8-2014

Multistep Kinetic Monte Carlo

Holly Nichole Johnson Clark
University of Tennessee - Knoxville, hclark4@vols.utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Holly Nichole Johnson Clark entitled "Multistep Kinetic Monte Carlo." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Mathematics.

Tim P. Schulze, Major Professor

We have read this dissertation and recommend its acceptance:

Ohannes Karakashian, Steven Wise, Yanfei Gao

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Multistep Kinetic Monte Carlo

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Holly Nichole Johnson Clark

August 2014

Dedication

I would first like to dedicate this work to God. Without His guidance, love, and mercy in my life, I would have never come this far or completed this document. Secondly, I dedicate this work to the many family members and friends who have supported and encouraged me during my studies. I love you all! Finally, I would like to dedicate this work, this document, and the fruits which will come from all of this, to my son Daniel. He is my life and joy on this earth, and I would have given up long ago, without his precious smile, hugs, and kisses to remind me why I work so hard. I love you, Daniel!

Abstract

Kinetic Monte Carlo (KMC) uses random numbers to simulate the time evolution of processes with well-defined rates. We analyze a multi-step KMC algorithm aimed at speeding up the single-step procedure and apply the algorithm to study a model for the growth of a surface dendrite. The growth of the dendrite is initiated when atoms diffusing on a substrate cluster due to lower hopping rates for highly coordinated atoms. The boundary of the cluster is morphologically unstable when the flux of new atoms is supplied in the far field, a scenario that could be generated by masking a portion of a substrate that is subject to some kind of deposition process. We allow atoms far from the growing dendrite to take large hops while atoms near the dendrite follow a usual single-step KMC algorithm. We study how coarse-graining affects the distribution of waiting times for hops, and how to accurately couple the multi-step and single-step regions.

Table of Contents

1 Introduction **1**

1.1 Modeling the Process of the Solidification of a Dendrite 1

1.1.1 Thermodynamics Background 2

1.1.2 Solidification in One Dimension 4

1.2 Introduction to Monte Carlo 6

1.2.1 Monte Carlo Methods 6

1.2.2 Diffusion Monte Carlo 8

1.2.3 Kinetic Monte Carlo 8

1.3 One-Dimentional KMC Solidification Model 12

2 Multistep Kinetic Monte Carlo Method in Two-Dimensions **16**

2.1 Previous Work Simulating Dendrites in Two-Dimensions using Multistep or Large Step Methods 17

2.1.1 DMC with Phase-Field Model 17

2.1.2 Multiscale Kinetic Monte Carlo 19

2.2 New 2D Model 21

2.2.1 Gaussian Diffusion in the Far Field 24

2.2.2 Multistep Random Walks in the Far Field 25

2.2.3 Probability Distributions for Event Waiting Times 27

2.2.4 How the Far Field and the KMC Domains Interact: Merging Clocks 30

3 Simulations and Results **36**

3.1 Qualitative Results 36
3.2 Simulation Speed 47

4 Summary and Future Directions **50**

Bibliography **51**

Appendices **55**

A Other Approximations of Multistep Waiting Times **56**
A.1 No Decrease in Step Size Near the KMC Boundary 56
A.2 Approximate Gamma Distribution with Scaled Exponential Distribution . . 59
A.3 Approximate Gamma Distribution with Normal Distribution for Larger Step Sizes 63
A.4 A Note on Random Number Generation Computational Times 66

Vita **67**

List of Figures

1.1 Actual 1D solution compared to approximations 14

2.1 Inner and Outer Regions 23
2.2 Gaussian Random Walker Density 26
2.3 Examples of Multistep Probability Distributions 32
2.4 Snapshot while clocks are not merged 34
2.5 Snapshot after merging clocks 35

3.1 An Example of the Scatter Regime 39
3.2 An example of the equilibrium regime 40
3.3 An example of the perturbation regime 41
3.4 An example of the dendrite regime 42
3.5 A great example of a two-dimensional dendrite 43
3.6 An example of the bordered dendrite regime 45
3.7 An example of KMC trying to do what Multistep KMC can do 46
3.8 Regimes 47
3.9 KMC and multistep size versus computing time 48

A.1 Computation time using a simulation that did not decrease step size near the KMC boundary 57
A.2 Simulated dendrite when step size did not decrease near the KMC boundary 58
A.3 Simulated dendrite using actual method 58
A.4 Gamma distribution with various shape numbers 60

A.5 Computation time using a simulation that used a scaled exponential distribution instead of a gamma distribution 61
A.6 Simulated dendrite using a simulation that used a scaled exponential distribution instead of a gamma distribution 62
A.7 Simulated dendrite using actual method 62
A.8 Computation time for a simulation using the normal distribution as an approximation to the gamma distribution 64
A.9 Simulated dendrite using a simulation using the normal distribution as an approximation to the gamma distribution 65
A.10 Simulated dendrite using actual method 65
A.11 Computational time comparison 66

Chapter 1

Introduction

1.1 Modeling the Process of the Solidification of a Dendrite

Broadly speaking, Kinetic Monte Carlo (KMC) is a method for simulating inhomogeneous random walks. The technique originated from attempts to simulate crystal growth processes, and is especially well suited to that application. In particular, the research presented here is largely motivated by simulations of dendritic growth, like that presented in Schulze 2008 (Schulze, 2008b). During these simulations, and many like them, one typically encounters an interfacial region where the topography and transition rates vary greatly from one lattice site to the next. The interface often separates bulk regions where the diffusion process is essentially homogenous. If one applies the same KMC method to these regions, it becomes very inefficient due to the large number of steps a random walker must take to travel from the far field to the interface. A number of ways around this have been attempted, many of which seek to couple KMC to a continuum diffusion model. Examples can be found in Boettinger et al., 2002, Gibou et al., 2003, Shishkina and Wagner, 2006, and Plapp and Karma, 2000. An alternative, explored here, is to use coarse grained random walks.

In this first chapter, we provide some background information on crystal growth processes and Monte Carlo methods. We briefly explore a one dimensional toy problem as a way of illustrating how KMC can be used for more general solidification problems. Our principal

topic, however, shall concern the coupling of a coarse grained random walk simulation—referred to as Diffusion Monte Carlo—with a more detailed KMC model in the vicinity of an interace. We will develop, implement and evaluate this technique in the context of a prototype problem featuring the growth and instability of two dimensional crystal islands that are growing on a crystalline substrate.

1.1.1 Thermodynamics Background

Let us start by motivating and defining some useful thermodynamic constants, notations, and governing equations that we will be refering to and using. These can be found in most introductory thermodynamics books such as *Introduction to Chemical Engineering Thermodynamics* (Smith et al., 2001).

Suppose we are interested in heating a material that is in its solid phase. The amount of heat required to change a unit mass of that solid material into its liquid form at the temperature $T = T_m$, where T_m is the melting temperature of the material, is called the *latent heat* L. This material will also have a *density* equal to its mass per unit volume, which can be different in the solid and liquid phases. We will use the notations ρ^s and ρ^ℓ to represent the densities in the solid phase and liquid phase, respectively. For convenience we will assume that $\rho^s = \rho^\ell = \rho$. Now, combining the concepts of latent heat and density we can define *latent heat per unit volume* which is given by the notation L_ν and is defined to be $L_\nu = \rho L$.

Depending on the material, changing the temperature of the material could take differing amounts of energy (or heat). To quantify this, we start by defining the amount of heat required to raise, without change of phase, the temperature of a unit mass of solid or liquid by $1°C$ to be the *specific heat*, denoted c_P, where the subscript of P implies this is happening at a constant pressure. (Note that we could also define c_V, where the subscript of V would imply a constant volume.)

Another important relationship from thermodynamics that we will need to reference is Fourier's Law of Heat Conduction, which states that the heat flux through a medium is proportional to the negative gradient of the temperature field. Symbolically, if we let $\mathbf{q}$ be

the heat flux. Then

$$\mathbf{q} = -k\nabla T, \tag{1.1}$$

where k is the *thermal conductivity*, which quantifies the material's ability to conduct heat. For example, in one dimension Fourier's Law of Heat Conduction takes the form of

$$q_x = -k\frac{dT}{dx}. \tag{1.2}$$

It is worth noting that k can be different in the solid and liquid phases, so we will denote these different thermal conductivities as k^s and k^ℓ, respectively.

Now that we have defined important concepts and set notations for inportant quantities, we can express the governing equations for solidifing and melting. The temperature field, or thermal diffusion field, is governed by the heat equation

$$\frac{\partial T}{\partial t} = D\Delta T, \tag{1.3}$$

where $D = \dfrac{k}{\rho c_p}$ is called the *thermal diffusivity*. In a material with a large thermal diffusivity, heat travels fast since the substance conducts heat quickly relative to its heat capacity.

At the solid-liquid interface we have the Stefan condition,

$$\rho^s L V_n = (k^s \nabla T^s - k^\ell \nabla T^\ell) \cdot \mathbf{n}, \tag{1.4}$$

where V_n is the speed of the liquid-solid front, and $\mathbf{n}$ is a normal vector to the liquid-solid front. The Stefan condition insures that energy is conserved by forcing the velocity of the liquid-solid front to be determined by the change in heat flux at the interface.

At the interface of the solid and liquid regions we have $T_s = T_l$, and we also must apply the Gibbs-Thomson equation which takes interfacial effects into account. The equation is given by

$$T^I = T_m \left[1 + 2H\frac{\gamma}{L_\nu}\right], \tag{1.5}$$

where T^I is the interface temperature, T_m is the equilibrium melting temperature for a flat

surface, γ is the surface energy, L_ν is latent heat per unit volume, and $2H = -\nabla \cdot \mathbf{n}$ is the mean curvature of the interface with normal $\mathbf{n}$. Note that by convention, $2H$ should be positive if the surface curves in the direction of the normal. In one dimension this reduces to $T^I = T_m$ becuase $-\nabla \cdot \mathbf{n} = 0$ (since 0 is the derivative of either 1 or -1, where the sign denotes direction). In two dimensions, if the normal can be described by a function of one variable $h(x)$, then the equation is

$$T^I = T_m \left[1 + \frac{h_{xx}}{(1 + h_x^2)^{3/2}} \frac{\gamma}{L_\nu} \right]. \tag{1.6}$$

Finally, in three dimensions, if the normal can be described by a function of two variables $h(x, y)$, then the equation has the form

$$T^I = T_m \left[1 + \frac{\left(1 + (h_x)^2\right) h_{yy} - 2h_x h_y h_{xy} + \left(1 + (h_y)^2\right) h_{xx}}{\left(1 + (h_x)^2 + (h_y)^2\right)^{3/2}} \frac{\gamma}{L_\nu} \right]. \tag{1.7}$$

1.1.2 Solidification in One Dimension

We would now like to apply these thermodynamics concepts and equations to a simple example in one dimension for two reasons. First, so the reader is comfortable with the basic ideas, and second, so we can use this example that has an analytic solution as a benchmark later in this chapter.

We consider one dimensional solidification freezing upward from a cold boundary. We consider this taking place on the half line $(0, \infty)$, and the governing equations become the following:

$$T_t = DT_{zz} \quad \text{in} \quad z < h(t), \tag{1.8}$$

$$\rho^s L \dot{h} = (k^s \nabla T^s - k^\ell \nabla T^\ell) \quad \text{and} \quad T^s = T^\ell = T_m \quad \text{at} \quad z = h, \tag{1.9}$$

and

$$T^\ell = T_m \quad \text{in} \quad z > h(t), \tag{1.10}$$

which correspond to solving the diffusion equation in the solid region (1.8), applying the

Stefan condition and the Gibbs-Thomson equation at the liquid-solid interface (1.9), and fixing the temperature in the liquid region (1.10).

Initially we assume that the entire bulk is at the melting temperature which gives the initial condition,

$$T(z,0) = T_m \tag{1.11}$$

.

Then, finally, the boundary condition we will assume for solidification from a cold boundary is (Davis, 2001)

$$T(0,t) = T_B < T_m, \tag{1.12}$$

which implies a constant temperature, below the melting temperature, at the bottom boundary.

If we also assume that the thermal conductivity is the same in the solid and the liquid, i.e. $k^s = k^\ell = k$, and define $\Lambda = \dfrac{\rho^s L}{k}$, then at $z = h$ we have we have $\Lambda \dot{h} = T_z^s - T_z^\ell = T_z^s$. Note that $T_z^\ell = 0$ since the temperature in the liquid is remaining constant at T_m.

Now, we seek a similarity solution of the partial differential equation (1.8) by letting $\eta = \dfrac{z}{2\sqrt{Dt}}$. This reduces the problem to the ordinary differential equation

$$T_{\eta\eta} + 2\eta T_\eta = 0, \tag{1.13}$$

and this equation has the well-known solution (Holmes, 2009)

$$T(\eta) = A + B\,\mathrm{erf}(\eta), \tag{1.14}$$

where $\mathrm{erf}(z) \equiv \dfrac{2}{\sqrt{\pi}} \displaystyle\int_0^z e^{-s^2} ds$. Thus, $T(z,t) = A + B\,\mathrm{erf}\left(\dfrac{z}{2\sqrt{Dt}}\right)$ is a family of solutions to $T_t = DT_{zz}$.

Now, using the boundary conditions (bottom boundary and liquid-solid interface) we can solve for A

$$T_B = T(0,t) = A + B\,\mathrm{erf}(0) = A, \tag{1.15}$$

and find an equation for B,

$$T_m = T(h,t) = T_B + B\,\mathrm{erf}\left(\frac{h(t)}{2\sqrt{Dt}}\right). \tag{1.16}$$

To solve for B, we suppose that $h(t) = 2C\sqrt{Dt}$ for some constant C to be determined so that $B = \dfrac{T_m - T_B}{\mathrm{erf}(C)}$. Using the boundary condition $\Lambda\dot{h} = T_z^s$ and some algebra we get a transcendental equation for C

$$\sqrt{\pi}Ce^{C^2}\,\mathrm{erf}(C) = \frac{T_m - T_B}{D\Lambda}. \tag{1.17}$$

Note that we can solve for C numerically given the values of all the other constants, so we have found a final solution for the temperature

$$T(z,t) = T_B + \frac{T_m - T_B}{\mathrm{erf}(C)}\,\mathrm{erf}\left(\frac{z}{2\sqrt{Dt}}\right), \tag{1.18}$$

and for the height

$$h(t) = 2C\sqrt{Dt}. \tag{1.19}$$

1.2 Introduction to Monte Carlo

In order to introduce a model that will ultimately be used to solve the problem of interest in Chapter 2, we leave solidification completely (planning to return in section 1.3) to look at Monte Carlo methods.

1.2.1 Monte Carlo Methods

Shonkwiler and Mendivil say of Monte Carlo methods in the first sentence of their book *Explorations of Monte Carlo Methods* (Shonkwiler and Mendivil, 2009), "The Monte Carlo method is a technique for analyzing phenomena by means of computer algorithms that employ, in an essential way, the generation of random numbers." Since these methods explicity involve computers, Monte Carlo methods were developed alongside the development of computers. Over the years Monte Carlo methods have been used to solve many problems

(Doucet et al., 2001). For some of these, better methods were eventually found and Monte Carlo was surpassed, but in many applications Monte Carlo is still the method of choice.

The Monte Carlo method was given its name by Stanislaw Ulam and John von Neumann, during research at Los Alamos in the 1940s (Shonkwiler and Mendivil, 2009). Since then, the method has found uses in mathematics, various sciences, finance, etc. Uses have even been found in the sports entertainment industry because the method can be applied to make predictions about virtually anything having some random component. These applications which contain a random component are referred to as stochastic processes.

While the method became useful with the computer, many give credit for the first "Monte Carlo method" to Comte de Buffon in 1733 (Shonkwiler and Mendivil, 2009). He invented an experiment with the purpose of estimating π by simply using a needle and a lined (ruled) surface. The "Buffon Needle Problem" has been analyzed and written about by many, so the reader could reference Shonkwiler's and Mendivil's book to find out more. However, one aspect worth noting is that when this method is implemented, one finds that "the accuracy of the estimate only improves in proportion to $1/\sqrt{n}$, where n is the number of tosses." (Shonkwiler and Mendivil, 2009) Therefore, to get any accuracy out of this first Monte Carlo method, one needs to toss the needle many times. So many times, in fact, that the method should be considered to be more of a thought experiment in Monte Carlo methods than a good way to estimate π (unless you employ a computer, of course).

One can also use a Monte Carlo method to estimate a definite integral. This is simply called Monte Carlo Integration. To estimate $I = \int_a^b f(x)dx$, let us first recall Riemann sum approximations for integrals. While more recognized Riemann sum approximations would partition the x-axis into equal subintervals and use the function value at either left endpoints, right endpoints, or midpoints to approximate the function value in an interval, Monte Carlo Integration uses random x values and their corresponding function values. We use a sample of K values (K a positive integer), called x_i $(i = 1, \ldots, \mathrm{K})$, from a uniform distribution over $[a, b]$ and estimate $I \approx \frac{b-a}{K} \sum_{i=1}^{K} f(x_i)$.

From its "computer-aged"' beginning, the first, and arguably most difficult, aspect of all Monte Carlo Methods is finding a sufficient way for computers to generate the required

random numbers. What requirements should we expect from a Random Number Generator? According to Shonkwiler and Mendivil, there are five qualities we look for. We want it to

1) be fast (many random numbers will be needed),

2) be repeatable (for the sake of confirming accurate programming),

3) lend itself to analysis (to ensure we are getting the probability distribution we expect),

4) have a long period (i.e., doesn't repeat over short periods), and

5) be apparently random (obviously the point of a generator of "random numbers").

Now, most computer programming languages have built-in, pseudorandom number generators which meet all the desired requirements to a very high standard (Shonkwiler and Mendivil, 2009).

1.2.2 Diffusion Monte Carlo

One widely used Monte Carlo method is known as Diffusion Monte Carlo (DMC). This method is often used in chemistry for modeling the diffusion behavior of atoms or molecules by comparing the rates of individual steps (random movements of atoms or molecules, usually caused by bouncing off of each other) with random numbers (Holmes, 2009). One can prove that an atom or molecule's random movements satisfy the diffusion equation if we carefully take a continuum limit, moving from discrete jumps to continuous movements, while assuming that space is *infinite* (Haji-Sheikh and Sparrow, 1966). Diffusion Monte Carlo is mentioned here without detail, as we will go as in-depth as necessary, using a specific case in Chapter 2.

1.2.3 Kinetic Monte Carlo

In contrast to Diffusion Monte Carlo and other Monte Carlo methods, Kinetic Monte Carlo (KMC), will be more central to what is outlined in Chapter 2. In light of this, more detail of the method in general will be given here (Schulze, 2008a), in order to ensure clarity later, when outlining the complete model and algorithm we create and analyze.

Kinetic Monte Carlo (or KMC) is freqently used refering to simulating crystal growth and evolution via stochastic simulation, where new atoms are added or existing atoms hop on a lattice (Voter, 2007). These events occur with particular rates which are affected by configuation of the growing crystal locally. A simulation of crystal growth using KMC acts as a numerical method to approximate the solution of partial differential equations when a more microscopic scale (atomistic) is desired. In particular, for the diffusion equation, KMC can simulate the random movements which occur in a diffusion process. We will now retreat from speaking of the application to crystal growth, and focus on KMC itself as a numerical algorithm that, with each simulation, gives one possible outcome from a Markov process.

Kinetic Monte Carlo models are usually Markov processes with discrete space, but continuous time, in which the system changes through a sequence of states, say $x_{t_k} \in X$, that come from the state-space X at particular times, which we will denote $t_0 < t_1 < \ldots t_k < \ldots$. These t_k's correspond exactly to times when the state (configuration) changes in some way. The sequence $\{x_k\}$ can also be viewed as a Markov chain which is modelling an inhomogeneous Poisson process with rate $Q(t)$. The time between events in a Poisson process generates a sequence of waiting times, $\{\Delta t_k\}$, which are the intervals between changes to the system state. One KMC simulation will yeild one sample of these two sequences, $\{x_k\}$ and $\{\Delta t_k\}$, from the set of all possible outcomes. However, KMC needs to be efficient, so want to be sure the simulation runs as fast as possible.

The Poisson process with rate $Q(t)$ uses the Poisson distribution, given by the probability density curve

$$f(x) = \frac{\lambda^x e^{-\lambda}}{x!}, \tag{1.20}$$

which expresses the probability that *x number of events occur in a fixed period of time* when these events occur with a known average rate, λ, and independently of the time since the last event (Parzen, 1965). In KMC simulations the "known average rate" changes after every event, so the parameter $\lambda = Q(t_k)$ at the kth step. However, during KMC simulations, it is necessary that we know the probability distribution for the waiting time between two events of a Poisson process. Let T denote the waiting time until the first event occurs, and

G(t) deonte the cummulative probabilitiy distribution of waiting times, t. Since the waiting time is nonnegative, $G(t) = 0$ when $t < 0$. For $t > 0$,

$$G(t) = P(T \leq t) = 1 - P(T > t) = 1 - P(\text{no change occurs in [0,t]}). \tag{1.21}$$

Now, the probability that no change occurs in a *unit* interval of time is given by $f(0) = \frac{\lambda^0 e^{-\lambda}}{0!} = e^{-\lambda}$. But if the mean number of events per unit interval is λ, then the mean number of events in an interval of length t is λt. We now have

$$G(t) = 1 - e^{-\lambda t}, \tag{1.22}$$

and taking the derivative of the cummulative probability distribution G, gives g, the probability distribution,

$$g(t) = G'(t) = \lambda e^{-\lambda t}. \tag{1.23}$$

This implies that each number in the squence $\{\Delta t_k\}$ comes from the exponential distribution, where on average the waiting time for one event is $1/\lambda$ (Parzen, 1965). Again, here we take $\lambda = Q(t_k)$ at the kth step.

An example for the state-space in two dimensions would be an array

$$A = \{A(i,j) | A(i,j) = 1 \text{ or } A(i,j) = 0\}, \tag{1.24}$$

where the one and zero would indicate something about the state at location (i,j). This array will change with each state transition, so the sequence $\{A_{j_k}\}$ corresponds to the sequence $\{x_k\}$ above. If we are calling the state-space at time t_k, A_{j_k}, then we could also think about a corresponding array $q_{t_k}(j,l)$, which consists of numbers greater than or equal to zero, corresponding to the rates (predicted by a model) for the system to change from state A_k to another state-space A_l. Let us call the set of all possible states X, so that A_k and A_l are just two elements of X. We assume the set of all possible states which can be

attained are independent, so that these rates can be combined into one rate

$$Q = \sum_{A_l \in X} q_{t_k}(j,l), \tag{1.25}$$

for the whole system at the time step t_k. The rates $q_{t_k}(j,l)$ may or may not depend on time, but when they do, the Markov process is changing at each time step. However, instead of thinking in terms of rates, we could equivalently look at probabilities for state transitions,

$$p_{t_k}(j,l) = q_{t_k}(j,l)/Q, \tag{1.26}$$

which imply

$$\sum_{A_l \in X} p_{t_k}(j,l) = 1. \tag{1.27}$$

We also can take $q_{t_k}(j,j) = p_{t_k}(j,j) = 0$ since a transition from a state onto itself would imply that the probability of any spacial aspect of the state changing at that time step is 0.

In Kinetic Monte Carlo methods, the array $q_{t_k}(j,l)$ is usually sparse, since this lists the probability of getting to *any* state from state j, and most states are impossible to realize in only one time step. The consequence of this is that at a given time-step, we can look at a much smaller list, the set of events where the transition rate away from present state is greater than zero. In light of this, while events can be indentified by (j,l), meaning changing from a state j to a state l, it will be easier to considere only the states that could be accessed in one time-step and these could be counted and enumerated as $\{X_n \in X\}_{n=1}^{N}$. This has the added benefit of also corresponding to the rate list $\{q_n\}_{n=1}^{N}$ at the given time-step. We would, of course, expect this list to change at each different time-step.

We want the algorithm to be efficient, so it is necessary that we limit possible events. This model's possible events are labeled so that

1) the subset of rates that require changing after any given event is minimal, and

2) the event labels should also indicate to us where these "neighboring" events are located.

From now on, we shall assume the existence of a map $\{n \leftrightarrow X_n\}$ between event numbers and the events themselves, so we can refer to the n^{th} event as simply event n.

For KMC to perform all tasks involved in calculating each state sequence $\{x_k\}$, we must sample from the distibution of $\{q_n\}$ and then update all necessary data structures that change from the previous state to the new state when the chosen event occurs. All these tasks are then repeated every time step. Note that the step of updating data structures includes updating our list of rates $\{q_n\}$ and the state $\{x_k\}$. Since we will be looking at a system with a large number of events N and with a sequence of time steps having cardinality much greater than N, we need KMC to be performed via a fast algorithm. Since our rates are affected using some form of nearest-neighbor model, the number of rates, K, affected by any specific transition is much less than the total number of events N. This speeds up the algorithm by allowing us to only need to update locally at each time step. We also must quickly generate the sequence of waiting times $\{\Delta t_k\}$. To do this a random number, v is generated from the uniform distribution on $(0, 1]$ which is used to obtain an exponential random variable via $\Delta t_k = (-1/Q)ln(v)$. This formula comes from the idea of inverse transform sampling (Devroye, 1986). This method for sampling the exponential distribution requires first that we generate a uniform random number $v \in (0, 1]$ as stated above. Then note that $u\lambda \in (0, \lambda]$. Now choose the waiting time that corresponds to event $u\lambda$, i.e., solve $u\lambda = \lambda e^{-\lambda t}$ for t. This gives the time step $\Delta t = -\frac{\log u}{Q}$, where $1/\lambda = 1/Q$ is the mean waiting time. A KMC algorithm that does not need a precise time calculation could instead simply keep track of the number of iterations or move time forward using the expected waiting time $1/Q$ at each step.

Most KMC algortihms consists of two main steps: selecting an event and updating the data structures. There are many ways to algorithmically perform these steps, and the algorithm we choose for our purpose will be explained in detail in Chapter 2.

1.3 One-Dimentional KMC Solidification Model

The ultimate goal of this research is find a way to handle simulations in 3D, motivated by Schulze, 2008b. Therefore, before concluding this introductory chapter, we would like

to show how KMC can be used to solve a simple, very artificial, solidification problem in one dimension, specifically the problem described in Section 1.1.2. Recall, this problem has an analytical solution, so we will also show how the KMC algorithm compares to the real solution for the height, $h(t)$.

In order to implement a KMC algorithm to help solve this prolbem, we first divide the z-axis into unit segments, where each unit interval represents one atom. we begin with a couple solid atoms, say $z \in (0, 1]$ is one solid atom and $z \in (1, 2]$ is another, and let the rest of the positive$z$-axis represent liquid atoms. The only event we will consider is that a liquid atom attaches to the highest solid atom and becomes solid itself, since surface hopping can only occur in two or more dimensions. The rate at which this occurs can be determined by the Stefan condition, $\mathbf{R} = \dot{h} = C(T_z^s - T_z^\ell)$. Since we only consider this **one event**, we must only calculate the time between successive solidification events, and then step forward the heat equation in the liquid and/or solid region(s) between events, we calculate the time step $\Delta t = -\frac{\log u}{R_N}$, as explained in the previous section, where $1/\lambda = 1/R_N$ is the mean waiting time. In numerical simulations on this one dimensional system, we used a simple forward-time centered-difference scheme, which in one dimension is written as

$$\frac{T_j^{n+1} - T_j^n}{\Delta t} = \frac{T_{j+1}^n - 2T_j^n + T_{j-1}^n}{(\Delta x)^2} \tag{1.28}$$

to update the temperature field between events. As you can see in Figure 1.1, using KMC along with this numerical solver for the diffusion equation to obtain the temperature gives solutions for $h(t)$, the height of solid atoms, which agree with the numerical solution.

One should now note that this particular incarnation of a KMC algorithm spends the majority of its computing time solving the heat equation. This usage of computer time only compounds in higher dimensions, so it is clear that if speed is an issue, another (faster) method should be implimented for determining the temperature field, or (as we do) look at a particular problem where temperature is not the driving force. Thus, we created and analyzed a hybrid method, in two dimensions, of Diffusion Monte Carlo and Kinetic Monte Carlo which will not only simulate a growing dendrite which is similar to other methods,

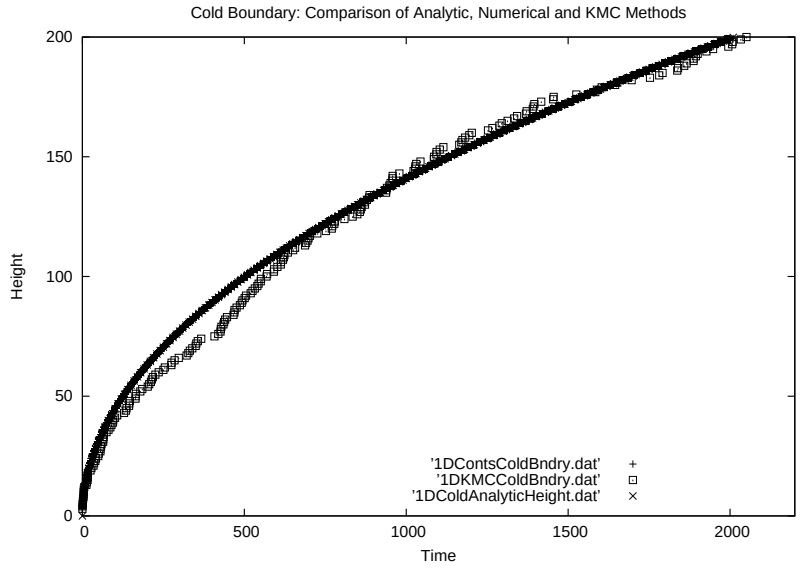

Figure 1.1: This graph compares the analytical solution, numerical solution, and KMC solution for the height of the one dimensional dendrite over time.

such as those in (Plapp and Karma, 2000) and (DeVita et al., 2005), but will also perform more quickly by eliminating the need to know a temperature field. The method that will be descibed will also allow the computational domain to be larger than what can be done (in a reasonable amount of time) with a pure KMC algorithm.

Chapter 2

Multistep Kinetic Monte Carlo Method in Two-Dimensions

In this chapter we develop a method in two-dimensions to simulate the growth of a surface dendrite. This growth is initiated when atoms diffusing on a substrate cluster due to lower hopping rates for highly coordinated atoms. The boundary of this cluster is morphologically unstable when the flux of new atoms is supplied in the far field. For our two-dimensional model, we supply these far field atoms by masking a portion of a substrate that is subject to some kind of deposition process, so that atoms can only be added to the system far from the growing cluster.

Since we are driving the growth of the cluster of atoms in this manner, our two-dimensional model we develop here is an epitaxy model. Epitaxy refers to the growth of crystals of one substance on a substrate of another substance. These atoms that are being added (for the growth of crystals) are called adatoms. In our model, we are simply covering the substate so that adatoms can only appear in the far field. Our model differs from most epitaxy models, in that they are usually used to simulate the process *growing* crystals *up* from a substrate, while our model with adatoms only supplied in the far-field, never allows the atoms to *stack* on top of each other.

2.1 Previous Work Simulating Dendrites in Two-Dimensions using Multistep or Large Step Methods

Much work has already been done with simulating a growing dendrite in two-dimensions (Chatterjee and Vlachos, 2007; Zheng et al., 2008). However, two works, (Plapp and Karma, 2000) and (DeVita et al., 2005), were essential in the forming of this new method, and we wish to detail some of their important results.

2.1.1 DMC with Phase-Field Model

The first work that we wish to detail is *Multiscale Finite-Difference-Diffusion-Monte-Carlo Method for Simulating Dendritic Solidification* (Plapp and Karma, 2000). Their model consisted of treating the interface dynamics using deterministic equations of motion, while the large-scale diffusion field is represented by random walkers and is evolved using Diffusion Monte Carlo. The deterministic model Karma and Plapp used was the Phase-Field Model. These equations can be integrated on a regular grid on the scale of the dendrite, without knowing explicitly where the solid-liquid interface is located. The interaction between the two regions takes place in a buffer layer. They start with a single nucleus and simulate dendrite solidification of a pure substance from an undercooled melt.

Their goal was to combine the phase-field method with the efficiency of a Diffusion Monte Carlo treatment for the diffusion field. This is achieved by dividing the simulation domain into an *inner* and *outer* region. In the inner region, consisting of the growing structure, and a thin *buffer layer* of liquid, the phase-field equations are integrated, while in the outer region, the diffusion field is represented by an ensemble of random walkers. Walkers are created and absorbed at the boundary between the inner and outer domains at a rate proportional to the local diffusion flux. The value of the diffusion field in the outer domain is related to the local density of walkers. The boundary conditions for the integration in the inner region are obtained by averaging this density over coarse-grained boxes close to the boundary. Now lets focus on how the Diffusion Monte Carlo was treated and implemted. Consider a single point particle performing Brownian motion in continuous

space and time. The conditional probability $P(\mathbf{x}', t'|\mathbf{x}, t)$ of finding the particle at position $\mathbf{x}'$ at time t', given that it started from position $\mathbf{x}$ at time t, is identical to the diffusion kernel,

$$P(\mathbf{x}', t'|\mathbf{x}, t) = \frac{1}{|4\pi D(t'-t)|^{d/2}} e^{\frac{-|\mathbf{x}'-\mathbf{x}|^2}{4D(t'-t)}} \tag{2.1}$$

where D is the diffusion coefficient and d is the spacial dimension. This kernel satisfies the convolution equation

$$P(\mathbf{x}'', t''|\mathbf{x}, t) = \int P(\mathbf{x}'', t''|\mathbf{x}', t')P(\mathbf{x}', t'|\mathbf{x}, t)d\mathbf{x}' \tag{2.2}$$

for all $t < t' < t''$. Therefore, the position of a walker as a function of time can be obtained by successive steps

$$\mathbf{x}(t+\tau) = \mathbf{x}(t) + \ell\xi \tag{2.3}$$

where the components of the random vector ξ are independent Gaussian random variables of unit variance. The time increment τ and the step size ℓ must satisfy the relation $\frac{\ell^2}{\tau} = 2D$. Since time is continuous and the convolution equation is not limited to uniform time increments, successive steps may have different time increments (and use different step lengths as long as $\frac{\ell^2}{\tau} = 2D$ is satisfied for each update).

The basic idea of this DMC simulation is to sample many realizations of diffusion paths, because in the limit as the number of walkers increases to infinity, the density of random walkers converges to the deterministic diffusion equation, thus giving them values for the temperature field near the conversion boundary. Step sizes are chosen to be approximately proportional to the distance d_{cb} of the walker from the conversion boundary between the inner and outer regions, i.e., $\ell \approx cd_{cb}$ with $c \leq 1$.

From Karma and Plapp we took the idea of creating a DMC/KMC hybrid. However, instead of using DMC to estimate a temperature field, we would use it to model longer random walks in the far-field. Unfortunately, as we will mention in more depth later this chapter, the problem in joining the inner KMC region with the outer DMC region smoothly and accurately was too great. We need the boundary to be (in a sense) invisible, in that

the density of the DMC atoms on the outside at the boundary would be essentially equal to the density of the KMC atoms immediately inside the boundary. In order to work around this problem, we looked at another paper, and we now wish to mention some important results from, "Multiscale Kinetic Monte Carlo for Simulating Epitaxial Growth" (DeVita et al., 2005).

2.1.2 Multiscale Kinetic Monte Carlo

The Multiscale Kinetic Monte Carlo method developed by Devita, Sander, and Smereka is essentially different from the multiscale approach of Plapp and Karma in that they are looking at an epitaxy problem. In the regime we are most interested in simulating, dendritic growth, the density of adatoms is too low to treat as a continuum of atoms. Therefore, we looked into ways of taking a KMC simulation and letting adatoms far from any nucleus or island of atom take longer steps. In their multiscale approach, they immediately ran into problems (which will be described in detail below) but found a good approximation that sped up the simulation by five to ten times at moderate temperatures and large D/F, where D is the diffusion constant and F is the flux driving the addition of more adatoms.

Before we can explain their results any further, we must first recal from Chapter 1 that a Kinetic Monte Carlo model includes modeling the event rates $\{q_{t_k}\}_{k=1}^{N}$ where N is the number of possible events at time t_k. In the type of model explained in this section and in the model we develop, the rate at which the adatoms hop from one location to another is given by

$$R(N) = De^{-\Delta E/k_B T}, \tag{2.4}$$

where D is the diffusion constant, k_B is Boltzmann's constant, T is temperature, and ΔE is an energy barrier, which is how much energy will be needed for an atom to hop from its current location to a position one hop away. Also, if the number of nearest neighbors (introduced in section 1.2.3) equals n, then the energy barrier is modeled to be proportional to the number of nearest neighbors, $\Delta E = nE_n$, where E_n is a contribution to the energy barrier from each of the n nearest neighbors. We also assume that in both models the

temperature remains constant, so that equation 2.4 can be written as

$$R(N) = De^{-n\tilde{E}}, \tag{2.5}$$

where $\tilde{E} = E_n/k_BT$, leaving only the parameters D and $\tilde{E}$. However, by scaling distance and time appropriately, we can take $D = 1$, and the rate calculation will only depend on the one parameter left in the exponential.

Devita, Sander, and Smereka noted that, for example, with a rate of a singly-bonded atom ($n = 1$) hopping at a rate given by $e^{-\tilde{E}} = e^{-5}$ ($\tilde{E} = 5$ simulates Cu at 600K), approximately 148 adatom diffusion events occur for every singly bonded atom hop, and approximately 22,000 adatom diffusion events occur for every doubly-bonded atom ($n = 2$) hop event (which has a rate of e^{-2*5}). This causes a bottleneck effect in any KMC simulation in this regime. The difficulty that they discovered when having adatoms make large hops was knowing how much "free room" an adatom has to move. For example, if an atom was going to make a 25 step hop there is a diamond shape of possible grid points the atom could land on, and in this example 1301 sites are possible. If this diamond is clear of other atoms all is fine, because the probability distribution for the 1301 possible sites could be computed in advance, but if any other atoms are in this diamond the probability distribution changes and would need to be recomputed each time based on where the other atoms are, and knowing where they are requires searching all 1301 spots. This basically takes away any computational gain that would be made by moving 25 steps at once.

One simplifying observation they made was that 98% of the probability density is contained in a circle of radius $2\sqrt{n}$, where n is the step size. For $n = 25$, this made the 1301 sites needed to be searched drop down to 314 and then a truncated probability distribution was sampled. This was, of course, faster, but still too slow for what they desired. Also, the option of truncating the distribution at a smaller radii than $2\sqrt{n}$ made the approximated simulation less accurate.

So they focused on sampling from as small a distribution as possible while still keeping

accurate KMC dynamics. All details are in the paper, but their optimal strategy for approximating the large hop was to let an "m^2-step" hop move to one of four places, (i+m,j), (i-m,j), (i,j+m), or (i,j-m) where (i,j) is the starting point for the "m^2-step" hop. Now in order to make sure no atoms were in the way of this move, a square $(2m-1) \times (2m-1)$ needs to be searched. The key now is that this search is faster than $n = m^2$ random number calls.

Also, by design, a move of m has rate of m^2. If an atom is found in the search of the above mentioned square, then the atom moves as far as it can, say s, which is less than m. This leaves $m^2 - s^2$ time left that this atom needs to move so the atom moves again until the maximum total step size is m.

This method for handling long hops for adatoms relies on *approximating* a n-step hop. In the next section, we put forth a method that allows an actual n-step random walk to occur.

2.2 New 2D Model

For the new model, we began with the knowledge that dendrite growth is initiated when atoms diffusing on a substrate cluster due to lower hopping rates for highly coordinated atoms, and the boundary of this cluster is morphologically unstable when the flux of new atoms is supplied in the far field (DeVita et al., 2005). We begin the simulation by placing a seed of atoms in the center of the simulation domain. The region comprised of only this seed plus additional area for the growing cluster to expand into, will be simulated by the usual (one-step) KMC model. This will be refered to as the inner region (see Figure 2.1) This process used for each time step is given by the following steps to be explained in full below:

1. Calculate the rate at which each atom in the domain moves.
2. Sum these rates along with the rate at which adatoms are added.
3. Generate a random number and use that to determine which atom hop event occurs or if an adatom is added.

4. Update the chosen atom's position or place the adatom, then update the number of nearest-neighbors for affected atoms.

5. Generate another random number and use it to determine how much time passed between this event and the previous event.

6. Update the total time passed.

7. Repeat steps 1-6 until a stopping condition is reached.

During each time step of the KMC algorithm we must decide which event occurs, by choosing one atom to hop. As explained earlier in the chapter, the hopping rate for each atom is given by

$$R(N) = De^{-\Delta E/k_B T} \tag{2.6}$$

where D is the diffusion constant, k_B is Boltzmann's Constant, T is temperature, and ΔE is an energy barrier. In our model, $\Delta E = E_N N$, where N is the number of neighbors, and temperature is kept constant, so we will refer to the exponent of e as $N \cdot C$, where is the constant $C = E_N/(k_B T)$ (DeVita et al., 2005). In order to make Steps 1-3 go quickly at each each time step, we group atoms in the simulation domain by the number of neighbors and put them into "bins" (Schulze, 2008a) which are updated as needed each time step. Then steps 1 and 2 are acheived by calculating a cumulative sum of the rates,

$$R_i = \sum_{N=0}^{i} K_N R(N) \tag{2.7}$$

for $i = 0, 1, 2, 3$, where K_N = number of atoms with N neighbors. (Note that in two dimensions, an atom with four neighbors is locked in and cannot move.) Once a bin is randomly chosen using a uniform random number between 0 and $R_3 + F$, where F is the fixed rate at which atoms are added to the boundary of the simulation domain, we will either be moving an atom or adding one to the boundary. Another uniform random number is then used to either chose a specific atom from the list in that "bin," or to find a location on the boundary for a new adatom. Step 4 will consist of two possible situations. If an atom from

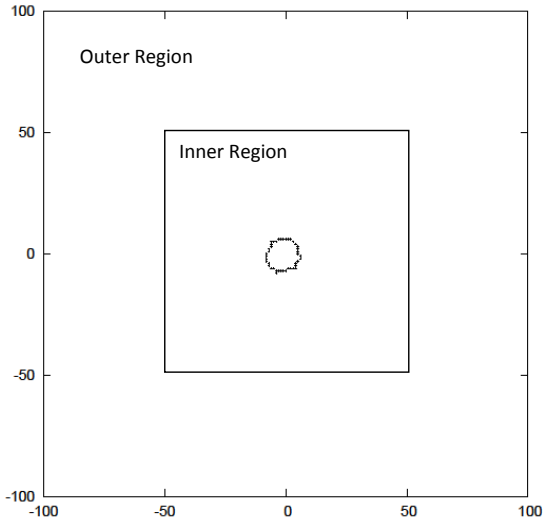

Figure 2.1: This shows the domain decomposition into Inner and Outer Regions.

a "bin" is chosen, that atom is randomly moved to one of the up to four unoccupied spaces adjacent to it, then the nearest-neighbors counts are updated as necessary. Otherwise, if an adatom is to be added to the boundary, a spot is randomly chosen. To accomplish Step 5, the random waiting time for this event to take place is then chosen using the exponential distribution with mean $1/(R_3 + F)$, via the inverse transform method described in the introduction chapter. Step 6 updates the total time passed, then we repeat.

So that we may update all data structures after an event, we also need to easlity know where in the "bin" list an moving atoms are. We can do this very quickly by keeping stored inverse lists (Schulze, 2002). These are simply a way of easily finding where in a "'bin"' an atom is by knowing only its location, since the inverse list inputs the location and outputs its place in the list.

The dendrite grows outward when the atoms diffusing in the far field eventually make contact and bond with the initial seed of atoms. Here is where the previous works mentioned in the last section were consulted and used as inspiration. In order to shorten the computational time used for the far field diffusion (necessary for the KMC algorithm described above), we would like to let atoms far from the growing dendrite take longer steps (just like both papers detailed above). We concentrated on two possible ways to accomplish these longer steps, Gaussian Diffusion (Plapp and Karma, 2000) and Multistep Random Walks (DeVita et al., 2005), and we will now examine the merits and demerits of each for use in the simulation we wish to create.

2.2.1 Gaussian Diffusion in the Far Field

Recall, the position of a Gaussian random walker as a function of time is obtained by successive steps

$$\mathbf{x}(t + \tau) = \mathbf{x}(t) + \ell\xi \tag{2.8}$$

where the components of the random vector ξ are independent Gaussian random variables of unit variance. Then the time increment τ and the step size ℓ must satisfy the relation $\frac{\ell^2}{\tau} = 2D$, where D is the thermal diffusivity. Our model requires that we choose and fix the flux of atoms entering the far field region of the simulation domain. However, since very

large hops *can* occur (from the Gaussian Distribution) in any direction, the atom density near the boundaries tends to stay lower than expected with the diffusion of atoms. Many hopped (far) out of the simulation domain on their first chance to move, while others were allowed to jump too far into the inner domain landing on top of atoms already growing on or positioned inside the growing cluster of atoms. Ignoring the effects of the latter problem, we ran a simulation where only Gaussian random walks were allowed in a square domain, $[-100, 100] \times [-100, 100]$, the goal being to fix the flux of atoms at the left and right boundaries, then start a simulation letting the atoms hop around the domain as Gaussian random walkers. We fixed the left boundary to a density of 0.05 and the right boundary to a density of 0.15, and we had the boundaries absorb any atom that hops out of the domain. If this truly simulated the diffusion field acurately, in the domain, we would expect a linear change in the column density from left to right corresponding to the flux at the left boundary and the flux at the right boundary throughout the simulation square. This, however, was not exactly the case. Because of the finiteness of the domain, we saw a graph that was not linear near the boundaries (see Figure 2.2 for an example).

This drop happened even when distance from the boundary was taken into account in the parameters of the Gaussion distribution. Gaussian random walks only appoximate the diffusion equation well if the domain is assumed to be infinite. The finite sized domain let atoms hop outside and disappear which changed the mechanics of the random walks enough that the density of the atoms (especially near boundaries) no longer simulated the diffusion equation correctly. Therefore, this method for speeding up the outer region's hops was abandoned in lieu of a more accurate accounting of the movement near boundaries.

2.2.2 Multistep Random Walks in the Far Field

To facilitate a multistep random walk, a fixed step size in the far field is chosen. In order to not have the difficulties of finding nearby atoms that the multistep method of Devita, Sander, and Smereka (DeVita et al., 2005) had, we used an aspect of the previously abandoned DMC method. We decided to not have the adatoms interact in the far field. In other words, we used long, noninteractive multisteps in the far field, meaning atoms could not

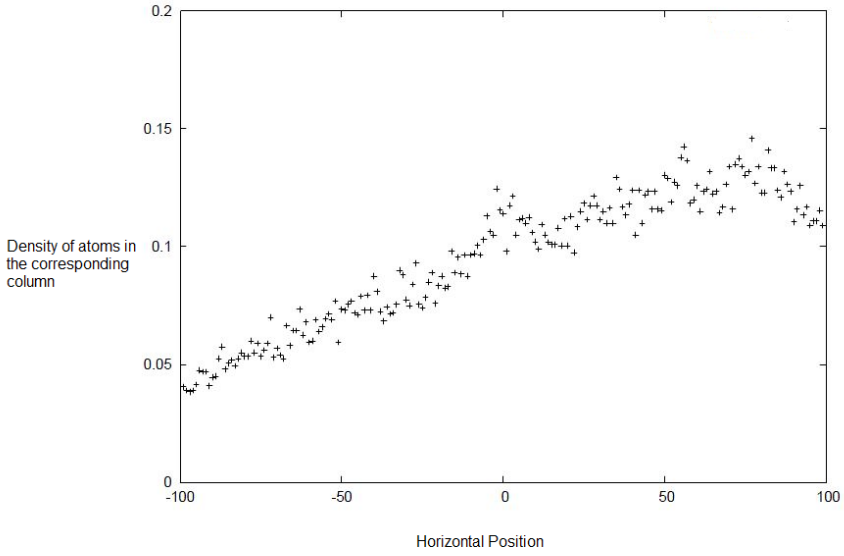

Figure 2.2: Graph of column density corresponding to a flux of 0.05 on the left and 0.15 on the right. If the simulation correctly modeled the diffusion, we would expect the left side to be near 0.05 and the right side to be near 0.15. and we expect the change in column density to be linear, which is clearly not the case, since the ends drop, especially on the right side where the density drops to a number close to 0.11.

form islands and multiple atoms could even occupy the same latice site.

Using noninteractive multisteps in the far field speeds up the adatom diffusion, but in order for the initial seed of atoms to grow, atoms must start interacting when their distance to the growing cluster is "close." Therefore, when an atom croses a predetermined boundary around the initial seed, the atom moves from the noninteracting, multistep region into the single-step, fully interacting KMC region. When this happens, multistep atoms must be converted (as far as the simulation is concerned) to have the properties needed in the new region (i.e. it cannot occupy the same site as another atom, the number of neighbors will change the atom's hopping rate, etc.). Also, single step KMC atoms could similarly move out into the (noninteracting) multistep domain. We will explain the exact conversion processes in the next section, so that in this section we can focus on the multistep walking adatoms only. On the other hand, multistep walking adatoms are allowed to hop out of the domain in an absorbing manner.

However, when these adatoms need to change regions (Multistep to/from Single-step or Multistep to/from the boundary of the simulation domain) care must be taken so that we do not have an incorrect density of adatoms near the boundaries. For example, when an atom is close to the boundary between the multistep region and the KMC region, the step size of the adatom must be decreased to equal the minimum distance to the first site *outside* the inner region, and a similar shortening of the step size must happen at the simulation domain boundary. This is necessary in order to keep the dynamics correct, since adatoms need to hop into the KMC region with a step size of exactly one. The reason for this will take some time to develop, and is fully explained in the section on how the far field and the KMC domains interact. However, one quick reason comes from remembering the problem we mentioned earlier that Devita, Sander, and Smereka had about knowing if any KMC atoms are in the way (DeVita et al., 2005) of a multistep hop.

2.2.3 Probability Distributions for Event Waiting Times

As described previously, for an event in single-step KMC, the waiting time between two successive events at times t_k and t_{k+1}, is given by $\Delta t = t_{k+1} - t_k$. This quantity Δt

was determined by randomly choosing a random time using the exponential distribution. However, the waiting time before the k^{th} event occurs does not enjoy the same probability distribution (unless $k = 1$, of course). The exponential distribution is actually a special case of the gamma distribution, which has a probability distribution function given by

$$f(x; k, \lambda) = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{\Gamma(k)}, \lambda \geq 0, \tag{2.9}$$

where

$$\Gamma(k) = \int_0^\infty y^{k-1} e^{-y} dy, k > 0 \tag{2.10}$$

is the gamma function evaluated at k, where k any positive real number (Larsen and Marx, 2000). The parameter λ is called the rate parameter (analogous to rate parameter, λ, in the exponential distribution), and the parameter k is called the shape parameter and will correspond to the length of the multistep an adatom is taking.

Let us note a few observations. First, we need only focus on when k is a natural number, so the gamma function simplifies (via $k - 1$ applications of integration by parts) to

$$\Gamma(k) = (k - 1)!. \tag{2.11}$$

This restriction to the natural numbers makes this an Erlang distribution which is a special case of the gamma distribution when the shape parameter is a natural number, but since the gamma distribution is more versitile, we will continue to refer to this distribution as the gamma distribution. Secondly, if we take $k = 1$ in the gamma distribution we see that,

$$f(x; 1, \lambda) = \frac{\lambda^1 x^{1-1} e^{-\lambda x}}{\Gamma(1)} = \frac{\lambda e^{-\lambda x}}{0!} = \lambda e^{-\lambda x}, \tag{2.12}$$

which shows that $k = 1$ is equivalent to the exponential distribution (as expected). Finally, when this method is carried out in simulation, it is important to note that each time a multistep atom is moved, sampling the gamma distribution to determine the time that passes is most quickly done (at the small step sizes we were analyzing) by generating n random variables from the exponential distribution, where n is the step size. This involves

generating a uniform random variable then calcuating its logarithm, n times. But this causes the method to be not all that significantly faster than KMC only. We sped up these simulations by generating n uniform random variables, and first taking their product, then only calculating one logartihm. This is of course mathematically equivalent since the logarithm of a product is the same as the sum of each of the corresponding logarithms of the factors. This led the simulations of our method to speed up by approximately a factor of two times the simulation time for KMC only.

This difference in waiting time probability distributions causes an immediate impact on the dynamics. Time is the problem. In DMC methods, different hop distances take place over different periods of time (see equation 2.8 and its following paragraph relating distance and time). In the Karma and Plapp paper detailed in a previous section, we noted that they minimized the effects of the walkers all being at different times by creating a buffer zone and solving the diffusion equation numerically in that region. We have no intention of using the random walkers to solve the diffusion equation in order to feed that information into another numerical method like Phase-Field, we simply want to speed up the computational time for the walking of far field adatoms. However, there is an elegant solution involving keeping track of different clocks.

Every atom in the KMC region and the event of adding a new adatom to the outer boundary can move forward using one clock (a random discrete timeline), which for the sake of reference we will call Clock Zero. We then use an idea from Karma and Plapp (Plapp and Karma, 2000), and give each multistepping adatom its own separate clock (on its own random discrete timeline). The inner region and adatom events can be on one clock because single-step KMC algorithms choose a random time from the exponential distribution to be the time before the first atom in the *entire* system can move (or get added). Clock Zero moves forward by sampling the exponential distribution with a rate parameter equal to the *sum of all rates*. This keeps the whole system at the same (random) discrete time. This combination of all rates into one rate parameter, however, was only possible because finding the first event to happen in the KMC region full of independent atoms with waiting times all coming from exponential distributions, also has an exponential

distribution with rate parameter equal to the sum of the rates involved (Evans et al., 2000). On the other hand, since the multistep random walks of length m have waiting times given by the gamma distribution with shape m, these cannot be on the same clock as the KMC region. In addition, the multistep atoms cannot all be on the same clock like all those in the KMC region can be since the gamma distribution does not have the same "memorylessness" property as the exponential distribution (explained further in the next section). In light of noting that the atoms in the far field domain each need their own clock, we decided to have our simulation algorithm choose which atom to move (or adatom to add to the boundary) based on which clock is the furthest behind.

2.2.4 How the Far Field and the KMC Domains Interact: Merging Clocks

So why can noninteracting, multistep atoms only be allowed to move into the KMC region with a one-step hop? First let's answer an easier question. If a KMC atom moves to the outer region or an adatom is placed on the boundary what should its clock start time be? Clearly, it should start with the time of the region it is coming from, the time on Clock Zero. This is clearly the right choice since that is how much time has passed according to this atom. However, when an atom hops from the outer region into the inner region, the two clocks (one belonging to the hopping atom and the other belonging to the entire KMC region) must somehow be merged so the atom can accurately become part of the KMC region, including syncing to Clock Zero.

When a multistep atom lands into the KMC region, one of three cases will occur:

1. the clocks will be equal,
2. the clock of the multistep atom will be ahead of clock zero, or
3. the clock of the multistep atom will still be behind clock zero.

The first case is clearly a nearly impossible feat when keeping time recorded to many decimal places. If it did happen then the multistep atom would hop in, convert to a KMC region atom, and the clocks would already be the same. The real question (and problem) is how to merge the two clocks when they are not equal (case 2 and case 3). The following

observation is important to be completely accurate. The atom needs to hop in with a step size of exactly one so that we can use the memorylessness property of the exponential distribution (a property the gamma distribution, does not enjoy) (Larsen and Marx, 2000). The "memorylessness" property of the exponential distribution says

$$P(T > s + t|T > s) = P(T > t) \tag{2.13}$$

In other words, suppose clock zero has time s. If a multistep atom would randomly get a new time of $T > s + t$, where $t > 0$, then the probability that the atom now has a clock equal to $s + t$ would be the same as it having a clock equal to s. If the multistep atom is still behind the KMC region clock when it moves in, then $T < s + t$, so we keep moving the atom one step at a time until its clock matches or supasses the KMC region clock. Now we are looking at the two cases in question becoming one, where the multistep atom's clock is ahead (unless, unlikely as it is, they match exactly, and in that case, there is nothing more to do with the clocks). Therefore, on the hop when the multistep atom's clock first surpasses the KMC domain's clock, we consider them synced. The multistep atom simply joins the KMC domain, is on the same clock, and becomes a complete member of the KMC domain.

We have now established that we need to be able to make multistep hops for any integer length up to the fixed step size. Therefore, we compute and store all the discrete probability distriubutions for a multistep random walk of each step size (see Figure 2.3 for examples for step sizes one, two, and three). Note that, since many of the nearby sites have probability 0 of the atom landing there, these were discarded and a list containing only possible landing sites (with probabilities greater than 0) is searched with the site of highest probability placed first in the list. All this combined speeds up the process of finding where a multistep atom lands by at least a factor of 2.

Poisson processes do not have a memory, i.e., the n^{th} random event does not influence the $(n + m)^{th}$ random event. Consider the first event of a Poisson process to be the hop that makes the atom's time more than the KMC domain's time. And consider that this

1-Step Probability Distribution

0	0.25	0
0.25	0	0.25
0	0.25	0

2-Step Probability Distribution

0	0	0.0625	0	0
0	0.125	0	0.125	0
0.0625	0	0.25	0	0.0625
0	0.125	0	0.125	0
0	0	0.0625	0	0

3-Step Probability Distribution

0	0	0	0.015625	0	0	0
0	0	0.046875	0	0.046875	0	0
0	0.046875	0	0.140625	0	0.046875	0
0.015625	0	0.140625	0	0.140625	0	0.015625
0	0.046875	0	0.140625	0	0.046875	0
0	0	0.046875	0	0.046875	0	0
0	0	0	0.015625	0	0	0

Figure 2.3: These are the probability distributions for a two-dimensional random with step sizes one, two, and three. Note that the center is where the atom starts, and each probability is placed on its corresponding landing position.

event happens at time T. Let t be equal to the KMC domain's clock. Then the probability distibutions for the waiting times are the same whether $T > t$ or $T = t$ making it possible to merge both clocks equal to the KMC domain's clock. To justify this, note that the probability density curve for the waiting time of an event happening at $T > t$ is given by (1.23) to be $\lambda e^{-\lambda t}$, while the waiting time for an event happing at $T = t$ is also given by the same formula. Thus, we can choose to move the multistep atom's clock back match the KMC domain clock without changing the probabilities of atom densities.

Now, when capturing pictures of the simulated dendrite, we realize that we must be careful. All atoms outside of the KMC region are on different clocks. At any given iteration of the simulation's program, we do not have an accurate picture of the dynamics that have taken place. Therefore, before a picture of the growing dendrite can be seen we must merge *all* clocks. If a time merge does not happen, we will see an inaccurate picture of the placement of all the atoms since they are not all on the same timeline.

To illustrate this (and be convinced that the differing clocks are a real problem), we set up a simulation on a $[-50, 50] \times [-50, 50]$ grid where the flux was fixed on the left and right (similar to the Gaussian diffusion simulation mentioned in a previous section). This time the flux on the left and right were equal, so we expected to see a column density that was essentially constant across all rows. We made atoms in the region $[-25, 25] \times [-25, 25]$ take steps of length two, while all atoms outside this region could only move one step at a time. Figure 2.4 shows what happens if we take a snapshot and look at the atom placement densities without merging all the atom clocks. Now compare that to Figure 2.5 after time merging has occured. The first is obviously not accurate, while the second is an accurate diffusion simulation on the whole domain up to the given time.

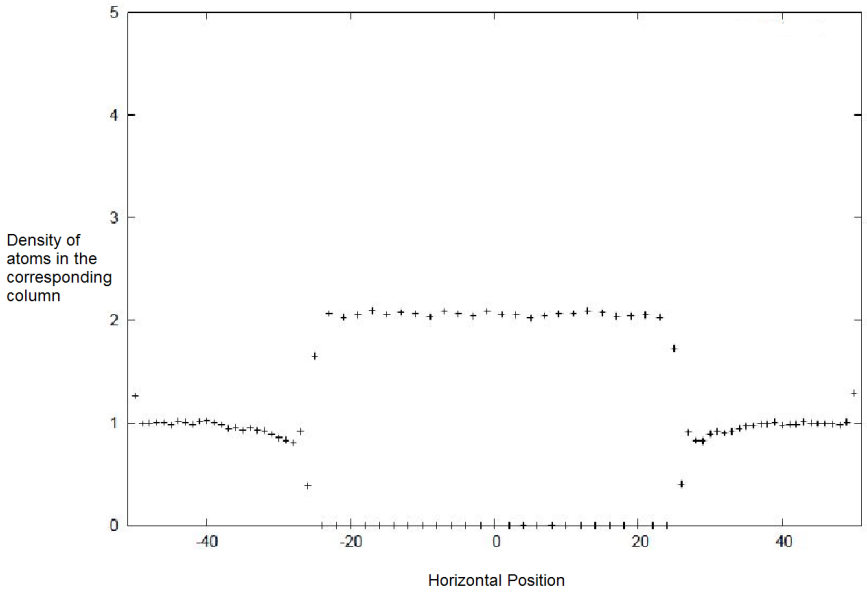

Figure 2.4: Graph of the column density before merging clocks. The adatom flux on the left and right sides are equal, but atoms inside $[-25, 25] \times [-25, 25]$ hop two steps at a time, while atoms not in that region hop one step at a time. One should expect basically a flat line, but the atoms in the two step region are all on different clocks (ahead or behind the ones outside that region). The confirms that clock merging is absolutely necessary to see an accurate picture. In addition, if an atom hops two steps at a time, there will be lattice positions that cannot be reached. Since only half the lattice spots are available when hopping 2-steps at a time, this is exactly why the atom densities inside the multistep region have densities approximately double that of atoms that perform one-step hops

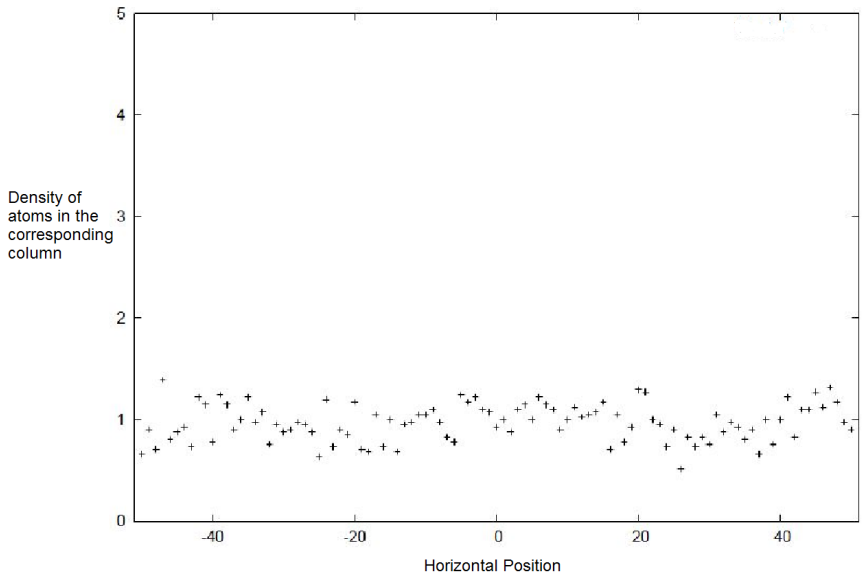

Figure 2.5: Graph of the column density after merging clocks. The atom flux on the left and right are equal, but atoms inside $[-25, 25] \times [-25, 25]$ hop two steps at a time, while atoms not in that region hop one step at a time. One expects basically a flat line (with random noise), and after merging that is exactly what is seen.

Chapter 3

Simulations and Results

3.1 Qualitative Results

We wish to begin this discussion of results by showing that the qualitative results one obtains can be separated naturally into five regimes. Which regime is obtained from a simulation depends on only two parameters in the model, the rate at which adatoms are added to the boundary in the far field, A, and a constant, C, which determines the hopping rate of KMC atoms (see Equation 2.5 with $C := \tilde{E}$) and can be viewed as a measure of how likely it is that a bonded atom will stay bonded to its neighbors. We analyzed how the qualitative results depended on these parameters and found five distinct parameter regimes. We will from now on refer to these five regimes as

1) the scatter regime,

2) the equilibrium regime,

3) the perturbation regime,

4) the dendrite regime, and

5) the bordered dendrite regime.

Also, as we fully describe the properties of each regime and give examples of simulation results, we will assume all of the following:

a. the $x-$ and $y-$ axes create a grid which represents where atoms can sit in the two-diminesional computational domain,

b. the computational domain is the region $CD = [-500, 500] \times [-500, 500]$,

c. the inner (KMC) domain is the region $I = [-50, 50] \times [-50, 50]$,

d. the outer (Multistep) domain is the region $M{=}CD - I$,

e. an initial seed of atoms covers the region $S = [-10, 10] \times [-10, 10]$ when the simulation begins,

f. $C = E_N/(k_B T)$ is constant in any particular simulation, and

g. the multi step size used was five (unless otherwise noted).

We call the first regime the scatter regime, since after a sufficient amount of time, all that can be seen is a scattering of mostly disconnected atoms. This occurs when the rate at which adatoms are added, A, is small, and in addition, the energy barrier, which needs to be surpassed for coordinated atoms to hop, is so small that it is relatively easy for a cluster of atoms to break apart. An example of such a simulation is shown in Figure 3.1. We call the second regime the equilibrium regime since it occurs when somewhat of an equilibrium is reached between the rate A and the rate at which atoms leave the computational domain (and in particular the KMC domain). More atoms can move back out to the multistepping region when the energy barrier is lowered. And more multistepping atoms combined with few coordinated atoms leads to an equilibrium state for certain pairs of parameters, A and C. This equilibrium regime leaves the initial seed of solid atoms virtually unchanged and no perturbations are formed no matter how long the simulation is left to run. An example of this result in simulation is shown in Figure 3.2.

The remaining three regimes are much more interesting for us since we are hoping to create a method that will extend to three-dimensions and simulate the growth of a dendrite. The first such regime we call the perturbation regime, since perturbations on the surface of the atom cluster lead to arms forming out of the initial seed of atoms, but these arms are few, large, and lack the tree-like branching structure of a dendrite. This regime occurs

when the deposition of atoms (determined by the parameter A) happens fast enough for large numbers of adatoms to reach the initial seed. However, the regime can only occur if one also chooses C appropriately so that the energy barrier between atoms is still low enough that perturbations are "smoothed out" over time. This formation is displayed as peninsulas of atoms seen growing out of the initial seed, but no dendritic growth forms over time (see Figure 3.3). The perturbation regime is very close to that of the dendrite regime, but less smoothing occurs in the outgrowing arms and more subbranches form out from them. This can be achieved when we take C to be large enough, so that once two atoms are bonded, the rate they contribute to the sum of rates in the KMC simulation is very small. For example, if C is taken to be 5, then the rate of hopping of a singly bonded atom is $e^{-5}(<< 0)$ and the rate of hopping of a doubly-bonded atom is $e^{-10}(<< e^{-5})$, which are very rare events. When right pairings of A and C are attained, a two-dimensional simulated dendrite is formed as shown in Figures 3.4 and 3.5.

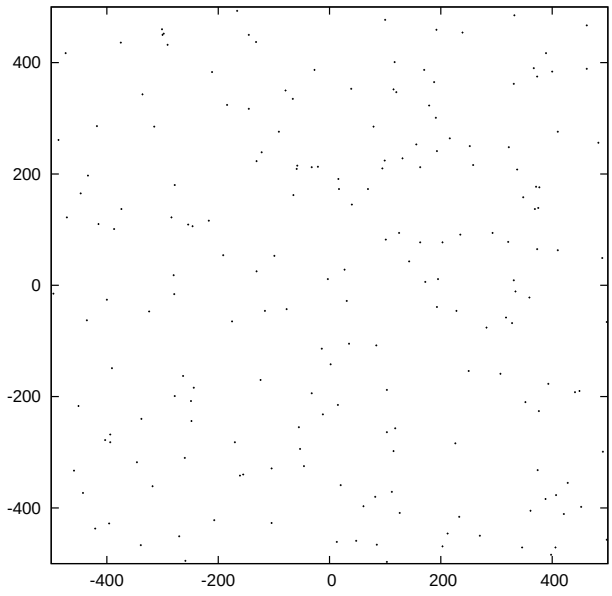

Figure 3.1: The parameters used for this particular simulation were $A = 0.50$, $C = 2.5$, and diffusion coeficient 1×10^5. The code ran until the KMC region reached a time of 500 seconds.

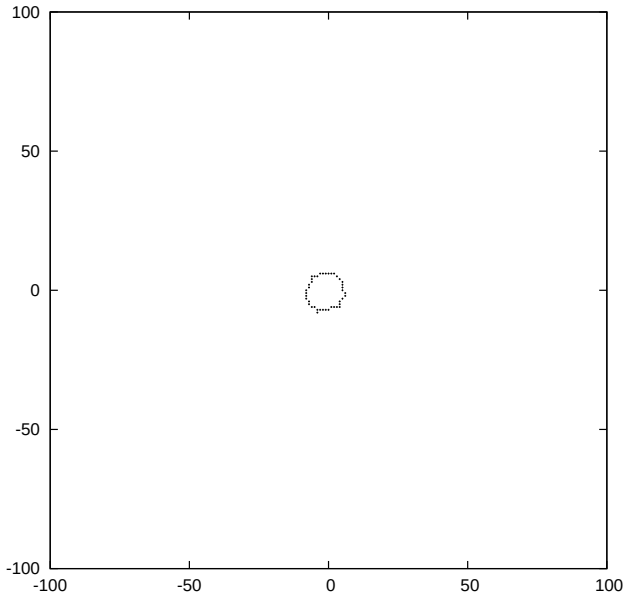

Figure 3.2: The parameters used for this simulation were $A = 0.05$, $C = 5.0$, and diffusion coeficient 1×10^5. The code ran until the KMC region reached a time of 500 seconds. Note that we are looking at the region zoomed into the square $[-100, 100] \times [-100, 100]$.

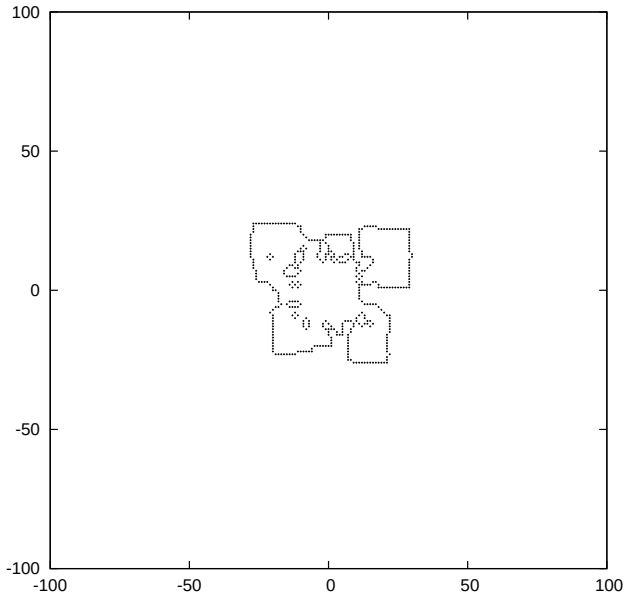

Figure 3.3: The parameters used for this simulation were $A = 0.005$, $C = 16.75$, and diffusion coeficient 1×10^5. The code ran until the KMC region reached a time of 500,000 seconds. Note that we are looking at the region zoomed into the square $[-100, 100] \times [-100, 100]$.

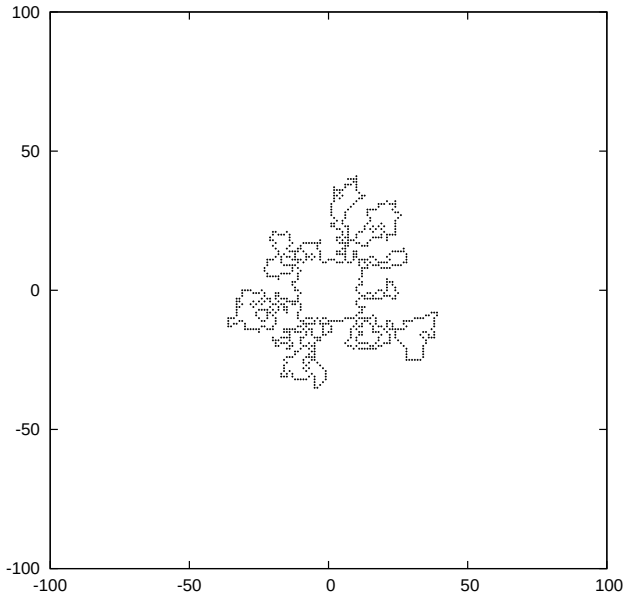

Figure 3.4: The parameters used for this simulation were $A = 0.002$, $C = 20.0$, and diffusion coeficient 1×10^5. The code ran until the KMC region reached a time of 50,000 seconds. Note that we are looking at the region zoomed into the square $[-100, 100] \times [-100, 100]$.

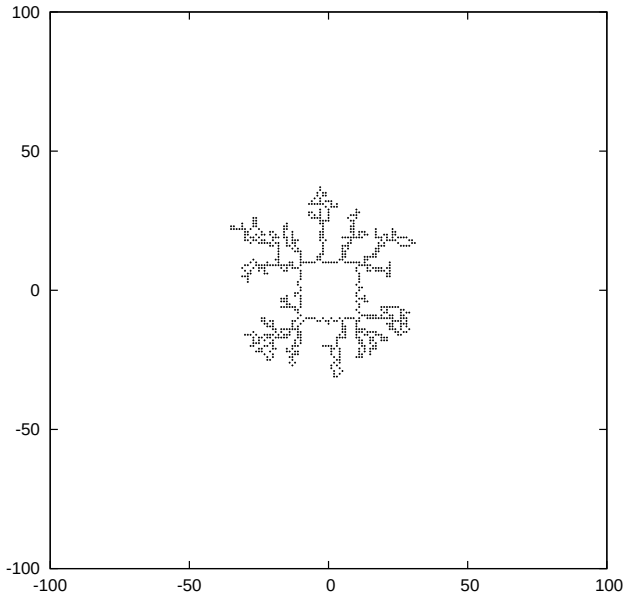

Figure 3.5: The parameters used for this simulation were $A = 0.001$, $C = 20.0$, and diffusion coeficient 1×10^5. The code ran until the KMC region reached a time of 50,000 seconds. Note that we are looking at the region zoomed into the square $[-100, 100] \times [-100, 100]$.

Finally, we should note that in executing this method, as well as a strict KMC simulation, some values of the parameters A and C will not form one growing cluster of atoms in the center, but will instead form many islands near the boundary of the KMC simulation domain. Once these islands occupy all boundary lattice sites, the simulation stops, since it is no longer possible to place new atoms on the boundary. Since the resulting distribution of atoms looks like a dendrite surrounded by a border of atoms, we call this the bordered dendrite regime. This can easily happen when the rate for adding atoms to the domain is too large. This causes the far field to be flooded with atoms (recall, in the far field more than one atom may occupy one latice site) which then causes the KMC boundary to fill with atoms which are bonded (since the KMC boundary is also surrounded by adatoms) and cannot move (note that an atom with four neighbors is completely landlocked with a hopping rate of 0). Since no new adatoms can be added to the region even when the simulation algorithm requires that move, the simulation stops. An example of this happening in our model can be seen in Figure 3.6. This also happens for a strict KMC code as well, and an example can be seen in Figure 3.7. Note that KMC cannot begin to grow the initial seed, because the outer boundary is surround by islands of atoms very quickly. Small dendrites are forming all around the boundary, but because of the size of A, they all attach to other atoms almost as soon as they are added, then they are virtually stuck in place.

In light of these qualitative observations, we wanted to know what values of A and C would give rise to which regimes, and which values should not be used because they cause this method to fail. To determine which parameters gave rise to which regimes, a systematic discrete sweep of the space containing reasonable values of A and C was used and simulations were ran. Then looking at the resulting images we classified them by regime. Figure 3.8 shows which values of A and C give rise to which regimes. This figure also includes a dotted line separating the true dendrite regime from where the method fails (but dendrites still do form). This analysis was subjective due to its qualitative nature, but we believe the parameter space shown is quite accurately separated as long as one understands the boundaries to be fuzzy.

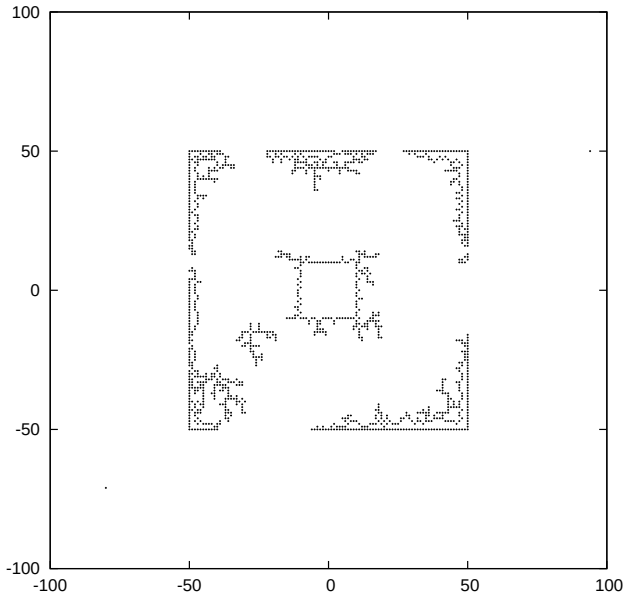

Figure 3.6: The parameters used for this simulation were $A = 0.05$, $C = 20.0$, and diffusion coeficient 1×10^5. The code ran until adatoms could not be added to the simulation domain due to a filling of the boundary between regions M and I.

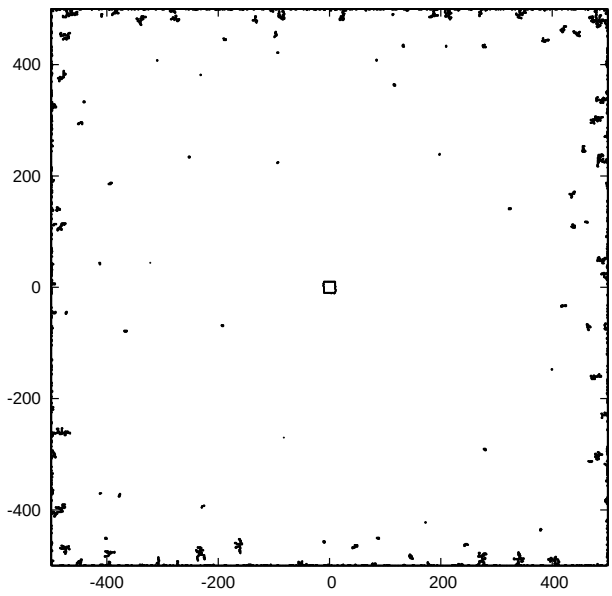

Figure 3.7: The parameters used for this simulation were $A = 0.05$, $C = 20.0$, and diffusion coeficient 1×10^5. The code ran until adatoms could not be added to the simulation domain due to a filling of the boundary.

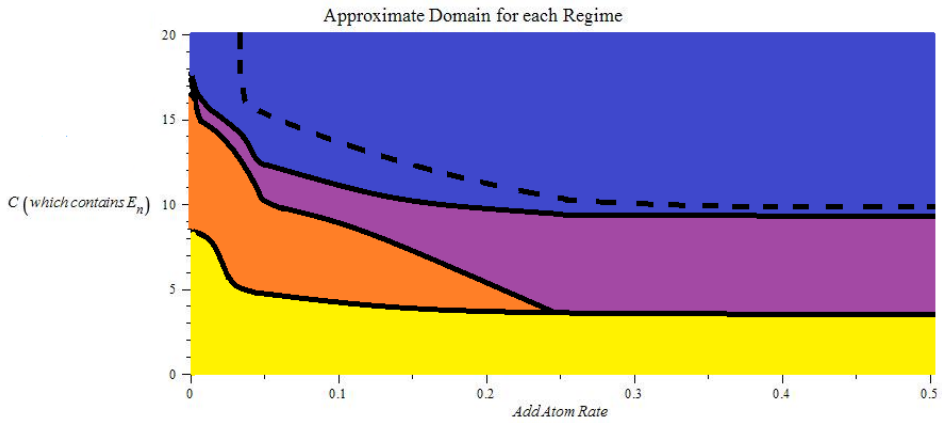

Figure 3.8: Using a step size of five for all simulations, the algorithm was applied to many pairs of parameters A and C. The five regimes are shown on a $A-$ $C-$ quarter plane (since $A > 0$ and $C > 0$). The yellow area is the scatter regime, orange area is the equilibrium regime, the purple area is the perturbation regime, and the blue area is the regime where dendrites are formed. Note that above the dotted line is the bordered dendrite regime.

3.2 Simulation Speed

Using fixed parameters of A and C (in the dendrite regime), we simulated 50,000 seconds of time for growing a dendrite using KMC only, followed by a simulation of our multistep method using step sizes of 1-10, 15, and 20 respectively in the outer region, M. Note that a multistep size of one is different from strict KMC simulation, since the multistep method ignores atom interactions in the far field region, M. The results are shown in Figure 3.9. Looking at Figure 3.9 one can see that the simulation becomes much faster than a strict KMC simulation when the step size is three and continues to have a similar simulation run times until the simulation time starts to rise around a step size of ten. As shown, the run times for the step sizes of 15 and 20 start to significantly slow down the computational time. This is occuring since more computational time is needed to deal with the increase in atoms within n hops from a boundary (and decreasing their step size appropriately) as well as determining where a multistep hop lands. Note that these times are still better than any simulation using less than three steps. In light of this, the method seems to be at its best use when step sizes are chosen between three and ten.

In looking to speed up the simulation even more, other approximations for sampling or approximating the gamma distribution were also tried, and two interesting approaches are

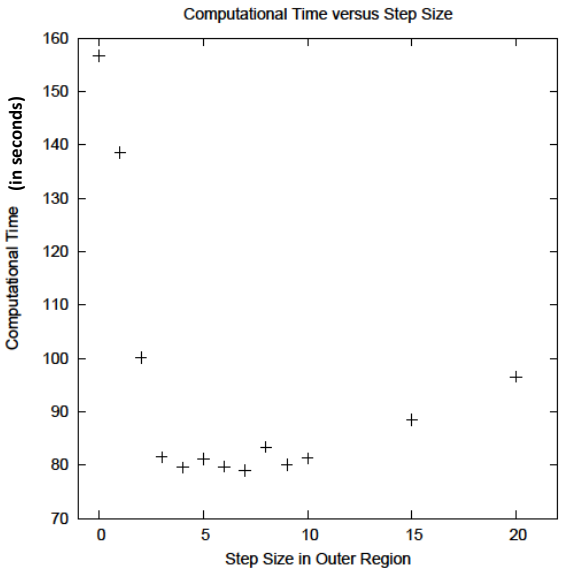

Figure 3.9: The graph shows the computational time for a KMC only simulation (plotted as step size 0). The average computing time (in seconds) for all other multistep sizes are plot versus the step size for a running 50,000 seconds of simulated time. The parameters used for these simulations were $A = .001$, $C = 20.0$, and diffusion coeficient 1×10^5.

in the Appendix. These approximations leave much to be desired in terms of approximating the gamma distribution 100% accurately, but made extremely similar dendrites to the ones found in our simulations in this section. One of the approximations in the Appendix gives a fastest time at the step size of 20, which is over four times faster than a KMC only simulation. Also in the Appendix is some analysis of what occurs when we ignore the need to decrease the step size near the Multistep to KMC boundary.

Chapter 4

Summary and Future Directions

Our long term goal of using a multistep KMC method to simulate dendritic growth in three-dimentions is the obvious next step. We have developed an algorithm that significantly speeds up the computational time needed to simulate the random walks of adatoms in the far field. This in turn reduces the computational time needed to grow a large two-dimensional dendrite using adatoms supplied in the far field. Naturally extending the idea to three-dimensions will be straight forward in theory, but coding the algorithm will take time. Future directions might also include an adaptive method where the inner KMC region grows/shrinks as the cluster of atoms grows/shrinks. This would require developing conditions that would force the algorithm to stop and redraw boundaries, certainly needing to instantly change some atoms from multistep to KMC or visa versa. This is very promising and could allow this simulation method to simulate the growth of much larger dendrites than previous kinetic monte carlo methods, while also completing the simulation in less time.

Bibliography

Boettinger, W., Warren, J., Beckermann, C., and Karma, A. (2002). Phase-field simulation of solidification. *Annual Review of Materials Research*, 32:163–194.

Bortz, A., Kalos, M., and Lebowitz, J. (1975). New Algorithm for Monte-Carlo Simulation of Ising Spin Systems. *Journal of Computational Physics*, 17(1):10–18.

Chatterjee, A. and Vlachos, D. (2007). An overview of spatial microscopic and accelerated kinetic monte carlo methods. *Journal of Computer-Aided Materials Design*, 14(2):253–308.

Chen, S., Merriman, B., Osher, S., and Smereka, P. (1997). A simple level set method for solving Stefan problems. *J. Comput. Phys.*, 135(1):8–29.

Davis, S. H. (2001). *Theory of Solidification*. Cambridge University Press.

DeVita, J. P., Sander, L. M., and Smereka, P. (2005). Multiscale kinetic monte carlo for simulating epitaxial growth. *Phys. Rev. B*, 72(20):205421.

Devroye, L. (1986). *Non-uniform random variate generation*. Springer-Verlag.

Doucet, A., De Freitas, N., and Gordon, N., editors (2001). *Sequential Monte Carlo methods in practice*.

Evans, M., Hastings, N., and Peacock, B. (2000). *Statistical Distributions*. Wiley-Interscience.

Gibou, F., Fedkiw, R., Caflisch, R., and Osher, S. (2003). A level set approach for the numerical simulation of dendritic growth. *J. Sci. Comput.*, 19(1-3):183–199. Special issue in honor of the sixtieth birthday of Stanley Osher.

Haji-Sheikh, A. and Sparrow, E. M. (1966). The floating random walk and its application to Monte Carlo solutions of heat equations. *SIAM J. Appl. Math.*, 14:370–389.

Holmes, M. H. (2009). *Introduction to the foundations of applied mathematics*. Texts in applied mathematics. Springer, Dordrecht, London.

Jansen, A. P. J. (2012). *An introduction to kinetic Monte Carlo simulations of surface reactions*, volume 856 of *Lecture Notes in Physics*. Springer, Heidelberg.

Larsen, R. J. and Marx, M. L. (2000). *An Introduction to Mathematical Statistics and Its Applications (3rd Edition)*. Prentice Hall, 3rd edition.

Parzen, E. (1965). *Stochastic processes*. Holden-Day San Francisco.

Plapp, M. and Karma, A. (2000). Multiscale finite-difference-diffusion-Monte-Carlo method for simulating dendritic solidification. *J. Comput. Phys.*, 165(2):592–619.

Saum, M. A., Schulze, T. P., and Ratsch, C. (2009). Inverted List Kinetic Monte Carlo with Rejection Applied to Directed Self-Assembly of Epitaxial Growth. *Communications in Computational Physics*, 6(3):553–564.

Schulze, T. P. (2002). Kinetic Monte Carlo simulations with minimal searching. *Physical Review E*, 65(3):036704+.

Schulze, T. P. (2008a). Efficient kinetic Monte Carlo simulation. *J. Comput. Phys.*, 227(4):2455–2462.

Schulze, T. P. (2008b). Simulation of dendritic growth into an undercooled melt using kinetic monte carlo techniques. *Phys. Rev. E*, 78:020601.

Shishkina, O. and Wagner, C. (2006). Adaptive meshes for simulations of turbulent Rayleigh-Bénard convection. *JNAIAM J. Numer. Anal. Ind. Appl. Math.*, 1(2):219–228.

Shonkwiler, R. W. and Mendivil, F. (2009). *Explorations in Monte Carlo Methods*. Undergraduate texts in mathematics. Springer, New York, Heidelberg, London.

Smith, J. M., Van Ness, H., and Abbott, M. (2001). *Introduction to Chemical Engineering Thermodynamics (The Mcgraw-Hill Chemical Engineering Series)*. McGraw-Hill Science/Engineering/Math, 6 edition.

Voter, A. F. (2007). Introduction to the kinetic monte carlo method. In Sickafus, K. E., Kotomin, E. A., and Uberuaga, B. P., editors, *Radiation Effects in Solids*, volume 235 of *NATO Science Series*, chapter 1, pages 1–23. Springer Netherlands, Dordrecht.

Zheng, Z., Stephens, R. M., Braatz, R. D., Alkire, R. C., and Petzold, L. R. (2008). A hybrid multiscale kinetic monte carlo method for simulation of copper electrodeposition. *Journal of Computational Physics*, 227(10):5184 – 5199.

Appendix

Appendix A

Other Approximations of Multistep Waiting Times

A.1 No Decrease in Step Size Near the KMC Boundary

For these simulations we simply ignored the need to hop into the KMC region with a step size of one for the clocks to remain accurate. Figure A.1 shows simulation run times of the same variety of steps given in the last chapter. The following two figures, Figure A.2 and Figure A.3, show the difference in the two dendrites formed using the two different methods. Note that the average number of atoms at the end of each simulation only have a difference of around 1%, which clearly shows this seems to actually be a very good approximation.

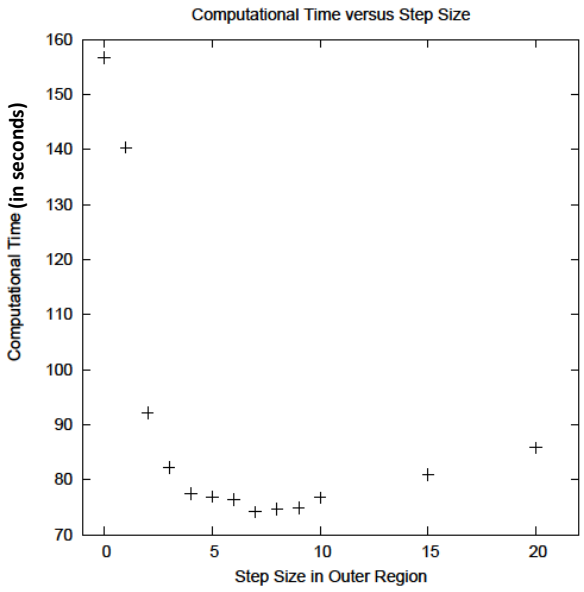

Figure A.1: Compuation time using a simulation that did not decrease step size near the KMC boundary. These dendrites were simulated using the parameters, $A = .001$, $C = 20.0$, and diffusion coeficient 1×10^5, and ran to 50,000 seconds of simulated time.

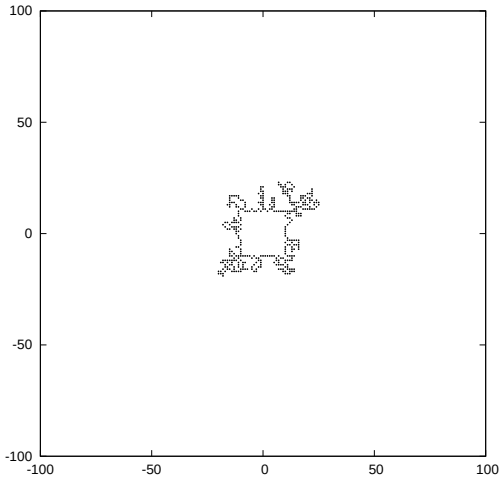

Figure A.2: This is a simulated dendrite when the step size did not decrease near the KMC boundary. This dendrite was simulated using the same parameters as the next figure, $A = .001$, $C = 20.0$, and diffusion coeficient 1×10^5, and ran for 50,000 seconds of simulated time. Hops of size five were allowed in the outer region. 827 atoms were in the simulation domain at the end.

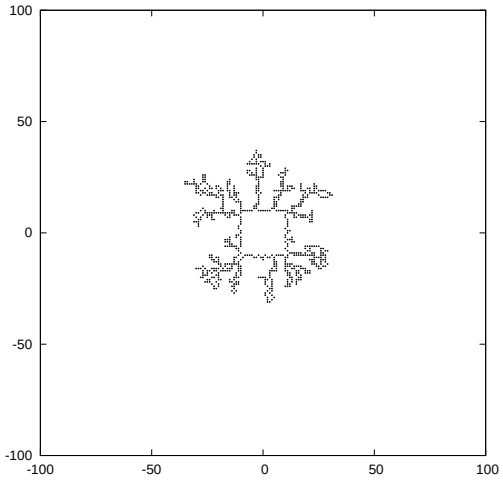

Figure A.3: This dendrite was simulated using the same parameters as the previous figure, $A = .001$, $C = 20.0$, and diffusion coeficient 1×10^5, and ran for 50,000 seconds of simulated time. Hops of size five were allowed in the outer region. 817 atoms were in the simulation domain at the end.

A.2 Approximate Gamma Distribution with Scaled Exponential Distribution

Even though the Gamma and Exponential probability distributions are different (for Gamma shape parameter not equal to 1), we wondered what would happen to the dendrite if the time was calculated using a scaled exponential distribution with the same mean, but different variance, which strays from the actual variance with a scaling equal to the step size. For these simulations, to find the time taken for a multistep atom to hop n steps, we generated one exponential random variable and multiplied that by the step size.

To see just how different these distributions are Figure A.4 shows the probability distribution function (PDF) graphed for various step sizes. As one would expect the computational time was greatly decreased since only one random number needed to be generated for each multistep hop, and a graph showing the time for KMC only and step sizes 1-10, 15, and 20 is shown in Figure A.5. Unexpectedly, the dendrites formed are still very similar, but the difference in number of atoms on average is significantly higher at approximately an 8% difference in the number of atoms in the domain when the simulation stopped (See Figure A.6 and Figure A.7).

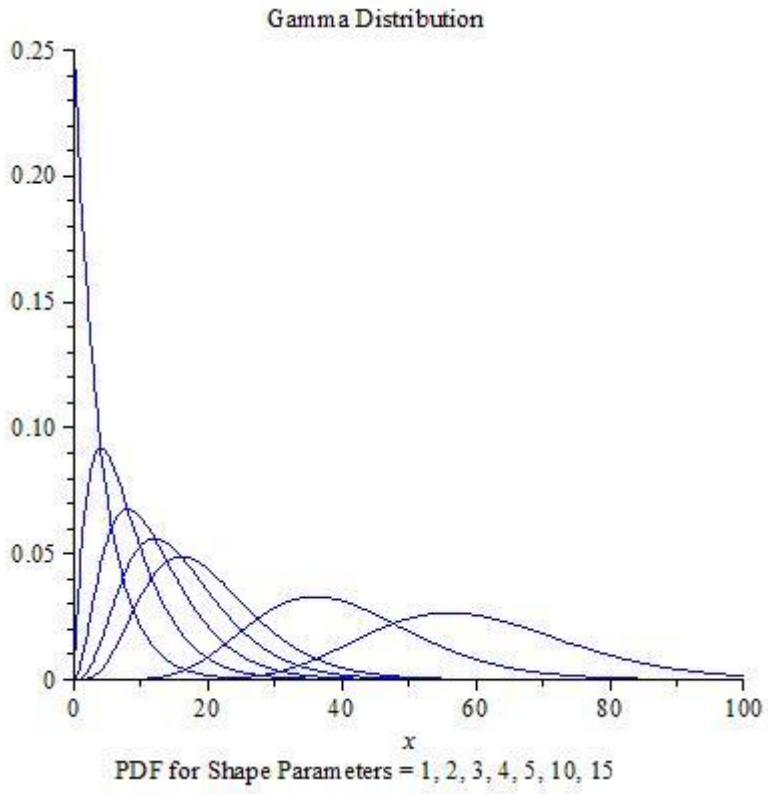

Figure A.4: Gamma distribution with various shape numbers compared to the exponential distribution (shown with shape number equal to one).

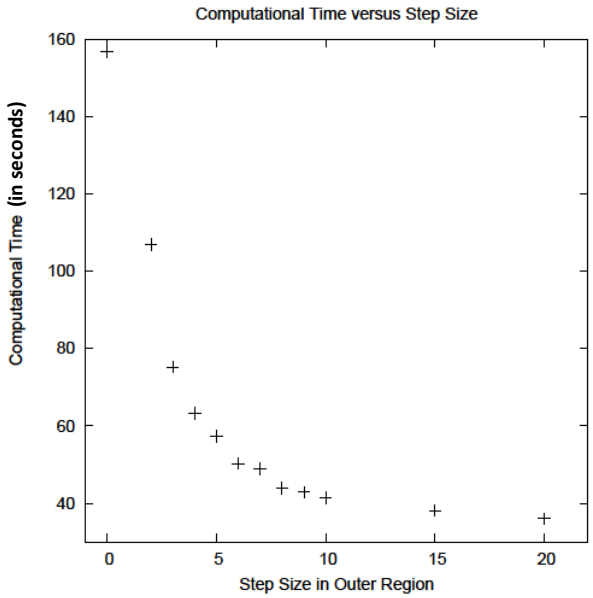

Figure A.5: Computation time using a simulation that used a scaled exponential distribution instead of a gamma distribution. These dendrites were simulated using the parameters, $A = .001$, $C = 20.0$, and diffusion coeficient 1×10^5, and ran to 50,000 seconds of simulated time.

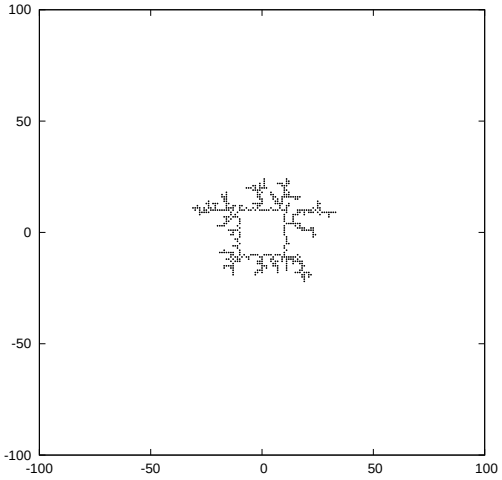

Figure A.6: This is a Simulated dendrite using a simulation that used a scaled exponential distribution instead of a gamma distribution. This dendrite was simulated using the same parameters as the next figure, $A = .001$, $C = 20.0$, and diffusion coeficient 1×10^5, and ran for 50,000 seconds of simulated time. Hops of size five were allowed in the outer region.

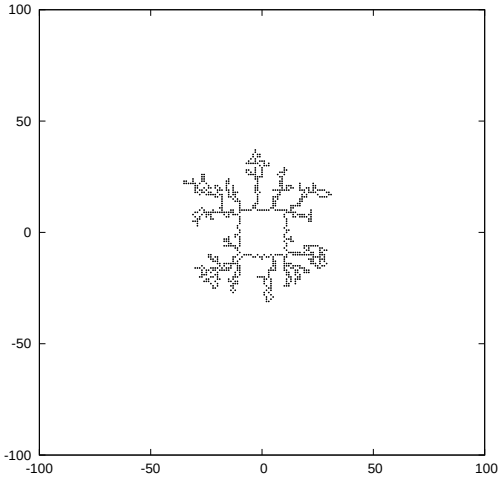

Figure A.7: This dendrite was simulated using the same parameters as the previous figure, $A = .001$, $C = 20.0$, and diffusion coeficient 1×10^5, and ran for 50,000 seconds of simulated time. Hops of size five were allowed in the outer region.

A.3 Approximate Gamma Distribution with Normal Distribution for Larger Step Sizes

The gamma distribution *can* be approximated by the normal distribution, but this approximation only gets better as the step size increases (see Figure A.4). We ran simulations updating the time of a multistep atom by generating one appropriate random variable from the normal distribution for step sizes 10, 15, and 20. The computation time is shown in Figure A.8. We also ran one simulation with step size 5 to be able to compare a sample simulated dendrite to my original model. The number of atoms in the simulation domains at the stopping time only differed by an average of approximately 1%, which makes this a good approximation that speeds up time, if large step sizes are used. See Figures A.9 and A.10 for a comparison of sample simulated dendrites.

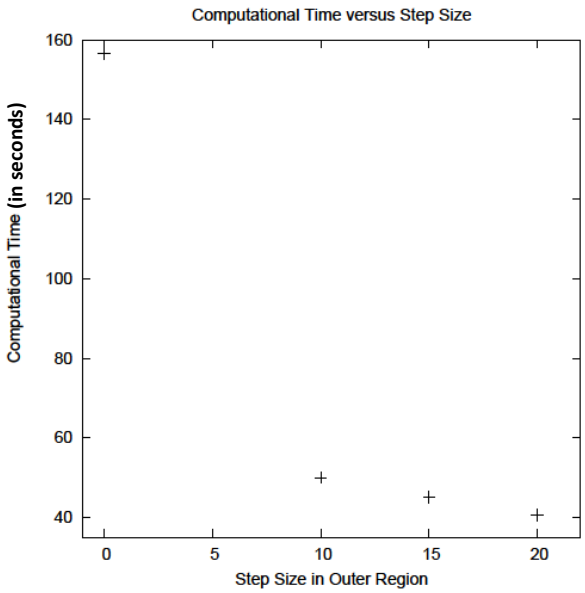

Figure A.8: Computation time for a simulation using the normal distribution as an approximation to the gamma distribution. This approximation is only good as the step size increases, so this was only done for a few step sizes with the KMC time still there for comparison. These dendrites were simulated using the parameters, $A = .001$, $C = 20.0$, and diffusion coeficient 1×10^5, and ran to 50,000 seconds of simulated time.

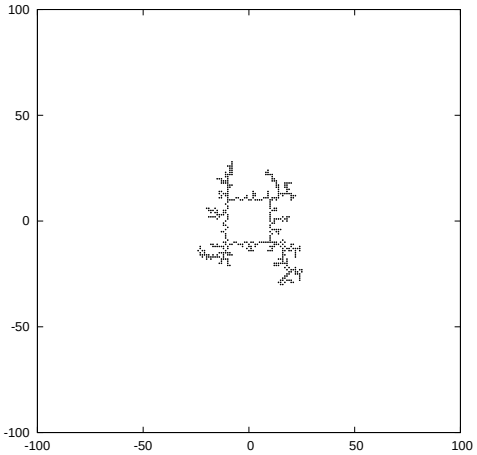

Figure A.9: Simulated dendrite using a simulation using the normal distribution as an approximation to the gamma distribution. This dendrite was simulated using the same parameters as the next figure, $A = .001$, $C = 20.0$, and diffusion coeficient 1×10^5, and ran for 50,000 seconds of simulated time. In order to compare with my model's dendrite, hops of size five were allowed in the outer region. 824 atoms were in the simulation domain at the stopping time.

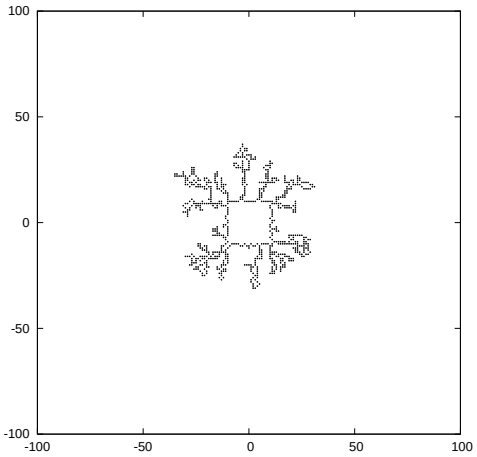

Figure A.10: This dendrite was simulated using the same parameters as the previous figure, $A = .001$, $C = 20.0$, and diffusion coeficient 1×10^5, and ran for 50,000 seconds of simulated time. Hops of size five were allowed in the outer region. 815 atoms were in the simulation domain at the stopping time.

A.4 A Note on Random Number Generation Computational Times

The different ways that we generated the random numbers needed for multistep hops took different amounts of computational time. Figure A.11 shows a comparison of time need to generate 500,000 such numbers for varying step sizes.

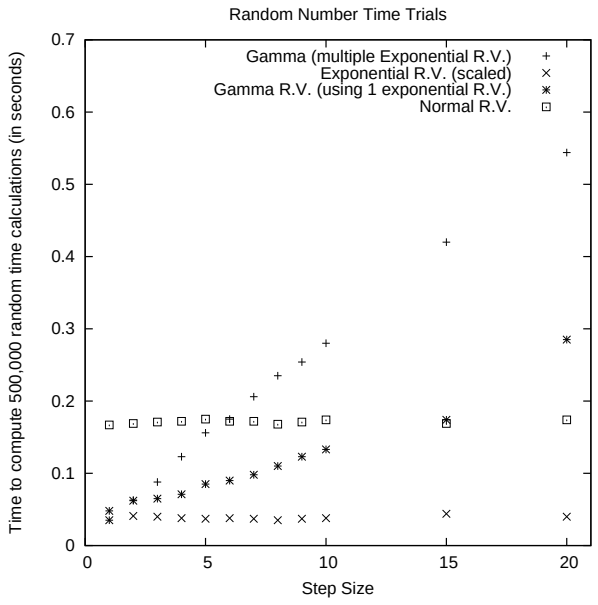

Figure A.11: Computational time comparison of four possible methods of calculating the random times needed in the simulation. Each did 500,000 generations.

Vita

Holly Nichole Clark was born in Dearborn, Michigan on April 9, 1982. In 2000 she graduated from Graves County High School in Mayfield, Kentucky. She received a Bachelor of Science degree in Applied Mathematics with an emphasis in Physics from Murray State University in 2004. She began the Ph.D. program in Mathematics at the University of Tennessee in August 2004, and received the degree in 2013 under the direction of Tim P. Schulze.