Kinetic Monte Carlo Simulations of Perovskite Crystal Growth with Long Range Coulomb Interactions

A thesis submitted in partial fulfillment of the requirement
for the degree of Bachelor of Science with Honors in
Physics from the College of William and Mary in Virginia,

by

Thomas J. Walls

Accepted for _____
(Honors, High Honors or Highest Honors)

_____
Advisor: Dr. Shiwei Zhang

_____

_____

_____

Williamsburg, Virginia
May 1999

Abstract

Computer simulations of crystal growth can provide valuable insights into the physical processes involved in actual crystal growth. These insights can then be used in theory and experiment to improve both our understanding of and our ability to create high quality crystals. Kinetic Monte Carlo methods have been developed which facilitate such computer simulations, but the models cannot be applied in all cases because of the assumptions made about the nature of the crystals. The standard computational model cannot be applied to the growth of perovskite crystals because long-range Coulomb interactions, which have been shown to greatly affect the structure of the crystal, are not included. In this thesis we propose a modification to the standard model which explicitly includes the long-range electrostatic interactions between the ions and develop the necessary mathematical and computational techniques required to implement our model. We generate some promising preliminary results from perovskite growth simulations and analyze the implications of these results.

Acknowledgments

I would sincerely like to thank my advisor, Dr. Shiwei Zhang, for his guidance on this project. His patience and insight were key elements in the completion of the work presented. I would also like to thank professor Henry Krakauer for his helpful insights and generously allowing the use of his office space and computers. Finally, I would like to thank the Physics and Applied Science departments of the College of William and Mary for providing the training and resources necessary to complete this work and all the members of my committee for their time and effort in evaluating this thesis.

Contents

Chapter 1

Introduction and Background

1.1 Introduction to Growth Simulations

Crystal structure has received much attention in the history of materials science. Crystals are used in nearly every branch of applied research, from high quality semi-conductor fabrication to military radar applications. While the art of manufacturing high-quality crystals is now well developed, our scientific understanding of their growth mechanisms is still lacking. Currently, theory, experiment and computer simulations are collaborating to better understand the underlying mechanisms behind crystal growth.

Computer simulations are often used to verify theoretical approaches or to help guide experimental procedures[1]. In turn, new experimental results or theoretical advances are then used to improve simulation models. Computer simulations can provide insights about the nature of crystal growth which are unobtainable through theory or experiment. The observations serve to advance our understanding of the growth processes involved. Most simulations we are aware of, however, are restricted in their usefulness because they assume overly simplified behavior of the molecules in the crystal. Although these models are restricted to simplified conditions, they have been able to demonstrate many properties

exhibited by real life crystals.

One way to do such simulations is the Kinetic Monte Carlo (KMC) method. KMC models are used to simulate the relaxation processes of systems away from equilibrium. KMC methods have proven very successful at simulating dynamic properties of non-equilibrium systems[2, 3]. The restriction of the KMC method is that only nearest neighbor molecular bonding inside the crystal is considered. In this paper we will implement the KMC method and demonstrate the agreement of these computational simulations with analytical theories.

We will then present a model of growth which is not restricted to local interactions, but which allows for long-range electrostatic interactions. This is motivated by the recent development of relaxor ferroelectric single crystals which exhibited tremendous technological potential[4]. Subsequent theoretical modeling of the structure of these crystal showed that the ion-ion long-range Coulomb potential is the dominating term in determining their structure. Since our goal is to model the growth processes of these single crystals to help guide experimentalists, these long-range interactions must be considered to obtain valid simulation results.

We will develop the necessary computational techniques to implement this kinetic growth model and use it to demonstrate properties of actual experimental crystals which were previously unobtainable. With further study, this model could prove to be a valuable tool in the understanding of more complex crystal structures.

1.2 Background on Perovskite Alloys

Electromechanical actuators are devices which convert electrical energy into mechanical energy and vice-versa. Actuators are used for a wide range of purposes varying from keeping time in a wrist watch to interpreting signals for cellular phone calls. Generally, these actuators use perovskite alloys because of their extraordinary dielectric and piezoelectric properties. Perovskite alloys are common compounds usually containing a mixture of alkaline earth metals and transition metals. These compounds were first studied because of their super-conducting properties.

Recently, a "new" class of relaxor ferroelectric crystals have been developed at Pennsylvania State University for actuator applications [4]. The class of single crystal ferroelectrics includes $Pb(Zn_{1/3}Nb_{2/3})O_3$ - $PbTiO_3$ (PZN-PT) and $Pb(Mg_{2/3}Nb_{2/3})O_3$ - $PbTiO_3$ (PMN-PT). These relaxor ferroelectric crystals have come to the forefront of research due to their unique piezoelectric properties.

When evaluating crystals for electromechanical actuator applications, the most important property is their strain versus electric field ratio. These new crystals have demonstrated strain versus electric field ratios *one order of magnitude higher* than those obtained from previous ceramic crystals. The piezoelectric properties of these crystals are critically dependent on the composition and orientation of the crystal[4]. At the present time, the mechanisms which regulate the crystallographic orientation observed in experimental crystals are not well understood. The underlying atomic structure and its effect on the piezoelectric properties of the new crystals is also not well understood.

Motivated to better understand the growth processes of perovskite crystals, we will

employ KMC techniques using long-range interactions. With an improved understanding of the growth processes of the crystals, we can assist experimentalists in determining what conditions provide crystals of the highest quality, the largest size, and the best growth rate. To our knowledge, this is the first model to explicitly include the long-range interactions in kinetic processes.

Eventually, this model could be used in conjunction with first principles calculations [6] and experimental data to further our understanding of the nature of these crystals. Simulations of this kind have proved valuable because they are often easier to manipulate, easier to analyze, and are often faster to perform. Once a model for the growth of the relaxor crystals has been fully developed, it can be used to compliment experimental data and guide experimental decisions about parameter modifications. This information will then provide important information on ways to maximize crystal quality and size for use in actuator applications.

The crystals are produced by the flux technique[4]. The crystal is grown by placing highly purified powders of the various desired components into a crucible, heating the powders into a melt phase and then allowing the melt to slowly cool into crystalline form. This method of growth has already been shown to be reproduced very well with standard KMC techniques. But the ionic nature of the new relaxor crystals plays a key role in their growth. The ionic interactions are not negligible, even at very long distances, thus the standard kinetic model does not provide valid results for perovskite crystal growth[6]. Understanding this infinite-range Coulomb interaction is a key element in motivating a new model.

1.3 Background on Monte Carlo Methods

Computational physics has come to the forefront of scientific research. The Monte Carlo approach has emerged as a key method in this rapidly evolving field. Many books have emerged which give a comprehensive introduction to this technique[2, 3]. Monte Carlo is a general method by which a **probability** (terms in bold-face are defined in Appendix A) is determined by repeated measurements of random points within a **state space**. This probability is then used to define an interval estimate for an actual value we wish to measure. In the limit that every point in the state space is measured, the value can be determined exactly. The value can be either **deterministic** or **stochastic** in nature. This freedom embodies a very profound idea; that real life systems, which are completely deterministic in nature can be simulated and reproduced using stochastic models.

To illustrate the basic Monte Carlo method, we show a simple example of the calculation of the fundamental constant π. Consider a unit circle inscribed inside a square (see fig. 1.1).

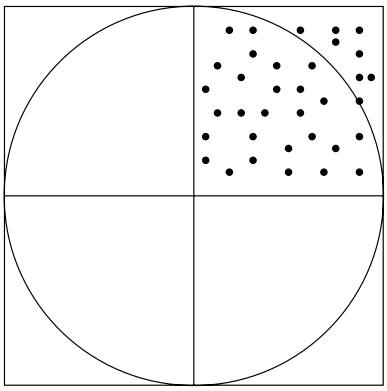

Figure 1.1: Random points generated in a square for the calculation of π

If we define **event** $\mathcal{A}$ as a randomly generated point in the upper right quadrant of the

square that falls inside the circle, then we can write:

$$Pr(\mathcal{A}) = \frac{\text{area of circle}}{\text{area of square}} = \frac{\pi \frac{x^2}{4}}{x^2} = \frac{\pi}{4}$$

Consequently, we can write $\pi = 4 * Pr(\mathcal{A})$. In our sample experiment we have generated thirty three points in the square (see fig 1.1). Twenty five of these points fall inside the circle, so we would calculate $\pi = 4 * \frac{25}{33} = 3.03$. Obviously, the more points we generate in the quadrant, the better our estimate for π is going to be. More exactly, from the Central Limit Theorem, the error in our estimate decreases as the square root of the number of **samples**. In the limit that the number of points approaches infinity, our answer becomes exact. This is known as *the law of large numbers*, and is a fundamental concept behind any Monte Carlo simulation.

As computational physics has become algorithmically more sophisticated, so have Monte Carlo techniques. Today, Monte Carlo methods are used to calculate everything from N dimensional integrals to determining the fractal order of various systems. The power of Monte Carlo techniques lies in the general principle that the computed interval estimate always converges to the actual answer as $\frac{1}{\sqrt{N}}$, regardless of the dimensionality of the system. Monte Carlo calculations have also been shown to accurately simulate kinetic non-equilibrium relaxation phenomenon. Bortz, Kalos and Lebowitz [7] have developed an effective algorithm for dealing with kinetic processes. The simulation of kinetic processes with KMC will be the backbone to our growth simulation.

1.4 Introduction to the Ising Model

The first computational model to successfully reproduce a system exhibiting a first order phase transition was the Ising model[5]. It is a model of the ferromagnetic properties of iron. It assumes that iron can be simulated by a **discrete** lattice of spin magnetic moments which can either be aligned up or down. In order to avoid boundary effects, periodic boundary conditions are imposed. This means that as you leave one side of the lattice, you reappear on the opposite side. The result of this condition (assuming your basic lattice is not TOO small) is that with finite size scaling and/or other analysis and extrapolation schemes, we can approach lattices of infinite length and width. We then define a set of rules by which the spins will 'flip' with a given probability from the state of neighboring spins and the temperature of the system based on the Metropolis algorithm[1]. The system is deemed in a ferromagnetic state based on the correlation of the spin alignments. After scanning the system for spin flips many times, we determine if the system is in a ferromagnetic state. It is interesting to note that even though ferromagnetism is a purely quantum mechanical result, the model still captures the essence of the physics involved without the required quantum mechanical calculations. A generalized version of the Ising model is the starting point for building KMC simulation models.

1.4.1 The Generation of Random Numbers

The accurate generation of uncorrelated random numbers is a key component for valid Monte Carlo results. As an extreme example, in our calculation of π, if the random points were skewed in such a way that they all fell inside the circle, we would calculate a value of

$\pi = 4$. Thus we are faced with the issue of how to accurately generate random numbers. There are many random number generators in use, some of which are statistically flawed. Our discussion here will be limited to a basic overview of "good" random number generation. For a comprehensive review of modern random number generators and the principles behind the generation of good random numbers see Park and Miller [8]. An ideal random number generator would produce a stream of numbers between 0.0 and 1.0 which are equally likely to occur and are reproducible.

To achieve this goal, we will use a pseudo random number generation program based on a variation of the Lehmer Generator, written and extensively tested by Park [9]. The basic algorithm for the number stream is:

$$x_{i+1} = a * x_i \bmod m$$

Where a and m are constants and x_i is the i_{th} value in the number stream. A very natural choice for the m value is $2^{31} - 1$ because it is the largest value that can fit into a single register on a 32 bit computer. We must be very careful about our choice of a to ensure that we generate all values between 1 and $2^{31} - 1$, otherwise our number stream will start repeating with a frequency that is lower than maximal. There are several appropriate values that can be chosen for a, of which we will choose $a = 48271$[9]. We can now forget the topic of random numbers altogether and focus on the details of the model at hand.

1.4.2 Equipment

All calculations done in this paper were performed on an Intel 300 MHz Pentium II processor. The programs were run in the Red Hat Linux environment and compiled using the SGI

compiler with maximum level of optimization. All computations performed were written either in C or C-C++ mixtures of code using standard math and Unix libraries. Appendix B and C contain the code used to generate the results presented in this thesis.

1.4.3 Goals of the Thesis

The remainder of this thesis will be dedicated to the development of a novel computational model for perovskite growth simulations. We begin by implementing the standard model for Kinetic Monte Carlo. We will review the mathematical specifications of the standard model and we will implement the model to generate data from a KMC simulation. This basic model will provide a good starting point for advancement. We will then motivate what modifications need to be made to the basic model to simulate perovskite crystals. We will discuss the current understanding of these crystals and show how this understanding can be used as a guide for our modifications. Specifically, the exact nature of the Coulomb forces within the crystal and how they need to be treated. We will then present some preliminary results obtained from our modified model and compare these results to those obtained from the standard model. We will also discuss the implication of these results to our model for perovskite crystal growth.

Chapter 2

The Standard Kinetic Monte Carlo Model

In this chapter we will review the concepts and implementation issues involved in standard kinetic growth simulations. We will discuss the basic assumptions made, present the equations of growth and discuss the comparison between approximate analytical solutions and computational results.

2.1 Development of the Basic Model

This section will be devoted to motivating the standard KMC model we will use for growth simulations. We will first define the model on a conceptual level, and examine what parameters need to be specified in order to properly evaluate growth kinetics.

2.1.1 The Conceptual Model

We want to start with the most basic conception of crystal growth. We will start by imagining a crystal surface at the front of a liquid-solid interface. The solid surface itself is an infinite plane of molecules locally bonded to each other. The surface is initially flat, a boundary condition we will impose. The liquid interface is a super-cooled region of the melt

phase of the solid. It has been suggested that during growth from a liquid, the molecules form a pre-crystalline order in the liquid at the interface, before impinging on the crystal surface [10]. We will assume these pre-crystalline molecules impinge with equal probability across the surface at random intervals (see fig 2.1).

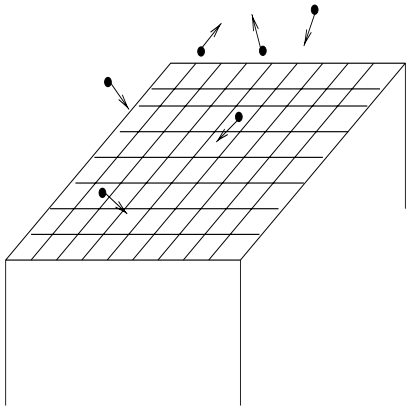

Figure 2.1: Drawing of pre-crystalline molecules near the crystal surface.

For simplicity we will assume that the surface at the interface is a crystalline arrangement of molecules which can only exist on discrete sites. In order to remove the effects of the edge of the lattice, we will impose periodic boundary conditions. For now, we assume only two possible events at the surface. The first is a molecule leaving the liquid interface and attaching itself to the crystal surface (adsorption). The other possibility is that internal energies within the crystal (ie thermal fluctuations or electrostatic potentials) provide a surface molecule enough energy to remove itself and enter the liquid phase (evaporation). The discrete sites can now be seen as "stacks" of crystal molecules which will either grow or shrink by one every time an event occurs. It is easy to see that this model is simply a two dimensional version of the Ising spin model discussed in section 1.4. By evolving the system with time, we can examine the kinetic growth properties exhibited by the advancement of

the surface of the crystal.

2.1.2 Specifications of the Model

We now need to define the exact rules by which events occur. The idea is to use a Monte Carlo approach to measure the probability that the system is in a given configuration and relate that probability to the growth kinetics. The basic model we will advance here was first introduced by G. Gilmer and P. Bennema [11, 12]. The notation and derivation in this section will follow closely the treatment by Saito [13].

We will assume a square lattice of L discrete lattice stacks per row or column. We will denote the number of molecules in stack i as $h(i)$ and the configuration of all stacks as $\{h\} \equiv h(1), \cdots, h(N)$, where $N = L^2$ is the number of lattice stacks. $h'(i)$ will denote a stack that differs from $h(i)$ by one unit and $\{h\}_i$ will denote a configuration the differs from $\{h\}$ by 1 unit at stack i. We are interested in studying the probability $P()$ that the system is in a certain configuration. We can write the probability that the system is in any configuration at time $t + \Delta t$ as:

$$P(\{h\}, t + \Delta t) =$$
$$P(\{h\}, t) - \sum_{i=1}^{N} w(h(i) \rightarrow h'(i)) P(\{h\}, t)\Delta t + \sum_{i=1}^{N} w(h'(i) \rightarrow h(i)) P(\{h\}_i, t)\Delta t \quad (2.1)$$

Where $w(h(i) \rightarrow h'(i))$ is the rate of transition from $h(i)$ to $h'(i)$, and $w(h'(i) \rightarrow h(i))$ is the rate of transition from $h'(i)$ to $h(i)$. If we look at the continuous time limit as $\Delta t \rightarrow 0$ we obtain the master equation of the probability:

$$\frac{\partial P(\{h\}, t)}{\partial t} = -\sum_{i=1}^{N} w(h(i) \rightarrow h'(i)) P(\{h\}, t) + \sum_{i=1}^{N} w(h'(i) \rightarrow h(i)) P(\{h\}_i, t) \quad (2.2)$$

In order to verify that the system reaches thermodynamic equilibrium (as $t \to \infty$), the Boltzmann distribution should be a solution to the master equation:

$$P_{\mathsf{eq}} = Z^{-1}\mathsf{exp}\left[-\frac{\mathcal{H}(\{h\})}{k_B T}\right]$$

Where $\mathcal{H}(\{h\})$ is the Hamiltonian of the system defined as:

$$\mathcal{H}(\{h\}) = \phi \sum_{<ij>} |h(i) - h(j)| + \Delta\mu \sum_i h(i) = E(\{h\}) + \Delta\mu \sum_i h(i) \tag{2.3}$$

Where ϕ is the strength of the bond between neighboring sites, and $\Delta\mu = \mu_s - \mu_l$ is the potential energy gained by one molecule crossing the interface. The summation over $< ij >$ is the summation over all nearest neighbor pairs.

When the system is in equilibrium, the rate of growth is zero. This implies that in equilibrium the adsorption rate $w(h(i) \to h(i)+1)$ must equal the evaporation rate $w(h(i) \to h(i) - 1)$. The transition rates must then satisfy the detailed balance condition:

$$\frac{w(h(i) \to h'(i))}{w(h'(i) \to h(i))} = \frac{P_{\mathsf{eq}}(\{h\}_i)}{P_{\mathsf{eq}}(\{h\})} = \mathsf{exp}\left[-\frac{\mathcal{H}(\{h\}_i) - \mathcal{H}(\{h\})}{k_B T}\right] \tag{2.4}$$

We will choose the adsorption rate to be constant and define:

$$w(h(i) \to h(i) + 1) = e^{\frac{\Delta\mu}{k_B T}} \tag{2.5}$$

Given this value for the adsorption rate, we can determine the evaporation rate from the detailed balance condition by:

$$\begin{aligned} w(h(i) \to h(i) - 1) &= \frac{P_{\mathsf{eq}}(h(i) - 1)}{P_{\mathsf{eq}}(h(i))} w(h(i) - 1 \to h(i)) \qquad (2.6) \\ &= \exp\left[\frac{[E(\{h\}_{i^-}) + \Delta\mu(\sum_i h(i) - 1)] - [E\{h\}_i + \Delta\mu \sum_i h(i)]}{k_B T}\right] e^{\frac{\Delta\mu}{k_B T}} \\ &= \exp\left[\frac{E(h(i) - 1) - E(h(i))}{k_B T}\right] \\ w(h(i) \to h(i) - 1) &= e^{-\frac{\Delta E}{k_B T}} \qquad (2.7) \end{aligned}$$

Here ΔE is the energy cost to lower site i and is written as:

$$\Delta E = -2\phi(n - \frac{z}{2}) \tag{2.8}$$

where n is the number of neighbors to site i and z is the maximum number of neighbors i can have.

Now that we have defined rates for both events of interest, we are ready to examine the time evolution of the system.

2.2 Analytical Models

While an analytical solution to the master equation 2.2 is ideal, one does not exist at the present time. Various approximate solutions have been proposed[15]. We will examine one such approximate approach in section 2.2.2.

2.2.1 Ideal Phase Transition

In this section we will use thermodynamics results derived by L.D. Landau[14]. According to the second law of thermodynamics, for a given p and T, the equilibrium state of a system is determined such that the Gibbs free energy of the system is minimized. If we examine a plot of the isobaric free energy versus temperature, it is easy to see how this state is selected.

It is readily apparent from figure 2.2 that as you lower the temperature of the system below T_M, it is energetically favorable for the system to be in the solid phase.

Using this basic relationship, and the relationships between the Gibbs free energy and

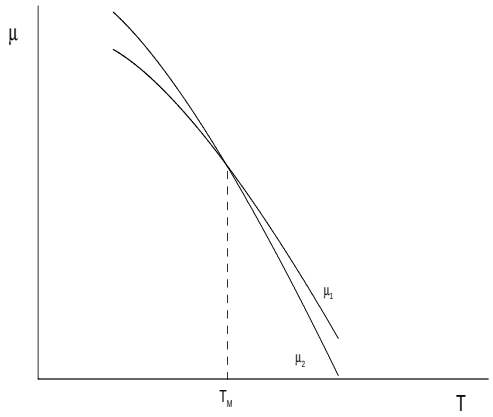

Figure 2.2: Isobaric variations of the chemical potential as a function of temperature for the solid and liquid phases.

the entropy, Saito[13] derives the following expression for ideal growth from the melt. Saito defines the growth rate R in terms of the atomic height a and the lattice vibrations ν. The lattice vibration is the relative motion of the crystal molecule around an average lattice position. The growth rate of a crystal from the melt phase can be written as:

$$R = a\nu e^{-\frac{E_d}{k_B T}} \left[1 - e^{-\frac{\Delta\mu}{k_B T}}\right] \tag{2.9}$$

Where E_d the energetic barrier opposing evaporation and $\Delta\mu$ represents the difference in the chemical potentials of the solid and liquid states. While equation 2.9 is a theoretical expression for growth rate, its usefulness is limited because it only speaks to the average growth rate of the crystal.

2.2.2 Two Rate Model

An approximate solution to equation 2.2, proposed by Weeks, Gilmer and Jackson[15], has proven to model well the growth rate in regions of reasonable deposition rates. Again, the reader should refer to the derivation of the theory from these authors. They derived an

expression for the growth rate to be:

$$\frac{\overline{R}}{k_+} = \frac{2\sinh(\frac{\Delta\mu}{k_B T})}{e^{\frac{\Delta\mu}{k_B T}} + \cosh(\alpha\frac{\phi}{k_B T})} \tag{2.10}$$

Where α is an empirically determined value of 1.33 and k_+ is the probability of adsorption of a molecule. The authors choose α to best match equation 2.10 other simulation data, and we divide by the k_+ term for normalization purposes. This model is advantageous because it is analytic in nature, however, equation 2.10 is limited to simple crystals within well behaved domains of high temperatures and high deposition rates. We will use equation 2.10 as a computational benchmark for our standard model.

Chapter 3

Simulation of the Standard Kinetic Monte Carlo Model

In this chapter we will examine the exact procedures by which Kinetic Monte Carlo simulations are performed. We will open with the basic implementation issues of the standard KMC model, then examine results obtained by running simulations of the standard KMC model. We will close the chapter with a discussion of yet unexplored topics in characterizing the standard KMC model.

3.1 Improving the Algorithms

The question now is, how do we actually use the transition rates, described in section 2.1.2, to simulate a dynamic system. As discussed in Chapter 1, we are taking a Monte Carlo approach. We are going to convert the derived transition rates into relative probabilities and see how the system evolves with increased sampling of the state space. We will define our lattice as an L x L matrix of integers. The integers will represent the current height of the stack. We now want to create a general algorithm for increasing and decreasing

the height of the stacks. We have to consider two events in our simulation, adsorption and evaporation. With many realizations of these two events, we will be able to study the advancement of the crystal surface.

We want to convert the transition rates derived in section 2.1.2 into true probabilities. To help with notation, we will use S_{E_i} to represent the transition rate of event E_i at a stack in the lattice (eg E_1 = adsorption and E_2 = evaporation). S_{MAX} will then represent a largest possible rate of any event to be used for normalization. An obvious algorithm to run the simulation is described by Levi and Kortla[1]:

- `Choose a possible event` (E_i) `which can occur in configuration` $\{h\}$
- `Define` S_{MAX} `to be the largest possible rate for selected event` E_i
- `Calculate the probability of event` E_i`,` $P_{E_i} = \frac{S_{E_i}}{S_{\mathsf{MAX}}}$
- `Generate a random number` $r \, \epsilon \, [0, 1)$
- `If` $r \leq P_{E_i}$ `then generate new configuration` $\{h\}_i$ `from event` E_i

Although intuitive, this algorithm is also extremely slow. The reason is that the rate difference between selected events can often be orders of magnitude, so you can get many failed attempts at a configuration change before you have a success.

A much more efficient algorithm, was proposed by Bortz, Kalos, and Lebowitz[7]. It utilizes a method of importance sampling, whereby an **event list** containing all possible events is created, and at each time step, events are selected based on their relative importance. This approach greatly improves computational time by eliminating nearly all possible time when nothing happens.

Now, we will define S_{MAX} as the sum of the largest rate for each possible event. In our model $S_{\mathsf{MAX}} = S_+ + S_- = e^{\frac{\Delta\mu}{k_B T}} + e^{\frac{z\phi}{k_B T}}$ because the adsorption rate is constant, and the maximum value $w(h(i) \to h(i) - 1)$ can have is $e^{\frac{z\phi}{k_B}}$. This will allow us to generate a list of events which can be selected can be selected with one realization of a random number. We can then define the probability of an event P_{E_i} in relation to this absolute maximum S_{MAX}. The algorithm can then proceed as:

- `Randomly select a stack` (i) `on the lattice square`
- `Generate list of possible events at stack` i
- `Define` $P_E = \frac{S_E}{S_{\mathsf{MAX}}}$
- `Generate a random number` $r\epsilon[0,1)$ `and choose the first event` E_s `such that` $\sum_i^s P_{E_i} \geq r$
- `Generate new configuration` $\{h\}_i$ `from event` E_s

The selection processes can be enumerated as follows. If random number r is less than the probability of the first event in the list (order is not important), then that event is selected. If not, then if r is less than the first event plus the second event, then the second event is selected. In this way, events from the event list are selected. After attempting N events, each site has been visited, on average, one time. We define this as one Monte Carlo Step. Monte Carlo Steps will represent the units of relative time in our model. For calculational simplicity, we will define $k_B T = 1$. Since we are not studying the system as a function of temperature (which can done [16, 15, 13]), the temperature term serves only as a constant scaling factor which can ignored in the current discussion.

3.2 Results of the Standard KMC Model

Lattice sizes $L = 10$, $L = 20$ and $L = 40$ were studied, all of which produced data with error ranges within statistical acceptability. To obtain good statistics, the system was allowed to grow for 1000 Monte Carlo Steps per simulation. Each simulation was then run 5 times and an average value was calculated. A typical simulation would require approximately 5-10 hours to complete on a dedicated processor, depending on lattice size chosen. To verify the validity of the basic model, we will compare the calculated Monte Carlo growth rates from the $L = 20$ lattice to that defined in section 2.2.2. We will define the growth rate R to be the average number of layers grown divided by the number of Monte Carlo steps. Since the maximum possible rate of growth depends on the adsorption probability, we will plot the rate of growth divided by the adsorption probability to normalize the range. The results for a wide range $\Delta\mu$ and ϕ are presented in figure 3.1.

According to the assumptions made to derive equation 2.10, we expect the greatest agreement in regions of high temperature and high deposition rates[15]. In our model, the temperature serves as a uniform scaling factor of the Hamiltonian. Thus, regions of high temperatures correspond to low chemical potentials. Therefore we expect the best agreement in the upper left portion of a graph of R versus chemical potential. Figure 3.1 verifies this expectation.

Using the standard KMC computational model we have been able to demonstrate the proper growth rate kinetics. Our basic model also produces surface configurations which obey the expected temperature behavior (see section 3.2.1). In light of these success we can now make the necessary modifications to investigate the desired type of growth.

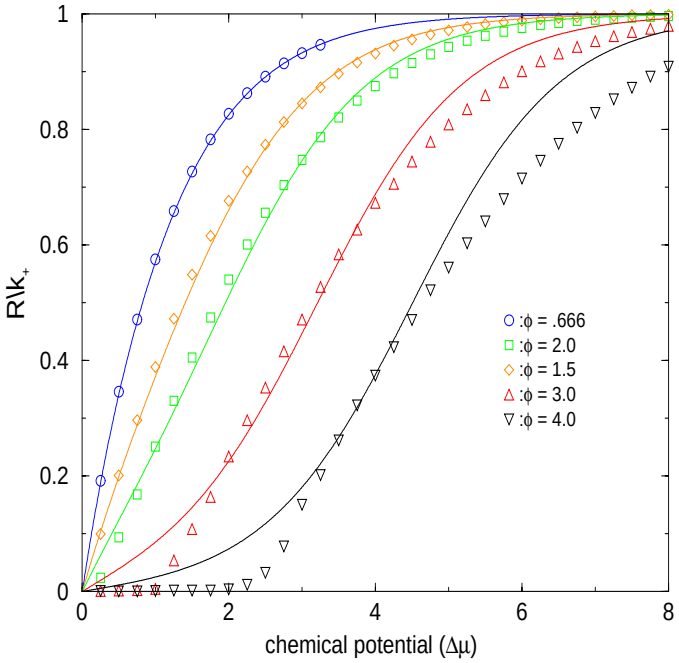

Figure 3.1: Growth rate versus chemical potential ($\Delta\mu$) for various bond strengths (ϕ) at $k_BT = 1$. The points are KMC results and the solid lines are theoretical results.

3.2.1 Surface Configurations of the Standard KMC Model

One major success of the standard KMC model was the ability to demonstrate the transition of the surface structure between two types of growth. At low temperatures, the surface grows largely by full layers. At any point in time, the surface looks like a converging collection of large islands. This is referred to as nucleated growth. At high temperatures however, molecules are much more likely to evaporate, and the surface grows with many different layers simultaneously. This is referred to as layered growth[1].

A typical surface configuration from a kinetic growth simulation at low temperatures (or high chemical potential and bond strength) is shown in figure 3.2, while figure 3.3 demonstrates a typical surface at high temperatures. The transition between these two

types of growth is a first order phase transition. One of the primary uses of the standard KMC model was the calculation of the exact point where the roughening transition occurs. At the transition point, the fundamental mechanism of growth at the surface changes, a fact which should be reflected by our model. Our basic model clearly demonstrates this roughening behavior, as demonstrated by figures 3.2 and 3.3.

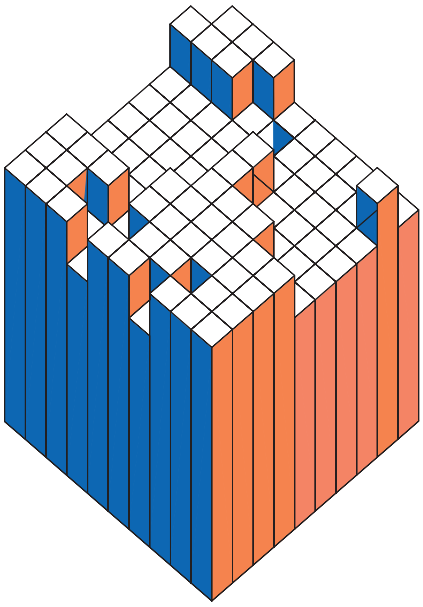

Figure 3.2: Typical surface configuration at low temperatures for a 10 x 10 surface.

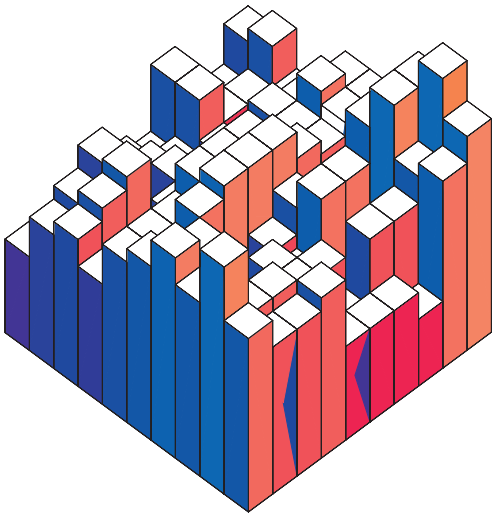

Figure 3.3: Typical surface configuration at high temperatures for a 10 x 10 surface.

3.3 Unexplored Modifications

In this section we will discuss components of the model which will not be fully explored in this paper, but which could have interesting effects on the simulated growth. We will discuss the assumptions made for the purposes of our modeling, explain the reasoning behind these

assumptions and explore possible methods by which the assumptions could be changed.

3.3.1 The Solid-on-Solid Restriction

Until now, we have implicitly assumed one very subtle but important point. We have assumed that each crystalline molecule will always exist above another crystalline molecule. This means that a particle adsorbing at site k in figure 3.4 is not allowed.

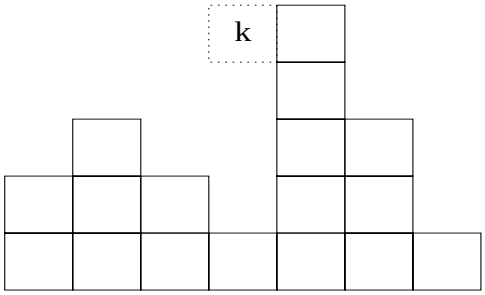

Figure 3.4: Site 'k' not allowed for molecule adsorption.

This has come to be referred to as the Kossel model or the Solid-on-Solid (SOS) restriction [17]. This restriction implies that there can be no vacancies or defects within the bulk of the crystal lattice. While this assumption is valid in regions of slow growth (ie small $\Delta\mu$) it may not be realistic for other growth regions.

Because non-SOS models have received very little attention in literature and thus are not very well characterized, we have chosen to maintain the SOS assumption. For the purposes of perovskite alloys, the actual method of growth is such that there is little chance for bulk defects within the crystal[4]. It has been suggested, however, that whatever defects do appear, could play a vital role in determining quality factors of the crystal[18]. The study of such defect phenomena could therefore provide very useful insights for increased

crystal quality.

In order to implement a model which includes bulk defects, changes need to be made to the basic algorithm. We propose two possible changes in implementation of the standard KMC model which may prove useful in the study of bulk vacancies for this type of growth.

As a first approach, we can view the first step in the basic algorithm (random selection of a stack) as a random selection from two sets. The first is the set of sites available for adsorption and the second is the set of sites available for evaporation. In the Kossel model, these two sets are the same; the set of all surface atoms. One way to include vacancies, therefore, is to maintain separately a list of all sites available for adsorption (include all 'k' sites). The downside of this approach is that the list of sites available for adsorption becomes very difficult to maintain correctly, and if not implemented cleverly carries a large increase in computational time.

A second approach would utilize a local search method. Select a stack at random as in the original algorithm, then scan the sites above the surface molecule to see if they are available for adsorption and select from this list of sites as appropriate. This approach carries with it a uniform increase in computational time, and could be very effective in implementing this model.

3.3.2 Surface Diffusion

We have also assumed that once a crystalline molecule adsorbs, it stays where it is. In real systems molecules tend to travel along the surface of the crystal until becoming bonded to two or more other molecules. This process is referred to as surface diffusion and as in previous studies, we have ignored it. Although surface diffusion is critical to growth

processes, it has been shown that the inclusion of diffusion with the SOS restriction serves only to scale the results of the standard KMC model in a constant manner. An interesting future topic is to study the effect of surface diffusion in our long-range model.

Chapter 4

Long-Range Interactions

While the standard model has achieved great success in some areas, it is still overly simplified. Depending on the application, various improvements to the standard model are necessary. In our case, as we will demonstrate below, we will need to account for the ionic long-range interactions. In this chapter we will examine recent studies which have outlined the importance of long-range interactions in perovskite crystals[19]. We will develop the required modifications to the standard model to include these interactions in our Kinetic Monte Carlo simulations.

4.1 Perovskite Alloys Revisited

As mentioned in section 1.2, perovskite alloys are well known oxide compounds usually containing a mixture of alkaline and transition metals. The composition of these alloys can be described most generally by the chemical formula ABO_3, where A is a usually a fixed Group II metal, and B has fractional compositional freedom containing metals from Groups II - VII. The crystal's symmetry is cubic, with element A occupying the body of the unit cell (see fig 4.1).

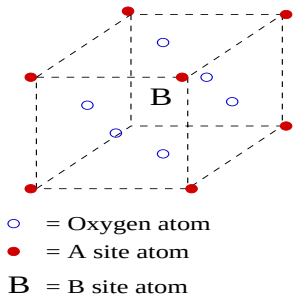

Figure 4.1: Unit cell of typical perovskite alloy.

One phenomenon not fully understood at present is the ordering within the crystals of the B sites. This ordering has proven to have a critical effect on the piezoelectric properties of the crystal[23]. B atom ordering has been observed to be very sensitive to composition and temperature and can dominate the nature of the crystal. Because the basic properties of the crystal are inherently dependent on this ordering, this feature must be a direct result of any model developed. Advanced theoretical models have emerged which explain the ordering of the B site with limited success[24]. These theories generally employ a method of cluster expansion about short range electrostatic forces.

Recent computational work, however, has taken a much different approach. Bellaiche and Vanderbilt have proposed a model which explicitly includes the long-range electrostatic interactions between ions[19]. The idea is that by directly calculating the infinite-range Coulomb interactions within the crystal, we can reproduce the ordering observed within different systems. While even the authors expected only limited success, the model has provided accurate predictions of the ordering experimentally measured in various different perovskite systems[19]. With this infinite-range sum in mind, will need to change a fundamental assumption of our basic KMC model.

4.2 The New Conceptual Model

We have to revise our fundamental conception of the crystal structure. Instead of distinct molecules covalently bonding to their neighbors, we will now view the crystal as a structure of ionic spheres arranged in a valence electron sea. No longer is a molecule simply affected by its neighbors, but now it sees a potential from the all the ions in the lattice. This conceptual change will affect a number of components in our model. First, we will need to redefine the Hamiltonian of the system. Secondly, we will need to redefine the normalization constant and evaporation rates for the system. Finally, since perovskite crystals are multi-species systems, we will need a slight modification of our implementation to account for the different species.

Bellaiche and Vanderbilt demonstrated, that if you write the energy of the system in terms of a variation on the average ionic charge of the B site, then you can reduce the system to the sub-lattice of B sites. Since this approach greatly simplifies computations, we will only consider the B sub-lattice in our simulation. We then define the new Hamiltonian as a slightly modified version of the Bellaiche and Vanderbilt expression for the energy of the system[19]:

$$\mathcal{H}(\{h\}) = C \sum_{(ij)} \left(\sum_{l=1}^{\infty} \frac{\Delta q_i \Delta q_j}{|\vec{r}_{ij} - \vec{R}_l|} \right) + \sum_i \Delta\mu h(i) \tag{4.1}$$

Where C contains all necessary constants, Δq_i represents the variation of site i from the average B site charge, the sum (i, j) is the sum over all lattice pairs and the sum l is over all lattice vectors and R_l is the lattice vector.

4.3 Computational Model with Long-Range Interactions

In this section we will derive the new specifications and implementation of the standard model including long-range interactions.

4.3.1 Specifications

Since the energy contribution from the chemical potential term has remained unchanged, the only term we need to reconsider is the summation over lattice pairs. From the detailed balance condition (Eq 2.4) and the constant adsorption rate (Eq 2.5), the evaporation rate can now be written as:

$$\begin{aligned} w(h(i) \to h(i)-1) &= \frac{P_{\mathsf{eq}}(h(i)-1)}{P_{\mathsf{eq}}(h(i))} w(h(i)-1 \to h(i)) \qquad (4.2) \\ &= \exp\left[\frac{C\left(\sum\limits_{(ij)-}\sum\limits_{l=1}^{\infty}\frac{\Delta q_i \Delta q_j}{|\vec{r}_{ij}-\vec{R}_l|} - \sum\limits_{(ij)}\sum\limits_{i=1}^{\infty}\frac{\Delta q_i \Delta q_j}{|\vec{r}_{ij}-\vec{R}_l|}\right) - \Delta\mu}{k_B T}\right] e^{\frac{\Delta\mu}{k_B}} \\ w(h(i) \to h(i)-1) &= C\Delta q_o \exp\left[\sum_i \left(\sum_{l=1}^{\infty}\frac{\Delta q_i}{|\vec{r}_{io}-\vec{R}_l|}\right)\right] \qquad (4.3) \end{aligned}$$

Where Δq_o represents the charge of the currently selected site and Δq_i is the charge at site i. Once again we have evaluated rates for the two events of interest.

4.3.2 Implementation

Implementation of the algorithm will remain similar to the standard KMC model. Now, since we need to keep track of multiple species, we will abandon the two dimensional integer lattice for a three dimensional lattice. One major technical issue remains to be addresses before any computations can be done. The problem lies in the summation of all lattice

vectors for the $(\frac{1}{\vec{r}})$ potential. This summation converges very slowly and convergence is not always guaranteed. For the moment lets assume the $(\frac{1}{\vec{r}})$ sum can be computed in constant time, we will still need to calculate this potential for *each* occupied site, so we have increased our algorithm complexity from $\mathcal{O}(N)$ to $\mathcal{O}(N^2)$. Unfortunately, however, the $(\frac{1}{\vec{r}})$ sum can not be done in constant time using brute force methods. Moreover, the rate of convergence of the sum is never known before the calculation is done. Since the accurate calculation of the long range term is a key element of our model, we cannot use brute force methods to calculate the potential. In the next chapter we will develop a method of doing infinite $(\frac{1}{\vec{r}})$ summations in constant time.

Chapter 5

The Ewald Summation for Finite Height –Mathematical Development

5.1 Introduction

The problem of lattice summations of long range interactions has received much attention in the history of condensed matter physics and computational science. This type of summation plays a key role in many areas of computational physics. The problem was first considered by P. P. Ewald[22]. The Ewald method was generalized by B. Nijober and F. de Wette to any N dimensional case[20]; their results have been used successfully in many applications[21]. We cannot use any previously developed method because we are calculating the potential between a particle in a plane and a particle that has a vector component perpendicular to the plane. In addition we cannot apply any periodic conditions to the plane of the surface because individual crystal layers have compositional variation.

In this chapter, we will develop the Ewald method for any dimensionality between two and three. Although the derivation in this section is restricted to this specific situation, it should be trivial to generalize it to a configuration of any dimensionality. It has already

been demonstrated that results from this chapter reduce to the two dimensional case when the perpendicular component equals zero [27].

The difficulty with the summation is that the standard formulas for the Ewald summation are inherently dependent on expanding the direct vector between two sites in all dimensions. We must develop a general method by which the particle's height above the plane can be accounted for in the Ewald method. We have essentially committed ourselves to calculating an out of plane potential with infinite contributions (see fig 5.1). In this chapter we will derive our own formulation of the Ewald method and demonstrate its usefulness in computer simulations.

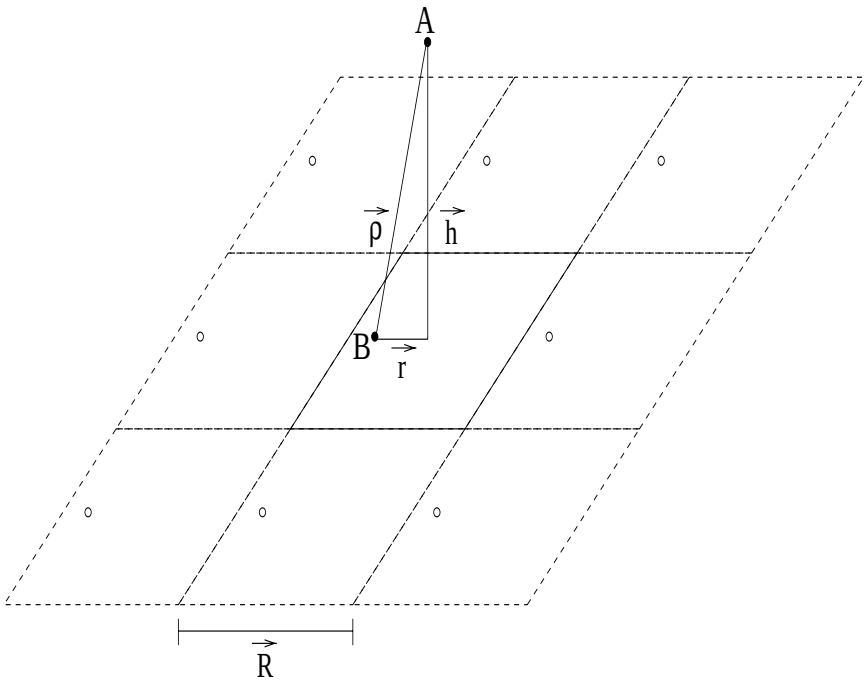

Figure 5.1: Vectors connecting ions A and B plus the periodic image charges due to B.

We want to calculate the Coulomb potential between charged particles A and B plus all the periodic image charges of B in the plane of B. The particles are assumed to reside

on a lattice of fixed sites, where $\vec{R}$ represents the size of the lattice vector, $\vec{h}$ represents the height of particle A above the plane of B and $\vec{\rho}$ represent the vector from A to B. $\vec{r}$ will then be the vector connecting A's image in the plane of B to site B (see fig 5.1). For simplicity we will assume A exists at the origin and assume B is a positive distance from A.

The basic idea behind the Ewald method is to replace the slowly converging sum ($\frac{1}{\vec{r}}$) with a combination of two rapidly converging complimentary sums. For a good introduction to general Ewald techniques, see Allen and Tildesley[25].

To begin, we will multiply the $\frac{1}{\vec{r}}$ term by the sum of the Incomplete Gamma and Complementary Incomplete Gamma functions. As their names suggest, the addition of these two functions equals unity. We can now represent the Coulomb potential of one single pair (not including any periodic images) as:

$$F(\rho) = C[\frac{1}{\rho}(P(\frac{1}{2}, \alpha\rho^2) + Q(\frac{1}{2}, \alpha\rho^2))] \tag{5.1}$$

Where C contains all necessary constants, α denotes some tunable parameter, ρ denotes the length of $\vec{\rho}$, $P()$ denotes the Incomplete Gamma function and $Q()$ denotes the Complementary Incomplete Gamma function. To reduce equation clutter, we will factor the constant term out of the rest of the derivation, making sure to put it back into the main simulation.

We now want to sum the contribution of all periodic images of B, to get a total potential energy for particle A. The total Coulomb potential (including images) is then:

$$\boxed{\phi_+(\vec{\rho}) = \sum_{\lambda} F(|\vec{\rho} - \vec{R}_\lambda|) = \sum_{\lambda} \overline{f}(|\vec{\rho} - \vec{R}_\lambda|) + \sum_{\lambda} f(|\vec{\rho} - \vec{R}_\lambda|)} \tag{5.2}$$

where

$$\overline{f}(x) = \frac{1}{x}Q(\frac{1}{2}, \alpha x^2) \tag{5.3}$$

$$f(x) = \frac{1}{x}P(\frac{1}{2}, \alpha x^2) \tag{5.4}$$

Here we notice that $\sum \overline{f}$ converges very rapidly, while $\sum f$ converges inversely proportional to $\sum \overline{f}$. Since 'flat' functions correspond to sharp peaks in reciprocal space, if we perform the $\sum f$ in reciprocal space, it will converge very rapidly. The parameter α can then be viewed as a point in the $\sum \frac{1}{r}$ where the real space summation ends and the reciprocal space summation begins (see fig 5.2).

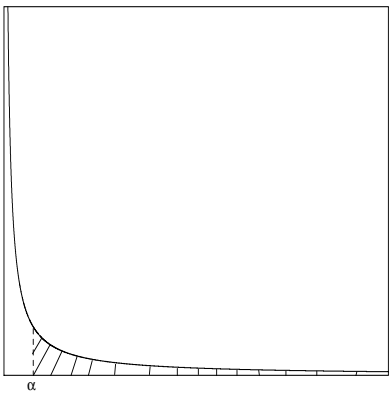

Figure 5.2: Graph of $\frac{1}{r}$ convergence and α cutoff of real space summation.

5.2 The Reciprocal Space Sum

The idea is to perform a Fourier transform of $\sum_{\lambda} f(|\vec{\rho} - \vec{R}_{\lambda}|)$. Let's start by defining $g(\vec{k})$ as follows:

$$g(\vec{k}) \equiv \int dr\ e^{-i\vec{k}\cdot\vec{r}'} f(r')\, dr' \tag{5.5}$$

Now, we will multiply both sides by $\frac{1}{L^2}e^{i\vec{k}\cdot\vec{r}}$ and sum both sides over the entire reciprocal space.

$$\frac{1}{L^2}\sum_{\lambda} e^{i\vec{k}_\lambda\cdot\vec{r}}g(\vec{k}) = \int \frac{1}{L^2}\sum_{\lambda} e^{i\vec{k}_\lambda\cdot(\vec{r}-\vec{r}')}f(r')\,dr' \tag{5.6}$$

Now, we will use the relation $\frac{1}{L^2}\sum\limits_{\lambda} e^{i\vec{k_\lambda}\cdot\vec{r}} = \sum\limits_{\lambda}\delta(|\vec{r}-\vec{R}_\lambda|)$[14, 20]. With this relation, we can write:

$$\begin{aligned}\frac{1}{L^2}\sum_{\lambda} e^{i\vec{k}_\lambda\cdot\vec{r}}g(\vec{k}) &= \int\sum_{\lambda}\delta(|\vec{r}'-\vec{r}-\vec{R}_\lambda|)f(r')\,dr' \\ \frac{1}{L^2}\sum_{\lambda} e^{i\vec{k}_\lambda\cdot\vec{r}}g(\vec{k}) &= \sum_{\lambda} f(|\vec{r}-\vec{R}_\lambda|)\end{aligned} \tag{5.7}$$

With this relation, we have expressed the slow converging real space summation as a summation in reciprocal space. In the following section we will derive $g(\vec{k})$ into a form that is computationally meaningful.

5.2.1 Derivation of the $g(k)$ Term

Again, to avoid equation clutter, we will let $r = |\vec{r}|$, $k = |\vec{k}|$, $h = |\vec{h}|$, and $\rho = |\vec{\rho}|$. Plugging equation 5.4 into equation 5.5, we obtain:

$$g(\vec{k}) = \int dr\; e^{-i\vec{k}\cdot\vec{r}}\frac{1}{\rho}\frac{1}{,\,(\frac{1}{2})}\int_0^{\alpha(r^2+h^2)} e^{-t}t^{-\frac{1}{2}}\,dt \tag{5.8}$$

let $t = \frac{k^2(r^2+h^2)}{4y}$, then $y = \frac{k^2(r^2+h^2)}{4t}$

$$g(\vec{k}) = \frac{1}{,\,(\frac{1}{2})}\int dr\, e^{-i\vec{k}\cdot\vec{r}}\frac{1}{\sqrt{(r^2+h^2)}}\int_{\infty}^{\frac{k^2}{4\alpha}} e^{-\frac{k^2(r^2+h^2)}{4y}}\left(\frac{k}{2}\sqrt{\frac{r^2+h^2}{x}}\right)^{-1}\left(-\frac{k^2(r^2+h^2)}{4y^2}\right)dy$$

Extracting terms from the dy integral gets:

$$\left(\frac{k\sqrt{r^2+h^2}}{2}\right)^{-1}\left(\frac{-k^2(r^2+h^2)}{4}\right)\int_{\infty}^{\frac{k^2}{4\alpha}} e^{\frac{-k^2(r^2+h^2)}{4y}}\frac{1}{y^{\frac{3}{2}}}\,dy$$

Consolidating all terms and simplifying, we can now the the $g(\vec{k})$ integral as:

$$g(\vec{k}) = \frac{1}{,\,(\frac{1}{2})}\frac{1}{\sqrt{(r^2+h^2)}}\left(\frac{2}{k\sqrt{r^2+h^2}}\right)\left(\frac{-k^2(r^2+h^2)}{4}\right)\int d\vec{r}\,e^{-i\vec{k}\cdot\vec{r}}\int_{\infty}^{\frac{k^2}{4\alpha}}\frac{e^{\frac{-k^2(r^2+h^2)}{4y}}}{y^{\frac{3}{2}}}\,dy$$

$$g(\vec{k}) = \frac{1}{,\,(\frac{1}{2})}\int_{\frac{k^2}{4\alpha}}^{\infty}\frac{k}{2y^{\frac{3}{2}}}\int e^{\frac{-k^2(r^2+h^2)}{4y}-i\vec{k}\cdot\vec{r}}\,d\vec{r}\,dy \tag{5.9}$$

We choose to do the inside integral first. We will start by adding and subtracting $\left(\frac{k\vec{r}}{2\sqrt{y}}\right)^2$ in the exponential:

$$\begin{aligned}\int e^{-(\frac{k^2(r^2+h^2)}{4y}+i\vec{k}\cdot\vec{r})}\,d\vec{r} &= \int e^{-\left(\left(\frac{k\vec{r}}{2\sqrt{y}}\right)^2+\frac{k^2h^2}{4y}+i\vec{k}\cdot\vec{r}+\left(i\frac{\sqrt{y}}{k}\vec{k}\right)^2+y\right)}\,d\vec{r}\\ &= \int e^{-\left(\left(\frac{k\vec{r}}{2\sqrt{y}}\right)^2+i\vec{k}\cdot\vec{r}+\left(i\frac{\sqrt{y}}{k}\vec{k}\right)^2\right)-y-\frac{k^2h^2}{4y}}\,d\vec{r}\\ &= \int e^{-\left(\frac{k\vec{r}}{2\sqrt{y}}+i\frac{\sqrt{y}}{k}\vec{k}\right)^2}\,d\vec{r}\cdot e^{-\frac{k^2h^2}{4y}-y}\\ &= \pi\left(\frac{2\sqrt{y}}{k}\right)^2 e^{-\frac{k^2h^2}{4y}-y}\end{aligned} \tag{5.10}$$

Where we have used the definition of a 2 dimensional Gaussian integral over all space to evaluate the integral. Putting equation 5.10 into equation 5.9, we obtain:

$$g(\vec{k}) = \frac{k}{,\,(\frac{1}{2})}\int_{\frac{k^2}{4\alpha}}^{\infty}\frac{1}{2y^{\frac{3}{2}}}\left(\pi\left(\frac{2\sqrt{y}}{k}\right)^2 e^{-\frac{k^2h^2}{4y}-y}\right)\,dy \tag{5.11}$$

Simplifying, we obtain the final result:

$$\boxed{g(|\vec{k}|) = \frac{2\pi}{k,\,(\frac{1}{2})}\int_{\frac{k^2}{4\alpha}}^{\infty} y^{-\frac{1}{2}}e^{-\frac{k^2h^2}{4y}-y}\,dy} \tag{5.12}$$

5.3 Converging the Sum

So far we have considered only one set of ionic nuclear charge in our model. Implicit in this argument is that this ionic lattice is a sub-lattice in an overall charge neutral system. If it were not, we should get an infinite potential at site A for a lattice of infinite charged particles! In order to guarantee the converge of the summation, we will subtract a constant term from one sub-lattice and add the same term back into the second sub-lattice.

On a conceptual level, this term has can be thought as a free electron contribution or the interaction of site A with its own images[21]. For our purposes, we will view this term as nothing more than a mathematical trick to allow us to do the computation.

We can now write the contribution to be subtracted from the ionic charge as:

$$\sum_{\lambda} F(|\vec{R}_\lambda + \vec{h}|) = \sum_{\lambda} \overline{f}(|\vec{R}_\lambda + \vec{h}|) + \sum_{\lambda} f(|\vec{R}_\lambda + \vec{h}|) - \frac{1}{,\ (\frac{1}{2})h} \int_0^{\alpha h^2} e^{-t} t^{-\frac{1}{2}}\, dt \tag{5.13}$$

The last term accounts for the inherent inclusion of the $R = 0$ term in the reciprocal space portion of the sum. Since this term corresponds to an infinity in real space, we need to remove it from the computation. For computation, we will also need to evaluate the last term at $h = 0$. The limit of the last term as $h \to 0$ is:

$$\lim_{h \to 0} \left(\frac{1}{,\ (\frac{1}{2})h} \int_0^{\alpha h^2} e^{-t} t^{-\frac{1}{2}}\, dt \right) = 2\sqrt{\frac{\alpha}{\pi}} \tag{5.14}$$

5.4 The Total Potential

We can now write the total potential of the site pair plus images, including equation 5.13 as:

$$\phi_{\text{tot}} = \phi_+(\vec{\rho}) - \sum_{\lambda \neq 0} F(|\vec{R}_\lambda + \vec{h}|)$$

$$\phi_{\text{tot}}(\vec{\rho}) = \sum_\lambda \overline{f}(|\vec{\rho} - \vec{R}_\lambda|) - \sum_{\lambda \neq 0} \overline{f}(|\vec{R}_\lambda + \vec{h}|) + \sum_{\lambda \neq 0} f(|\vec{k}_\lambda|)(e^{i\vec{k}_\lambda \cdot \vec{r}} - 1) + \frac{1}{,\,(\frac{1}{2})} \frac{1}{h} \int_0^{\alpha h^2} e^{-t} t^{-\frac{1}{2}}\, dt$$

where

$$\overline{f}(x) = \frac{1}{x} erfc(\sqrt{\alpha} x) \tag{5.15}$$

$$f(x) = \frac{1}{|\vec{R}|^2} \frac{2\pi}{x,\,(\frac{1}{2})} \int_{\frac{x^2}{4\alpha}}^{\infty} t^{-\frac{1}{2}} e^{-(\frac{x^2 h^2}{4t} + t)}\, dt \tag{5.16}$$

5.5 Evaluating the Integral

Now that we have an expression for the potential energy of the pair, we are left to evaluate the terms. Numerical evaluation of the *erfc* is accomplished using a standard algorithm from Press et. al[28]. The $f(x)$ integral, however has no closed form solution and is not a standard computational term. General numerical steepest descent methods are available to evaluate such integrals[26], but these methods are complicated to implement, so we will develop a straightforward analytic method for evaluating this term. Since α is a free parameter, we will assume that we can choose α to be small. Since α can be chosen small, the lower limit of the integral will be large, which in turn will correspond to small values of $\frac{x^2 h^2}{4t}$ in the exponential. With this in mind, we will do a Taylor expansion of $e^{-\frac{x^2 h^2}{4t}}$ about 0. Let $\beta = -\frac{x^2 h^2}{4}$, then we can re-write the integral in $f(x)$ as:

$$\int_a^\infty t^{-\frac{1}{2}} e^{-t} e^{\frac{\beta}{t}}\, dt \tag{5.17}$$

After a Taylor expansion about 0, we can express the integral as:

$$\int_a^\infty f(t)\,dt = \sum_{i=0}^{N} \frac{\beta^i}{i!} \int_a^\infty t^{-i-\frac{1}{2}} e^{-t}\,dt \tag{5.18}$$

We are now left to evaluate any integral of the general form:

$$\int_a^\infty t^{-\frac{n}{2}} e^{-t}\,dt \tag{5.19}$$

where $n = 1, 3, 5...$

We will now attempt to solve this integral with an integration by parts, after one iteration we obtain:

$$\int_a^\infty e^{-t} t^{-1\frac{n}{2}} = \tfrac{1}{1-\frac{n}{2}} e^{-t} t^{1-\frac{n}{2}} |_a^\infty + \tfrac{1}{1-\frac{n}{2}} \int_a^\infty e^{-t} t^{1-\frac{n}{2}}\,dt$$

After $\frac{n-1}{2}$ iterations, we will arrive at the Complementary Incomplete Gamma function $Q()$, which can be readily computed using standard algorithms from Press et al[28]. The integral is then equal to:

$$\int_a^\infty e^{-t} t^{-\frac{n}{2}} = -e^{-a} a^{-\frac{n}{2}} \sum_{i=1}^{\frac{n-1}{2}} \left(\prod_{k=1}^{i} \frac{1}{k-\frac{n}{2}} \right) a^i + \prod_{i=1}^{\frac{n-1}{2}} \left(\frac{1}{i-\frac{n}{2}} \right) \left(Q(a, -\frac{1}{2}) \cdot, (\frac{1}{2}) \right) \tag{5.20}$$

Plugging equation 5.20 back into equation 5.18 we find that the integration in the $f(x)$ term can be computed as:

$$\int_a^\infty f(t)\,dt = \sum_{i=0}^{N} \frac{-\beta^i}{i!} e^{-a} a^{-\frac{n}{2}} \sum_{i=1}^{\frac{n-1}{2}} \left(\prod_{k=1}^{i} \frac{1}{k-\frac{n}{2}} \right) a^i + \prod_{i=1}^{\frac{n-1}{2}} \left(\frac{1}{i-\frac{n}{2}} \right) \left(Q(a, -\frac{1}{2}),\ (\frac{1}{2}) \right)$$

Figure 5.3 demonstrates the convergence of the value of the computed integral as a function of the number of terms included in the Taylor expansion.

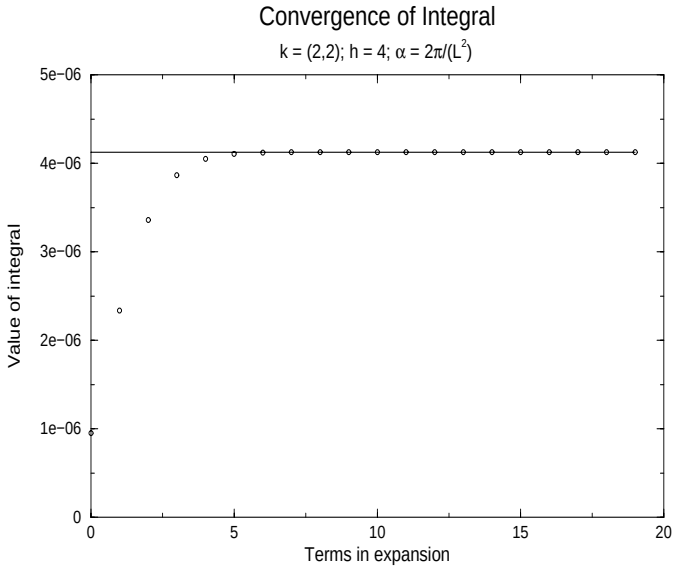

Figure 5.3: Convergence of computed value to actual value with increasing number of expansion terms.

5.5.1 Computing the Imaginary Term

The only term left to evaluate is $\sum_{\lambda \neq 0} (e^{i\vec{k}_\lambda \cdot \vec{r}} - 1)$, and then we will freely be able to calculate the potential for the site pair. We will re-write the exponential term as:

$$e^{i\vec{k}\cdot\vec{r}} = \cos(\vec{k} \cdot \vec{r}) + i \sin(\vec{k} \cdot \vec{r}) \tag{5.21}$$

We will assume that we are summing over a complete set for all lattice vectors, and using the identity $\sin(-x) = -\sin(x)$, the $\sin(x)$ terms cancel. We can then write the summation as a function of only real terms:

$$\sum_{\lambda \neq 0} (e^{i\vec{k}_\lambda \cdot \vec{r}} - 1) = \sum_{\lambda \neq 0} (\cos(\vec{k}_\lambda \cdot \vec{r}) - 1) \tag{5.22}$$

See Appendix C for the exact code used implement equation 5.22. We are now ready to compute the potential energy for any site pair on in the lattice space.

Chapter 6

The Ewald Summation –Implementation and Results

In this chapter we will analyze the computational improvements made by developing the Ewald method. We will demonstrate that without developing this method, performing the required calculations for the growth simulation would not have been possible.

6.1 Advantages of the Ewald Sum

As mentioned above, the choice of α is not arbitrary. As you decrease α you increase the number of terms needed in the real space sum. Notice that in the limit of α approaching 0, you recover the strict brute force summation. We have chosen alpha to be $\left[\frac{\pi}{8|\vec{R}|^2}\right]$, which is partially empirically chosen to balance the number of terms necessary in the real space term and the necessity to keep α small. We were able to implement straightforwardly the computations discussed above. With this choice of α we have been able to demonstrate fast and accurate calculations for the potential energy of the infinite-range sum. The convergence of the brute force summation to the value calculated by Ewald method is shown in figure 6.1.

The computation time for the brute force term was approximately 5 hours. The computation time for all possible 64 site pairs using the Ewald method required under 1 second. The important thing to note here is that in order to increase the brute force accuracy by an order of magnitude, you need exponentially increasing time. This increase is also demonstrated by figure 6.1. Since the rate of convergence for the brute force summation is never guaranteed, a brute force approach is not computationally feasible.

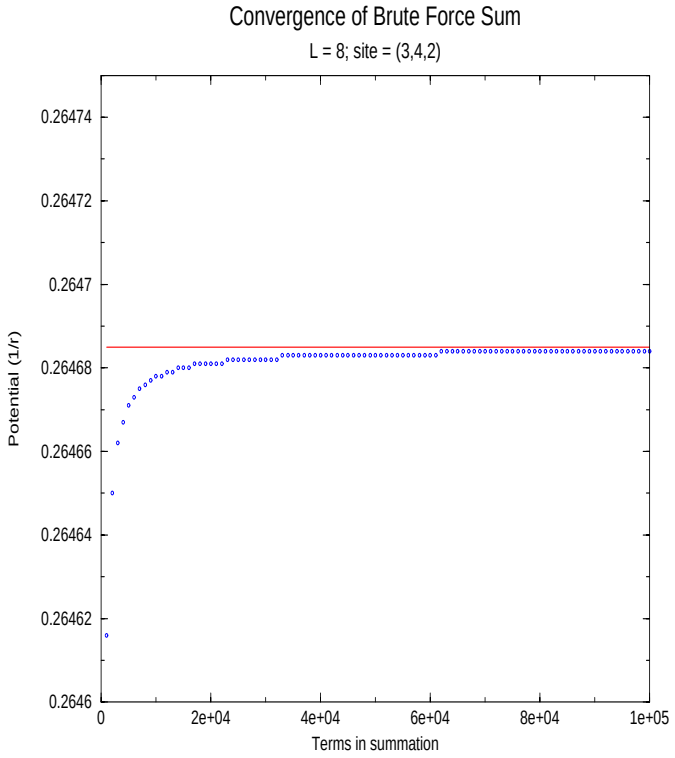

Figure 6.1: Convergence of brute force method to Ewald summation method.

Table 6.1 shows the absolute (ie time active in system) computation times for our brute force summation to produce the equivalent of table 6.2 as a function of system size. The brute force summation included 5000 terms. Again the convergence rate for each term in the brute force varies, but on average the terms are accurate to 10^{-4}. To demonstrate the

Lattice Vector (L)	Computation Time For All Pairs (hh:mm:ss)
4	00:11:40
6	00:31:07
8	01:04:50
10	01:56:43
12	03:10:38
14	04:50:30
16	07:00:13
18	09:43:37
20	13:04:39

Table 6.1: Brute force total computational time for all possible site pairs (with 5000 terms).

relative accuracy of these brute force tables, 5000 terms represents the second data point from the origin in figure 6.1. Since the 8 x 8 calculation required over an hour for 5000 terms (see table 6.2), the computational increase associated with improving the answers for even an 8 x 8 by an order of magnitude is extremely large.

6.2 Results of the Ewald Method

In this section will provide some more general data concerning the Ewald summation method. We will examine the energy as a function of lattice size and present the potential terms for an 8 x 8 lattice.

6.2.1 Characterizing the Lattice

Because the Ewald summation calculates a sum to infinity of all lattice vectors, we would like to get a sense that the total energy of our system does not change with lattice size. To demonstrate this point, we will examine a 2 dimensional lattice with all sites occupied by

a +1 charge. A plot of the average energy per pair as a function of lattice size is shown in figure 6.2.

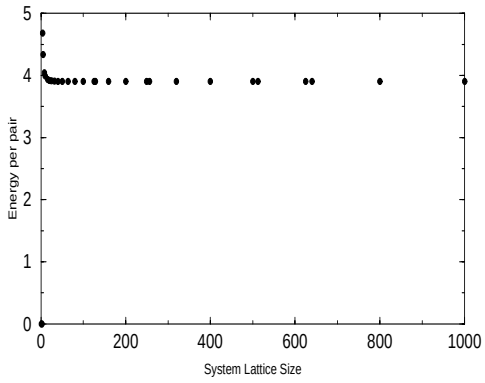

Figure 6.2: Average energy per pair plotted versus lattice size.

As we had expected, for reasonably large system sizes, the average energy per pair remains unchanged.

6.2.2 Ewald Summation Data

Because the calculation of long-range potentials is of such general applicability, we will include a table for all possible site pairs in a 8 x 8 x 8 lattice, disregarding the constant term. It is important to note here, that due to the periodic requirement, we only needed to calculate bonds for up to half the lattice vector in any given direction. Also, the table shown is the potential between a site at $(0, 0, 0)$ and site (i, j, k) away. Because of translational invariance, the choice of the origin is arbitrary. Table 6.2 then describes every possible situation in the lattice.

Distance (x,y,z)	Potential
(0,0,1)	1.000000000000
(0,0,2)	0.500000000000
(0,0,3)	0.333333333333
(0,0,4)	0.250000000000
(0,1,0)	1.004489711297
(0,1,1)	0.711418857478
(0,1,2)	0.451049546723
(0,1,3)	0.319424385300
(0,1,4)	0.245069564385
(0,2,0)	0.518940145984
(0,2,1)	0.465339784006
(0,2,2)	0.369527117693
(0,2,3)	0.290496674772
(0,2,4)	0.233897759260
(0,3,0)	0.379966540102
(0,3,1)	0.360549195848
(0,3,2)	0.315714410737
(0,3,3)	0.266567709263
(0,3,4)	0.223640622847
(0,4,0)	0.344737958438
(0,4,1)	0.331421567987
(0,4,2)	0.298140023461
(0,4,3)	0.257787634510
(0,4,4)	0.219629740765
(1,1,0)	0.715789186607
(1,1,1)	0.585695672920
(1,1,2)	0.415688760333
(1,1,3)	0.307731802530
(1,1,4)	0.240651599158
(1,2,0)	0.469424986952
(1,2,1)	0.429547316092
(1,2,2)	0.352205581585
(1,2,3)	0.282912741838
(1,2,4)	0.230573680339
(1,3,0)	0.364353577568
(1,3,1)	0.347369970899
(1,3,2)	0.307234269073

Distance (x,y,z)	Potential
(1,3,3)	0.261883918469
(1,3,4)	0.221239739445
(1,4,0)	0.335110872777
(1,4,1)	0.322889737723
(1,4,2)	0.292040048310
(1,4,3)	0.254057744745
(1,4,4)	0.217569021221
(2,2,0)	0.386561698881
(2,2,1)	0.365039592679
(2,2,2)	0.316901073980
(2,2,3)	0.266101888138
(2,2,4)	0.222869696858
(2,3,0)	0.331439074071
(2,3,1)	0.319068022010
(2,3,2)	0.288318839367
(2,3,3)	0.251082876492
(2,3,4)	0.215583314225
(2,4,0)	0.313330929686
(2,4,1)	0.303466411831
(2,4,2)	0.277949158399
(2,4,3)	0.245308533662
(2,4,4)	0.212677885545
(3,3,0)	0.303693247101
(3,3,1)	0.294698853564
(3,3,2)	0.271275160979
(3,3,3)	0.240932289316
(3,3,4)	0.210114549824
(3,4,0)	0.293379278507
(3,4,1)	0.285528717997
(3,4,2)	0.264684572296
(3,4,3)	0.236901180514
(3,4,4)	0.207901518436
(4,4,0)	0.285590254400
(4,4,1)	0.278489057927
(4,4,2)	0.259412999436
(4,4,3)	0.233513527001
(4,4,4)	0.205955612669

Table 6.2: Possible potential terms for an 8 x 8 x 8 lattice.

Chapter 7

Simulation Results and Conclusions

In this chapter we will discuss some preliminary results from our simulation model. We will then compare these results to those obtained with the standard model.

7.1 Surface of the Advanced Model

While full analysis of results from our advanced model has not yet been possible, preliminary results are very promising. Figure 7.1 demonstrates a calculated surface configuration from our advanced model for values chosen near equilibrium for a B site sub-lattice with 2 possible species, $\Delta q_1 = +1$ and $\Delta q_2 = -2$. A 10 x 10 x 30 lattice was run for 300 Monte Carlo steps. The the adsorption probabilities of each species were weighted to guarantee overall charge neutrality. Because of the constant term used in the Ewald method, we are able to calculate the potential at intermediate steps (when the overall charge is not necessarily neutral), because the overall contribution of this constant term will reduce to 0.

It seems that this region represents growth above the roughening transition, but to make any more meaningful statements about the configuration, much further analysis is required.

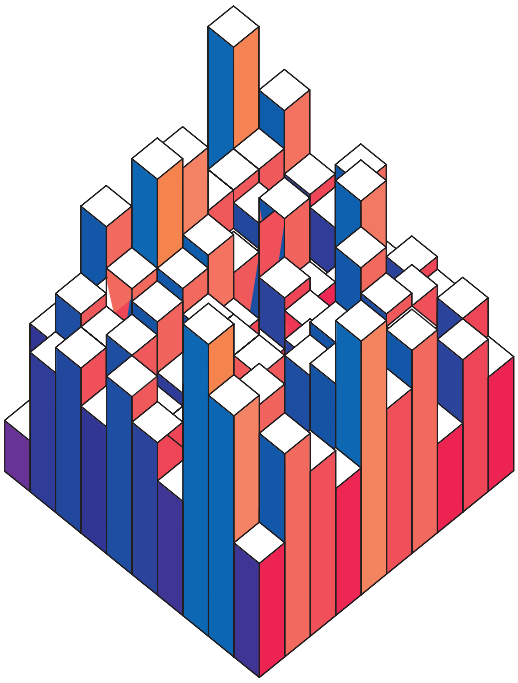

Figure 7.1: Typical surface configuration from advanced simulation for values near equilibrium.

7.2 Conclusions

In summary, we have implemented the standard Kinetic Monte Carlo model for the growth of crystal structures. We demonstrated the agreement of this model to analytic expectations for growth kinetics. We examined various strengths and weakness of the standard model and proposed the introduction of long-range terms as a replacement for local interactions. Motived to calculate the infinite-range summations, we developed a unique method by which general calculations of this type can be implemented in reasonable computer time. We were able to implement our model including long-range terms to produce some preliminary results. We compared of these results to those produced from the standard model. While further analyses of this preliminary data is required, we are confident that this model can

serve as a useful tool to further our understanding of the growth mechanisms in complex crystal structures.

Appendix A

Definition of Selected Terms

deterministic : An event such that the outcome can be known given the initial conditions.

discrete : Finite or countable by nature.

event : Any sub-set of the state space for an experiment.

event list : A list of possible events from the state space.

probability : Likelihood that an event will occur. Must be a number on the range $[0, 1]$ and the sum of all complimentary events within a state space must equal unity.

sample : Selection of an event from the event list.

state space : The set of all possible outcomes of an experiment.

stochastic : An event such that the outcome is random by nature.

Appendix B

Kinetic Growth Code

```
/* cr_pt ==========================================================================
 This is a simulation of crystal growth including long range ionic
   interactions. This file must me used in conjunction with an external
   random number generator and Numerical Recipes routines.
   Programmer: TJ Walls

 This program will run a simulation of a L x L x H lattice.
   It will initialize the lattice by filling in the lattice from 0
   to BASE with a random mixture of the desired components. It will
   then proceed by repeated realizations of  creation and evaporation
   events according the probabilities described in a Kinetic Monte Carlo
   simulation.

 RETURN VALUES:
  On successful completion, the simulation return, the simulation will
    place the integer heights in an L x L array in file `LRO.height` and
    will display the final growth rate on STDOUT. If
    the simulation attempts to grow or recede beyond the bounds of the
    static array used, it will return an error message on STDERR.
   ================================================================================ */
#include <stdio.h>
#include <string.h>
#include <iomanip.h>
#include <fstream.h>
#include <iostream.h>
#include <stdlib.h>
#include <math.h>
#include "rvgs.h"
#include "rngs.h"

/* Constants ------------------------------------------------
   Now we will define the required constant terms for the simulation.
------------------------------------------------------------------ */
#define  RUNS             2              // Number of times to run simulation
#define  MC_STEPS        30              // How long to grow crystal
#define  kT             500.0            // Boltzmann's const * temp
#define  BASE           5                // Minimum number of occupied rows
#define  TEST_SOS         0              // Impose sos restriction
```

```
#define  L               10              // Size of base square
#define  H               30              // Height of crystal
#define  N_SITES     (L*L*H)             // Number of possible sites in lattice
#define  qB               4              // Average B site charge
//---------------------------------------------------------------------
// These constants are required for the calculation of the potential energies.
#define   PI      3.14159265359
#define   root2   1.41421356237
#define   rootpi  (sqrt(PI))
#define   unitk   (2.0 * PI / (double)L)
#define   CUTOFF  (pow(10,-15))
#define   ALPHA   (PI/(8.0*(double)(L*L)))
#define   root_A  (sqrt(ALPHA))


// Prototypes ================================================

// Column will contain the information for an L x L array of columns
struct column
{
  int stack[H];
  long top;
  long num_sites;
};


double   Initialize(column[][L], double[L/2+1][L/2+1][H/2+1]);
void   Ewald25(double[L/2+1][L/2+1][H/2+1]);
long   Check_pbc(long, long);
long   Conv_coord(long**, long, long);
void   Map(long[][N_SITES]);
void   MC_step(column[][L], double V_clmb[L/2+1][L/2+1][H/2+1],
       double, double, long&, long&);
int    Select_site(int);
double   Prob_calc(column[][L], double[L/2+1][L/2+1][H/2+1], long, long);
double Sim(double, double&, double[L/2+1][L/2+1][H/2+1]);
double Standard_deviation(double&, double[]);
void   Data_dump(column lattice[][L]);


/* main() -------------------------------------------------
 This is the main control body of the code. It will initialize
   the array of all possible potential terms. It then runs
   Monte Carlo simulation for a range of chemical potentials
   defined in the control loop.

   *** This routine includes an extraneous computation of
         standard deviation. This is a feature left over from
 the standard model, and is intended to be included in
 latter versions of the simulation for statistical
 calculations.
```

```
-------------------------------------------------------*/
int main()
{
  double prob;                          //Adsorption probability
  double V_clmb[L/2+1][L/2+1][H/2+1];   //Matrix of all possible potentials
  double fin_rate;                      //Final rate of growth
  double stor_ans[RUNS],                //Storage for computed growth rates
    std_dev;                            //Standard deviation of rates
  double i;                             //Variable for control rates


  // Initialize the array of required potentials.
  Ewald25(V_clmb);

  for (i = 630; i <= 630; i = i + .25)
    {
      for (int good = 0; good < RUNS; good++)
{
  fin_rate = Sim(i, prob, V_clmb);
  fin_rate = fin_rate / prob;
  stor_ans[good] = fin_rate;
      } //end for

      std_dev = Standard_deviation(fin_rate, stor_ans);

      cout << setprecision(6) << i << '\t' << fin_rate << '\t' << std_dev
   << '\t' << (std_dev / sqrt(RUNS - 1)) << endl;
    } //end for

  return 0;
} //end main

/* Sim()-----------------------------------------------------
   This is the main simulation section of the code. It takes
 the chemical potential and returns through the reference parameter
 p_adsorb the probability of adsorption. It also takes the
 array of Coulomb potentials as an argument.
-------------------------------------------------------------*/
double Sim(double chem,        //Chemical potential driving force
   double &p_adsorb,   //Adsorption probability
   double V_clmb[L/2+1][L/2+1][H/2+1])
{
//Variable declarations
  column lattice[L][L];        //Lattice of crystal sitess
  long tot_height;             //Total height of crystal
  double V_energy,             // Energy sum of bonds
    distrib,                   // Re-normalization constant
    max_evap;                  // Maximum evaporation rate

  long adsorb;                 //Number of creation events for elapsed time
  long evap;                   //Number of evaporation events
```

```
  long i, j, k;                //Loop variables.

//Initialize Variables
  PlantSeeds(123456789);
  max_evap = Initialize(lattice, V_clmb);

  p_adsorb = exp(chem / kT);
  distrib = (p_adsorb + exp(max_evap / kT));
  p_adsorb = p_adsorb / distrib;


//Code Body
  for (i = 0; i < MC_STEPS; i++) {
    for (j = 0; j < (L * L); j++)
      MC_step(lattice, V_clmb, p_adsorb, distrib, adsorb, evap);
  } //end for

  tot_height = 0;
  for (i = 0; i < L; i++)
    for (j = 0; j < L; j++)
      tot_height = tot_height + lattice[i][j].num_sites;


  Data_dump(lattice);
  return (tot_height - (L*L*BASE)) / (double)(L*L*MC_STEPS);
}

/* Initialize()--------------------------------------------
   This function will initialize the simulation. It will place
a random distribution of the desired elements into a lattice
positions below BASE. It will return to maximum possible rate
of evaporation to be used in normalization.
-------------------------------------------------------------*/
double Initialize(column lattice[][L],
  double V_clmb[L/2+1][L/2+1][H/2+1])
{
  long i,j,k;                     // Loop variables
  long nx, ny, nz;                // Loop variables
  double max_evap = 0.0;          // Maximum evaporations rate
  long **coord;                   // Array which map numbers 0 to N to
                                  // their relative positions

  coord = new (long *)[3];
  for (i = 0; i < 3; i++)
    coord[i] = new long[N_SITES];

  //Initialize coordinate map
  Conv_coord(coord,L,H);

  for (i = i; i < N_SITES; i++) {
      nx = Check_pbc((coord[2][0] - coord[2][i]), L);
```

```
      ny = Check_pbc((coord[1][0] - coord[1][i]), L);
      nz = Check_pbc((coord[0][0] - coord[0][i]), H);

      if (nx != 0 || ny != 0 || nz != 0)
max_evap = max_evap + V_clmb[nx][ny][nz];
    }
  max_evap = max_evap * Select_site(1);

  for (i = 0; i < 3; i++)
    delete(coord[i]);
  delete(coord);

  for (i = 0; i < L; i++)
    for (j = 0; j < L; j++) {
      lattice[i][j].top = BASE - 1;
      lattice[i][j].num_sites = BASE;
      for (k = 0; k < H; k++) {
if (k < BASE) {
  lattice[i][j].stack[k] = Select_site(0);
} else
  lattice[i][j].stack[k] = -1000;
      } //end for
    } // end for

  return max_evap;
} //end Initialize()

/* Prob_calc()-------------------------------------------------------
   This function will return the non-normalized probability
of evaporation for a given configuration.
-----------------------------------------------------------------*/
double Prob_calc(column lattice[][L],
 double V_clmb[L/2+1][L/2+1][H/2+1],
 long   i,
 long   j)
{
  long nx, ny, nz;
  long ni, nj, nk;

  double p_evap = 0.0;

  for (ni = 0; ni < L; ni++)
    for (nj = 0; nj < L; nj++)
      for (nk = 0; nk <= lattice[ni][nj].top; nk++) {
nx = Check_pbc((ni - i), L);
ny = Check_pbc((nj - j), L);
nz = Check_pbc((nk - lattice[i][j].top), H);
if (nx != 0 || ny != 0 || nz != 0) {
  p_evap = p_evap + (lattice[ni][nj].stack[nk] * V_clmb[nx][ny][nz]);

  if (fabs(lattice[ni][nj].stack[nk]) > (float)Select_site(1) + .1) {
```

```
    fprintf(stderr, "(%ld,%ld,%ld)\t%ld\n", ni, nj, nk,
                    lattice[ni][nj].stack[nk]);
    exit(55);
  }
}
      }


  return fabs(p_evap);
} //end Prob_calc


/* MC_step()-------------------------------------------------
   This function will perform 1/Nth of 1 MC step. (here
a MC step is defined as complete when each of the N sites
has had an opportunity to change, on average).
-------------------------------------------------------------*/
void MC_step(column lattice[][L],      //Lattice of crystal heights
     double V_clmb[L/2+1][L/2+1][H/2+1],
     double p_adsorb,         // Prob of adsorption
     double distrib,          // Normalization constant
     long &adsorb,       //Number of sites added to system
     long &evap)         //Number of sites evaporated from system
{
  int type;
  long i, j;                                 // Loop variables

  double trans_prob,                        //probability of transition
    p_evap,                                 // Prob of evaporation
    try_1;                                  //Generated number

  SelectStream(1);
  i = Equilikely(0, (L - 1));
  SelectStream(2);
  j = Equilikely(0, (L - 1));            //lattice column to change

  type = Select_site(0);

  p_evap = Prob_calc(lattice, V_clmb, i, j);
  p_evap = exp(p_evap / kT);
  p_evap = p_evap / distrib;

  trans_prob = p_adsorb + p_evap;
  SelectStream(3);
  try_1 = Random();

//---------- Range Test
  if (lattice[i][j].top == (H-1)) {
    cerr << "H = " << H << endl;
    cerr << "Error: Grown sites exceed array size!\n";
    fprintf(stderr, "Stack (%ld,%ld) crashed\n", i, j);
```

```
    printf("Adsorbed:%ld\tEvaped:%ld\n", adsorb,evap);
    exit(57);
  } else if (lattice[i][j].top == 0) {
    cerr << "Error: Negative growth rate, lost all crystal!\n";
    printf("Stack (%ld,%ld) crashed\n", i,j);
    printf("Adsorbed:%ld\tEvaped:%ld\n", adsorb,evap);
    exit (58);
  }
  if (lattice[i][j].top >= H || lattice[i][j].top < 0) {
    cerr << "THIS SHOULD NOT HAVE HAPPENED!";
    exit(59);
  }

//---------- Adsorption
  if (try_1 < p_adsorb) {
    //if (i == 3 && j == 3)
    //  fprintf(stderr, "(%6.4f,A->%ld)", p_adsorb, lattice[i][j].top+1);
    lattice[i][j].top++;
    lattice[i][j].stack[lattice[i][j].top] = type;
    lattice[i][j].num_sites++;
    adsorb++;

    return;
  }

//----------- Desorption
  else if (try_1 < trans_prob) {
    //if (i == 3 && j == 3)
    //  fprintf(stderr, "(%6.4f,D->%ld)", p_evap, lattice[i][j].top-1);
    lattice[i][j].stack[lattice[i][j].top] = -1000;
    lattice[i][j].top--;
    lattice[i][j].num_sites--;
    evap++;

    return;
  }

//------------- Nothing
  else
    return;
} //end MC_step()

/* Select_site() -------------------------------------------
  This function will return the type of site selected based on
compositional probability.
------------------------------------------------------------ */
int Select_site(int flag)
{
  if (flag == 1) {
    return 2;
  } else if (flag == 0) {
```

```
    SelectStream(28);

    if (Random() < .3333333333)
      return (-2);
    else
      return 1;
  }
}

/* Data_dump()-------------------------------------------------
   This function will output the integer lattice heights to
a L x L integer array in file `LRO.height`
   ------------------------------------------------------------*/
void Data_dump(column lattice[][L])
{
  FILE *fp;
  long i, j,k;
  long p, m, error;

  fp = fopen("LRO.height", "w");


  p = 0;
  m = 0;
  error = 0;
  for (i = 0; i < L; i++) {
    for (j = 0; j < L; j++) {
      fprintf(fp, "%ld\t", lattice[i][j].num_sites);
      for (k = 0; k <= lattice[i][j].top; k++) {
if (lattice[i][j].stack[k] == -2) {
  m++;
} else if (lattice[i][j].stack[k] == 1) {
   p++;
} else {
  error++;
}
      }
    }
    fprintf(fp, "\n");
  }

  fprintf(stdout, "-2 sites: %ld\n1: sites: %ld\n", m, p);
  if (error != 0)
    fprintf(stdout, "%ld errors!\n", error);

  fclose(fp);
}


/* Standard_deviation()---------------------------------------------
   This code is taken from and modified from "Numerical Recipes in C"
```

```
 Second Edition by Press, Teukolsky, Vetterling, and Flannery.
 It will calculate the mean, and standard deviation. The mean is
passed back in the first parameter by reference and the deviation
is returned.
-------------------------------------------------------------------*/
double Standard_deviation(double& mean,      //Mean value of data
  double data[])    //Array of data
{
  int j;                       //loop counter

  double ep = 0.0,              //Storage info
    s = 0.0,                   //Storage info
    var;                       //Variance info

  if (RUNS <= 1)
    {
      cout << "Not enough runs for deviation info!\n";
      return 0.0;
    } //end if

  for (j = 1; j <= RUNS; j++)
    s += data[j-1];

  mean = s / (float)(RUNS);

  for (j = 1; j <= RUNS; j++)
    {
      s = fabs(data[j-1] - mean);
      ep = ep + s;
      var = var + (s*s);
    } //end for

  var = (var - ep * ep / RUNS)/(RUNS-1);

  return sqrt(var);

} //end Standard_deviation()
```

Appendix C

Ewald Summation Code

```
/* Ewald_sum25 ============================================
  This program will calculate the long range potential between
sites on a 3-D lattice of infinite width and finite height using
the method of the Ewald sum. It creates a 3-D table of 1/|r|
potentials from the origin to L/2 (due to pbc).
  Programmer: TJ Walls
------------------------------------------------------------
  Consider the Coulomb potential between a charged particle at the
origin and one at site i plus all the images of site i due to pbc.
Let p denote the vector connecting the two sites and R be the lattice
vector in the (001) plane of i:
V = sum_R{1/|r-R|} - sum_{R!=0}{1/|R|}
  where the latter accounts for a uniform background form the unit cell.
Using the Ewald method we can write the above equation as:
V = sum_R{f(r-R)} - [sum_{R!=0}{f(R)} - (1/z)*erf(sqrt(alpha)*z)]
  + sum_{k!=0}{g(k)*[exp(ik.r)-1]}
   +++ NOTE +++ (1/z)*erf(sqrt(alpha)*z) -> 2*sqrt(alpha/pi) as z -> 0
  where
    f(r) = 1/|r| * erfc(sqrt(alpha) * r)
    g(k) = 2*pi/[|k| * L^2 * sqrt(pi)] *
           int([x^(-1/2)*exp([-k^2*z^2]/[4*x] - x)],
                x = [k^2]/[4*alpha]..infinity)
  We have chosen alpha to be pi/[8*L^2]. The choice of alpha is NOT
arbitrary and must be made substantially larger to accommodate for
an expansion made in evaluating the g(k) integral.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  A few notes:
-This routine is only applicable in 2.5-D space and can NOT be
 used for space with integral dimensionality.
-Ceperley {PRB 18,3126 (1978)} derived general formulas for
 the Ewald method in integral dimensionality.
-It is a slight computational waste to loop through all sites
 and only perform
-This program must be accompanied by the file int_gk.c which
 contains code to perform the g(k) integral.
============================================================ */
#include <stdlib.h>
#include <math.h>
```

```
#include <stdio.h>

#define   L       8
#define   H       8
#define   PI      3.14159265359
#define   root2   1.41421356237
#define   rootpi  (sqrt(PI))
#define   unitk   (2.0 * PI / (double)L)
#define   CUTOFF  (pow(10,-15))
#define   ALPHA   (PI/(8.0*(double)(L*L)))
#define   root_A  (sqrt(ALPHA))

double Erfcc(double);
long   Check_pbc(long);
double Int_gk(double,long,double);

int main()
{
  double V_clmb[L/2+1][L/2+1][H/2+1];   // Map of Coulomb potentials
  double V_temp;                        // Temp storage for potential
  long kk;                              // nx^2 + ny^2
  long nx, ny;                          // k vectors for i space
  long pi, pj, pk;                      // Coords of p vector
  double x, y = 0.0;                    // Coords of (p-R) vector
  double r, r2;                         // Lengths of (p-R) vector
  double k_dot_r;                       // Value of k dot r vector
  double passk;                         // Length of k vector
  double max;                           // Current maximum value
  long max_r_terms;                     // max number of real space terms
  long max_i_terms;                     // max number of i space terms

  /* ------------------------------------------------------------
    First lets initialize the V array.
  ------------------------------------------------------------ */
  for (pi = 0; pi <= L/2; pi++)
    for (pj = 0; pj <= L/2; pj++)
      for (pk = 0; pk <= H/2; pk++)
V_clmb[pi][pj][pk] = -1.0;

  for (pi = 0; pi <= L/2; pi++)
    for (pj = pi; pj <= L/2; pj++)
      for (pk = 0; pk <= H/2; pk++) {
 /* ------------------------------------------------------------
    Now lets find our cutoff terms for the real and reciprocal
 space sums. We will drop all cell series which contribute a maximum
 of less than the cutoff value. We are going by full series,
 ie (-n->n, -n->n) inclusive.
 ------------------------------------------------------------ */
max = 1.0;
max_r_terms = 0;
while(max > CUTOFF) {
```

```
  max_r_terms++;
  x = (double)(pi+(max_r_terms*L));
  y = (double)(pj);
  r2 = (x*x)+(y*y)+(double)(pk*pk);
  r = sqrt(r2);
  max = Erfcc(root_A * r);
}

max = 1.0;
max_i_terms = 0;
while (fabs(max) > CUTOFF) {
  max_i_terms++;
  passk = max_i_terms*unitk;
  max = Int_gk(passk,pk,ALPHA);
}

  /* Background Potential --------------------------
    Now lets compute the uniform background potential
  for each site, including the subtraction of the
  over counting term in the reciprocal sum.
  ------------------------------------------------*/
V_temp = 0.0;
for (nx = -1*max_r_terms; nx <= max_r_terms; nx++)
  for (ny = -1*max_r_terms; ny <= max_r_terms; ny++) {
    if (nx != 0 || ny != 0) {
      r2 = (double)((nx*nx)+(ny*ny));
      r = sqrt((r2*(double)(L*L))+(double)(pk*pk));
      V_temp += 1/r*Erfcc(r * root_A);
    }
  }

if (pk == 0)
  V_temp -= 2.0 * root_A / rootpi;
else
  V_temp -= (1.0 - Erfcc(root_A * (double)pk)) / (double)pk;

V_temp = -1.0*V_temp;
  /* ------------------------------------------------
    Now we do the f(|r-R|) sum.
  ------------------------------------------------ */
for (nx = -1*max_r_terms; nx <= max_r_terms; nx++) {
  for (ny = -1*max_r_terms; ny <= max_r_terms; ny++) {
    x = (double)(pi+nx*L);
    y = (double)(pj+ny*L);
    r2 = (x*x)+(y*y)+(double)(pk*pk);
    r = sqrt(r2);
    V_temp = V_temp + Erfcc(r * root_A) / r;
  } // end for
} // end for
 /* ------------------------------------------------
     Now we'll do the sum k!=0: g(|k|)*(exp(ikr)-1)
```

```
  Notice that since we are only considering full series of
  cells that the imaginary parts of exp(ik.r) will cancel
  and only the cos() terms will remain if we sum over the
  positive half of the cell and multiply by 2.
  ------------------------------------------------- */
for (nx = 0; nx <= max_i_terms; nx++) {
  if (nx == 0)
    ny = 1;
  else
    ny = -1*max_i_terms;
  for (; ny <= max_i_terms; ny++) {
    k_dot_r = (double)((ny*pi)+(nx*pj))*unitk;
    passk = sqrt((nx*nx)+(ny*ny))*unitk;
    kk = (nx*nx)+(ny*ny);
    V_temp = V_temp + (Int_gk(passk, pk, ALPHA)*(2.0*(cos(k_dot_r)-1)))
                        / (double)(rootpi * L * sqrt((double)kk));
  } //end for(ny < max)
} //end for(nx < max)

V_clmb[pi][pj][pk] = V_temp;
V_clmb[pj][pi][pk] = V_temp;
printf("(%ld,%ld,%ld)\t%9.12f\n", pi, pj,pk, V_temp);
      } //end for

  return 0;
} //end main

/* Erfcc() -------------------------------------------------
  Complimentary error function (1-erf)
  This is taken largely from Numerical recipes, with slight
 modification by Prof. Zhang
 The fractional error is claimed to be less than 1.2x10^(-7)
 EVERYWHERE
-----------------------------------------------------------*/
double Erfcc(double x)
{
  double z, t;
  double ans = 0.0;

  z = fabs(x);
  t = 1.0 / (1.0 + (z / 2.0));

  ans = t * exp((-1*z)*z-1.26551223+t*(1.00002368+t*(.37409196+t
               *(.09678418+t*(-.18628806+t*(.27886807+t*(-1.13520398
               +t*(1.48851587+t*(-.82215223+t*.17087277)))))))));
  if (ans < 0.0)
    ans = 2.0 - ans;

  return ans;
} // end Erfcc()
```

```
/* Check_pbc() ------------------------------------------------
   This function will take the difference between 2 points as
an argument and if that distance is greater than L/2 (therefore
overlapping in the pbc) converts and returns that value as the shortest
possible distance in 1 dimension.
------------------------------------------------------------ */
long Check_pbc(long dist)
{
  long temp;
  if (dist < 0)
    dist = -1*dist;

  temp = dist % L;
  if (temp > L/2)
    return (L - temp);
  else
    return temp;
} //end Check_pbc
```

Bibliography

[1] A. Levi and M. Kortla, Journal of Physics: Condensed Matter **9**, 299 (1997).

[2] K. Binder and D. Heermann *Monte Carlo Simulation in Statistical Physics: an introduction*. Springer-Verlag, Berlin, 1992.

[3] M. Kalos and P. Whitlock *Monte Carlo Methods*. J Wiley and Sons, New York, 1986.

[4] S. Park and T. Shrout, Journal of Applied Physics **82**, 1804 (1997).

[5] N. Giordano *Computational Physics*. Prentice Hall, New Jersey, 1997.

[6] S. Zhang and H. Krakauer, Proposal to ONR

[7] A. Bortz, M. Kalos and J. Lebowitz, Journal of Computational Physics **17**, 10 (1975) .

[8] S. Park and K. Miller, Communications of the ACM **31**, 1192 (1988).

[9] S. Park and L. Leemis *Discrete-Event Simulation: A First Course*. pre-print edition 1998.

[10] L. Mikheev and A. Chernov, Journal of Crystal Growth **112**, 591 (1991).

[11] G. Gilmer and P. Bennema, Journal of Crystal Growth **13/14**, 148 (1972).

[12] G. Gilmer and P. Bennema, Journal of Applied Physics **40**, 1347 (1972).

[13] Y. Saito *Statistical Physics of Crystal Growth*. World Scientific, New Jersey, 1996.

[14] L. Landau and E. Lifshitz *Statistical Physics – Part 1*. Pergamon Press, New York, 1980.

[15] J. Weeks and G. Gilmer, Advances in Chemical Physics **40**, 157 (1979).

[16] H. Leamy, G. Gilmer and K. Jackson in J. Blakeley ed., *Surface Physics of Materials I*. Academic Press, New York, 1975.

[17] H. Muller-Krumbhaar in K. Binder ed., *Monte Carlo Methods in Statistical Physics*. Springer, Berlin, 1986.

[18] R. Waser and D. Smyth in C. Araujo ed., *Ferroelectric Thin Films: Synthesis and Basic Properties*. Gordon and Breach, New Jersey, 1996.

[19] L. Bellaiche and D. Vanderbilt, Physical Review Lettters **81**, 1318 (1998).

[20] B. Nijober and F. de Wette, Physica **23**, 309 (1957).

[21] D. Ceperley, Physical Review B **18**, 3126 (1978).

[22] P. Ewald, Annals de Physik **64**, 253 (1921).

[23] N. Setter and L. Cross, Journal of Applied Physics **51**, 4356 (1980).

[24] B. Burton and R. Cohen, Physical Review B **52**, 792 (1995).

[25] M. Allen and D. Tildesley *Computer Simulations of Liquids*. Oxford Science, New York, 1987.

[26] C. Bender and S. Orszag *Advanced Mathematical Methods for Scientists and Engineers.* McGraw-Hill, New York, 1978.

[27] S. Zhang, private communication.

[28] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery *Numerical Recipes in C, 2nd ed.* Cambridge Press, New York, 1992.