms were added. To fix this one can build up the sum in succesive shells surrounding the primary cell, forming a spherical arrangement. This will solve the issue of the sum not being convergent but the rate of convergence will still be slow due to the $1/r$-dependence. To remedy this we will rewrite the potential at an ionic site $\mathbf{x}_i$ as

$$\Phi(\mathbf{x}_i) = \Phi(\mathbf{x}_i) + \Phi_r(\mathbf{x}_i) + \Phi_k(\mathbf{x}_i) - \Phi_s \tag{A.2}$$

Where the terms $\Phi(\mathbf{x}_i)+\Phi_r(\mathbf{x}_i)$ and $\Phi_k(\mathbf{x}_i)$ correspond to sums in real- and reciprocal space respectively which both converge very rapidly and Φ_s is a constant. This expansion is easy to understand from a physical viewpoint, the term Φ_r comes from an imagined set of smeared charges centered on the ionic sites, where the total charge of each of the

smeared charges is equal to the corresponding ionic charge but with the sign reversed. The potential contribution from these smeared charges is then Φ_r. To compensate for adding this to Φ in Eq. (A.2) we must then add the contribution from a new set of smeared charges, this time with the same sign as their corresponding ions. Expanding this contribution in reciprocal space and compensating for a self-interaction term yields the last two terms $\Phi_k(\mathbf{x})$ and Φ_s. Clearly, the equality in Eq. (A.2) must then hold since all we have done is add potential contributions from two sets of smeared charges with opposite sign but otherwise identical.

To be more precise, consider first the smearing distribution given by a dimensionless gaussian distribution

$$\sigma(\mathbf{x}) = \frac{\alpha^3}{\pi^{3/2}} e^{-\alpha^2 \mathbf{x}^2}. \tag{A.3}$$

Note that this distribution is normalised to unity and that α controls the width of the gaussian peak (α is often referred to as the splitting parameter for reasons that soon will become apparent). First we will consider the real-space summation, to that end imagine thus a set of charges, smeared according to σ and centered on the ionic sites $\mathbf{x}_j$, each with a total charge of $-q_j$. To determine the potential contribution at an arbitrary observation point $\mathbf{x}$ (rather than an ionic site $\mathbf{x}$) we can begin by considering the simpler problem of determining the potential Φ_g from a single gaussian charge distribution $q\sigma$ located at the origin. Since σ is spherically symmetric this potential is just the solution to the radial Poisson equation[1]

$$\frac{1}{r}\frac{\partial^2}{\partial r^2}\left(r\Phi_g(r)\right) = -4\pi q\sigma(r) \tag{A.4}$$

which can be solved by integrating twice. We find that

$$\Phi_g(r) = q\frac{\mathrm{erf}(\alpha r)}{r} \tag{A.5}$$

where $\mathrm{erf}(r) = \frac{2}{\sqrt{\pi}}\int_0^r ds e^{-s^2}$ is known as the error function. The potential Φ_r from our set of smeared, reversed charges located at $\mathbf{x}_j + \mathbf{n}$ can thus be written

$$\Phi_r(\mathbf{x}) = -\sum_{\mathbf{n}}\sum_{j=1}^{N} \frac{q_j \mathrm{erfc}\left(\alpha\left|\mathbf{x}-\mathbf{x}_j+\mathbf{n}\right|\right)}{\left|\mathbf{x}-\mathbf{x}_j+\mathbf{n}\right|}. \tag{A.6}$$

Adding this contribution to Eq. (A.1) we get

$$\Phi(\mathbf{x}) + \Phi_r(\mathbf{x}) = \sum_{\mathbf{n}}\sum_{j=1}^{N} \frac{q_j\left(1-\mathrm{erf}\left(\alpha\left|\mathbf{x}-\mathbf{x}_j+\mathbf{n}\right|\right)\right)}{\left|\mathbf{x}-\mathbf{x}_j+\mathbf{n}\right|} = \sum_{\mathbf{n}}\sum_{j=1}^{N} \frac{q_j \mathrm{erfc}\left(\alpha\left|\mathbf{x}-\mathbf{x}_j+\mathbf{n}\right|\right)}{\left|\mathbf{x}-\mathbf{x}_j+\mathbf{n}\right|} \tag{A.7}$$

[1]Note that in atomic units the usual SI $1/\epsilon_0$ factor in the Poisson equation is replaced by a factor 4π.

where we have introduced the complementary error function $\text{erfc}(r) = 1 - \text{erf}(r)$. We have now replaced the $1/r$ decay by $\text{erfc}(r)/r$ which falls off rapidly as r increases. Consider now the second set of smeared charges, this time with the same total charges as their corresponding ions i.e q_j. The total charge distribution for the entire set of smeared charges can be written

$$\rho_\sigma(\mathbf{x}) = \sum_{j=1}^{N} q_j \sigma(\mathbf{x} - \mathbf{x}_j) \tag{A.8}$$

while the total charge distribution for the initial set of point charges is

$$\rho(\mathbf{x}) = \sum_{j=1}^{N} q_j \delta(\mathbf{x} - \mathbf{x}_j). \tag{A.9}$$

Combining Eq. (A.9) and Eq. (A.8) we can write $\rho_\sigma(\mathbf{x})$ as a convolution

$$\rho_\sigma(\mathbf{x}) = \int d\mathbf{x}' \rho(\mathbf{x} - \mathbf{x}')\sigma(\mathbf{x}'). \tag{A.10}$$

Now, because of the periodic boundary conditions, ρ_σ has the same periodicity as the simulation box and can thus be expand in k-space. Recalling that the Fourier transform of a convolution is just the product of the individual transforms we get

$$\rho_\sigma(\mathbf{x}) = \frac{1}{V} \sum_{\mathbf{k}} c(\mathbf{k})\hat{\sigma}(\mathbf{k}) e^{-i\mathbf{k}\cdot\mathbf{x}} \tag{A.11}$$

where $\mathbf{k} = \frac{2\pi}{L}(n_{k_x}, n_{k_y}, n_{k_z})$ with $n_{k_i} \in \mathbb{Z}$, $c(\mathbf{k})$ is the Fourier coefficients of ρ and $\hat{\sigma}$ is the (continuous) fourier transform of σ given by

$$\hat{\sigma}(\mathbf{k}) = \int d\mathbf{x} \sigma(\mathbf{x}) e^{i\mathbf{k}\cdot\mathbf{x}} = e^{\frac{-|\mathbf{k}|^2}{4\alpha}} \tag{A.12}$$

as can be confirmed by consulting any table of common Fourier transforms. Using Eq. (A.11) gives the the following expression for the electrostatic potential corresponding to ρ_σ

$$\Phi_k(\mathbf{x}) = \int d\mathbf{x}' \frac{\rho_\sigma(\mathbf{x}')}{|\mathbf{x} - \mathbf{x}'|} = \frac{1}{V} \sum_{\mathbf{k}} c(\mathbf{k})\hat{\sigma}(\mathbf{k}) \int d\mathbf{x}' \frac{e^{-i\mathbf{k}\cdot\mathbf{x}'}}{|\mathbf{x} - \mathbf{x}'|}. \tag{A.13}$$

Since integral to the right is over all of space we can conveniently choose $\mathbf{x}$ as the origin and evaluate the integral in spherical coordinates. The manipulations are standard and the result is simply

$$\int d\mathbf{x}' \frac{e^{-i\mathbf{k}\cdot\mathbf{x}'}}{|\mathbf{x}'|} = \frac{4\pi}{|\mathbf{k}|^2}. \tag{A.14}$$

Furthermore, the Fourier coefficients $c(\mathbf{k})$ in Eq. (A.13) are

$$c(\mathbf{k}) = \int d\mathbf{x}\rho(\mathbf{x})e^{i\mathbf{k}\cdot\mathbf{x}} = \sum_{j=1}^{N} q_j e^{i\mathbf{k}\cdot\mathbf{x}_j} \tag{A.15}$$

since ρ is a sum of Dirac δ-functions. Finally, substituting Eq. (A.12), Eq. (A.14) and Eq. (A.15) into Eq. (A.13) yields a final expression for Φ_k

$$\Phi_k(\mathbf{x}) = \frac{4\pi}{V} \sum_{\mathbf{k}\neq 0} \sum_{j=1}^{N} q_j \frac{e^{\frac{-|\mathbf{k}|^2}{4\alpha}}}{|\mathbf{k}|^2} e^{i\mathbf{k}\cdot(\mathbf{x}_j - \mathbf{x})} \tag{A.16}$$

Note that if $\mathbf{x} = \mathbf{x}_i$ for an ionic site this term contains the unphysical contribution from a smeared charge located at $\mathbf{x}_i$ which turns out as

$$\Phi_s = \int d\mathbf{x}' \frac{q_i \sigma(\mathbf{x}')}{|\mathbf{x}'|} = \frac{2\alpha}{\sqrt{\pi}} q_i. \tag{A.17}$$

We have now obtained explicit expressions for the terms Φ_r, Φ_k and Φ_s in Eq. (A.2) through Eqs. (A.7), (A.16) and (A.17). The total electrostatic potential at an ionic site $\mathbf{x}_i$ can thus be expanded as

$$\begin{aligned} \Phi(\mathbf{x}_i) = &\sum_{j=1}^{N} \frac{q_j \mathrm{erfc}\left(\alpha\left|\mathbf{x}_i - \mathbf{x}_j + \mathbf{n}\right|\right)}{\left|\mathbf{x}_i - \mathbf{x}_j + \mathbf{n}\right|} \\ &+ \frac{4\pi}{V} \sum_{\mathbf{k}\neq 0} \sum_{j=1}^{N} q_j \frac{e^{\frac{-|\mathbf{k}|^2}{4\alpha}}}{|\mathbf{k}|^2} e^{i\mathbf{k}\cdot(\mathbf{x}_j - \mathbf{x}_i)} \\ &- \frac{2\alpha}{\sqrt{\pi}} q_i \end{aligned} \tag{A.18}$$

and it follows immediately that the total electrostatic energy of the system is

$$\begin{aligned} \mathcal{V}_{\mathrm{coul}} = &\frac{1}{2} \sum_{\mathbf{n}} \sum_{i=1}^{N} \sum_{j=1}^{N} q_i q_j \frac{\mathrm{erfc}(\alpha\left|\mathbf{x}_{ij} + \mathbf{n}\right|)}{\left|\mathbf{x}_{ij} + \mathbf{n}\right|} \\ &+ \frac{2\pi}{V} \sum_{\mathbf{k}\neq 0} \sum_{i=1}^{N} \sum_{j=1}^{N} q_i q_j \exp\left(i\mathbf{k}\cdot\mathbf{x}_{ij}\right) \frac{\exp\left(\frac{-k^2}{4\alpha}\right)}{k^2} \\ &- \frac{\alpha}{\sqrt{\pi}} \sum_i q_i^2 \end{aligned} \tag{A.19}$$

which we recognise as the expression given in Eq. (3.9) in Chapter 3. Eqs. (A.18) and (A.19) look quite daunting but are in fact simple to calculate on a computer. Indeed, since both the sums over $\mathbf{n}$ and $\mathbf{k}$ respectively have a very high rate of convergence we can truncate the sums by introducing two sperical cut-offs, one for the $\mathbf{n}$-sum and one for the $\mathbf{k}$-sum (much like we did for the Buckingham potential in Section 3.3).

Hence, using r_{cut} and k_{cut} to denote these two cut-offs we only include terms for which $r = |\mathbf{x}_{ij} + \mathbf{n}| < r_{\text{cut}}$ and $k = |\mathbf{k}| < k_{\text{cut}}$ in our summation. The question remaining then is how to choose these cut-offs to get an optimal rate of convergence, this issue turns out to be intimately related to the one free parameter in Eqs. (A.18) and (A.19), namely the splitting parameter α. An analysis orginally made by [18] gives a simple relation between the Ewald summation parameters (r_{cut},k_{cut},α) and a fourth parameter $\epsilon = \exp(-p)$ which represents the smallest value a term in the real space summation can have and still be included:

$$\alpha = \frac{\sqrt{p}}{r_{\text{cut}}} \tag{A.20}$$

$$k_{\text{cut}} = \frac{2p}{r_{\text{cut}}} \tag{A.21}$$

$$r_{\text{cut}} = \sqrt{\pi} L N^{1/6} \tag{A.22}$$

In a simulation one can regard ϵ as a measure of the truncation error and set this equal to a small number, say 1×10^{-7}, and then compute $p = -\log(\epsilon)$ at which point the ewald summation parameters can be directly determined from Eqs. (A.22), (A.21) and (A.20). An implementation of the Ewald summation in the form of Eq. (A.18) can be found in appendix B. To show the significance of the splitting parameter α we have conducted a very limited convergence study. Here, a number of simulations were performed on a $BaZrO_3$ system where the total electrostatic energy for the $T = 0$ equilibrium configuration was calculated for several different values of the Ewald summation parameters. The results are displayed in Fig. A.1 where the total electrostatic energy has been plotted as a function of the dimensionless combination αL for several different cases where we have varied the number of terms included in the real- and reciprocal space summations.

The values $n_{\max}$ and $n_{k,\max}$ given in Fig. A.1 should be understood as the largest $n = |\mathbf{n}|$ such that $|\mathbf{x}_{ij} + \mathbf{n}| < r_{\text{cut}}$ and the largest $n_k = (n_{k_x}^2 + n_{k_y}^2 + n_{k_z}^2)^{1/2}$ such that $n_k < k_{\text{cut}}/(2\pi/L)$. From Fig. A.1 it is clear that values of αL in the range $3 - 8$ yield the best results. We can also infer since the calculated energy should be independent of the value of α that a failure to converge at small values $\alpha L \approx 1$ occurs because not enough terms have been included in the real space summation. Similarily, failure to converge at larger values $\alpha L \approx 10$ is due to not enough terms having been included in the reciprocal space summation. In conclusion, by adjusting the value of α one sets the relative convergence rate of the two sums, hence the name splitting parameter. A good reivew on the Ewald summation and related techniques can be found in [19].

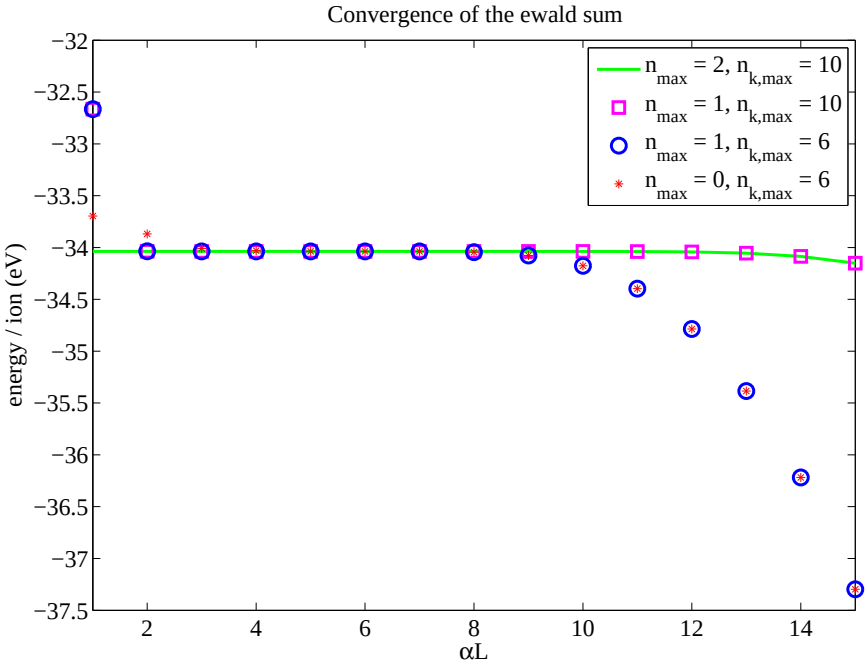


Figure A.1. *A small convergence study of the Ewald summation. The total electrostatic energy for a $BaZrO_3$ system is shown as a function of the dimensionless number αL. The different markers represent simulations where the number of terms included in the real- and reciprocal space summations have been varied. Note that for small αL runs with only a few terms included in the real space sum fail to convergence while for larger αL the same holds true for runs which included a smaller number of terms in the reciprocal space summation.*

APPENDIX B

NOTES ON PROGRAMMING

This appendix includes the source code for the most important subroutines used to perform the PIMC simulations as well as some brief general notes on the programming aspect of the thesis. Fortran 90 (F90) has been our language of choice for implementing the algorithms described throughout this report. The main reason behind this choice is that Fortran is specifically designed for scientific computations and as such the compilers are highly optimised in order to generate fast executing code. There are also excellent array handling capabilities, dynamic memory allocation as well as a large collection of useful intrinsic functions.

B.1 Programming in Fortran 90

As an example of how to code in Fortran 90 consider implementing the algorithm described in Chapter 3 Section 3.4 for calculating the total short-range ínteraction of the system given by the Buckingham pontential. As usual the basic simulation domain is a cubic box with side L, defined so that the lower leftmost corner coincides with the origin i.e. the box is entirely located in the positive octant. The positions of the particles are stored in an $3\times N$ array called `x`. If a particle with label i has moved outisde the box, the following line of code will periodically translate the particle back into the box in agreement with the periodic boundary conditions:

```
x(:,i) = x(:,i) - L * floor(x(:,i) / L)
```

Note here the use of the colon operator which, in almost all cases, has the same meaning in Fortran as in Matlab. Thus whenever we move particles in the code we should use the line of code above to make sure all paritcles are inside the simulation box. Calculating the potential will require us to compute the minimum image distance between two particles, this can be accomplished with the following lines of code:

```
xij = x(:,i) - x(:,j)
xij = xij - L * nint(xij / L)
```

Here the `nint` function rounds the argument to the nearest integer. Keeping this common contruction in mind we can now write the main loop which calculates the total interaction energy for the system.

```
! A FORTRAN 90 loop for calculating the energy
! Assumes variables/array have been declared and
! initialised with appropriate values.
energy = 0.0
Linv = 1.0 / L
do i = 1, N - 1
  xi = x(:,i)
  do j = i + 1, N
    xij = xi - x(:,j)
    xij = xij - L * nint(xij * Linv)
    rSq = xij(1)*xij(1) + xij(2)*xij(2) + xij(3)*xij(3)
    if(rSq < rCutSq) then
      r = sqrt(rSq)
      energy = energy + A * exp(-r/rho) - C / rSq**3
    end if
  end do
end do
```

In this we have introduced a number of common optimisations that the reader should be aware of. Firstly, to reduce the number of references into the matrix `x`, individual particle coordinates are temporarily stored in `xi` already in the outermost loop. Secondly, we try to avoid any divisions that are not necessary since multiplications are faster and hence we choose multiply with the inverse of the box length when calculating the minimum image distance. Lastly, we calculate the squared minimum image distance and compare the result to the squared cut-off radius to avoid calculating the square root unless it is actually needed (in general, transcendental functions are relatively expensive to calculate). On older workstations tricks like these were crucial for writing fast executing code but nowadays one can, to a certain extent, approach the optimisation procedure with a bit more leniency. Although this example was rather basic, it includes many important features such as the computation of the minmum image distance which are ubiquitous is the larger code found in Section B.3. Note also the similariy between the snippet of code above and Matlab syntax. It is the author's hope that most readers are familiar with Matlab and are thus able to decipher the code above as well the subroutines found in Section B.3 even if they have had little or no previous experience with Fortran.

We also briefly note the coding convention used here which differs from many other routines or libraries written in Fortran where the author has noticed a predisposition towards cryptic naming of important variables, sparse commentary and other practices which are not very helpful. To make the code more accessible for anyone that wishes to reproduce the results found in this work, build on the results or use certain parts of the code et cetera we have elected to use a coding convetion common to e.g. java programmers. In this convention, a variable name always start with a small letter, has no spaces or underscores and begins each new word after the first one with a captial letter e.g. `energyDifference`. An important special case is variables used as indices into arrays, these begin with the letter i, sometimes followed by a descriptive word e.g. `iAtom`. Function or subroutines on the other hand always begin with a capital letter

e.g. `IntialiseLattice`. The general rule of thumb is that variable names should be as descriptive as possible without obfuscating the code although exceptions are frequently made for temporary or auxiliary variables.

B.2 Random number generation

Monte Carlo simulations in general require that one can generate random numbers. In particular, in this thesis we have seen the need to generate real numbers and integers which are uniformly distributed as well as normally-distributed real number. Of course, a computer is fundamentally unable to generate true random numbers and the closest one can get is an algorithm which generates a deterministic sequence of numbers which resemble a random sequence. Such an algorithm is referred to as a pseudo random number generator (PRNG). For Monte Carlo simulations and many other applications the go-to choice of PRNG is the *Mersenne Twister* (MT) [13]. The hallmark of a good PRNG is a long period i.e. the longest unrepreated sequence and a low correlation between the numbers generated. The Mersenne Twister excels in both of these aspects and has passed several difficult benchmarking tests for PRNGs, notably the Diehard tests. For our PIMC simulations we have thus used a Fortran 90 version of the MT based on a Fortran 77 version by the inventors of the algorithm, subsequently adapted to F90 in [14].

B.3 Source code

Below we provide the source code for most of the programs inplemented in this thesis as a reference for the curious reader. Things from the main code which are **not included** here are some simple io-routines to write results and a module containing a long list of variables for global use. Also note that for the bisection sampling version of the main PIMC program some parts, where the code did not change much, have been omitted for brevity. The main aim here is to illustrate how one goes about implementing the concepts described in the thesis rather than providing a ready-to-use code for path intregral calculations.

```
!******************************************************************************
!                                InitialiseLattice.f90
!******************************************************************************
! Initialises a nxCells*nyCells*nzCells supercell, constructed out of Barium
! zirconate unitcells, each of which contains 1 Ba atom, 1 Zr atom and 3 O atoms.
! Also create two arrays atomType and charge containing a number to identify the
! type of atom and charge, respectively.
!******************************************************************************
subroutine InitialiseLattice()
  use GlobalVariables
  implicit none

  !------------------ Local variables
  integer :: k, i, j, m, ind
  integer, parameter :: nBasis = 5
  double precision, dimension(3,nBasis) :: xBasis
  double precision, dimension(3) :: xTrans, xPrim1, xPrim2, xPrim3
```

```
nAtoms = nxCells * nyCells * nzCells * nBasis

boxDim = latticeConst * (/dble(nxCells), dble(nyCells), dble(nzCells)/)
Lx = boxDim(1)
Ly = boxDim(2)
Lz = boxDim(3)
volume = Lx * Ly * Lz
Linv = 1.0d0 / Lx

!------------------- Define atoms in basis
xBasis(:,1) = (/0.5d0, 0.5d0, 0.5d0/) !Ba
xBasis(:,2) = (/0.0d0, 0.0d0, 0.0d0/) !Zr
xBasis(:,3) = (/0.5d0, 0.0d0, 0.0d0/) !O
xBasis(:,4) = (/0.0d0, 0.5d0, 0.0d0/) !O
xBasis(:,5) = (/0.0d0, 0.0d0, 0.5d0/) !O
xBasis = latticeConst * xBasis

!------------------- Define primitive lattice vectors
xPrim1 = latticeConst * (/1.0d0, 0.0d0, 0.0d0/)
xPrim2 = latticeConst * (/0.0d0, 1.0d0, 0.0d0/)
xPrim3 = latticeConst * (/0.0d0, 0.0d0, 1.0d0/)

!------------------- Construct the position vectors for the atoms
ind = 0
do k = 1, nzCells
  do j = 1, nyCells
    do i = 1, nxCells
      xTrans = (i-1)*xPrim1 + (j-1)*xPrim2 + (k-1)*xPrim3
      do m = 1, nBasis
        x0(:,m + ind) = xTrans + xBasis(:,m)
      end do
      ind = ind + nBasis
    end do
  end do
end do

!------------------- Create vectors containing atom type, mass and charge
do i = 1, nAtoms
  select case(mod(i,nBasis))
    case(1)
      charge(i) = 2
      atomType(i) = 1
      mass(i) = 137.327d0 * massFac - 2.0d0
    case(2)
      charge(i) = 4
      atomType(i) = 2
      mass(i) = 91.224d0 * massFac - 4.0d0
    case(3)
      charge(i) = -2
      atomType(i) = 3
      mass(i) = 15.9994d0 * massFac + 2.0d0
    case(4)
      charge(i) = -2
      atomType(i) = 3
      mass(i) = 15.9994d0 * massFac + 2.0d0
    case(0)
      charge(i) = -2
      atomType(i) = 3
      mass(i) = 15.9994d0 * massFac + 2.0d0
  end select
end do
```

```
end subroutine InitialiseLattice

!*******************************************************************************
!                                  Potential.f90
!*******************************************************************************
! Calculates the energy of atom iAtom in the configuration specified by x.
! Includes both the Buckingham potential and Ewald summation. Must not be called
! before an intial call to InitialiseLattice() and requires that the potential
! parameter rPairCut and ewaldError have been set.
!*******************************************************************************
function Potential(x, iAtom) result(energy)
  use GlobalVariables
  implicit none

  !------------------- Local variables
  integer :: iAtom, j, chargei, chargej, nx, ny, nz, nMax, atomTypei, atomTypej
  integer :: kMax, kx, ky, kz, nZeros
  double precision, dimension(3) :: xi, xij
  double precision :: energy, energyShift13, energyShift23, energyShift33
  double precision :: r, rSq, rPairCutSq, rCoulCutSq
  double precision, dimension(3, nAtoms) :: x
  double precision :: alpha, timeFac, L, p
  double precision :: ewaldReci, selfEnergy
  double precision :: a, b, bSq, kCut, kSq, kCutSq, symmFac, cosFac

  rPairCutSq = rPairCut**2
  L = boxDim(1)
  energyShift13 = A13 * exp(-rPairCut/rho13)
  energyShift23 = A23 * exp(-rPairCut/rho23)
  energyShift33 = A33 * exp(-rPairCut/rho33) - C33 / rPairCutSq**3

  !------------------- Choose optimal Ewald parameter values
  L = boxDim(1)
  p = - log(ewaldError)
  timeFac = 1.0d0
  alpha = sqrt(pi) * (timeFac * nAtoms / volume**2)**(1.0d0/6.0d0)
  rCoulCut = sqrt(p) / alpha
  rCoulCutSq = rCoulCut**2
  kCut = 2.0d0 * alpha * sqrt(p)
  nMax = floor(rCoulCut / L)
  kMax = ceiling(kCut / (2.0d0 * pi / L))

  xi = x(:,iAtom)
  chargei = charge(iAtom)
  atomTypei = atomType(iAtom)
  !------------------- Compute the buckingham potential and real part of ewald sum
  energy = 0.0d0
  do nx = -nMax, nMax
    do ny = -nMax, nMax
      do nz = -nMax, nMax
        do j = 1, nAtoms
          if((abs(nx) + abs(ny) + abs(nz) /= 0) .OR. j /= iAtom) then
            xij = xi - x(:,j)
            xij = xij - L * nint(xij / L)
            rSq = (xij(1) + L*nx)**2 + (xij(2) + L*ny)**2 + (xij(3) + L*nz)**2
            chargej = charge(j)
            atomTypej = atomType(j)

            if(rSq < rPairCutSq) then
              r = sqrt(rSq)
              if(atomTypei == 3 .AND. atomTypej == 3) then
                energy = energy + A33 * exp(-r/rho33) - C33 / rSq**3 - energyShift33 !O-O interactions
```

```
            endif
            if((atomTypei == 1 .AND. atomTypej == 3)  &
            .OR. (atomTypei == 3 .AND. atomTypej == 1)) then
              energy = energy + A13 * exp(-r/rho13) - energyShift13 !Ba-O interactions
            endif
            if((atomTypei == 2 .AND. atomTypej == 3) &
             .OR. (atomTypei == 3 .AND. atomTypej == 2)) then
              energy = energy + A23 * exp(-r/rho23) - energyShift23 !Zr-O interactions
            endif
            energy = energy + chargei*chargej * erfc(alpha*r) / r
          elseif(rSq < rCoulCutSq) then
              r = sqrt(rSq)
              energy = energy + chargei*chargej * erfc(alpha*r) / r
          else
          endif
        endif
      end do
    end do
  end do
end do

!------------------- Compute the reciprocal part of the ewald sum
ewaldReci = 0.0d0
b = 2.0d0 * pi / L
bSq = b**2
kCutSq = kCut**2
cosFac = 0.0d0
a = 1.0d0 / (4.0d0*alpha**2)
nZeros = 0
symmFac = 0.0d0
do kx = 0, kMax
  do ky = 0, kMax
    do kz = 0, kMax
      kSq = bSq * (kx**2 + ky**2 + kz**2)
      if(kSq >= kCutSq .OR. (abs(kx) + abs(ky) + abs(kz)) == 0) then
        cycle
      endif
      nZeros = 0
      if(kx == 0) then
        nZeros = nZeros + 1
      endif
      if(ky == 0) then
        nZeros = nZeros + 1
      endif
      if(kz == 0) then
        nZeros = nZeros + 1
      endif
      if(nZeros == 0) then
        symmFac = 8.0d0
      elseif(nZeros == 1) then
        symmFac = 4.0d0
      else
        symmFac = 2.0d0
      endif
      do j = 1, nAtoms
        xij = x(:,j) - xi
        cosFac = cos(b*kx * xij(1)) * cos(b*ky * xij(2)) * cos(b*kz * xij(3))
        ewaldReci = ewaldReci + symmFac / kSq * exp(-kSq*a) * chargei*charge(j)*cosFac
      end do
    end do
  end do
end do
```

```
  ewaldReci = 4.0d0 * pi * ewaldReci / volume

  !------------------- Compute self energy and add together
  selfEnergy = -2.0d0 * alpha / sqrt(pi) * chargei**2
  energy = energy + selfEnergy + ewaldReci

end function Potential

!*****************************************************************************
!                                 Metropolis.f90
!*****************************************************************************
! Implements the PIMC method with single slice moves and COM displacements.
! Uses auxiliary subroutines InitialisePath() to initialise the path and SquaredDistance
! to compute the minimum image distance squared between two beads. There are also
! subroutines for initialising, updating and writing the averages and histograms
! for the various properties. An initial guess for the parameters controlling then
! ratio of accepted single bead/COM moves is required but are subsequently controlled
! automatically by the program.
!*****************************************************************************
subroutine Metropolis()
  use GlobalVariables
  use MersenneTwister
  implicit none

  !------------------- Local variables
  integer :: iRandAtom, iRandSlice, iAtom, iRight, iLeft, iSlice, moveType, iDum, nSpUpdates
  integer :: seed, iCycle, ind, nAcceptCOM, nAcceptBead, nTotalCOM, nTotalBead, iHist
  integer :: nMovesCOM, nMovesBead, nTotalAcceptCOM, nTotalAcceptBead, nCheck, nCheckCOM
  double precision, dimension(3,nAtoms) :: x
  double precision, dimension(3) :: xNew, xOld, xRight, xLeft, displaceCOM
  double precision, dimension(3, nSlices) :: xSave
  double precision :: Potential, potActionOld, potActionNew, kinActionOld, kinActionNew
  double precision :: potActionDiff, kinActionDiff, actionDiff, weightRatio, SquaredDistance
  double precision :: currentEnergy, lambda, targetRatio, acceptRatioCOM, acceptRatioBead
  double precision :: totalEnergy

  beta = 1.0d0 / (kBoltz * temperature)
  timeStep = beta / nSlices

  !------------------- Initialise the Mersenne Twister PRNG
  call system_clock(seed)
  call init_genrand(seed)

  !------------------- Initialise a random path
  call InitialisePath()

  !------------------- Main loops
  nAcceptBead = 0
  nAcceptCOM = 0
  nMovesCOM = 0
  nMovesBead = 0
  nTotalAcceptCOM = 0
  nTotalAcceptBead = 0
  nTotalBead = 0
  nTotalCOM = 0
  nCheck = 1
  nCheckCOM = 50
  targetRatio = 0.50d0
  acceptRatioCOM = 0.0d0
  acceptRatioBead = 0.0d0
  ind = 1
  totalEnergy = 0.0d0
```

```
call InitialiseProperties()

do iCycle = 1, nCycles

   !----- COM move once per cycle
  nTotalCOM = nTotalCOM + 1
  nMovesCOM = nMovesCOM + 1
  iRandAtom = grndi(nAtoms)
  displaceCOM(1) = deltaCOM * (2.0d0 * grnd() - 1.0d0)
  displaceCOM(2) = deltaCOM * (2.0d0 * grnd() - 1.0d0)
  displaceCOM(3) = deltaCOM * (2.0d0 * grnd() - 1.0d0)

  potActionDiff = 0.0d0
  do iSlice = 1, nSlices
    x = path(:,1:nAtoms,iSlice)
    potActionOld = Potential(x, iRandAtom)
    xNew = path(:,iRandAtom,iSlice) + displaceCOM
    xNew = xNew - boxDim * floor(xNew / boxDim)
    x(:,iRandAtom) = xNew
    xSave(:,iSlice) = xNew
    potActionNew = Potential(x,iRandAtom)
    potActionDiff = potActionDiff + potActionNew - potActionOld
  end do

  !----- Metropolis rejection step
  actionDiff = timeStep * potActionDiff
  if(actionDiff < 0.0d0) then
    do iSlice = 1, nSlices
      path(:,iRandAtom,iSlice) = xSave(:,iSlice)
      end do
      nAcceptCOM = nAcceptCOM + 1
      nTotalAcceptCOM = nTotalAcceptCOM + 1
  elseif(actionDiff > 75.0d0) then
    !if the exponent is too big automatically reject
  else
    weightRatio = exp(-actionDiff)
    if(grnd() <= weightRatio) then
      do iSlice = 1, nSlices
        path(:,iRandAtom,iSlice) = xSave(:,iSlice)
      end do
      nAcceptCOM = nAcceptCOM + 1
      nTotalAcceptCOM = nTotalAcceptCOM + 1
    else
      !reject
    endif
  endif

  do iAtom = 1, nAtoms
    !------- Single bead moves
    do iDum = 1, nSlices
      nTotalBead = nTotalBead + 1
      nMovesBead = nMovesBead + 1
      iRandSlice = grndi(nSlices)
      x = path(:,1:nAtoms,iRandSlice)
      iRandAtom = grndi(nAtoms)
      xOld = x(:,iRandAtom)
      potActionOld = Potential(x, iRandAtom)

      !----- Generate a trial state
      xNew(1) = xOld(1) + deltaBead * (2.0d0 * grnd() - 1.0d0)
      xNew(2) = xOld(2) + deltaBead * (2.0d0 * grnd() - 1.0d0)
```

```
      xNew(3) = xOld(3) + deltaBead * (2.0d0 * grnd() - 1.0d0)
      xNew = xNew - boxDim * floor(xNew / boxDim) !translate back into box
      x(:,iRandAtom) = xNew
      potActionNew = Potential(x, iRandAtom)
      potActionDiff = timeStep*(potActionNew - potActionOld)

      !----- Ring polymer condition
      if(iRandSlice == nSlices) then
        iLeft = nSlices - 1
        iRight = 1
      elseif(iRandSlice == 1) then
        iLeft = nSlices
        iRight = 2
      else
        iRight = iRandSlice + 1
        iLeft = iRandSlice - 1
      endif
      xRight = path(:, iRandAtom, iRight)
      xLeft = path(:, iRandAtom, iLeft)

      !----- Compute the kinetic (spring) energy difference
      kinActionOld = SquaredDistance(xOld,xLeft) + SquaredDistance(xRight,xOld)
      kinActionNew = SquaredDistance(xNew,xLeft) + SquaredDistance(xRight,xNew)
      kinActionDiff = kinActionNew - kinActionOld
      lambda = 0.5d0 / mass(iRandAtom)
      kinActionDiff = kinActionDiff / (4 * lambda * timeStep)
      actionDiff = potActionDiff + kinActionDiff

      !----- Metropolis rejection step
      if(actionDiff < 0.0d0) then
        !always accept downhill moves
        path(:,iRandAtom, iRandSlice) = xNew
        nAcceptBead = nAcceptBead + 1
        nTotalAcceptBead = nTotalAcceptBead + 1
      elseif(actionDiff > 75.0d0) then
        !if the exponent is too big automatically reject
      else
        weightRatio = exp(-actionDiff)
        if(grnd() <= weightRatio) then
          path(:,iRandAtom, iRandSlice) = xNew
          nAcceptBead = nAcceptBead + 1
          nTotalAcceptBead = nTotalAcceptBead + 1
        else
          !reject
        endif
      endif
    end do !------ end do idum

  end do !---- end loop over atoms

   !------------------- Adjust to get a 50 percent acceptance ratio
  if(mod(iCycle, nCheck) == 0) then
    acceptRatioBead = dble(nAcceptBead) / dble(nMovesBead)
    if(acceptRatioBead < targetRatio) then
      deltaBead = deltaBead * 0.98d0
    else
      deltaBead = deltaBead * 1.02d0
    endif
    nAcceptBead = 0
    nMovesBead = 0
   endif
```

```
    if(mod(iCycle, nCheckCOM) == 0) then
      acceptRatioCOM = dble(nAcceptCOM) / dble(nMovesCOM)
      if(acceptRatioCOM < targetRatio) then
        deltaCOM = deltaCOM * 0.98d0
      else
        deltaCOM = deltaCOM * 1.02d0
      endif
      nAcceptCOM = 0
      nMovesCOM = 0
     endif

    !--------- Start updating averages after equilibration
    if(iCycle > nBurnIn .AND. pflagSpDensity == 1) then
      call TrailMethod()
    endif

    if(pflagPairCorr == 1) then
       do iSlice = 1, nSlices
          call PairCorrelation(path(:,:,iSlice), 1)
          nPCorrUpdates = nPCorrUpdates + 1
       end do
    endif

    if(iCycle > nBurnIn .AND. pflagEnergy == 1) then
      call EnergyEstimator(currentEnergy)
      totalEnergy = totalEnergy + currentEnergy
      energyVec(ind) = totalEnergy / dble(iCycle - nBurnIn)
      ind = ind + 1
    endif


  end do !---- end loop over cycles

  call AverageProperties()
  call WriteProperties()

end subroutine Metropolis

!*****************************************************************************
!                                SquaredDistance.f90
!*****************************************************************************
! Returns the minimum image distance squared between two beads with
! coordinates x1, x2
!*****************************************************************************
function SquaredDistance(x1, x2) result(rSq)
  use GlobalVariables
  implicit none
  double precision :: rSq, xDist, yDist, zDist
  double precision, dimension(3) :: x1, x2

  xDist = x1(1) -  x2(1)
  yDist = x1(2) -  x2(2)
  zDist = x1(3) -  x2(3)
  xDist = xDist - Lx * nint(xDist / Lx)
  yDist = yDist - Ly * nint(yDist / Ly)
  zDist = zDist - Lz * nint(zDist / Lz)
  rSq = xDist*xDist + yDist*yDist + zDist*zDist

end function SquaredDistance

!*****************************************************************************
!                                InitialisePath.f90
```

```
!*****************************************************************************
! Initialises a path to be used in the main PIMC simulation
! using the T = 0 equilibrium lattice configuration, requires that
! InitialiseLattice() has already been called.
!*****************************************************************************
subroutine InitialisePath()
  use GlobalVariables
  use MersenneTwister
  implicit none

  !------------------- Local variables
  integer :: iSlice, iFirst, iLast, i
  double precision, dimension(3, nAtoms) :: x
  double precision :: dx

  !------------------- Initialise a path
  path = 0.0d0
  dx = 0.5 * lengthFac

  do iSlice = 1,nSlices
    x = path(:,1:nAtoms,iSlice)
    do i = 1, nAtoms
        x(1,i) = x0(1,i)
        x(2,i) = x0(2,i)
        x(3,i) = x0(3,i)
    end do
    path(:,1:nAtoms,iSlice) = x
  end do
  path(:,1:nAtoms,nSlices+1) = 0.0d0

end subroutine InitialisePath

!*****************************************************************************
!                                  EnergyEstimator.f90
!*****************************************************************************
! Implements the direct (Hamiltonian) estimator for the internal energy
!*****************************************************************************
subroutine EnergyEstimator(energy)
  use GlobalVariables
  implicit none

  !------------------- Local variables
  double precision :: energy, kinEnergy, potEnergy, constEnergy, lambda, Potential, SquaredDistance
  double precision :: kinTmp
  integer :: iSlice, iAtom, idm, iRight
  double precision, dimension(3,nAtoms) :: x
  double precision, dimension(3) :: xRight, xCurr

  !------------------- Calculate the total spring energy
  kinEnergy = 0.0d0
  kinTmp = 0.0d0
  do iAtom = 1,nAtoms
    kinTmp = 0.0d0
    lambda = 0.5d0 / mass(iAtom)
    do iSlice = 1, nSlices
      iRight = iSlice + 1
      if(iRight == nSlices + 1) then
        iRight = 1
      endif
      xRight = path(:,iAtom,iRight)
      xCurr = path(:,iAtom,iSlice)
      kinTmp = kinTmp + SquaredDistance(xCurr, xRight)
```

```
    end do
    kinTmp = kinTmp / lambda
    kinEnergy = kinEnergy + kinTmp
  end do
  kinEnergy =  kinEnergy / (4.0d0 * timeStep**2) / dble(nSlices) / dble(nAtoms)

  !------------------- Calculate the total potential energy
  potEnergy = 0.0d0
  do iSlice = 1, nSlices
    x = path(:, 1:nAtoms, iSlice)
    do iAtom = 1, nAtoms
      potEnergy = potEnergy + Potential(x, iAtom)
    end do
  end do
  potEnergy = potEnergy / nSlices
  if(nAtoms > 1) then
    potEnergy = 0.5d0 * potEnergy
  endif

  constEnergy = 3.0d0 / timeStep / 2.0d0
  energy =  constEnergy + potEnergy - kinEnergy
end subroutine EnergyEstimator

!*****************************************************************************
!                                TrailMethod.f90
!*****************************************************************************
! Computes the single-particle density matrix using the Trail metod
! Adapted to F90 (and slightly modified) from the F77 version found in Brualla (2002).
! The dispFac parameter must be set manually, as an intial
! guess start with a number on the order of the thermal wavelength and then tune
! by hand.
!*****************************************************************************
subroutine TrailMethod()
  use MersenneTwister
  use GlobalVariables
  implicit none

  !------------------- Local variables
  integer :: iCut, iSlice, iAtom, typeSelected, iIter, iTmp
  integer :: iLeft, iHist, nTypeSelected
  double precision :: typeMass, thermalWavel, potActionOld, kinActionOld, Potential
  double precision :: kinFac, lambda, mu, dispFac, dx, dy, dz, SquaredDistance
  double precision :: kinActionNew, potActionNew, actionDiff, rMax, dr, rMaxSq, r, rSq, L2Sq
  double precision, dimension(3, nAtoms) :: x
  double precision, dimension(3,nSlices + 1) :: xNew
  double precision, dimension(3) :: xLeft, xCurr, xEndDist
  double precision :: kinActionDiff, potActionDiff

  !------------------- Initialise
  typeSelected = 3
  if(typeSelected == 3) then
    nTypeSelected = nxCells * nyCells * nzCells * 3
  elseif(typeSelected < 3) then
    nTypeSelected = nxCells * nyCells * nzCells
  endif

  typeMass = mass(typeSelected)
  lambda = 0.5d0 / typeMass
  kinFac = 1.0d0 / (4.0d0 * lambda * timeStep)
  dispFac = 0.075d0 * lengthFac

  rMax = dispFac * 2.0d0
```

```
dr = rMax / nHist
rMaxSq = rMax * rMax

!------------------- Begin main iteration loop
do iIter = 1, nTypeSelected
  iCut = grndi(nSlices)
  iAtom = grndi(nAtoms)
  do while(atomType(iAtom) /= typeSelected)
    iAtom = grndi(nAtoms)
  end do

  xNew = 0.0d0
  !------------------- Compute old potential action
  potActionOld = 0.0d0
  iSlice = iCut
  do iTmp = nSlices+1, 1, -1 !note contribution from extra bead
    x = path(:,1:nAtoms,iSlice)
    potActionOld = potActionOld + Potential(x, iAtom)
    iSlice = iSlice - 1
    if(iSlice == 0) then
      iSlice = nSlices
    endif
  end do
  potActionOld = timeStep * potActionOld

  !------------------- Compute old kinetic action
  kinActionOld = 0.0d0
  iSlice = iCut
  do iTmp = nSlices, 1, -1
    iLeft = iSlice - 1
    if(iLeft == 0) then
      iLeft = nSlices
    endif
    xLeft = path(:,iAtom,iLeft)
    xCurr = path(:,iAtom,iSlice)
    rSq = SquaredDistance(xLeft,xCurr)
    kinActionOld = kinActionOld + rSq
    iSlice = iSlice - 1
    if(iSlice == 0) then
      iSlice = nSlices
    endif
  end do
  kinActionOld = kinActionOld * kinFac

  !------------------- Trail displacement
  dx = dispFac * 2.0d0 * (grnd() - 0.5d0)
  dy = dispFac * 2.0d0 * (grnd() - 0.5d0)
  dz = dispFac * 2.0d0 * (grnd() - 0.5d0)
  xNew(:,nSlices + 1) = path(:,iAtom,iCut) !new bead, doesn't move
  iSlice = iCut
  do iTmp = nSlices, 1, -1
    xNew(1,iSlice) = path(1,iAtom,iSlice) + dx * dble(iTmp) / nSlices
    xNew(2,iSlice) = path(2,iAtom,iSlice) + dy * dble(iTmp) / nSlices
    xNew(3,iSlice) = path(3,iAtom,iSlice) + dz * dble(iTmp) / nSlices
    xNew(:,iSlice) = xNew(:,iSlice) - boxDim * floor(xNew(:,iSlice) / boxDim)
    iSlice = iSlice - 1
    if(iSlice == 0) then
      iSlice = nSlices
    endif
  end do

  !------------------- Compute new kinetic action
```

```
    iSlice = iCut
    kinActionNew = 0.0d0
    do iTmp = nSlices, 1, -1
      iLeft = iSlice - 1
      if(iLeft == 0) then
        iLeft = nSlices
      endif
      if(iTmp == 1) then
        iLeft = nSlices + 1
      endif
      xLeft = xNew(:,iLeft)
      xCurr = xNew(:,iSlice)
      rSq = SquaredDistance(xLeft,xCurr)
      kinActionNew = kinActionNew + rSq
      iSlice = iSlice - 1
      if(iSlice == 0) then
        iSlice = nSlices
      endif
    end do
    kinActionNew = kinActionNew  * kinFac

    !------------------- Compute new potential action & action difference
    potActionNew = 0.0d0
    iSlice = iCut
    do iTmp = nSlices, 1, -1
      x = path(:,1:nAtoms,iSlice)
      x(:,iAtom) = xNew(:,iSlice)
      potActionNew = potActionNew + Potential(x, iAtom)
      iSlice = iSlice - 1
      if(iSlice == 0) then
        iSlice = nSlices
      endif
    end do
    x = path(:,1:nAtoms,iCut)!contribution from extra bead
    x(:,iAtom) = xNew(:,nSlices + 1)
    potActionNew = potActionNew + Potential(x, iAtom)
    potActionNew = timeStep * potActionNew
    actionDiff = potActionNew + kinActionNew - &
                 potActionOld - kinActionOld

    !------------------- Compute end-to-end distance & add to histogram
    xEndDist = path(:,iAtom,iCut) - xNew(:,iCut)
    xEndDist =  xEndDist - boxDim * nint(xEndDist / boxDim)
    rSq = xEndDist(1)**2 + xEndDist(2)**2 + xEndDist(3)**2
    if(rSq <= rMaxSq) then
      r = sqrt(rSq)
      iHist = int(r / dr) + 1
      if(iHist < nHist) then
        actionDiff = exp(-actionDiff)
        if(actionDiff > tmpMax) then
          tmpMax = actionDiff
        endif
        endHist(iHist) = endHist(iHist) + 1
        actionHist(iHist) = actionHist(iHist) + actionDiff
      endif
    endif

  end do !end main iteration loop

end subroutine TrailMethod

!*******************************************************************************
```

```
!                                MetropolisBisection.f90
!*****************************************************************************
! Same as the Metropolis.f90 program but with bisection moves instead of single
! slice moves.
!*****************************************************************************
subroutine MetropolisBisection()
  use GlobalVariables
  use MersenneTwister
  implicit none

  !------------------- Local variables
  integer, parameter :: intKind = selected_int_kind(11)
  integer(kind = intKind) :: nTotalAcceptCOM, nTotalCOM, nAcceptBisect, nTotalBisect
  integer :: iRandAtom, iRandSlice, iAtom, iLeft, iSlice, moveType, iDum, nSpUpdates
  integer :: seed, iCycle, ind, nAcceptCOM, iHist
  integer :: nMovesCOM, nCheck
  double precision, dimension(3,nAtoms) :: x
  double precision, dimension(3) :: xNew, xOld, xLeft, displaceCOM
  double precision, dimension(3, nSlices) :: xSave
  double precision :: Potential, potActionOld, potActionNew, kinActionOld, kinActionNew
  double precision :: potActionDiff, kinActionDiff, actionDiff, weightRatio, SquaredDistance
  double precision :: currentEnergy, lambda, targetRatio, acceptRatioCOM, acceptRatioBead, totalEnergy

  integer, parameter :: nLevels = 2
  integer, parameter :: nClip = 2**nLevels + 1
  integer :: iLevel, iClipFirst, iTmp, iBead, iStep, iFirst, iLast, iLow, iHigh, iRight, iClip
  integer :: flagProceed, nCurrLevel, nMoved
  double precision, dimension(3, nClip) :: xClip, xClipSave
  double precision, dimension(3) :: xLow, xBead, xHigh, xRight, xCurr
  double precision :: rnorm, variance, timeDiff, kinFac, potActionDiffPrev

  beta = 1.0d0 / (kBoltz * temperature)
  timeStep = beta / nSlices

  !------------------- Initialise the Mersenne Twister PRNG
  call system_clock(seed)
  call init_genrand(seed)

  !------------------- Initialise a random path
  call InitialisePath()

  !------------------- Main loops
  nAcceptCOM = 0
  nMovesCOM = 0
  nTotalAcceptCOM = 0
  nTotalCOM = 0
  nCheck = 100
  targetRatio = 0.50d0
  acceptRatioCOM = 0.0d0
  nAcceptBisect = 0
  nTotalBisect = 0
  ind = 1
  totalEnergy = 0.0d0
  nPCorrUpdates = 0

  call InitialiseProperties()

  do iCycle = 1, nCycles

   !--------------------------
   !Do COM moves
   !--------------------------
```

```
if(nSlices /= 1) then
!----------- Bisection moves
do iAtom = 1, nAtoms
    nTotalBisect = nTotalBisect + 1
    !iRandAtom = grndi(nAtoms) !polymer to bisect
    iRandAtom = iAtom
    lambda = 0.5d0 / mass(iRandAtom)
    iClipFirst = grndi(nSlices) !starting bead of the clip
    iSlice = iClipFirst
    do iClip = 1, nClip !map the clip from path to xClip
      xClip(:,iClip) = path(:, iRandAtom, iSlice)
      xClipSave(:,iClip) = xClip(:,iClip)
      iSlice = iSlice + 1
      if(iSlice == nSlices + 1) then
        iSlice = 1
      endif
    end do

    !------ Loop over levels, starting at the coarsest level
    potActionDiffPrev = 0.0d0
    flagProceed = 0
    do iLevel = nLevels, 1, -1
      iStep = 2**(iLevel-1)
      timeDiff =  timeStep * dble(iStep)
      variance = sqrt(lambda * timeDiff)
      nMoved = 2**(nLevels - iLevel) !#beads sampled in current level

      !--- Sample new beads in current level
      iClip = 1 + iStep !index of first bead to be sampled in current level
      do iTmp = 1, nMoved
        iLow = iClip - iStep
        iHigh = iClip + iStep
        xCurr = xClip(:,iClip)
        xLow = xCurr - xClip(:,iLow)  !sample new bead
        xHigh = xClip(:,iHigh) - xCurr
        xLow = xLow - Lx * nint(xLow / Lx)
        xHigh = xHigh - Lx * nint(xHigh / Lx)
        xLow = xCurr - xLow
        xHigh = xCurr + xHigh
        xClip(1,iClip) = 0.5d0 * (xLow(1) + xHigh(1)) + variance * rnorm()
        xClip(2,iClip) = 0.5d0 * (xLow(2) + xHigh(2)) + variance * rnorm()
        xClip(3,iClip) = 0.5d0 * (xLow(3) + xHigh(3)) + variance * rnorm()
        xClip(:,iClip) =  xClip(:,iClip) - Lx * floor(xClip(:,iClip) / Lx)
        iClip = iClip + 2 * iStep !proceed to next bead in current level
      end do

      !--- Calculate potential action difference
      nCurrLevel = 2**(nLevels - iLevel + 1) - 1 !beads in current level
      potActionOld = 0.0d0
      potActionNew = 0.0d0
      iClip = 1 + iStep
      iSlice = iClipFirst + iStep
      if(iSlice > nSlices) then
        iSlice = iSlice - nSlices
      endif
      do iTmp = 1, nCurrLevel
        x = path(:,1:nAtoms,iSlice)
        potActionOld = potActionOld + Potential(x, iRandAtom)
        x(:,iRandAtom) = xClip(:,iClip)
        potActionNew = potActionNew + Potential(x, iRandAtom)
        iClip = iClip + iStep
```

```
          iSlice = iSlice + iStep
          if(iSlice > nSlices) then
            iSlice = iSlice - nSlices
          endif
        end do
        potActionDiff = timeDiff * (potActionNew - potActionOld)

        !--- Metropolis rejection step
        actionDiff = potActionDiff - potActionDiffPrev
        potActionDiffPrev = potActionDiff

        if(actionDiff < 0.0d0) then
          flagProceed = 1 !proceed to next level

        elseif(actionDiff > 75.0d0) then
          flagProceed = 0 !reject entire move
          exit
        else
          weightRatio = exp(-actionDiff)
          if(grnd() <= weightRatio) then
            flagProceed = 1 !proceed to next level
          else
            flagProceed = 0
            exit
          endif
        endif

      end do !---- end loop over levels

      if(flagProceed == 1) then !bisection move was accepted
        nAcceptBisect = nAcceptBisect + 1
        iSlice = iClipFirst
        do iClip = 1, nClip
          path(:,iRandAtom,iSlice) = xClip(:,iClip)
          iSlice = iSlice + 1
          if(iSlice == nSlices + 1) then
            iSlice = 1
          endif
        enddo
      else !else revert to old positions
        iSlice = iClipFirst
          do iClip = 1, nClip
            path(:,iRandAtom,iSlice) = xClipSave(:,iClip)
            iSlice = iSlice + 1
            if(iSlice == nSlices + 1) then
              iSlice = 1
            endif
          enddo
      endif

  end do !---- end loop over atoms

  endif !--- end if nSlices /= 1

   !-----------------------------------------
   !Adjust to get a 50 percent acceptance ratio for COM moves
   !--------------------------------------

   !-----------------------------------------
   ! Calculate and update properties
   !--------------------------------------
```

```
  end do !---- end loop over cycles

  call AverageProperties()
  call WriteProperties()

end subroutine MetropolisBisection

!*****************************************************************************
!                                 OpenChain.f90
!*****************************************************************************
! The open chain algortihm for calculating the end-to-end distance distribution.
! Note that this basically includes the entire PIMC main code since using an
! open chain requires a specific simulaton and no other diagonal properties
! can be calculated at the same time.
!*****************************************************************************
subroutine MetropolisOpen()
  use GlobalVariables
  use MersenneTwister
  implicit none

  !------------------- Local variables
  integer :: iRandAtom, iRandSlice, iAtom, iRight, iLeft, iSlice
  integer :: moveType, iDum, nSpUpdates, nLast, iOpenAtom
  integer :: nPCorrUpdates, seed, iCycle, ind, nAcceptCOM
  integer :: nAcceptBead, nTotalCOM, nTotalBead, iHist, iCut
  integer :: flagCase, iCutNext
  double precision, dimension(3,nAtoms) :: x
  double precision, dimension(3) :: xNew, xOld, xRight, xLeft, displaceCOM, xEndDist
  double precision, dimension(3, nSlices+1) :: xSave
  double precision :: Potential, potActionOld, potActionNew, kinActionOld, kinActionNew
  double precision :: potActionDiff, kinActionDiff, actionDiff, weightRatio
  double precision :: currentEnergy, lambda, r, rMax, rSq, rMaxSq, dr, SquaredDistance

  beta = 1.0d0 / (kBoltz * temperature)
  timeStep = beta / nSlices

  !------------------- Initialise the Mersenne Twister PRNG
  call system_clock(seed)
  call init_genrand(seed)

  !------------------- Initialise a random path
  call InitialisePath()

  !------------------- Main loops
  nAcceptBead = 0
  nAcceptCOM = 0
  nTotalBead = 0
  nTotalCOM = 0
  flagCase = 0

  !open chain
  iOpenAtom = 34
  iCut = grndi(nSlices)
  iCutNext = iCut + 1
  if(iCutNext == nSlices + 1) then
    iCutNext = 1
  endif
  path(:,iOpenAtom,nSlices + 1) =  path(:,iOpenAtom,iCut)
  !rMax = boxDim(1) / 2.0d0 !for the histogram
  rMax = 0.6d0 * lengthFac
  dr = rMax / dble(nHist)
```

```
rMaxSq = rMax * rMax

do iCycle = 1, nCycles
  !------------------- Centre of mass displacement
  nTotalCOM = nTotalCOM + 1
  iRandAtom = grndi(nAtoms)
  displaceCOM(1) = deltaCOM * (2.0d0 * grnd() - 1.0d0)
  displaceCOM(2) = deltaCOM * (2.0d0 * grnd() - 1.0d0)
  displaceCOM(3) = deltaCOM * (2.0d0 * grnd() - 1.0d0)

  if(iRandAtom == iOpenAtom) then
    nLast = nSlices + 1
  else
    nLast = nSlices
  endif

  potActionDiff = 0.0d0
  do iSlice = 1, nLast
    x = path(:,:,iSlice)
    potActionOld = Potential(x, iRandAtom)
    xNew = path(:,iRandAtom,iSlice) + displaceCOM
    xNew = xNew - boxDim * floor(xNew / boxDim)
    x(:,iRandAtom) = xNew
    xSave(:,iSlice) = xNew
    potActionNew = Potential(x,iRandAtom)
    potActionDiff = potActionDiff + potActionNew - potActionOld
  end do

  !----- Metropolis rejection step
  actionDiff = timeStep * potActionDiff
  if(actionDiff < 0.0d0) then
    do iSlice = 1, nLast
      path(:,iRandAtom,iSlice) = xSave(:,iSlice)
      end do
      nAcceptCOM = nAcceptCOM + 1
  elseif(actionDiff > 75.0d0) then
    !if the exponent is too big automatically reject
  else
    weightRatio = exp(-actionDiff)
    if(grnd() <= weightRatio) then
      do iSlice = 1, nLast
        path(:,iRandAtom,iSlice) = xSave(:,iSlice)
      end do
      nAcceptCOM = nAcceptCOM + 1
    else
      !reject
    endif
  endif

  do iAtom = 1, nAtoms

    !------------------- Single bead moves
    do iDum = 1, nSlices
      nTotalBead = nTotalBead + 1
        iRandAtom = grndi(nAtoms)
      if(iRandAtom == iOpenAtom) then
        nLast = nSlices + 1
      else
        nLast = nSlices
      endif
      iRandSlice = grndi(nLast)
      x = path(:,:,iRandSlice)
```

```
xOld = x(:,iRandAtom)
potActionOld = Potential(x, iRandAtom)

!----- Generate a trial state
xNew(1) = xOld(1) + deltaBead * (2.0d0 * grnd() - 1.0d0)
xNew(2) = xOld(2) + deltaBead * (2.0d0 * grnd() - 1.0d0)
xNew(3) = xOld(3) + deltaBead * (2.0d0 * grnd() - 1.0d0)
xNew = xNew - boxDim * floor(xNew / boxDim) !translate back into box
x(:,iRandAtom) = xNew
potActionNew = Potential(x, iRandAtom)
potActionDiff = timeStep*(potActionNew - potActionOld)

if(iRandAtom /= iOpenAtom) then

  !----- closed chain (ring polymer condition)
  if(iRandSlice == nSlices) then
    iLeft = nSlices - 1
    iRight = 1
  elseif(iRandSlice == 1) then
    iLeft = nSlices
    iRight = 2
  else
    iRight = iRandSlice + 1
    iLeft = iRandSlice - 1
  endif
  xRight = path(:, iRandAtom, iRight)
  xLeft = path(:, iRandAtom, iLeft)

  !----- Compute the kinetic (spring) energy difference
  kinActionOld = SquaredDistance(xOld,xLeft) + SquaredDistance(xRight,xOld)
  kinActionNew = SquaredDistance(xNew,xLeft) + SquaredDistance(xRight,xNew)

  kinActionDiff = kinActionNew - kinActionOld
  lambda = 0.5d0 / mass(iRandAtom)
  kinActionDiff = kinActionDiff / (4 * lambda * timeStep)
  !actionDiff = potActionDiff + kinActionDiff
else

  !----- Open chain
  if(iRandSlice == iCut) then !no bead to the right
    iLeft = iCut - 1
    if(iLeft == 0) then
      iLeft = nSlices
    endif
    xLeft = path(:,iOpenAtom,iLeft)
    kinActionOld = SquaredDistance(xOld,xLeft)
    kinActionNew = SquaredDistance(xNew,xLeft)

  elseif(iRandSlice == nSlices + 1) then !no bead to the left
    iRight = iCutNext
    xRight = path(:,iOpenAtom,iRight)
    kinActionOld = SquaredDistance(xOld,xRight)
    kinActionNew = SquaredDistance(xNew,xRight)

  elseif(iRandSlice == iCutNext) then !nSlices + 1 to the left, iCut + 2 to the right
    iLeft = nSlices + 1
    iRight = iCutNext + 1
    xLeft = path(:,iOpenAtom,iLeft)
    xRight = path(:,iOpenAtom,iRight)
    kinActionOld = SquaredDistance(xOld,xLeft) + SquaredDistance(xRight,xOld)
    kinActionNew = SquaredDistance(xNew,xLeft) + SquaredDistance(xRight,xNew)
```

```
          else  !beads in both directions
            iRight = iRandSlice + 1
            iLeft = iRandSlice - 1
            if(iRight == nSlices + 1) then
              iRight = 1
            endif
            if(iLeft == 0) then
              iLeft = nSlices
            endif
            xLeft = path(:,iOpenAtom,iLeft)
            xRight = path(:,iOpenAtom,iRight)
            kinActionOld = SquaredDistance(xOld,xLeft) + SquaredDistance(xRight,xOld)
            kinActionNew = SquaredDistance(xNew,xLeft) + SquaredDistance(xRight,xNew)
          endif
          lambda = 0.5d0 / mass(iRandAtom)
          kinActionDiff = kinActionNew - kinActionOld
          kinActionDiff = kinActionDiff / (4 * lambda * timeStep)
        endif !----- End open / closed chain

        actionDiff = potActionDiff + kinActionDiff
        !----- Metropolis rejection step (regardless of open/closed chain)
        if(actionDiff < 0.0d0) then
          !always accept downhill moves
          path(:,iRandAtom, iRandSlice) = xNew
          nAcceptBead = nAcceptBead + 1
        elseif(actionDiff > 75.0d0) then
          !if the exponent is too big automatically reject
        else
          weightRatio = exp(-actionDiff)
          if(grnd() <= weightRatio) then
            path(:,iRandAtom, iRandSlice) = xNew
            nAcceptBead = nAcceptBead + 1
          else
            !reject
          endif
        endif
      end do !---- end do idum
    end do !---- end loop over atoms

    if(iCycle > nBurnIn) then
      !end to end distance histogram
      xEndDist = path(:,iOpenAtom,iCut) - path(:,iOpenAtom,nSlices + 1)
      xEndDist = xEndDist - boxDim * nint(xEndDist / boxDim)
      rSq = xEndDist(1)**2 +  xEndDist(2)**2 +  xEndDist(3)**2
      if(rSq <= rMaxSq) then
        r = sqrt(rSq)
        iHist = int(r / dr) + 1
        if(iHist < nHist) then
          endHist(iHist) = endHist(iHist) + 1
        endif
      endif
    endif

  end do !---- end loop over cycles

end subroutine MetropolisOpen
```