University of Pennsylvania
ScholarlyCommons

Senior Design Reports (CBE)

Department of Chemical & Biomolecular Engineering

4-14-2009

Design and Control Using Stochastic Models of Deposition Reactors

Peter J. Beltramo
University of Pennsylvania

Christina L. Bodarky
University of Pennsylvania

Helen M. Kyd
University of Pennsylvania

Follow this and additional works at: http://repository.upenn.edu/cbe_sdr

Part of the Chemical Engineering Commons

Beltramo, Peter J.; Bodarky, Christina L.; and Kyd, Helen M., "Design and Control Using Stochastic Models of Deposition Reactors" (2009). *Senior Design Reports (CBE)*. 8.
http://repository.upenn.edu/cbe_sdr/8

This paper is posted at ScholarlyCommons. http://repository.upenn.edu/cbe_sdr/8
For more information, please contact libraryrepository@pobox.upenn.edu.

Design and Control Using Stochastic Models of Deposition Reactors

Abstract
The financial feasibility of the creation of a start-up company to sell software developed for the optimization and in-line control of thin film growth in deposition processes was investigated. An analysis of the current marketplace revealed potential for a small start-up company to be competitive with this novel product. The investigation concluded an IRR of 20% for a five year period before possible sale of the company. The kinetic Monte Carlo method was employed as the basis for all simulations in this work. This method retains atomic scale information while enabling simulation of process relevant features such as roughness, growth rate and efficiency. A model predictive controller was designed to reproducibly generate thin films with desired properties under a variety of initial condition disturbances for both single component and multi component systems. The substrate temperature and gas flux were employed as control variables. The control algorithms were investigated using a sensitivity analysis and shown to be robust under a wide range of conditions.

Disciplines
Chemical Engineering

This working paper is available at ScholarlyCommons: http://repository.upenn.edu/cbe_sdr/8

Design and Control Using Stochastic Models of Deposition Reactors

Chemical and Biomolecular Engineering 459
Professor Leonard A. Fabiano
April 12, 2009

Peter J. Beltramo
Christina L. Bodarky
Helen M. Kyd
Department of Chemical and Biomolecular Engineering
University of Pennsylvania

Project Advisor: Dr. Talid R. Sinno
Project Recommended By: Dr. Talid R. Sinno, University of Pennsylvania

April 14, 2009
Department of Chemical and Biomolecular Engineering
School of Engineering and Applied Sciences
University of Pennsylvania
220 S. 34th Street
Philadelphia, PA 19104

Dear Dr. Talid Sinno and Professor Leonard Fabiano,

Enclosed in this book is the final copy of our Senior Design Project on Design and Control using Stochastic Models of Deposition Reactions. A software program was designed to model deposition processes through use of kinetic Monte Carlo simulations. The software employs model predictive control to dynamically optimize and control single and multiple component thin film growth by manipulating deposition parameters. The results obtained from studying the behavior of and the control of these systems were used to determine the economic feasibility of a start-up software company to sell this product. An analysis of the current marketplace revealed that although a few firms dominate the market, there is potential for a small start-up company to be competitive with a novel product. Depending on the growth of sales of this product, charging between $10,000 and $20,000 per license per year for this software will result in an IRR of 20% for the five year time horizon.

Sincerely,

Peter J. Beltramo Christina L. Bodarky Helen M. Kyd

Table of Contents

Abstract ... 1

1 Introduction ... 2

1.1 Thin Film Deposition Processes ... 2
1.2 Products of Deposition Processes ... 3
1.3 Surface Roughness Measurement Techniques ... 3
1.4 The Kinetic Monte Carlo Method ... 4
1.5 Validation for Use of Kinetic Monte Carlo Method ... 5
1.6 Project Charter ... 7
1.7 Innovation Map ... 9

2 Single Component Thin Film Growth Modeling ... 10

2.1 Single Component Model Assumptions ... 10
2.2 KMC Algorithm ... 13
2.3 Quantification of Deposition Process Optimality ... 16

3 Results of Single Component Thin Film Growth ... 18

3.1 Effect of Lattice Size ... 18
3.2 Effect of Temperature ... 21
3.3 Effect of Flux ... 24

4 Single Component Control of Thin Film Growth ... 27

4.1 Control Goals ... 27
4.2 Optimal Profile ... 27
4.3 Control Strategy ... 32

5 Results of Single Component Control of Thin Film Growth ... 35

5.1 Control Problem ... 35
5.2 Finding the Optimal Profile ... 35
5.3 Control Tests ... 39

6 Multiple Component Thin Film Growth Modeling ... 62

7 Results of Multiple Component Thin Film Growth 66
7.1 Effect of Lattice Size 66
7.2 Effect of Temperature 67
7.3 Effect of Flux 69
7.4 Effect of Gas Phase Composition 70
8 Multiple Component Control of Thin Film Growth 71
8.1 Control Goals 71
8.2 Optimal Profile 71
9 Results of Multiple Component Control of Thin Film Growth 77
10 Financial Analysis 85
10.1 Semiconductors Market Overview 85
10.2 Competitive Environment 87
10.3 Business Model 91
11 Conclusions 98
12 Acknowledgements 100
13 Bibliography 101
Appendix A Fundamental Equations 103
A.1 Master Equation 103
A.2 Arrhenius Relationship 103
Appendix B Financial Details 104
B.1 Reactor Quote 104
B.2 Clean Room Quote 105
B.3 Glove Box Quote 106
B.4 Aggressive Growth Cash Flow Summary 107
B.5 Conservative Growth Cash Flow Summary 108

Appendix C MATLAB Code 109

C.1 Single Component Code 109
C.2 Single Component Controller Code 112
C.3 Multiple Component Code 117
C.4 Multiple Component Controller Code 127
C.5 Initiate Surfaces 144

Abstract

The financial feasibility of the creation of a start-up company to sell software developed for the optimization and in-line control of thin film growth in deposition processes was investigated. An analysis of the current marketplace revealed potential for a small start-up company to be competitive with this novel product. The investigation concluded an IRR of 20% for a five year period before possible sale of the company. The kinetic Monte Carlo method was employed as the basis for all simulations in this work. This method retains atomic scale information while enabling simulation of process relevant features such as roughness, growth rate and efficiency. A model predictive controller was designed to reproducibly generate thin films with desired properties under a variety of initial condition disturbances for both single component and multi component systems. The substrate temperature and gas flux were employed as control variables. The control algorithms were investigated using a sensitivity analysis and shown to be robust under a wide range of conditions.

1 Introduction

1.1 Thin Film Deposition Processes

Thin film deposition is widely used to deposit a layer of solid material onto the surface of a substrate. Physical Vapor Deposition (PVD) and Chemical Vapor Deposition (CVD) are common variants of deposition used extensively in the semiconductor and coatings industries. CVD refers to a process in which gaseous reactive precursors are used to deposit a thin film of solid material on a substrate. A gaseous mixture containing atoms flows continuously through a controlled reactor environment where it comes into contact with the substrate on which reaction and deposition will occur. PVD refers to a process in which atoms are deposited onto the substrate surface by condensation and in which no reaction takes place. The temperatures at which these processes occur, as well as the concentration of the inlet vapor, are extremely important factors in determining the way in which atoms are deposited onto the substrate surface. Many different reactor geometries exist for these types of processes; however, the simulation developed in this report does not focus on any specific reactor or deposition process, as these parameters may be adjusted to fit the customers' needs.

1.2 Products of Deposition Processes

Deposition processes are commonly used to produce Micro-Electro-Mechanical Systems (MEMS), solar cells, advanced semiconductor substrates and integrated circuits. The electrical and mechanical properties of these products are highly dependent on surface uniformity, composition and microstructure (Granneman, 1993). Due to the need for smooth and uniform surfaces, measuring and controlling surface roughness is necessary for quality production of these items. The growth rate and reactant conversion are also important in the design of deposition processes in order to maximize throughput and reduce the waste of expensive materials such as gallium arsenide, GaAs.

1.3 Surface Roughness Measurement Techniques

A number of methods currently exist to measure the surface roughness of a thin film, including Atomic Force Microscopy (AFM) and Ellipsometry. AFM is implemented by bringing a microscope cantilever with a sharp tip into close proximity of the surface in order to detect forces such as Van der Waals forces. The topology of the surface, and therefore the surface roughness, can be measured by this method due to the principle that the measured forces change as the distance between the tip and the surface change. The tip scans the surface while maintaining a constant force measurement by preserving a constant distance from the surface, and the amount the tip must move to maintain constant force and distance is used to determine surface morphology and roughness (Carpick and Salmeron). Ellipsometry is used to measure surface morphology by detecting the change in polarization of light as it is reflected off of a surface and relating it to height of the substrate. While these methods of real-time roughness measurement exist, it is difficult to implement these techniques into a feedback control system.

The time necessary for these measurement techniques is too great to compete with the rates of molecular movement and growth on the surface. This conflict has led to interest in a control system based on accurate modeling of the dynamics of thin film growth.

1.4 The Kinetic Monte Carlo Method

In a generic deposition process, thin film properties such as surface roughness and growth rate are highly dependent on macroscopic system inputs such as substrate temperature and inlet gas concentration. The macroscopic scale determines how these input process parameters will affect the overall growth dynamics of a system. This is often modeled using Partial Differential Equations (PDEs) to describe the relevant momentum, energy and mass balances. However, in order to obtain precise control of film properties, the microstructure of the surface must also be considered; these properties are functions of much smaller length scales, typically on the order of several atoms. This dramatic decrease in length scale renders the use of continuum type PDEs invalid, and a microscopic technique must be used to model the growth and development of the surface microstructure.

The Kinetic Monte Carlo (KMC) method is appropriate for modeling at the atomic scale, and in general can be coupled to the macroscopic reactor-scale continuum description (Lou and Christofides 2003). The KMC method is an efficient stochastic technique for numerically solving the underlying "master equation" system, which describes the rates of all atomic scale events in the system as a function of time. (Appendix A.1, Van Kampen 1992). KMC simulations are used to predict average properties of the thin film, and at increasing lattice sizes, give a numerical solution to the master equation (Kang and Weinberg 1992).

Two balance criteria must be satisfied by the KMC method: (1) the ability to calculate the lifetime of each event, and (2) the guarantee of the stochastic nature of the system by using a random number generator to make certain each event is independent (Fichthorn and Weinberg, 1991). The stochastic nature of the KMC method is incredibly important in modeling a real system, as the exact movement on the atomic scale is probabilistic. However, despite the random stochastic nature, the model must still be able to accurately predict the thin film growth on a small scale, which is proven by the convergence of the KMC method to the master equation at increasing lattice sizes.

1.5 Validation for Use of Kinetic Monte Carlo Method

The ability to successfully simulate complicated chemical processes on multiple length and time scales is limited strongly by the available computing power. There are many simulation methods currently in use by researchers, each having applicability to different areas of interest. The most rigorous method of simulation is *ab initio*, which makes minimal assumptions and calculates movement based on first principles (i.e. including quantum mechanical forces between individual atoms). Due to the computational demands of this method, only atomic scale simulations in the femtosecond time range can be carried out in most cases. Molecular dynamics (MD) simulations, based on classical force fields, are computationally cheaper but are still limited to the nanoscale, and are far too slow for use in real-time applications such as model-predictive process control. The KMC method, on the other hand, retains the overall atomic picture, but removes the need to consider atomic vibrations, greatly increasing the simulation scope. Furthermore, KMC requires the specification of every possible atomic event allowable in the simulation. These events must be specified in advance as any omission of important events

can lead to severe model error. The precision of the rates imputed to the KMC model determine the validity of the simulation. Finally, as mentioned in the previous section, continuum models such as PDE's can be used for longer time scales and lengths, although this scale of modeling no longer consider atomic configurations explicitly. A graphical representation of the various forms of modeling and their applicability can be seen in Figure 1 on page 6.

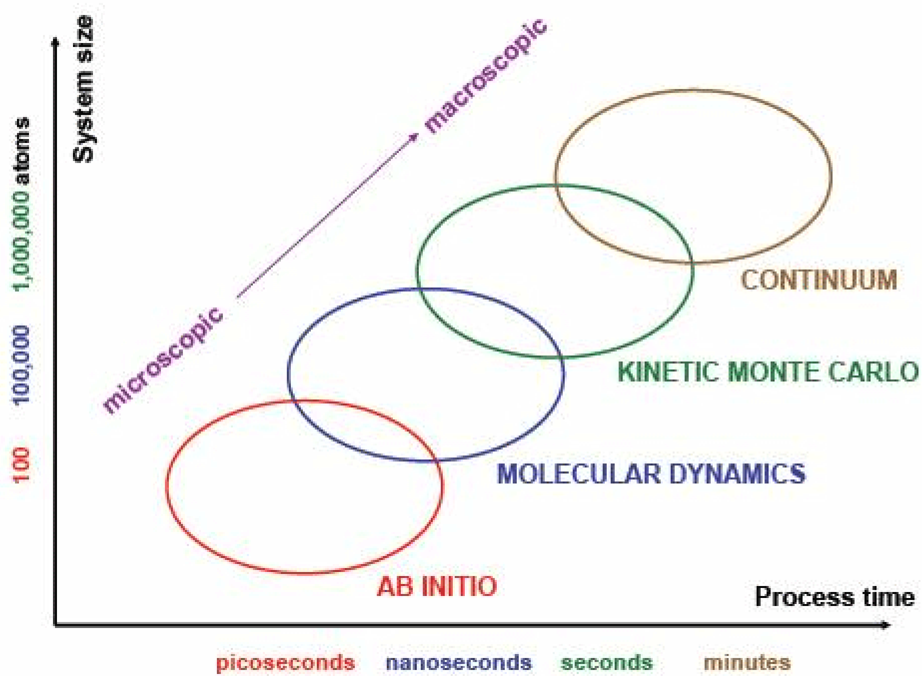


Figure 1: Simulation tools for chemical modeling (Nanostellar 2009).

1.6 Project Charter

The project charter describes briefly the goals and scope of the product designed in this report (Table 1). There are two primary goals of this project. The first is to develop software that can accurately model and control thin film growth properties of generic deposition processes consisting of single or multiple component gas phases. This piece of software is completely written and executed in MATLAB. The code is provided in Appendix C. The second goal is to provide a feasibility plan for a start-up company to develop and sell this software to companies that employ deposition reactors.

Project Name	Design and Control using Stochastic Models of Deposition Reactors
Project Champions	Dr. Talid Sinno
Project Leaders	Peter Beltramo, Christina Bodarky, and Helen Kyd
Specific Goals	Develop software to model and control single particle and multi-component thin film formation
Project Scope	In Scope • Simulate a generic deposition process • Observe effects of temperature, lattice size, concentration profiles, and interactions between molecules on growth rate and surface roughness • For multi-component systems model different atomic growth behaviors • Develop a controller to bring a system to a desired state • Evaluate product fidelity • Develop a business model to market software Out of Scope • Model with specific substrate and chemical properties • Model specific reactors and processes, such as CVD and PVD
Deliverables	• Software package to model and control thin film formation • User interface • Product testing
Timeline	12 weeks

Table 1: Project Charter

1.7 Innovation Map

Presented below is the innovation map for a control product for a deposition simulation.

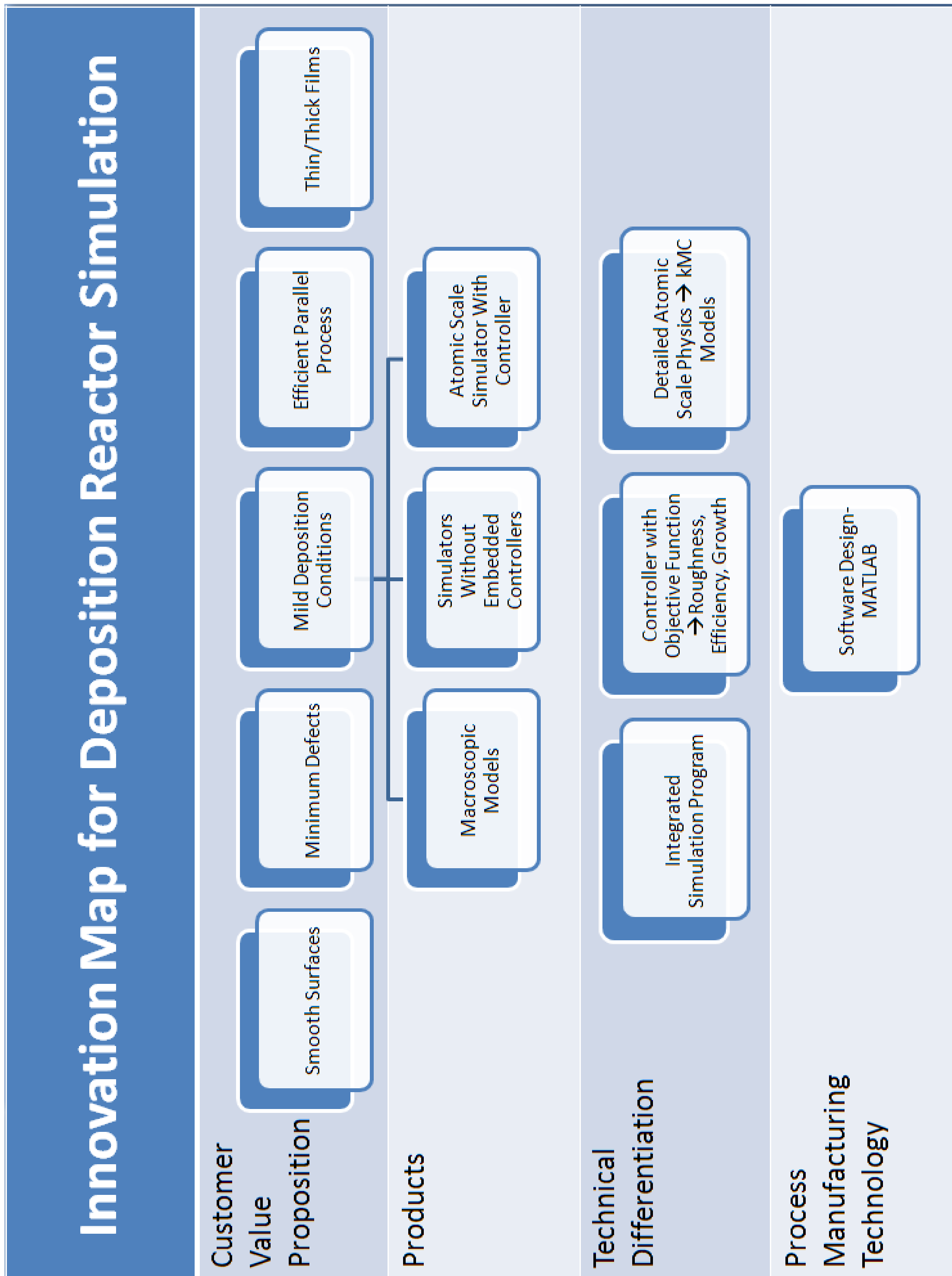


Figure 2: Innovation map.

2 Single Component Thin Film Growth Modeling

2.1 Single Component Model Assumptions

The KMC method used in this project is of the lattice variety in which all atomic positions are restricted to a rigid lattice in order to maximize computational efficiency. The simulation system consists of a regular square lattice containing N x N sites on which three distinct processes can occur; (1) adsorption of an atom from the gas phase onto a lattice point on the substrate surface, (2) desorption of an atom from a lattice point to the gas phase, and (3) migration of an atom from one lattice point to another on the substrate surface (Figure 3). In the proposed software simulation, different rates are generated for each of these three processes depending on their position on the lattice due to the effects of bonding to their nearest neighbors. On the square lattice employed in this work, a given atom (or molecule) can possess at most four nearest neighbors in the plane of the atom. Note that nearest neighbors also exist in the planes above and below a given atom, but these are considered separately. Second nearest neighbor interactions are not considered in the single component model, but such interactions can be readily included as required for specific material systems (Figure 4). In this section, a single component system is considered in which all atoms are assumed to be the same.

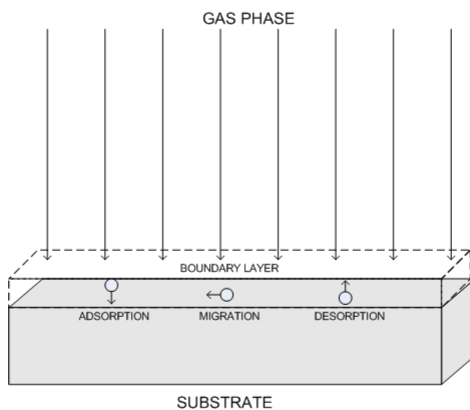


Figure 3: KMC events depicted from left to right: adsorption, migration, and desorption. Events occur within the boundary layer.

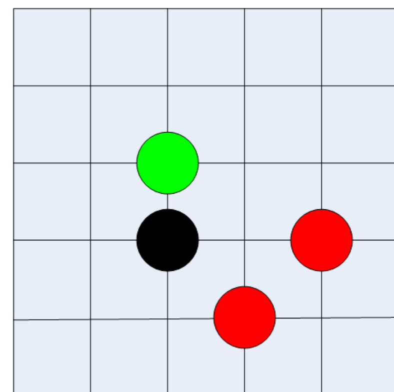


Figure 4: Neighbor interactions: primary neighbor (green) interactions are considered and secondary neighbor (red) interactions are not.

The adsorption flux, specified in units of monolayers/site-second (ML/site-sec), is a user defined system input. Note that in the current model, the sticking coefficient for adsorption is assumed to be one for all conditions; once again, this parameter can be modified according to specific material properties. Since the sticking coefficient is equal to one the adsorption rate is directly equal to the flux of atoms over the surface and can be described by:

$$R_{ads} [=] ML / site \cdot \sec \qquad [1]$$

and is assumed to be independent of lattice position.

The desorption flux, given in units of ML/site-sec, is determined by the probability that an atom has enough energy to overcome an energy barrier which arises from molecular bonding energies as described by the Arrhenius equation (Appendix A.2, Shitara 1992). The probability of a particle overcoming this barrier takes the form of a Boltzmann distribution, and is described explicitly by:

$$w_{des}^{A}(n) = k_{des}^{A} \exp\left(\frac{-E_{des}^{A} + nE_{n}^{A}}{kT}\right) \qquad [2]$$

where the pre-exponential term $k^{A}{}_{des}$ is a desorption frequency factor, $E^{A}{}_{des}$ is the desorption surface energy (which accounts for bonding to atoms in the plane below the one containing the desorbing atom), n is the number of nearest neighbors, $E^{A}{}_{n}$ is the (in-plane) neighbor bond energy, k is the Boltzmann constant, and T is the temperature of the substrate. Five different rates of desorption could possibly be described for a certain lattice point because the number of (in-plane) nearest neighbors could vary from zero to four. An increase in the number of nearest neighbors causes the atom to be bonded more strongly in its lattice position, and therefore the rate of desorption decreases. This rate is also temperature dependent, and therefore will vary in the model as temperature is manipulated to control the thin film properties. A larger temperature

will cause an increase in the rate of desorption, and therefore increase the probability of this event occurring leading to a decreased growth rate.

The rate of migration, given in units of particles/site-second, is determined by the same type of Boltzmann distribution and neighboring bond energies as desorption, because migration also involves the breakage of nearest neighbor bonds. The rate of migration is given by:

$$w_m^A(n) = \frac{k_m^A}{4}\exp\left(\frac{-E_s^A + 0.5nE_n^A}{kT}\right) \quad [3]$$

where the pre-exponential term $k^A{}_m$ is a diffusion frequency factor, $E^A{}_s$ is the diffusion surface energy barrier (which reflects the breakage of bonds to atoms lying in the plane below the migrating atom), n is the number of nearest neighbors, $E^A{}_n$ is the (in-plane) neighbor bond energy, k is the Boltzmann constant, and T is the temperature of the substrate. Four different rates of migration could possibly exist for a certain lattice point. These rates occur because the number of (in-plane) nearest neighbors could vary from zero to four for the lattice point in which the atom currently exists. Atoms that are surrounded by four in-plane neighbors are "blocked" within their plane and can only migrate up unless one of the neighboring particles moves.

In the model employed here, an atom may migrate in four directions (i.e. to an adjacent unoccupied nearest neighbor position). The assumption that no diffusion can occur to occupied neighboring sites reflects the fact climbing up to a higher atomic plane is an energetically costly process. Moreover, atoms that hop to adjacent sites that unoccupied for more than five monolayers are assumed to become desorbed. However, the atom may only migrate within the same monolayer, down a maximum of five monolayers, or migrate up one monolayer. It is kinetically favorable for a migrating atom to move in the direction with the lowest energy barrier, and therefore the rate of migration will decrease as the number of nearest neighbors

increases. All rates also exhibit temperature dependence, and therefore will vary in the model as temperature is manipulated to control the thin film properties. A larger temperature will cause an increase in kinetic energy, therefore the rate of migration will increase due to the increase in the probability of migration occurring. Generally, higher molecular mobility will result in smoother surfaces because particles tend to diffuse from lower bonded environments to higher ones, increasing the flatness of the deposited film.

For both the rate of migration and the rate of desorption, the pre-exponential factors as well as the values for the bond energies are dependent on the material. Commonly used parameters, parameters for a created gallium arsenide (GaAs) model, and values for the proposed model are provided in Table 2 (Shitara, 1992).

Common Parameters		GaAs	Proposed Model
Frequency Factor	k^A_m, k^A_{des}	$5.8\text{x}10^{13}$	$1\text{x}10^{13}$
Diffusion Surface Energy Barrier (eV)	E^A_s	1.82	1.58
Neighbor Bond Energy (eV)	E^A_n	.27	.27
Desorption Surface Energy (eV)	E^A_{des}	2.32	2.32
Temperature (K)	T		500-900
Adsorption Rate (ML/site-sec)	r_{ads}		1-9

Table 2: GaAs and proposed model specific KMC algorithm parameters.

2.2 KMC Algorithm

The KMC algorithm implemented for this study begins by specifying a rate for every possible event once an initial configuration is chosen. Each of the NxN lattice sites has up to six

rates: one adsorption rate, one desorption rate, and four migration rates (one for each adjacent lattice point). This vector of rates is then converted to a vector of wait times. The wait time contains two parts, the current system (elapsed) time and an additional time that represents the time it will take for the next event to take place. The system time is the sum of the times for each previously executed event. The wait time is described by the equation:

$$wait_time = t + \frac{-\ln(u)}{r} \quad [4]$$

where τ is the system time, *u* is a random number, and *r* is the rate of the specific event from the rate matrix. The random number, *u*, is uniformly distributed from zero to one, so that the ln(*u*) term represents a Poisson distribution in which each of the individual events is an independent random process.. The random number is generated using the Mersenne Twister random number generator which is built into MATLAB. The wait time vector is then ordered from smallest to largest time, and events are selected in that order. This algorithm naturally introduces stochasticity into the system, and therefore captures the noise and fluctuations associated with dynamics at the atomic or molecular scales. Note that despite the element of randomness introduced into the wait-time, eq. (4) still tends to shift events with larger rates to the front of the wait-time vector and therefore correctly allows for faster events to be executed more often. . It should be noted that for small systems the average obtained from KMC simulations is not necessarily the same as the value obtained from deterministic models based on ODEs or PDEs – this is an important motivator for employing stochastic models.

Once the rate on the top of the wait time vector is executed, the time for this event is added to the system time, the rates for the affected sites are regenerated, and the wait time vector is updated. The process is then repeated until the system time meets the total time set by the user

or a specified height has been reached. A block diagram is presented in Figure 3 to depict the flow of the simulation as it models the system with KMC kinetics.

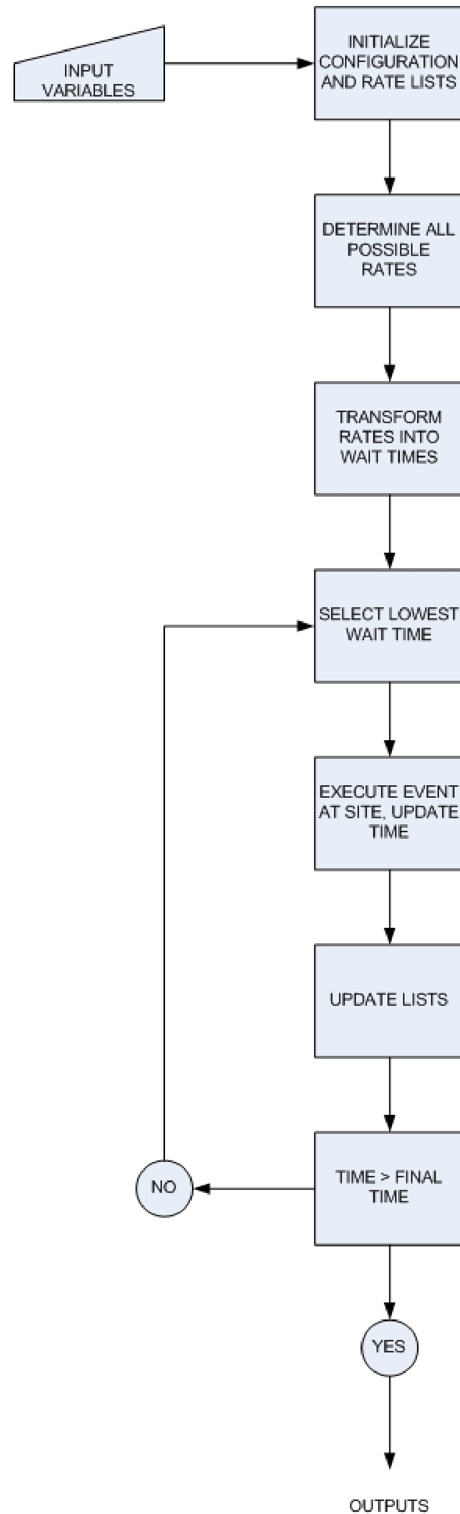


Figure 3: A block diagram showing the execution of the kinetic Monte Carlo method.

2.3 Quantification of Deposition Process Optimality

There are several factors that must be considered in the design of a deposition process. Deposition film properties are of course critical for any given application. For example, semiconductor substrates must be extremely flat and defect-free given the very small scale of microelectronic devices. In this report, the focus is solely on the surface roughness as a measure of film quality, although other morphological properties can also be computed with more complex deposition models. In addition to film quality, process throughput and reactant use efficiency also must be considered in the design of operating conditions. Capital expenditures for each reactor system are substantial and high throughput can be an important factor in determining whether the process is profitable or not. Finally, in the case of expensive or highly toxic reactants, care must be ensured to utilize as much of the feed atoms as possible is utilized during the deposition process. Often, contamination issues prohibit the use of a simple recycle of the feed stream making. Each of these factors must be considered in a controller designed to optimize a deposition process. In the following discussion, each of these factors is quantified so that they may be incorporated into a control algorithm.

Roughness in this model is defined to be the standard deviation of the height, where the height at each lattice point is compared to the average height of all lattice points, or more explicitly:

$$r = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}[h_{ij}-\bar{h}]^2}{NxN}} \quad [5]$$

where N is the total number of lattice points, h_{ij} is the height at a given lattice point located at position i and j, on the square lattice and h_{avg} is the average height of all lattice points. Note that this definition of roughness is atomically resolved, and therefore cannot be compared to actual

measurements, particularly in real time. A more accurate simulation of feedback control based on actual measurements may be realized by defining roughness on a somewhat coarser scale. For example, to more closely mimic the output of an AFM or other tool, roughness can be calculated in the KMC simulations by using average height of varying sized patches of the lattice, instead of each lattice point height itself.

The film growth rate, given in units of ML/second, is a measure of how quickly the thin film is developing. An objective of deposition is to grow the film as quickly as possible with the least amount of wasted material, without severely compromising the roughness of the surface. In this model, the average height at the current time is compared to the average height at a previous time, typically one second before the current time, to give the rate at which the thin film is growing. The equation used to describe this rate is:

$$gr(t) = \bar{h}(t) - \bar{h}(t-1) \qquad [6]$$

where $\bar{h}(t)$ is the average height at the current time and $\bar{h}(t-1)$ is the average height at a second previous to the current time. A typical value for growth rate for a deposition process of atomic materials ranges from 2-10 ML/second.

The amount of material that is lost due to desorption is measured by the efficiency, which is defined as the percentage of particles that remain adsorbed to the surface. This efficiency also affects the growth rate. When a significant number of particles are desorbing, growth can either be slowed or even reversed. Explicitly,

$$efficiency = \frac{n_a - n_d}{n_a} \qquad [7]$$

where n_a is the number of particles that adsorbed and n_d is the number of particles that desorbed. The efficiency can be calculated per second or over the course of the entire deposition process.

3 Results of Single Component Thin Film Growth

Prior to studying how to control the various aspects of thin film growth, it is necessary to perform test simulations without control to gain an understanding of how the system behaves under different operating conditions. For all of the simulations that follow, the previously discussed model physics were used and kept constant. Also, to establish a universal basis to compare each simulation a design specification of depositing a 100 nm film on the substrate surface was chosen. Using a bond length of 2.8 Å for a GaAs film as a rough guide, this corresponds to a 357 monolayer deposition (Azevedo 2005). Based on this criterion, the effect of lattice size, temperature, and flux on the dependent system parameters, roughness, time, and efficiency were studied.

3.1 Effect of Lattice Size

A large constraint on modeling methods such as the simulation described in this report is the computational demand, and thus the time required to fully simulate a deposition process to the desired deposition height. The simulation must converge to the same average thin film properties given by the master equation in order to accurately describe the surface morphology and microstructure. This convergence occurs at larger NxN lattices sizes, however increasing the lattice size increases computational demand. It was determined that a simulation carried out on a 100x100 lattice accurately depicted thin film growth without extreme computational demands. This was determined by running multiple simulation trials on 10x10, 20x20 and 100x100 matrices and examining one of the sensitive output variables, roughness. Ten simulation trials on the 10x10 matrix yielded an average standard deviation of roughness over all time steps of 1.04. Five simulation trials on the 20x20 matrix yielded an average standard

deviation of roughness over all time steps of 0.36. Four simulation trials on the 100x100 matrix yielded an average standard deviation of roughness over all time steps of .08. The low standard deviation of roughness indicates that the 100x100 matrix more accurately captures the thin film properties of the generic deposition process, where as the 10x10 and 20x20 do not as accurately capture these properties. Figure 4 and 7 on the following page show the average 10x10 and 100x100 roughness plots with the standard deviation of each time step. The average computation times for each matrix are shown in Table 3. It can be seen that as the lattice size increases, the time for computation increases dramatically, however the 100x100 matrix is still within a reasonable time constraint.

Lattice Size	**Computation Time**
10x10	32 seconds
20x20	125 seconds
100x100	1 hour and 40 minutes

Table 3: Computation times for single component KMC simulation on different lattice sizes.

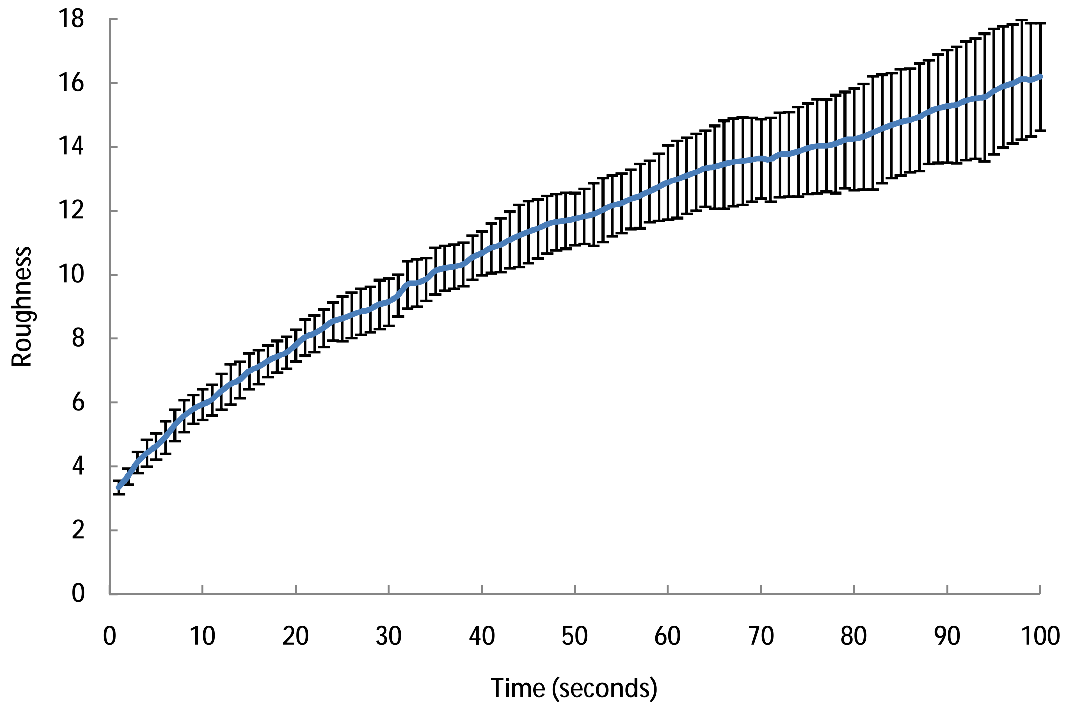


Figure 4: Average roughness vs. time for ten 10x10 KMC simulations carried out at 500K and adsorption rate 5 particle/site-second. Error bars depict the standard deviation of all ten simulations at each second.

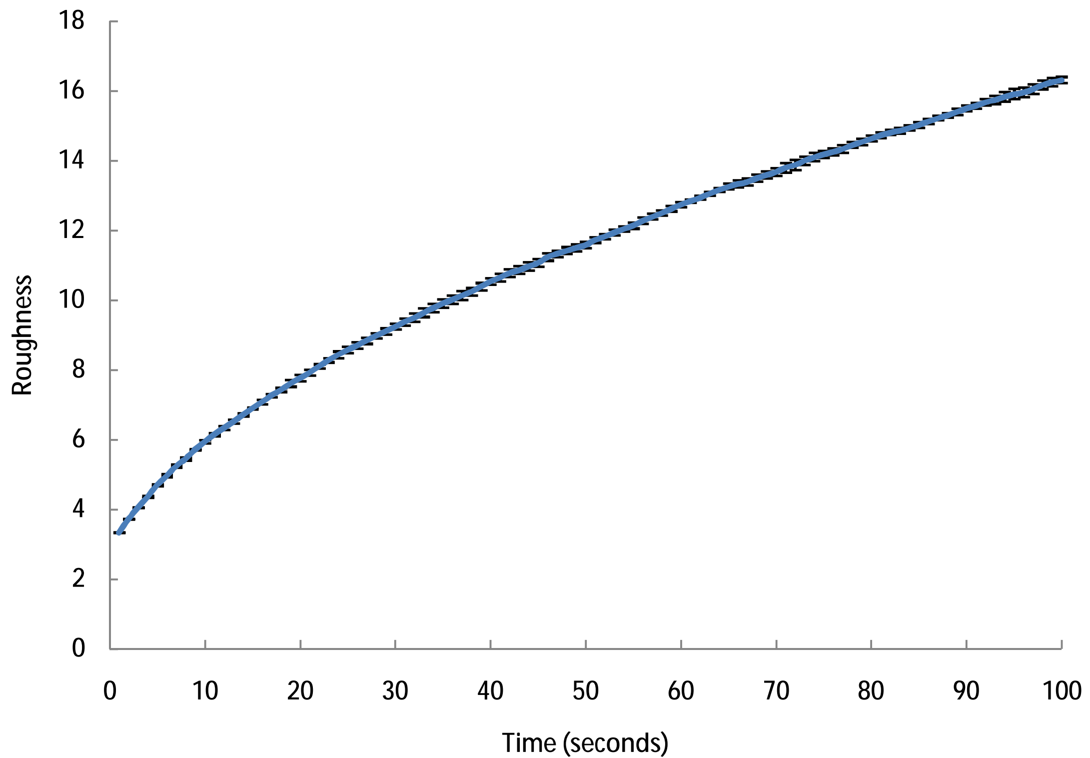


Figure 5: Average roughness vs. time for four 100x100 KMC simulations carried out at 500K and adsorption rate 5 particle/site-second. Error bars depict the standard deviation of all four simulations at each second.

3.2 Effect of Temperature

The temperature of the substrate affects the roughness, efficiency, and growth rate by changing the rates of desorption and migration, described previously by equation 2 and 3. Since these rates exhibit Arrhenius behavior, increasing the temperature increases the rate of migration and the rate of desorption exponentially. With the above proposed values for bonding energy (see Table 2 on page 13) simulations carried out within a temperature range of 500 – 900K provided a broad spectrum of rates, as shown in Figure 6.

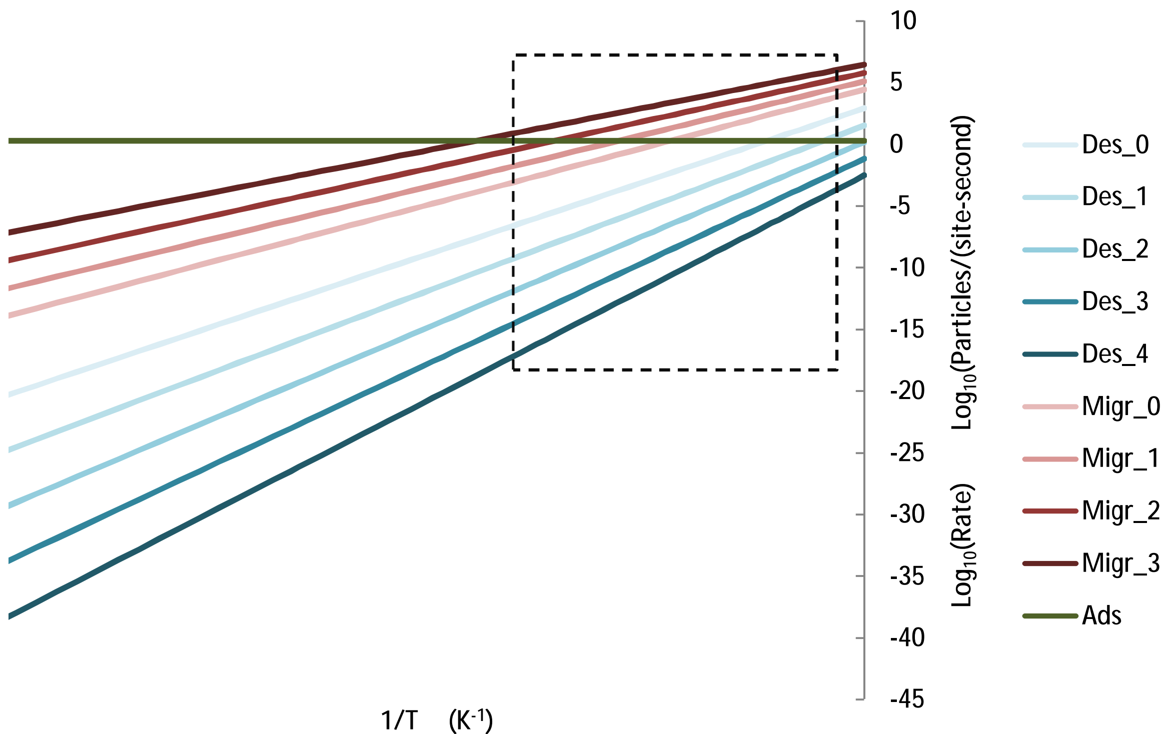


Figure 6: Log of rate vs inverse temperature for different rates of adsorption, migration and desorption. Des_0-Des_4: denotes desorption with zero through four neighbors, Migr_0-Migr_3 denotes migration with zero through three neighbors, Ads denotes rate of adsorption which on a log scale remains almost constant as the values vary only from one to nine. The dotted box indicates the temperature range (500-900K) where all simulations were carried out.

As shown, increasing the temperature increased the rates of migration and desorption on the surface of the thin film. The increased mobility of the surface atoms has a smoothing effect on the surface over time. In Figure 7, it can be seen that increasing the temperature caused decreased roughness and smoother surfaces at a fixed rate of adsorption. Although desorption increases with temperature, over the range of temperatures studied desorption was still minimal compared to adsorption, therefore there was not a significant change in the time required to deposit the film. For example, when increasing the temperature from 500 K to 800 K at a constant adsorption rate of five ML/site-sec the time to deposit a 100 nanometer film only increased from 76 to 77 seconds. The quantified change in roughness at higher temperatures can be easily seen in Figure 8 by plotting the final surface height on a constant color bar scale.

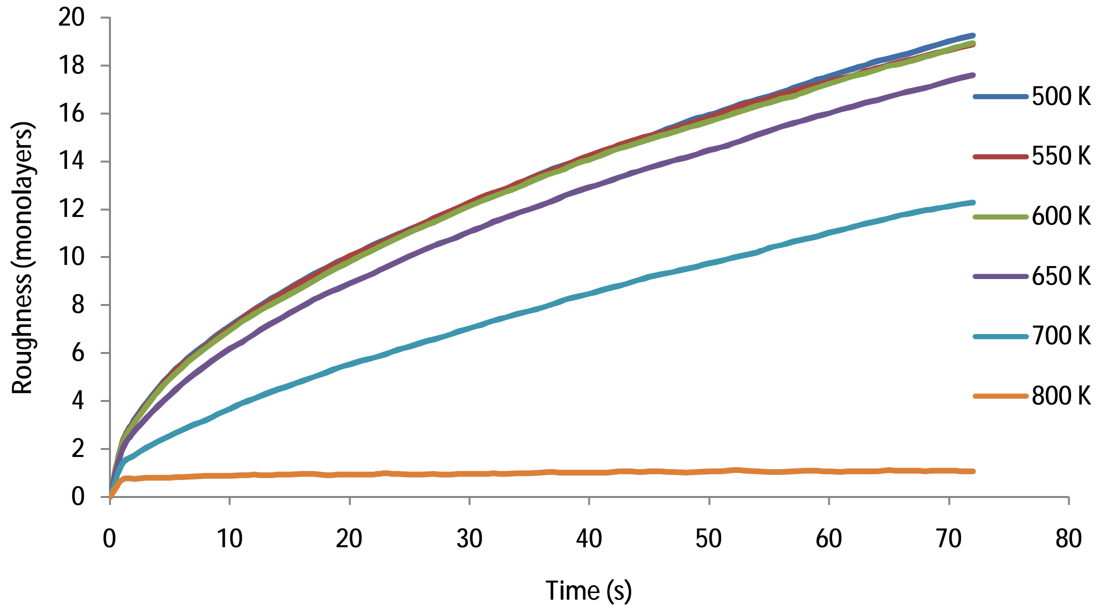


Figure 7: Roughness versus time for varying temperatures and a constant flux (5 ML/site-sec).

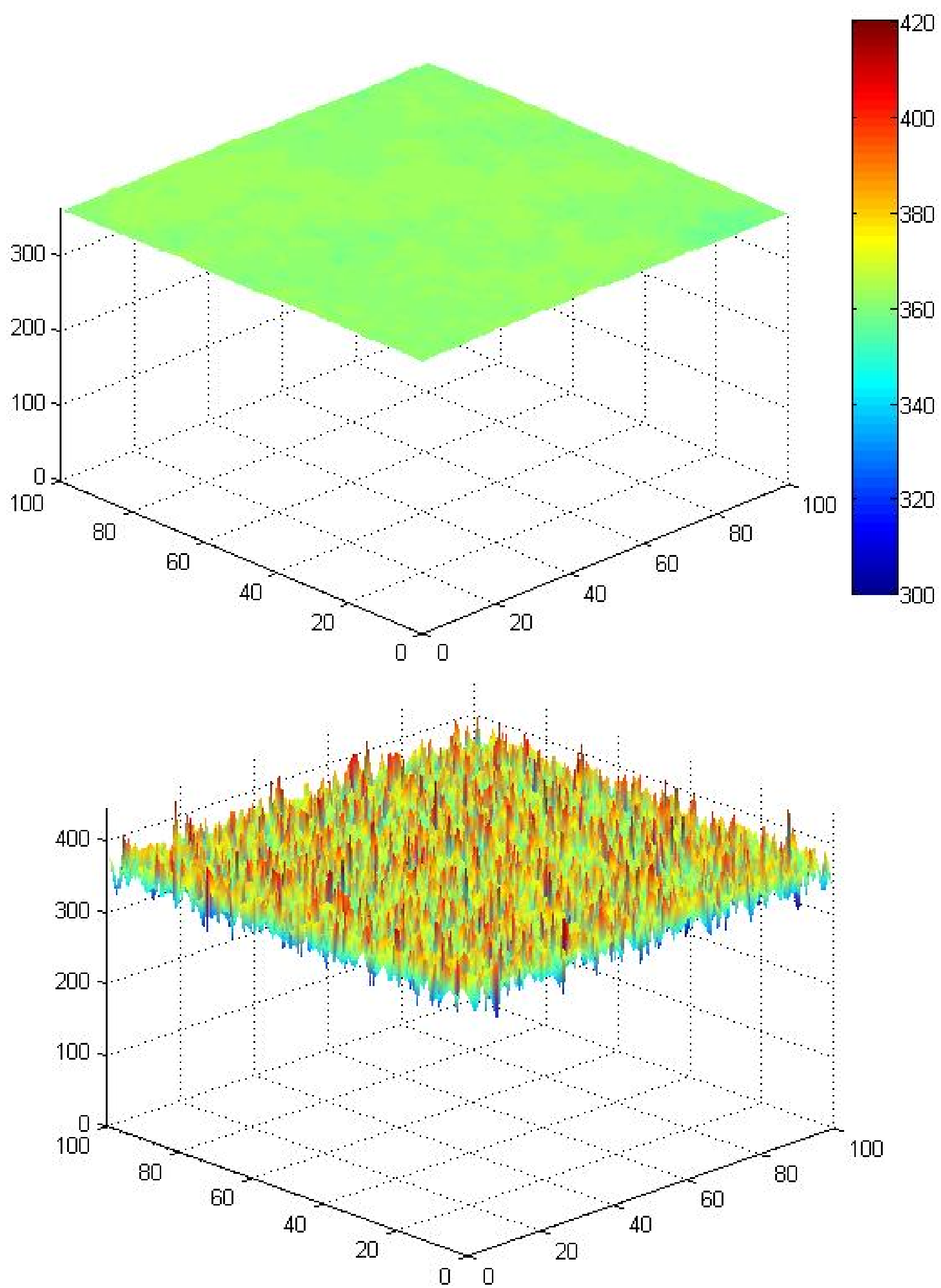


Figure 8: Surface plots of a 100 nanometer film deposited with constant adsorption rate (5 ML/site-sec) at 800 K (top) and 500 K (bottom).

3.3 Effect of Flux

The gas phase flux directly influences the rate of adsorption of particles onto the surface as the user specifies the number of particles to be adsorbed onto the surface per site per second. The relationship between external mass flow rates in a reactor vessel and the flux experienced on the surface is dependent on the specific reactor geometry and therefore only representative values were used in the present simulations. These values ranged from 1 to 9 deposition events per site per second of simulation time.

In general, higher fluxes lead to faster growth rates, therefore a decreased amount of time necessary to produce a 100 nanometer film. Simulation results show the time to deposit a 100 nanometer film was inversely proportional to the adsorption rate, as seen in Figure 9.

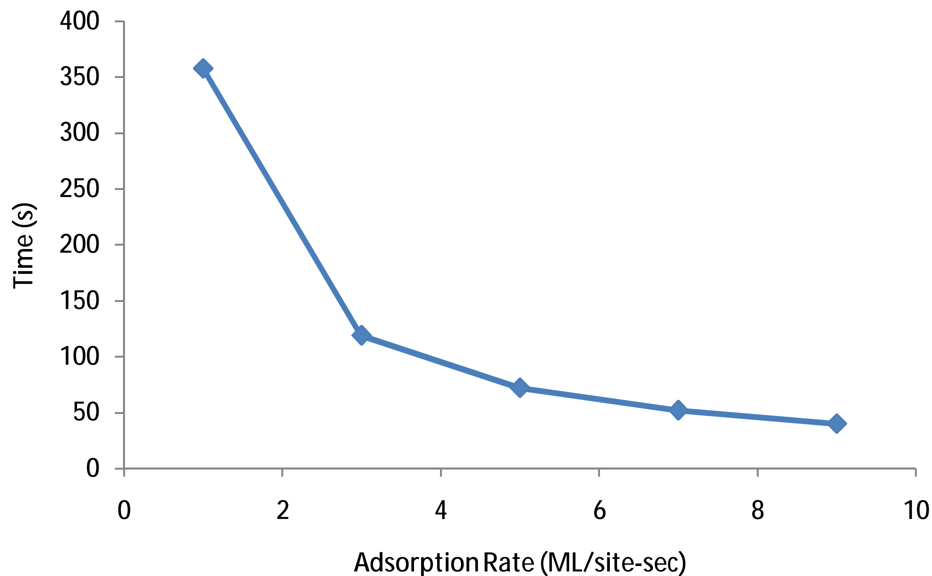


Figure 9: Adsorption rate versus time for constant temperature.

Based on the data, the relationship between flux and time to deposit a specified monolayer thickness can be expressed by the following equation, where t is time in seconds, N is the desired monolayer thickness and r_{ads} is the monolayers deposited per site per second:

$$t = \frac{N}{r_{ads}} \quad [8]$$

The flux does not directly affect the rate of migration or desorption, however it does affect the amount of lost material and thus the efficiency of the system. As discussed previously, atoms were constrained to migrate to a neighboring site that was less than six monolayers down or one monolayer up. Low fluxes promoted neighbor relationships that did not fit into this constraint resulting in increased desorption. However, at lower temperatures the overall rate of desorption was depressed making the relationship between flux and efficiency weaker. As shown in Figure 10, for 100 nanometer layer growth at constant temperature increasing the flux at constant temperature increasing the flux decreased the amount of desorption most significantly in the high temperature regime.

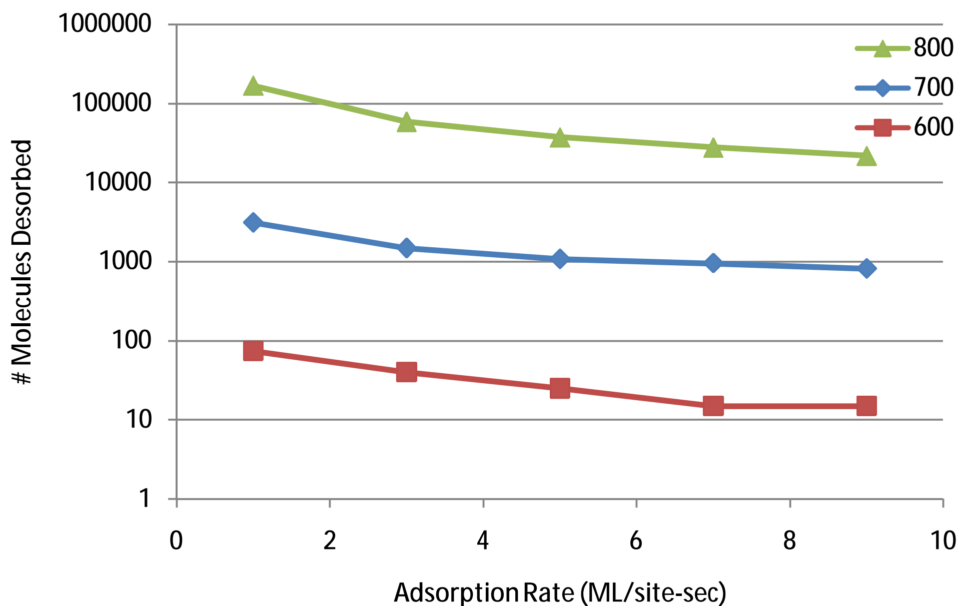


Figure 10: Effect of flux on efficiency at varying temperatures.

The relationship between roughness and flux is more nuanced and depends on the temperature. At low temperatures the system is dominated by adsorption; migration and desorption events are trivial. Random adsorption on the surface resulted in similar roughness evolution, regardless of the adsorption rate, at 500 K and 600 K. At high temperatures, the adsorption rate again had a minimal effect on the surface roughness. Increased kinetic energy in this regime allowed atoms to migrate across the surface very rapidly, smoothing the surface immediately regardless of the flux of atoms adsorbing. At temperatures where adsorption and migration were comparable, the flux had a direct effect on the surface roughness. This is shown clearly in Figure 11, where the roughness increased with adsorption rate at 700 K.

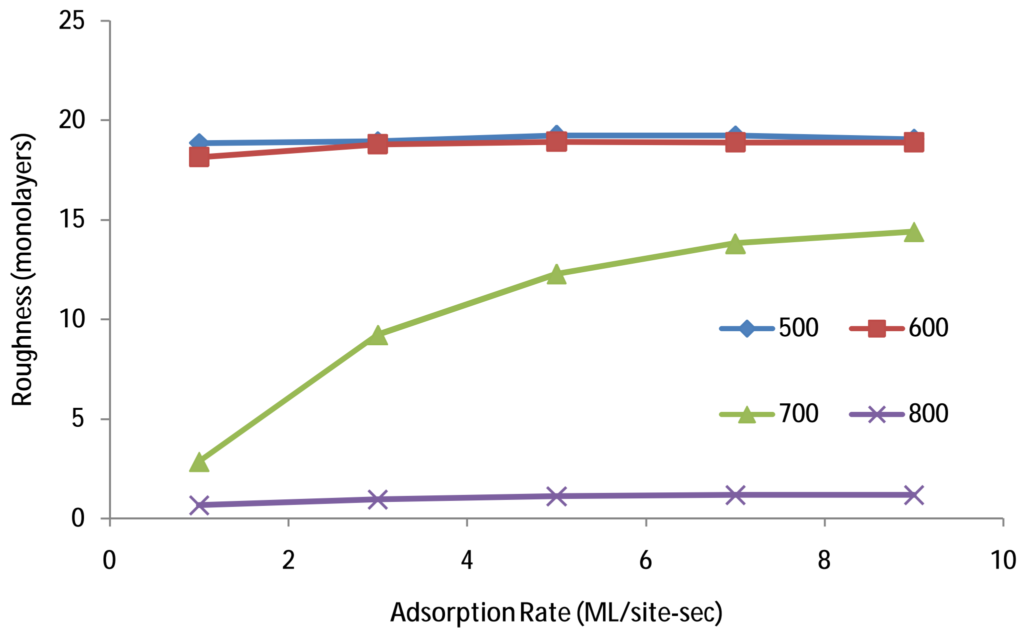


Figure 11: Effect of flux on roughness at varying temperatures.

4 Single Component Control of Thin Film Growth

4.1 Control Goals

Deposition processes have a wide range of potential applications, and as a result each process has its own set of optimal conditions. For instance, junctions between two layers of chemicals may need to be very well defined, material must be deposited to a precise height in a specific timeframe, or the surface of the material must be very smooth. Each characteristic can have an effect on the optical, electronic, and chemical properties of the thin film; therefore it is necessary to develop rigorous control schemes that can be applied to a wide range of deposition systems. Three dependent system parameters have been established for the control system: roughness, growth rate, and efficiency. Roughness must be minimized in a controller for a system requiring a smooth surface. In industrial settings where throughput is directly correlated with profits, it is desirable to have rapid thin film deposition growth. When depositing an expensive chemical, efficiency, or the number of atoms that remain adsorbed to the surface, must be maximized. With these control goals in mind, the general deposition goal of depositing a 100 nanometer film was analyzed.

4.2 Optimal Profile

The first step in developing a control system is to establish an optimal profile curve for the controller. In order to accomplish this, simulations were performed for two cases. In the first, a constant temperature and flux were applied throughout the process, while in the second the temperature and flux were allowed to change once. Clearly, more flexibility in process parameters should generally lead to more optimal evolution, but computational limitations

restricted the scope of our study in the present report. All initial surfaces were perfectly smooth with an initial height of zero monolayers. For accurate comparisons between simulations, an objective function was created to turn the dependent parameters of roughness, growth rate, and efficiency into a quantitative score for each simulation. Since only optimal process conditions were being identified, the variables were measured at the end of each simulation. The objective function has the following general equation:

$$OF = \frac{A \times (t_f)}{(t_{max} - t_{min})} + \frac{B \times (r_f)}{(r_{max} - r_{min})} + \frac{C \times (d_f)}{(d_{max} - d_{min})} \qquad [9]$$

Each parameter, time (t), roughness (r), and desorption (d) was converted into a fraction based on the minimum and maximum values observed in the data set. As discussed in section 3.3, time is directly correlated with growth rate. Roughness is defined in equation 9. The number of desorbed atoms is used to evaluate efficiency. Since each simulation completed when the same average height was reached, the total number of adsorbed atoms remained relatively constant from simulation to simulation, therefore desorption was the only contributor to the inefficiency of the system. The weighting factors A, B, and C can be altered to developed different objective function surfaces based on the relative priorities placed on each parameter.

The optimal profile was found at the point where the objective function is minimized in a given set of data. To show how the objective function surface changes with each parameter, plots of the objective function surface evaluated with time, roughness, and efficiency prioritized independently follow in Figure 12, Figure 13 and Figure 14. Simulations were performed with constant adsorption rates (1, 3, 5, 7, and 9 ML/site-sec) at constant temperatures (500, 550, 600, 650, and 700 K). As seen in the surface plots, the minimization of the three different parameters, time, roughness, and material lost, require different operating conditions. To minimize process

time, high adsorption rates are necessary, while temperature has no effect. Roughness is minimized at low adsorption rates and high temperatures. Efficiency is maximized at low temperatures, regardless of adsorption rate.

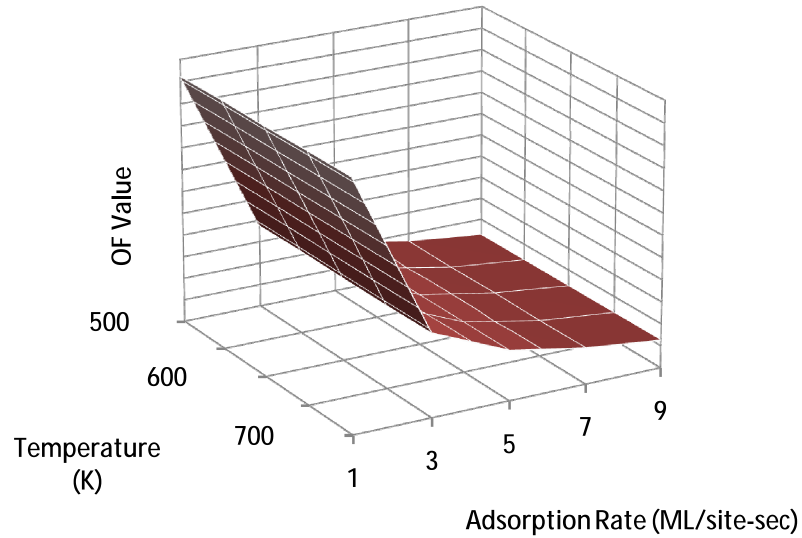


Figure 12: Objective function surface plot with time prioritized (A = 1, B = C = 0).

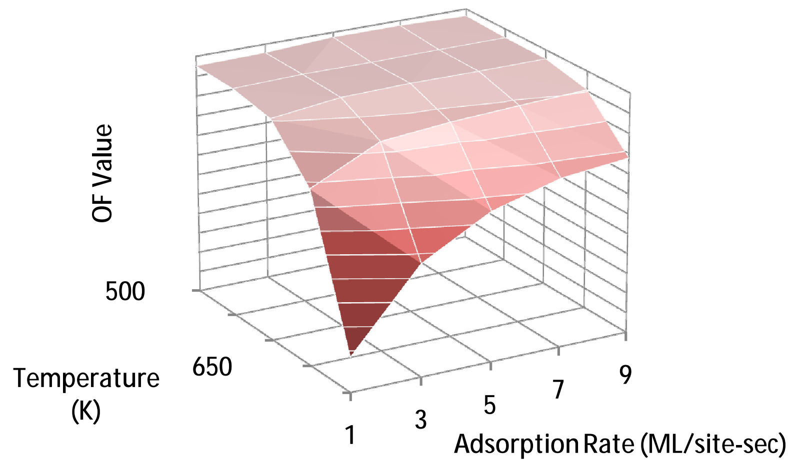


Figure 13: Objective function surface plot with roughness prioritized (B = 1, A = C = 0).

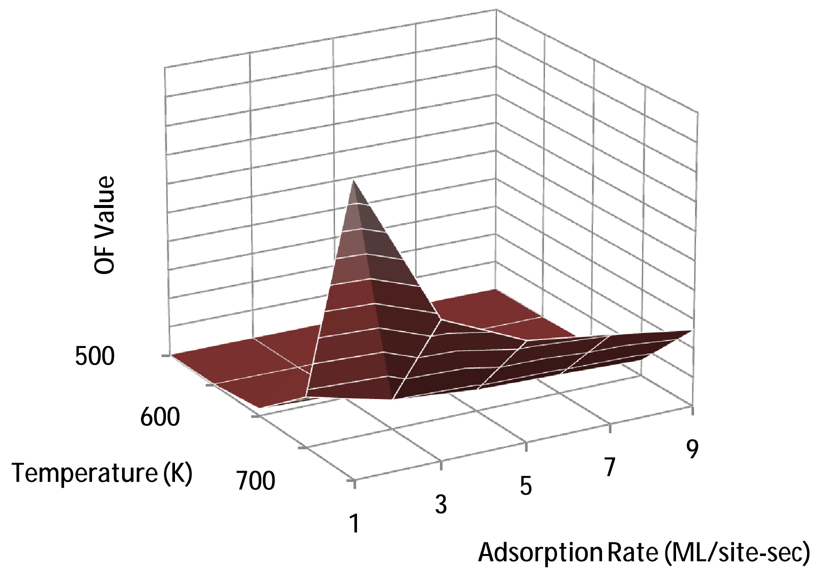


Figure 14: Objective function surface with priority efficiency prioritized (C = 1, A = B = 0).

By manipulating the parameters A, B, and C the objective function surface changes to potentially reveal minimum values at different process conditions. A situation where roughness, growth rate and efficiency are weighted equally (A = B = C = .33) is shown in Figure 15. The objective function is minimized with an adsorption rate of 9 ML/site-sec at a temperature of 700 K.

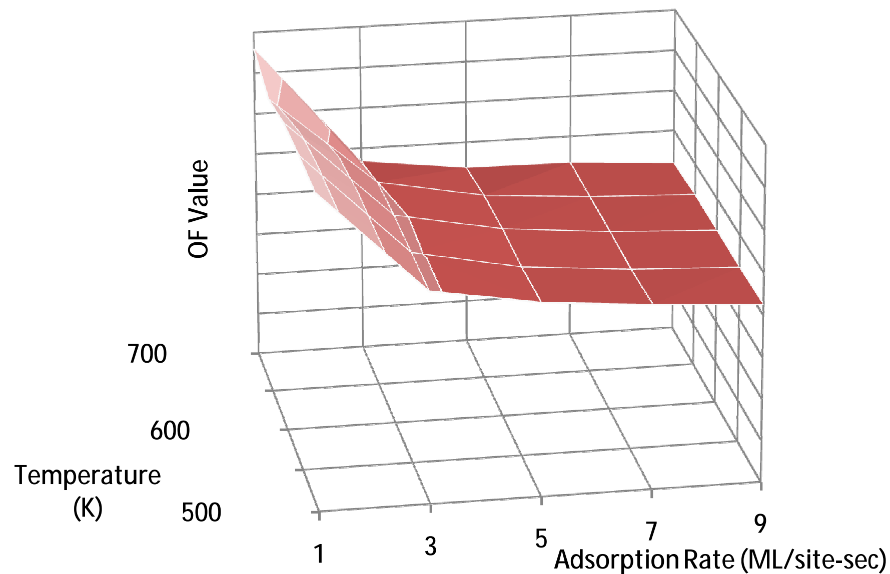


Figure 15: Objective function surface with equal priority on time, efficiency and roughness (A = B = C).

Once the objective function is minimized with a given set of parameters, the surface evolution from that process condition is used for the optimal profile in the controller. Since the objective function was minimized, the height and roughness evolution over time for the optimal conditions are the ideal curves that any control implementation should direct the system towards. A method for approaching the optimal curve has been developed and is presented in the next section.

4.3 Control Strategy

In order to appropriately control the complex deposition process a model predictive controller was developed. This type of control strategy predicts how the dependant variables will behave in response to changes in the controlled independent process parameters. The prediction is obtained by running a small scale simulation and determining the best parameters to select for the next time period (i.e. before the next control action is taken). A reference trajectory, the optimal evolution profile, was first developed to decrease the computation time required by solving the receding horizon problem over a short time scale rather than to an end point value. During each control cycle the controller first samples the current surface configuration to determine its roughness, height and molecular desorption events. It then compares these values to the expected values on the developed optimal profile. If the error is less than a set value, the controller leaves the temperature and flux at the previous value. If the error is greater than a set value it then manipulates the flux and substrate temperature to see how the roughness, growth rate and efficiency are affected over the next second of growth. The controller performs these operations using ten 10x10 lattice samples for a variety of manipulations of the parameters. These manipulations are summarized in Table 4.

Test	Controller Action
1	T, A
2	T + 10, A
3	T - 10, A
4	T, A + 1
5	T, A - 1
6	T + 10, A + 1
7	T - 10, A + 1
8	T + 10, A - 1
9	T - 10, A - 1

Table 4: Controller test performed, T: no change in temperature, A: no change in adsorption, T + 10/T-10: current temperature plus/minus 10 K, A– 1/A+1: current adsorption rate plus/minus 1 ML/site-sec.

The use of ten 10x10 matrices is valid as the average results of these simulations converge to the result of a 100x100 matrix, as seen in Figure 16. Note that the choice of control actions listed in Table 4 is limited by the ability of the reactor to adjust operating conditions. In other words, it is not generally possible (nor desirable) to make very large temperature changes in a small amount of time. These restrictions can sometimes lead to limitations in the performance of the controller.

Once the simulations have been completed, the controller chooses the best parameters to run the model on for the next time period, which was chosen to be one second unless otherwise stated. Once the run has completed it once again compares the new surface to that expected with the developed optimal profile. This controller strategy is shown in Figure 17.

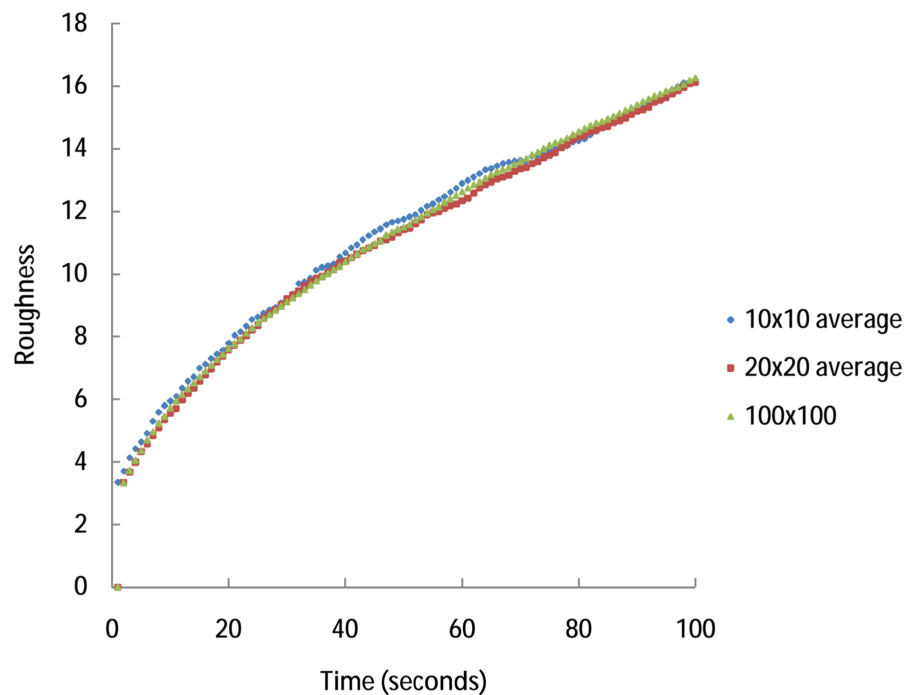


Figure 16: A plot of roughness versus time for a variety of lattice sizes. Ten trials on a 10x10 matrix and five trials on a 20x20 matrix average out to equal the same value of roughness over time as a 100x100 matrix which has been proved to accurately depict the surface morphology of the substrate.

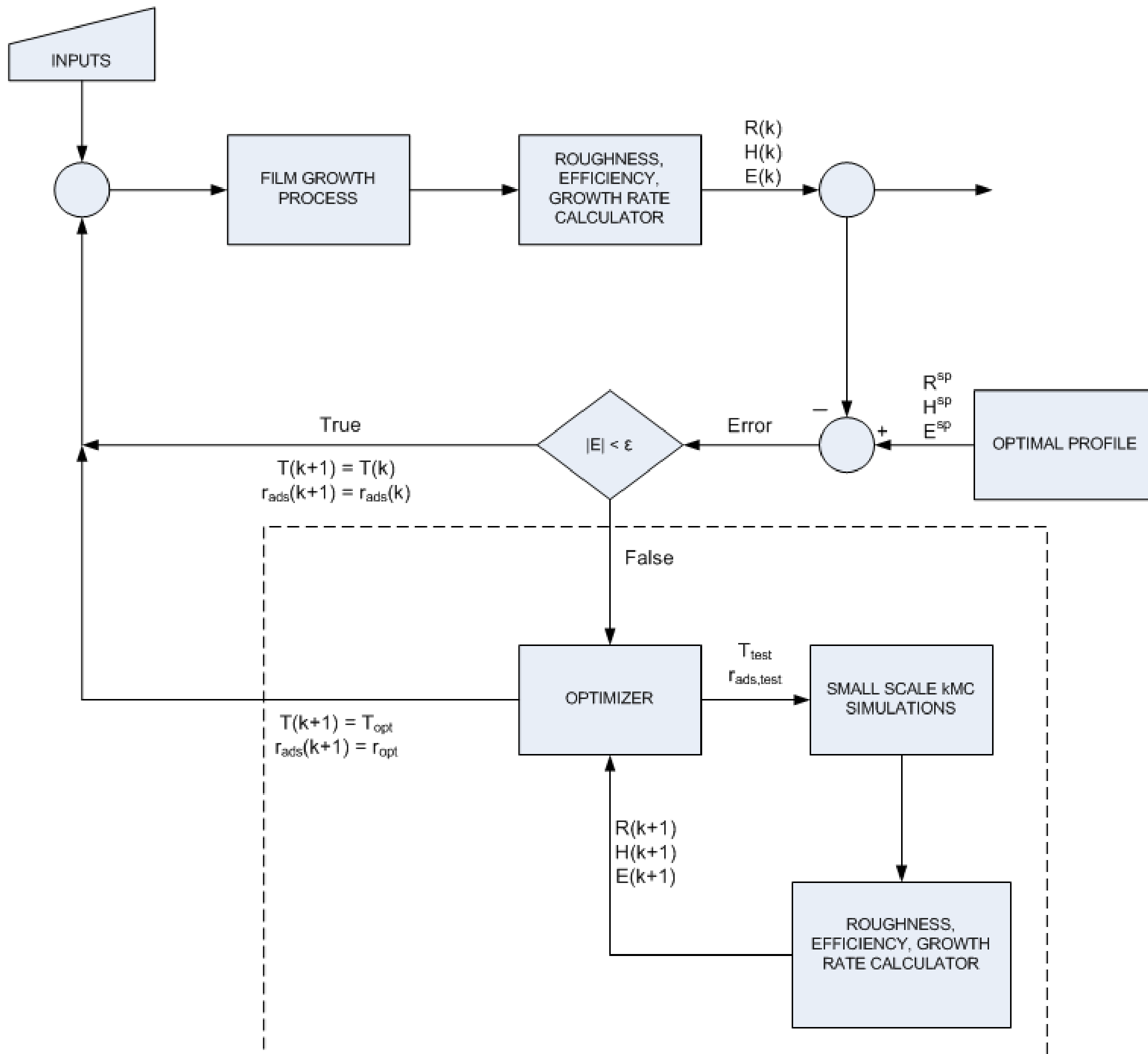


Figure 17: Flow Diagram of Control of Single Component KMC method.

5 Results of Single Component Control of Thin Film Growth

5.1 Control Problem

Using the control methodology outlined in Section 4 several different control problems were posed and the system response to the controller analyzed. First, the optimal control conditions were found by analyzing the objective function values for a variety of system characteristics, specifically the cases where 1) the adsorption flux and temperature remained constant and 2) the adsorption flux and temperature were allowed to change midway through simulation. The optimal conditions were then used to develop the optimal profile curves to be used by the controller. Keep in mind that the optimal control data was based on the initial conditions of zero height and a perfectly flat surface. As explained in the previous section, the goal of the controller was to eliminate any deviations from the optimal height and roughness profiles by manipulating the external parameters of adsorption flux and temperature. The controller was tested by challenging it with initial conditions that included 1) positive average height values and/or 2) rough surfaces. Analyzing the control response gave a better understanding of the system and allowed for suggestions for improved control strategies to be made.

5.2 Finding the Optimal Profile

In Section 4, the development of an objective function surface for constant deposition flux and temperature was presented. However, it is unlikely that the optimal solution for the entire deposition time to create a 100 nm thin film will be a constant temperature and flux deposition. In a real deposition reactor, the flux of atoms and reactor temperature would be able

to be tuned continuously with time to optimize the product to specifications. Since this creates a multi-dimensional problem that cannot be fully examined under the project timeframe, only a specific category of conditions with a single change in deposition flux and/or temperature was tested. To complement the constant deposition process, processes were examined that had constant properties for the first 50 nm of deposition, then upon reaching that specified height the deposition flux was increased or decreased by 2 ML/site-sec and/or the temperature was ramped up or down 100 K.

By using a recipe that allowed for the flux and temperature conditions to experience a single step change, the objective function was successfully minimized beyond that which was possible with the constant properties case. The objective function weighting factors A = 0.05, B = 0.30 and C = 0.65 were used for the following analysis and subsequent control simulations. These weighting factors correspond to a physical situation where efficiency is the most important factor, followed by roughness and finally deposition time. For the constant properties case, this objective function was minimized with a deposition flux of 9 ML/site-sec and a temperature of 500 K. However, when allowing a single step change to the external system parameters the objective function was further minimized with a) a deposition flux of 7 ML/site-sec and temperature of 600 K for the first 50 nm and b) a deposition flux of 9 ML/site-sec and 500 K for the final 50 nm deposition. A graphical representation comparing the deposition flux and temperature for the two different recipes follows in Figure 18 and Figure 19.

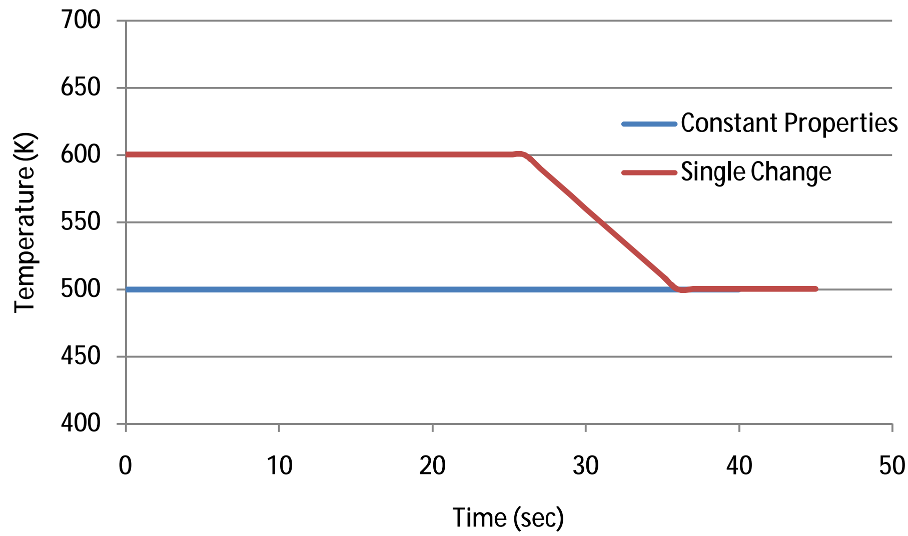


Figure 18: Temperature profile for optimum conditions when T is held constant or T is allowed a single 100 K change.

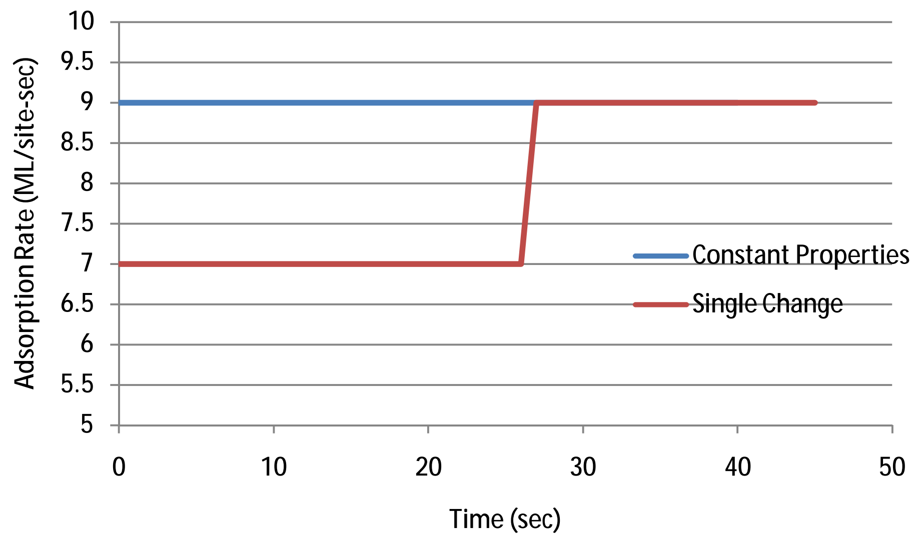


Figure 19: Deposition flux profile for optimal conditions when flux is held constant or flux is allowed a single 2 ML/site-sec change.

To prove that allowing the system to change parameters midway through the deposition process improved the quality of the thin film produced, the roughness and height evolution are shown below. The roughness curve for the single change case develops consistently below the constant properties case, and at 100 nm the roughness is improved from 19.1 to 18.8 monolayers.

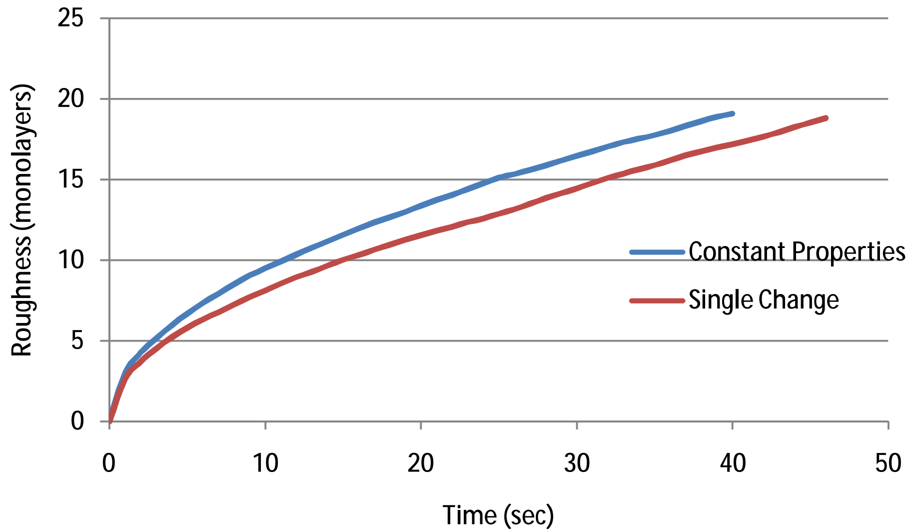


Figure 20: Roughness evolution for constant property deposition (blue) and allowing a single change (red). Final roughness: blue = 19.1, red = 18.8.

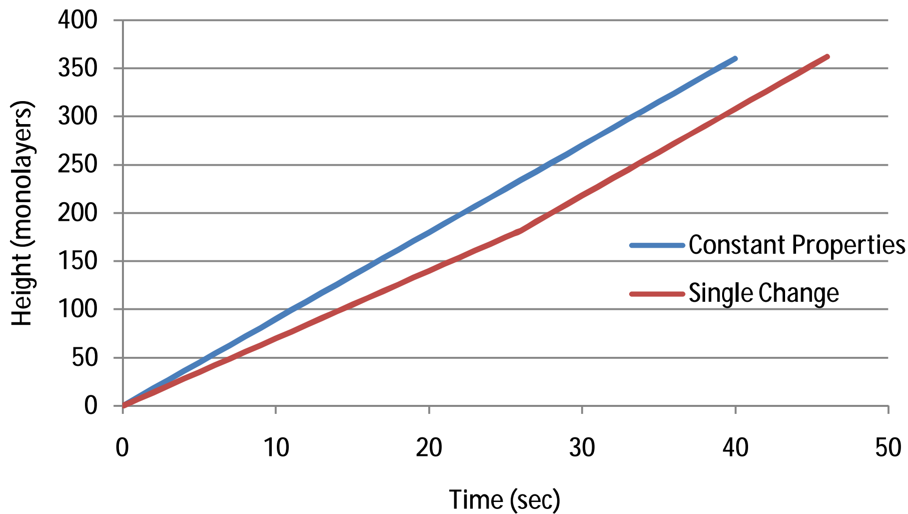


Figure 21: Height evolution for constant property deposition (blue) and allowing a single change (red).

The time to deposit the 100 nm thin film increased from 40 to 46 seconds when allowing a single step change. Both processes were able to complete the deposition with 100% efficiency, with zero atoms desorbing from the surface during the simulation. Since the efficiency was equal in both processes, the weighting factors dictated that the decrease in roughness was enough to tolerate an additional six seconds of deposition time. The OF value, as calculated by Equation 9, was decreased from 0.7966 to 0.7868 when allowing a single change in the deposition flux and temperature during growth, an improvement of 1.2%. The optimal profile obtained from the single change case, where 1) deposition flux of 7 ML/site-sec and temperature of 600 K was used for the first 50 nm and 2) a deposition flux of 9 ML/site-sec and 500 K was used for the final 50 nm deposition will be used in all subsequent single component control simulations. Using this profile also has an additional benefit over the constant properties case because responses of the controller to changes in set point (deposition flux, and temperature) can also be examined.

5.3 Control Tests

The optimum operating conditions were established based on a perfectly smooth surface with an initial height of zero monolayers. In a potential application, the controller would have to act in real time on wafers that possess different thickness and varying degrees of roughness. Using this as a basis, two different potential control problems were simulated. The first involved a situation where the initial height remained zero, but the surface had an initial roughness. This corresponds with the physical situation requiring the deposition of 100 nm of new material onto a surface, despite the initial roughness. The second case included situations where the thin film was already partially deposited and also had an initial roughness. In many applications the

thickness of a thin film is extremely important and affects many electronic and chemical properties. By simulating situations with a positive initial height and roughness, the control response can be analyzed to see how a surface with initial disturbances in roughness and height can be fixed in real time.

Three different initial surface morphologies with an average starting height of zero were simulated. First was the case where the surface possessed periodic hills and valleys, referred to hereafter as 'hills'. Hills had an initial range of height from -20 to 20 monolayers, was periodic every 20 lattice points in both directions and had an initial roughness of 10.0 monolayers. Next, a surface generated by weighted random height values between zero and 20 monolayers were studied, referred to hereafter as 'random'. More than 60% of this surface had an initial height between zero and five monolayers, so it did have a small average initial height of 5 monolayers. The surface had an initial roughness of 6.4 monolayers. Lastly, a surface with parallel grooves every five lattice points was introduced. This surface had an initial height range of -2 to 2 monolayers (for an average of zero), and had an initial roughness of 1.4 monolayers. Additionally, a flat surface with initial height of zero was simulated with control to make sure the optimal profile was being followed and to confirm that the controller was operating as intended. The three different initial surface morphologies are presented in Figure 22, Figure 23, and Figure 24.

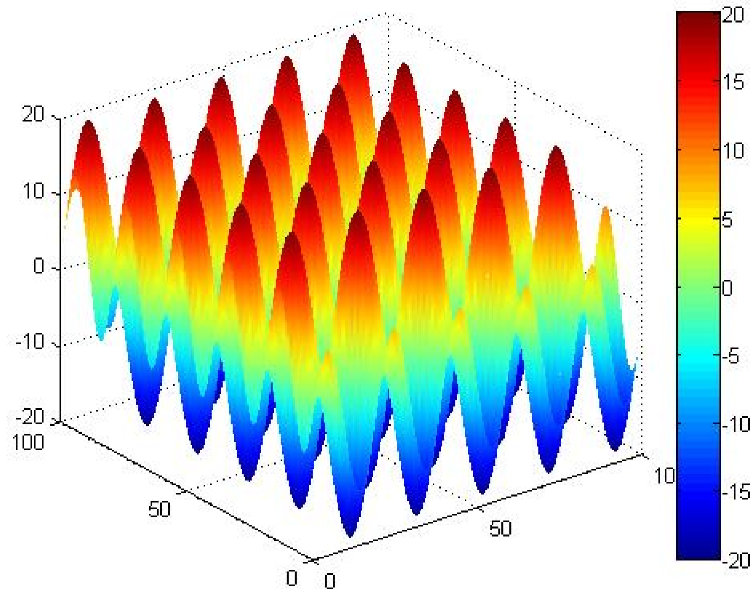


Figure 22: Initial surface, hills.

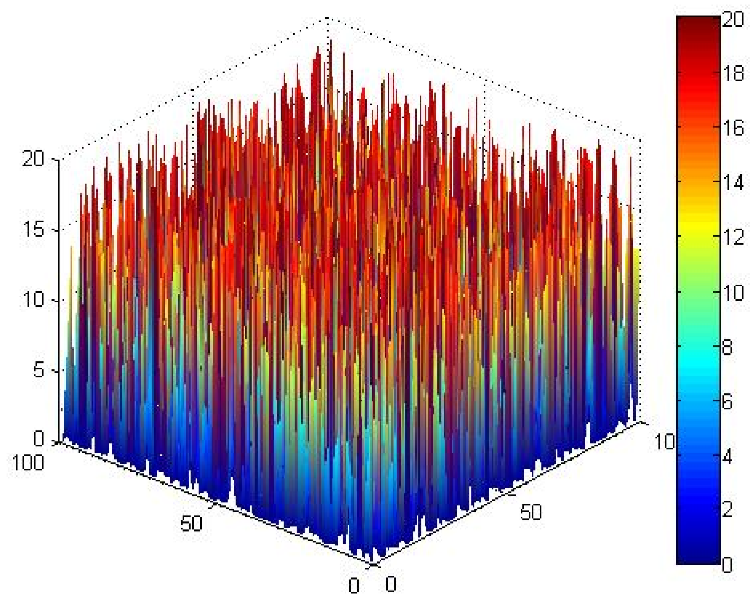


Figure 23: Initial surface, random.

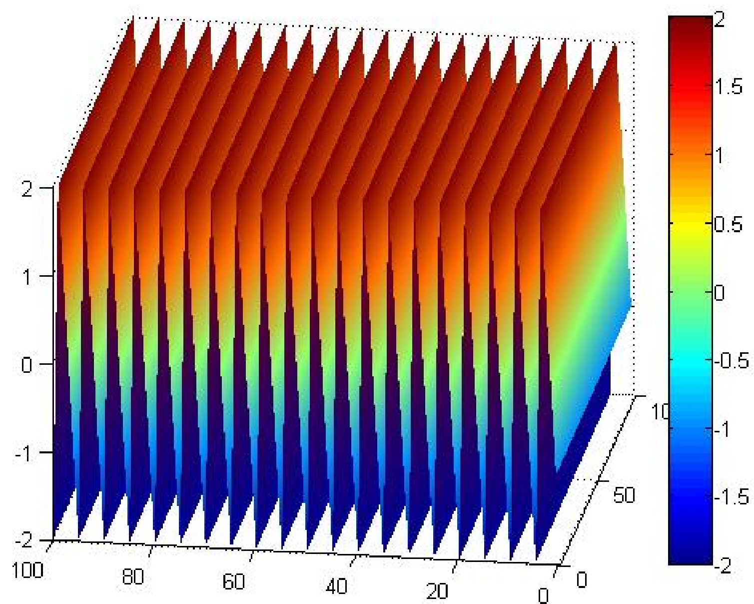


Figure 24: Initial surface, grooves.

The control simulations completed with varying results. In general, the deposition flux and the height were effectively manipulated to maintain the optimal profile curve. Due to the initial roughness, the temperature started at a higher initial value, while the deposition flux was lowered. This result was expected, since high temperature and minimal deposition lead to smoother surfaces in general. When the temperature was ramped up to minimize the difference between the roughness curve and the optimal profile, the roughness was able to be corrected within 10 seconds for all cases. However, although the optimal profile curve was attained there was a significant overshoot and recovery period. The limitations on the temperature ramp rate in the system prevented a quick return to the optimal temperature profile. Since the temperature was still relatively high, the surface continued to become smoother, moving away from the optimal curve. In the case of the hills and random surface, the optimal roughness profile was never reached again. The hills surface was also allowed to progress without control in a separate simulation for comparison purposes. It is clearly evident that the controller was successful in improving the quality of the overall thin film deposition from the following figures.

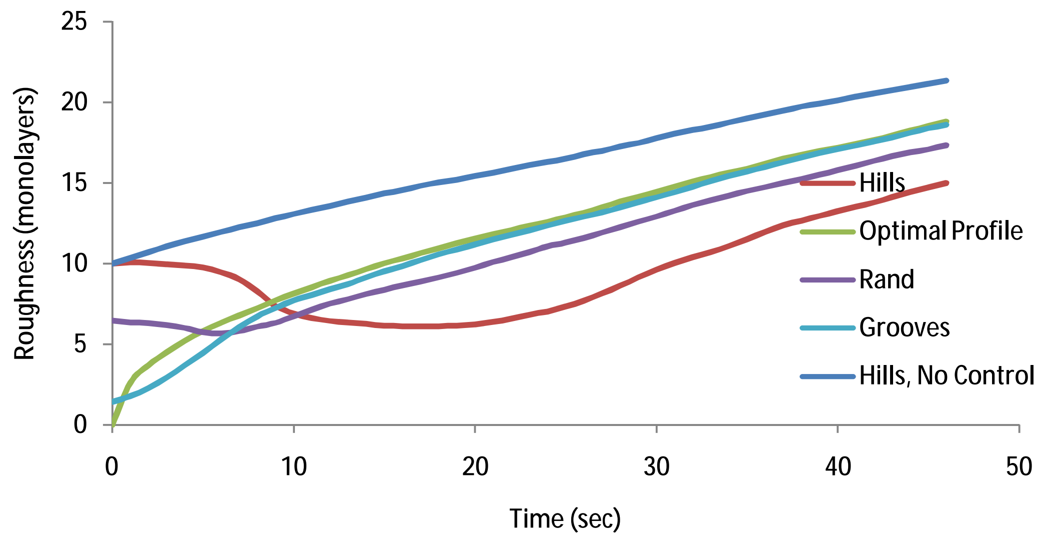


Figure 25: Roughness evolution, initial height = 0, various surface morphologies.

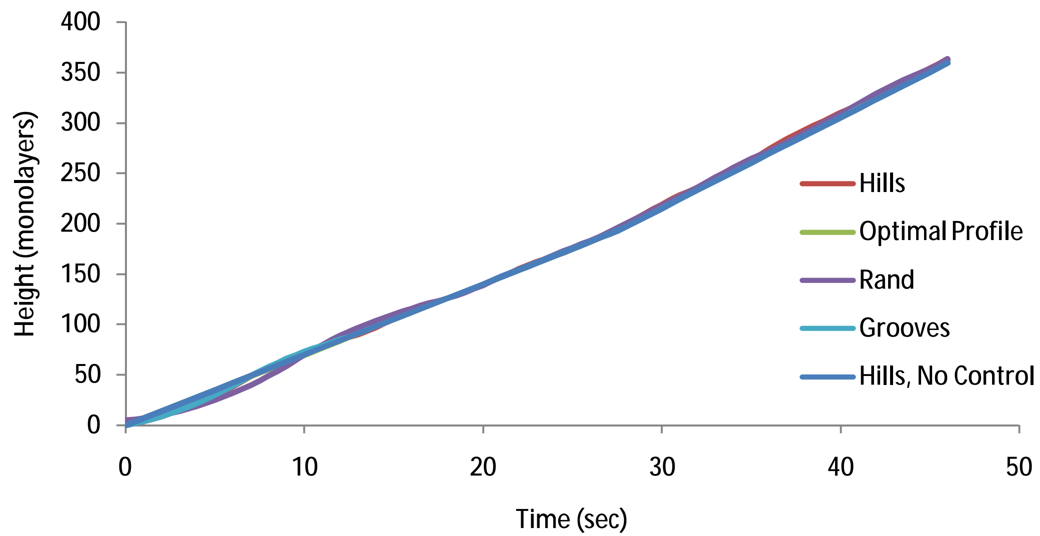


Figure 26: Height evolution, initial height = 0, various surface morphologies.

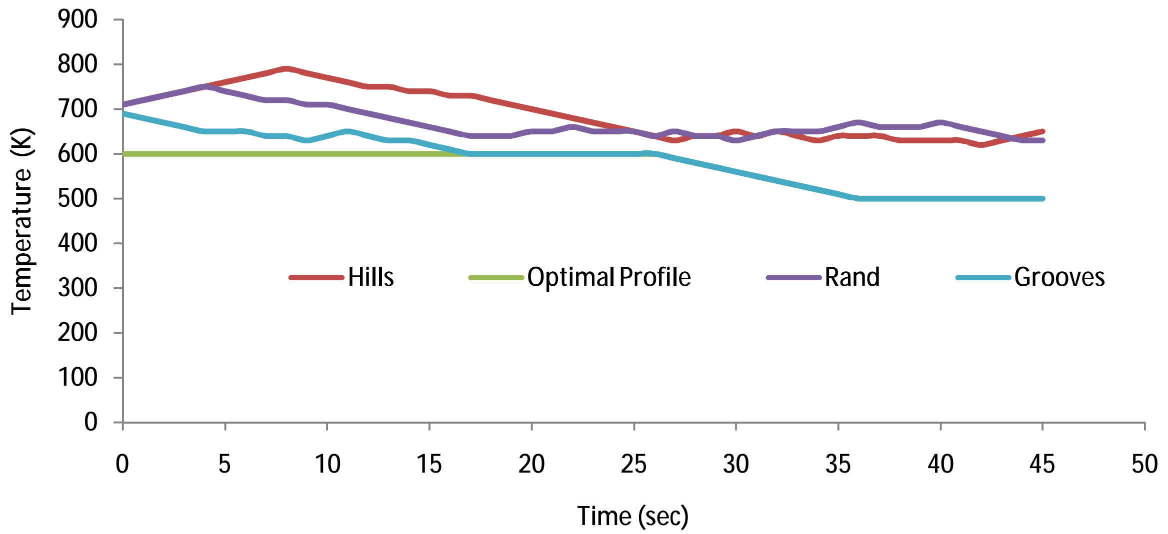


Figure 27: Temperature evolution with control, initial height = 0, various surface morphologies.

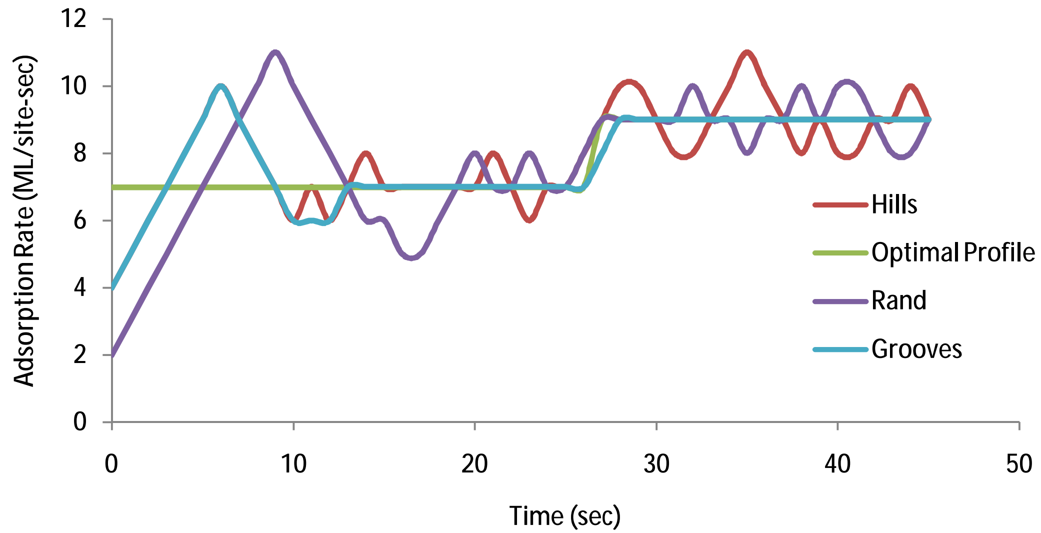


Figure 28: Deposition flux evolution, initial height = 0, various surface morphologies.

It is important to note that although the results of the control runs showing smaller roughness than the optimal profile with a 100 nm thin film deposition does not mean that it is better to start with an imperfect surface and apply control. The controller only attempts to reach the roughness and height optimal profiles, however, there is a third term in the objective function: efficiency. The control runs all ramped the temperature well above the values used in the optimal simulation to minimize the roughness deviation. The penalty for this was decreased efficiency, as seen in the number of atoms desorbed from the surface. In all cases the objective function was decreased towards the optimal profile objective function value. The grooved surface and the randomly generated surface were most successfully controlled, with the hills surface only decreasing the objective function slightly compared to having no control. Since hills had the largest initial roughness, it is obvious that the larger the deviation from ideal the harder the system is to control.

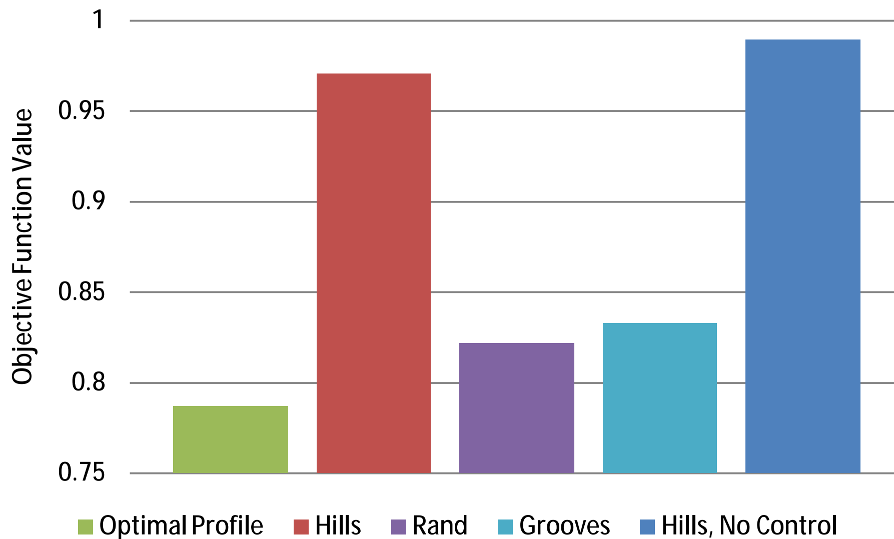


Figure 29: Comparison of final objective function values for different control experiments.

In the previous discussion it can be seen that the constraint on temperature ramp rate hindered the controller's ability to maintain the optimal profile. Depending on the reactor being used and on furnace technology, it may be possible to ramp the temperature at a greater rate than 10 K/sec. To see how the increased flexibility in temperature manipulation would affect control performance, the hills surface was once again simulated with the maximum temperature ramp rate increased to 30 K/sec.

This had several effects on the system, as shown in Figure 30 and Figure 31. The initial time required to reach the optimal profile for roughness decreased from 10 to 5 seconds because the increased ramp rate allowed higher temperatures to be used sooner, accelerating the smoothing process. The system was also able to ramp down much faster to decrease the time required to recover after the overshoot period. The roughness profile came much closer to the optimal profile by the end of the simulation with the increased ramp rate. There was a decrease in the controller's ability to maintain the optimal deposition flux. Finally, increased ramp rate decreased the final objective function value, coming much closer to the optimal value. It is safe to assume that increased flexibility in tuning the external system parameters will only lead to improved controller performance.

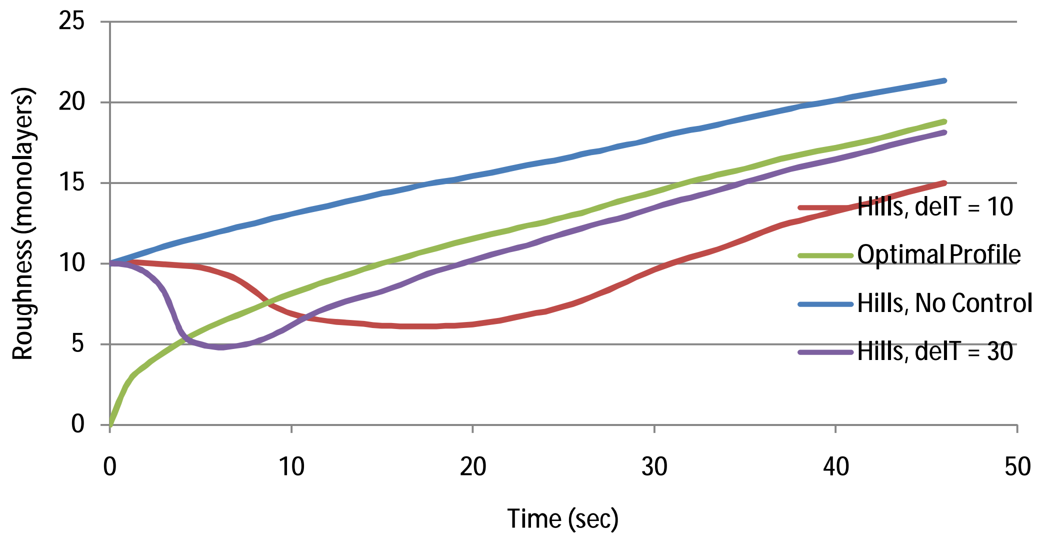


Figure 30: Roughness profile, effect of temperature ramp rate.

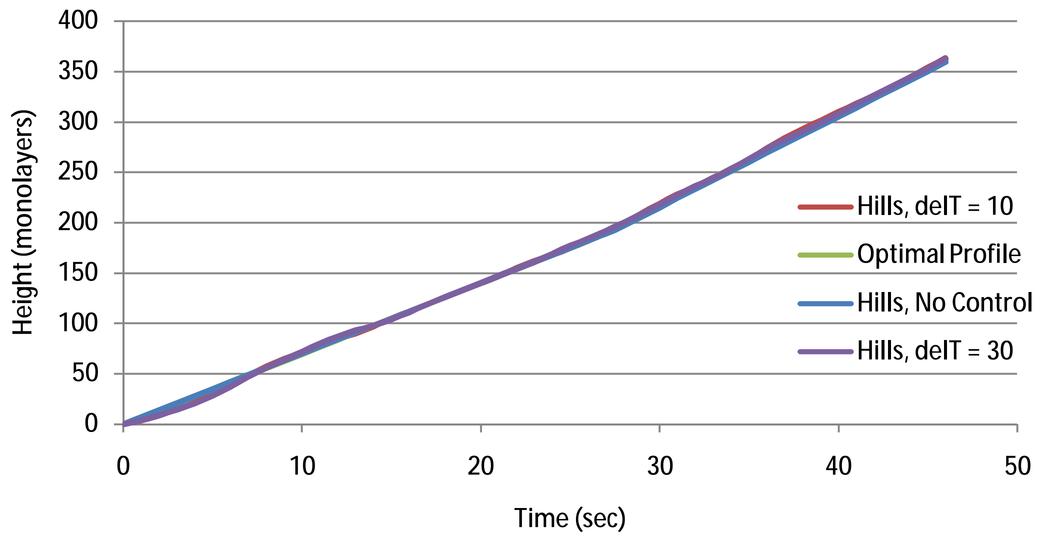


Figure 31: Height profile, effect of temperature ramp rate.

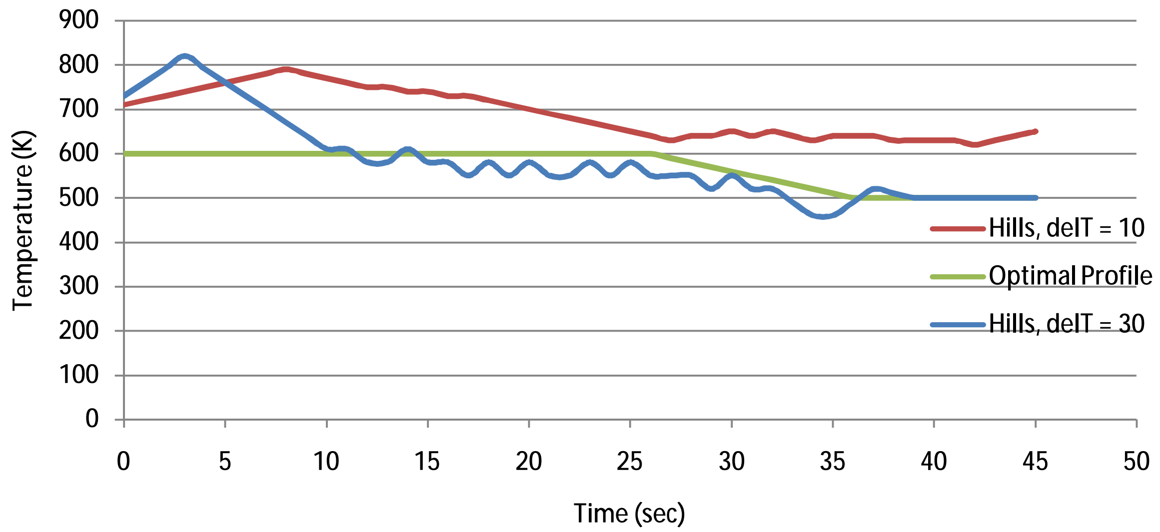


Figure 32: Temperature profile, effect of increased ramp rate.

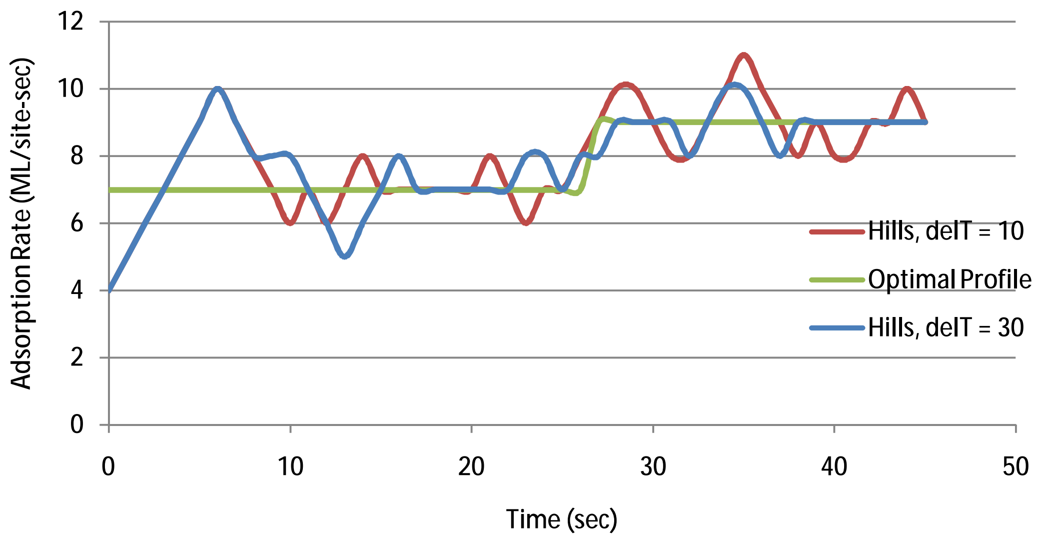


Figure 33: Deposition flux profile, effect of increased ramp rate.

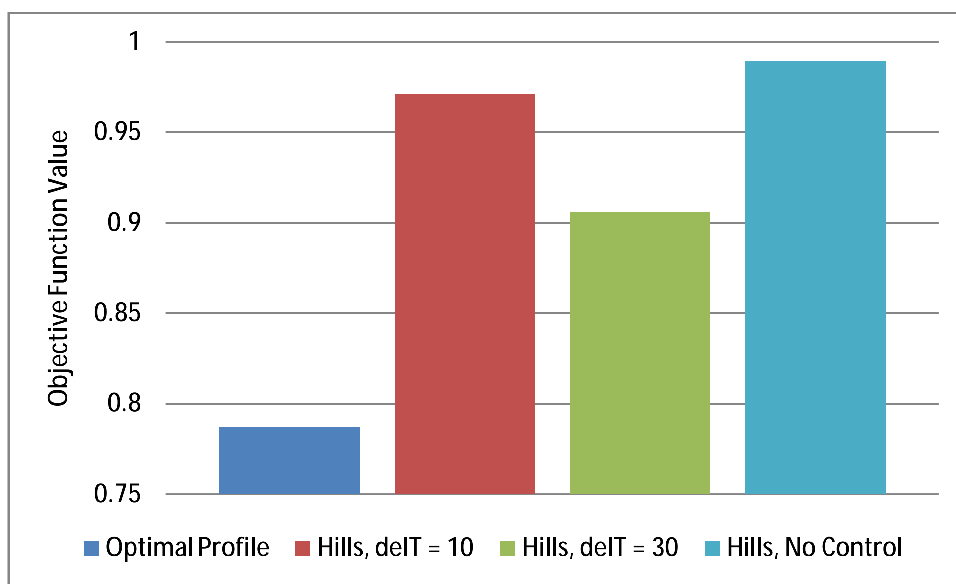


Figure 34: Final objective function, effect of increased ramp rate.

In addition to using different ramping rates for the temperature and deposition flux change through the controller, controller performance can be evaluated by manipulating two additional variables: 1) the lattice size used for control calculations and 2) the number of times the controller is called during the simulation. In all previous simulations the controller was called every second and a 20x20 lattice size was used. As stated earlier, the lattice size used in the controller is an area where error can be important. Smaller lattice sizes may not be able to capture larger morphologies on the surface and therefore may not give reliable control results. On the other hand, larger lattice sizes also require additional computational time; therefore there is an advantage to using a small lattice size if the reliability is not compromised. If unlimited computing power and time were available, using a lattice the size of the surface in the controller

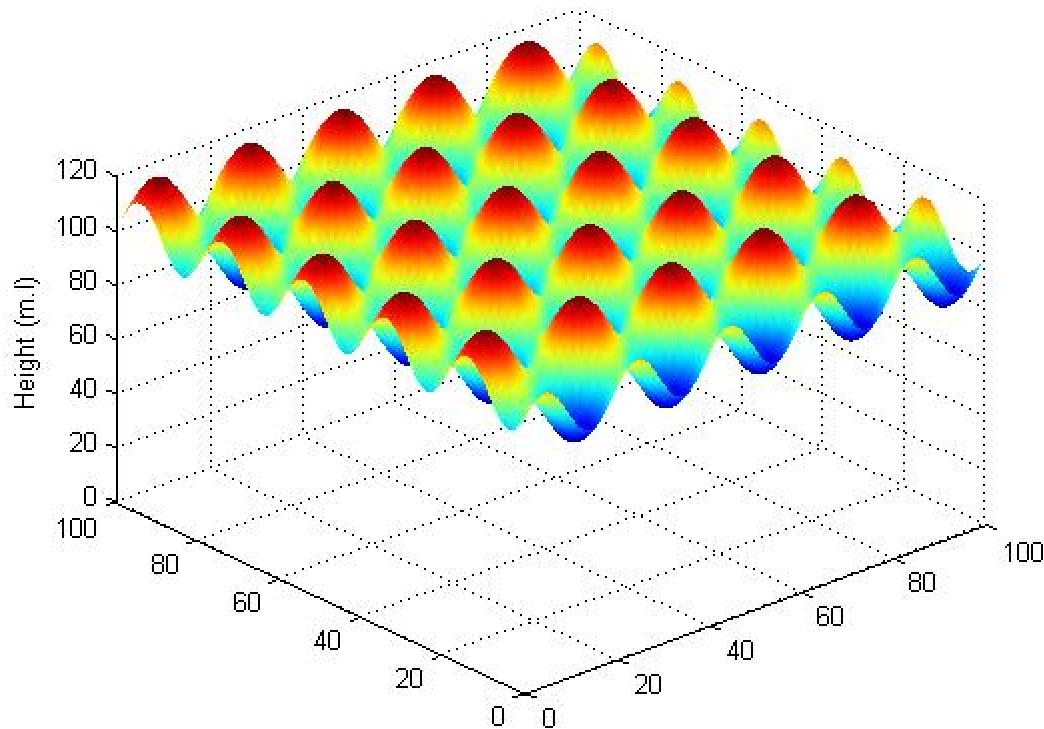


Figure 35: Initial surface morphology, hills.

would yield the best results. For the purposes of this study, lattice sizes of 5x5, 10x10, and 20x20 were used for the model predictive controller and a sensitivity analysis was performed to determine whether controller quality was impacted by this choice. The hills surface from the previous discussion was once again used, with the modification of an average initial height of 100 monolayers. This allows for the additional control problem to be addressed.

The roughness and height development using all three lattice sizes are shown in Figure 36 and Figure 37. For comparison purposes, a simulation with the controller off and the adsorption rate and temperature set to those corresponding to the optimal profile conditions was also performed. All three lattice sizes were successful in reaching the optimal profile curve. However, upon magnification of the first 30 seconds of deposition (Figure 38, Figure 39, page 54), differences between the three are clear. The small 5x5 lattice took the longest time to reach the optimal profile curve; however, once the optimal curve was reached it did not overshoot as much and remained near the curve for the remainder of the simulation. Both 10x10 and 20x20 lattice sizes reached the optimal curve approximately 3 seconds faster, but both could not stay on the curve initially. The 20x20 lattice more rapidly recovered the roughness to the set point. Additionally, the 20x20 lattice was within 1% of the optimal profile curve at the end of the simulation, while the 10x10 and 5x5 were within 2% and 3%, respectively. While these percentages seem small, the strict requirements of deposition processes call for very accurate and precise control procedures.

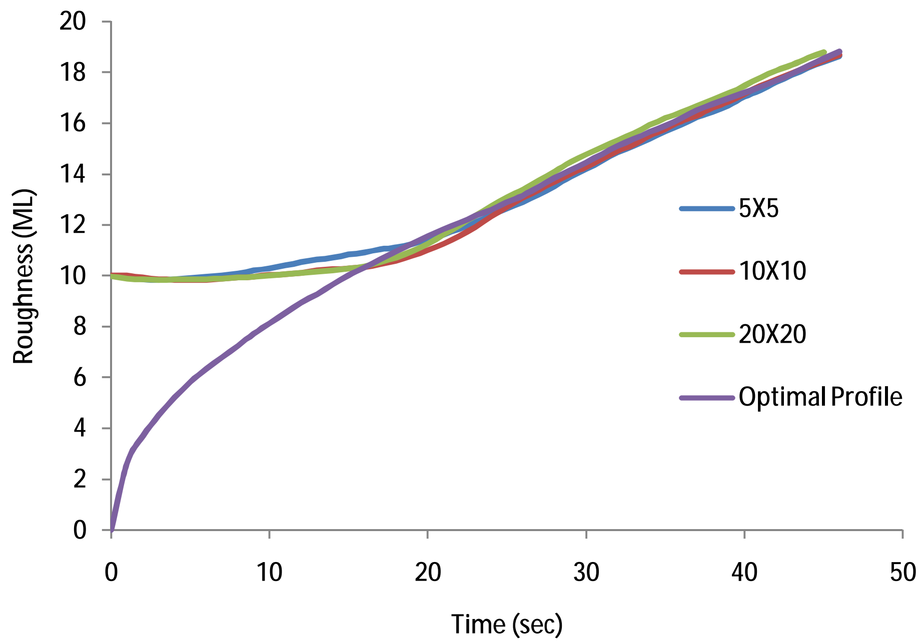


Figure 36: Roughness profile for varying controller lattice sizes.

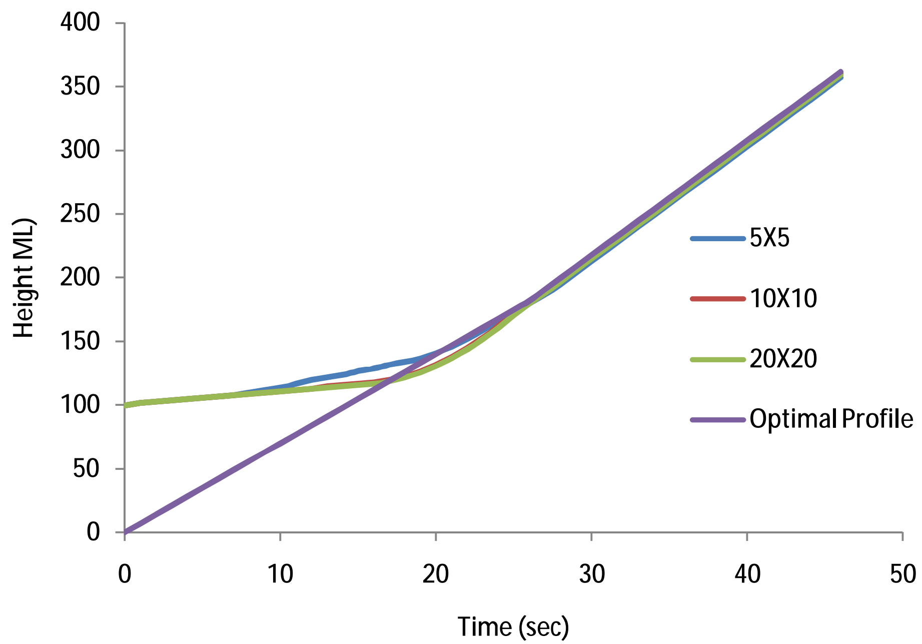


Figure 37: Height profile for varying controller lattice sizes.

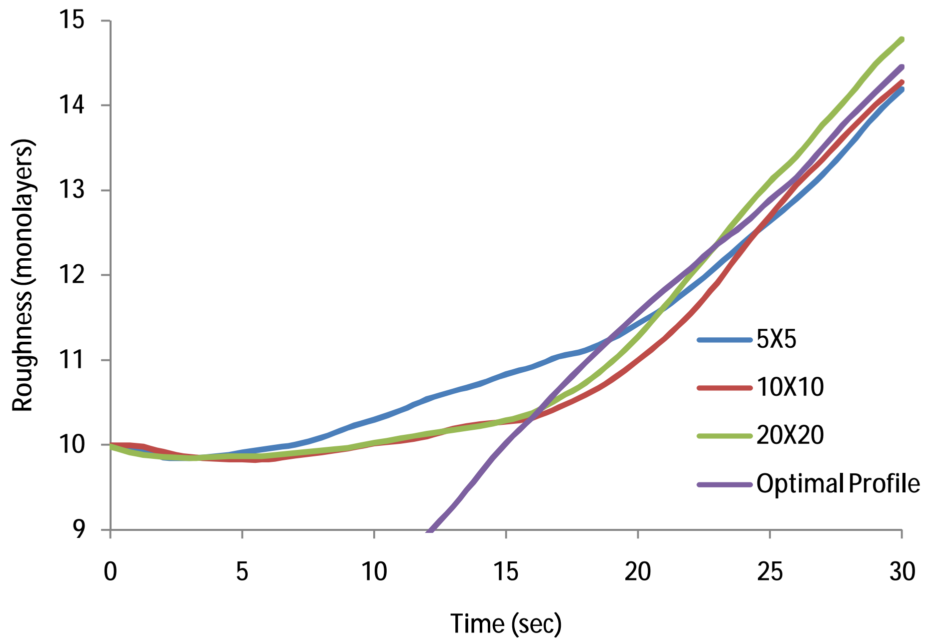


Figure 38: Magnified view of roughness profile with varying controller lattice sizes.

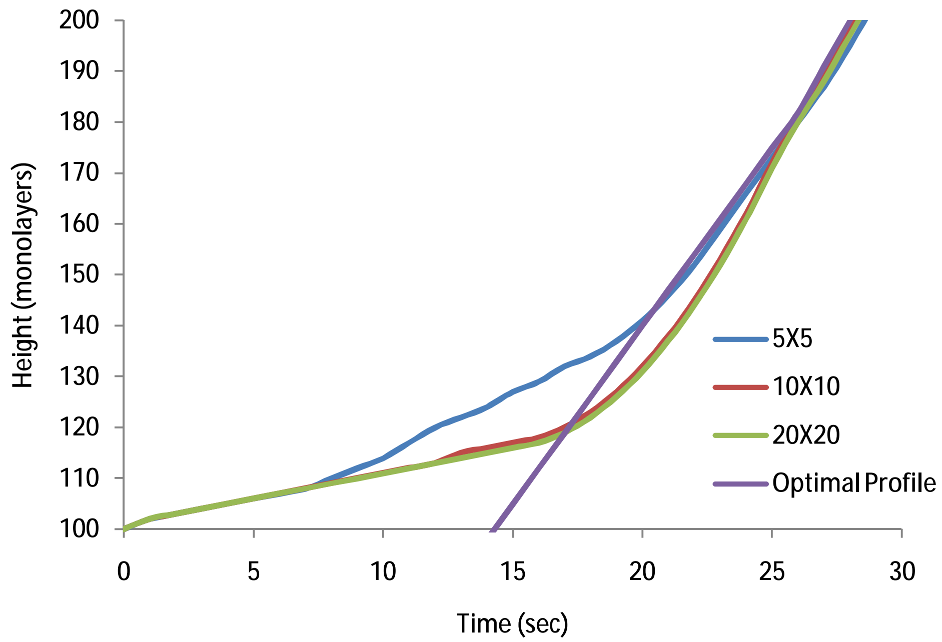


Figure 39: Magnified view of height profile with varying controller lattice sizes.

To gain a better understanding of how the controller acted on the system to direct it to the optimal profile, it is beneficial to display the temperature and deposition flux profiles for each control experiment (Figure 27, Figure 28, page 41). When analyzing this data, it is important to remember that the temperature and adsorption rate for the optimal profile curve are not the appropriate conditions for the controller to direct the surface in the most efficient manner. A 5x5 lattice once again reaches the set values for temperature and flux fastest, however, due to the initial morphology of the surface this may not be the best action to approach the optimal profile curve overall. This data explains why the control curves for the 10x10 and 20x20 lattice sizes reach the optimal profile curves for height and roughness first, but then need more time to recover to follow it tightly. Since the current control implementation only accounts for the best solution over the next one second time step, the solution has the potential to drive the values for adsorption rate and temperature far away from those that gave the optimal profile. Due to the constraints placed in changing these parameters in our simulation, additional time is required to maintain the optimal profile curve. Before the 5x5 lattice brought the system to the optimal profile curve, the adsorption rate varies significantly between time steps. This variation can be attributed to several factors. No matter what sections of the overall lattice the controller uses to perform the model predictive simulations, a 5x5 lattice does not capture the entire morphology of the hills and valleys surface. This can easily cause false conclusions to be made by the controller and compromise its effectiveness. With a periodicity of 20 lattice points, a 20x20 lattice is large enough to capture the general surface characteristics for the controller. As a result, the 20x20 is more effective in tuning both the temperature and the adsorption rate to fix the system to the optimal conditions and its use in previous simulations is justified.

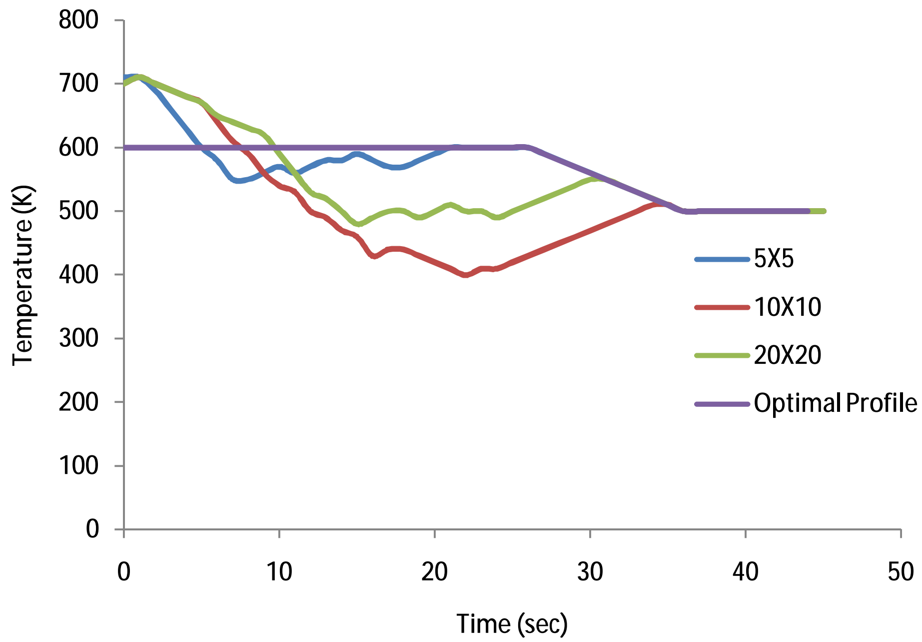


Figure 40: Temperature profile for varying controller lattice sizes.

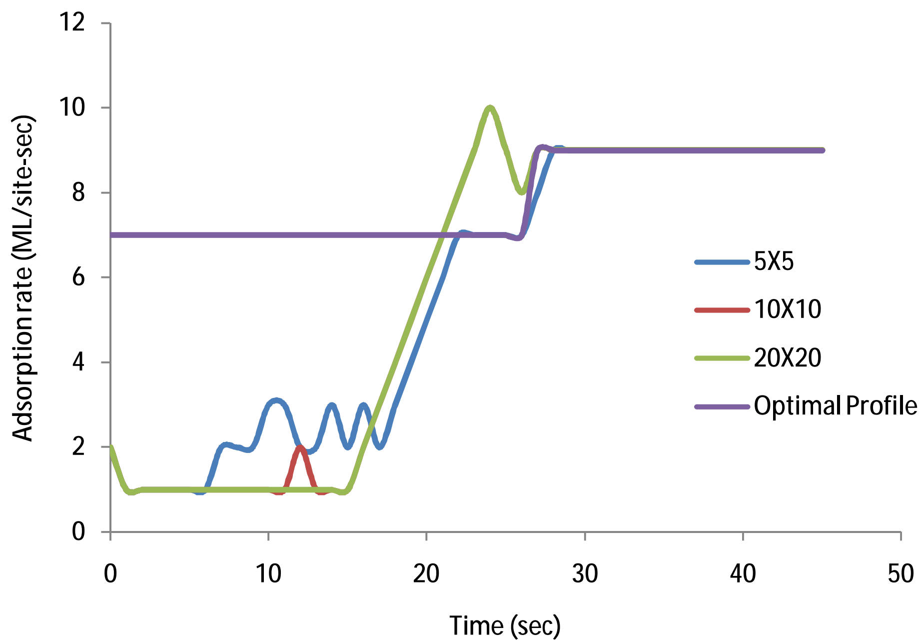


Figure 41: Growth rate profile for varying controller lattice sizes.

Although each controller was able to ultimately reach the optimal profile for height and roughness at a 100 nm thin film, the initial surface morphology was still seen in the final surface. As shown in Figure 42 - Figure 44 on pages 58 - 59, the controller was successful in decreasing the roughness compared to the simulation without control, but relics of the initial surface topography remain. Although the final surface of the controller simulation had the same roughness as the optimal profile simulation, the surface appears different to the naked eye. The primary cause for this was from calculating the roughness with only lattice point resolution. This is currently difficult to measure and monitor in real time. By calculating the roughness based on each lattice point height, the overall morphology of the system is lost. An improved control scheme would calculate roughness with different levels of resolution, make a conclusion on the morphology of the surface, and act appropriately based on those conclusions. This would be a more effective implementation to deal with larger surface features, and would result in a more versatile and efficient controller.

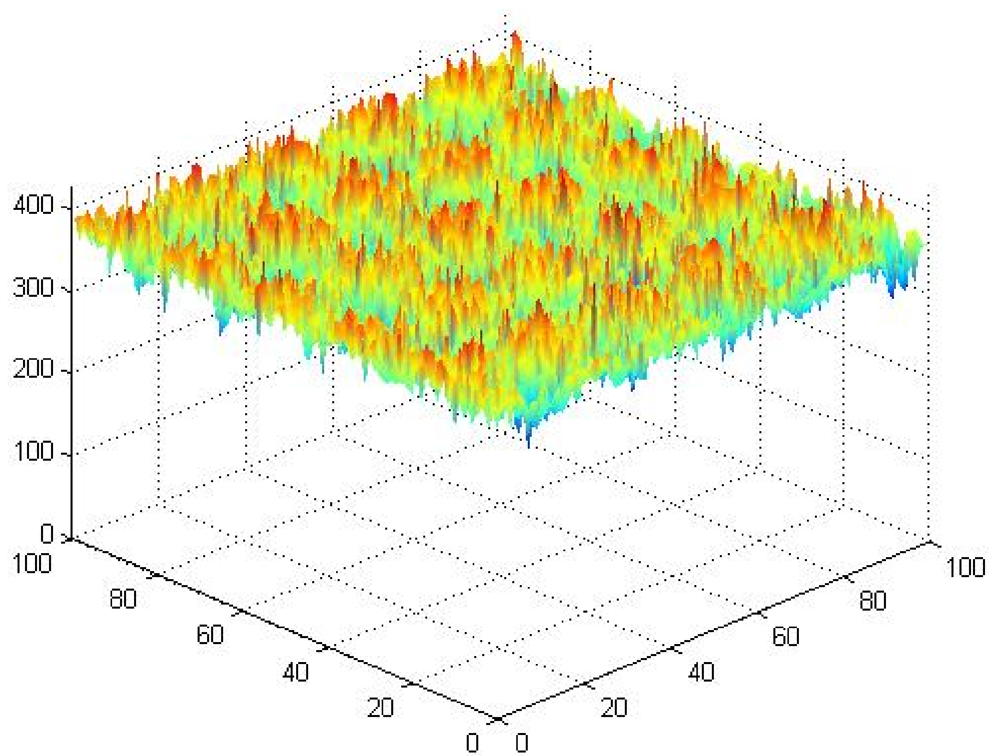


Figure 42: Final surface, control off.

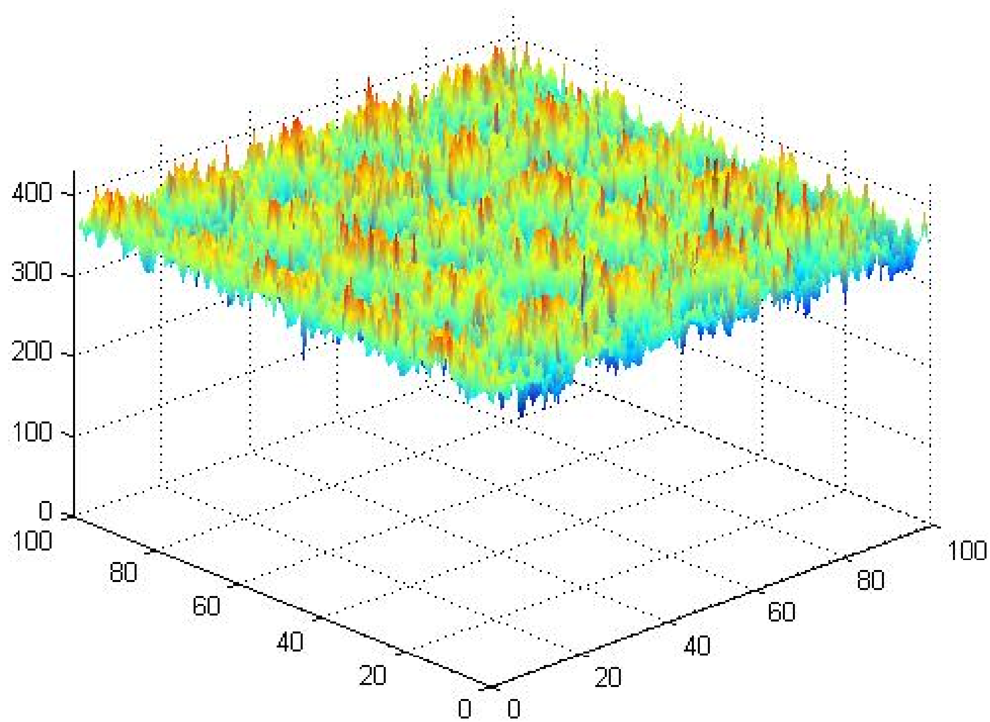


Figure 43: Final surface, control on, 20n20 lattice.

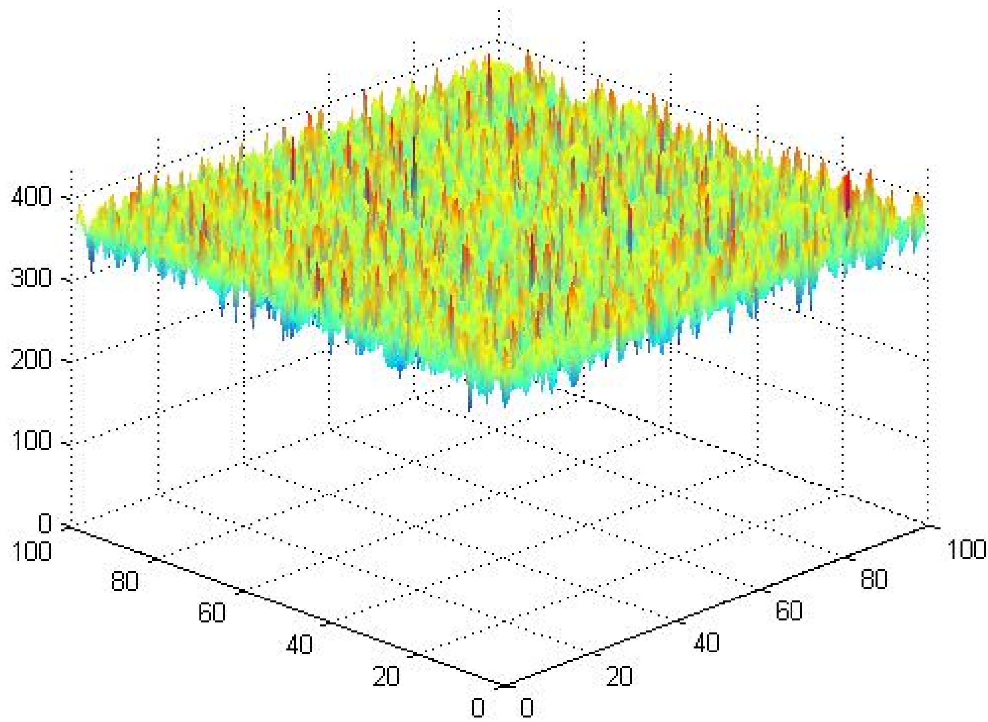


Figure 44: Final surface, optimal profile.

When implementing a model predictive control system on an actual deposition process occurring in real time, speed of calculation is a very important consideration. During implementation, there would be some dead time between measuring the process conditions and calculating the optimal conditions for the future time steps. Therefore, minimizing the amount of times the controller is used can potentially improve the overall efficiency of the system. To test this in the current simulation system, an additional experiment was performed where the controller action only took place every other second, compared to every second in previous experiments. The general control scheme remained the same, however, when the model predictive controller was called the lattice grids were simulated for two seconds, as opposed to for only one second. Accordingly, the control decision was implemented for the next two seconds of the actual simulation. This experimental setup decreased the number of control

actions by 50%. Unfortunately, there was a significant loss of effectiveness using this scheme. As shown in Figure 45 and Figure 46 on page 60, taking control action every other second caused a significant increase in the amount of time necessary to reach the optimal profile curves.

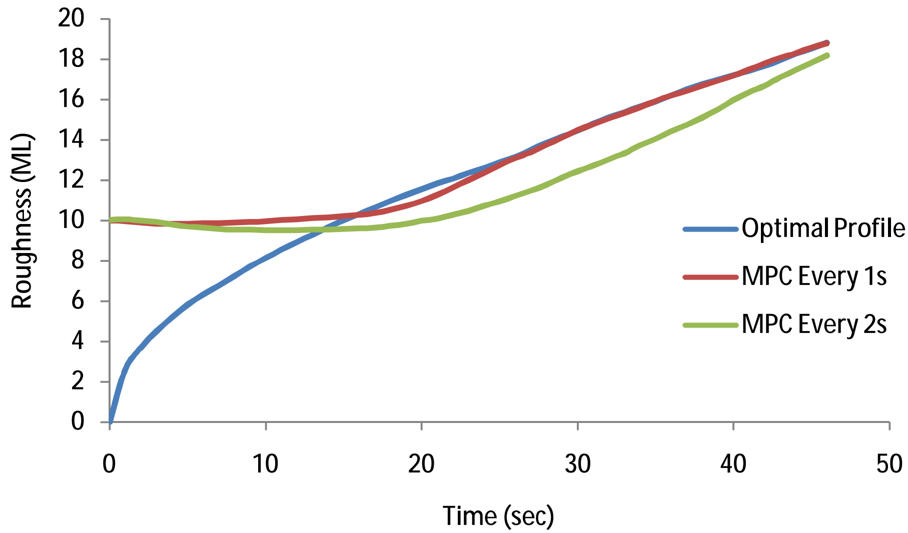


Figure 45: Roughness profile for varying control implementations.

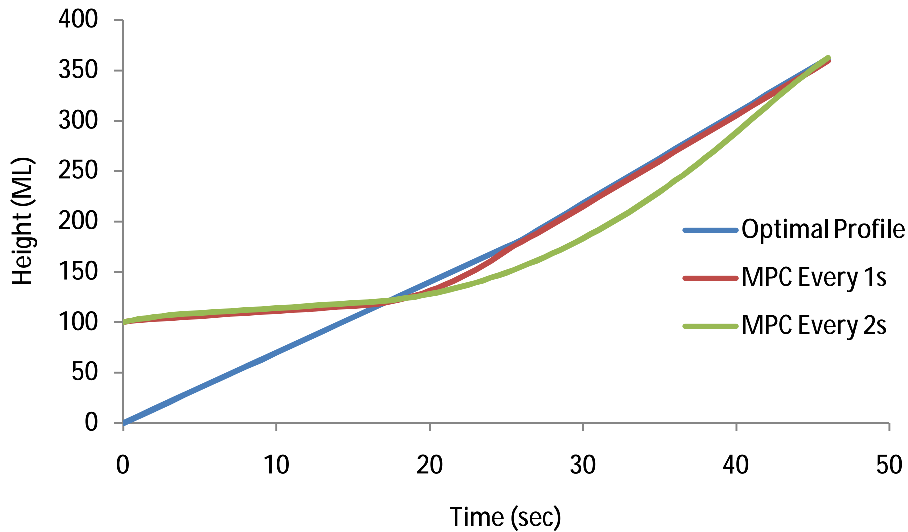


Figure 46: Height profile for varying control implementations.

This section has presented an analysis of model predictive control response to two different cases: 1) a surface with initial roughness and zero height and 2) a surface with initial roughness and nonzero height. The surface evolution of different morphologies was improved when control was implemented. However, improved control can be achieved by relaxing the constraints on changes to the temperature and deposition flux. Control with respect to different initial morphologies may also be improved as modifications are made in roughness calculation. Additionally, the lattice size used for control calculations can compromise controller fidelity. A 20x20 lattice size was deemed satisfactory for rigorous control experiments of the surfaces presented in this report. Using the current control setup, calling the controller at least once per second is necessary to obtain satisfactory control solutions.

6 Multiple Component Thin Film Growth Modeling

Many deposition processes involve multiple chemical components, which leads to additional degrees of freedom in the specification of the film structure. The single component KMC model described in the previous sections and the associated controller are expanded here to include two different types of atoms in the gas phase. Adsorption, surface migration and desorption are still considered to be the three events which may occur at the surface. The introduction of multiple components alters the kinetics by affecting the way in which the particles interact and migrate about the surface. An important property of binary systems is the distribution of atoms. In order to create a system with long range interactions that promote atom aggregates of like atoms, nearest neighbor interactions were expanded to include both first and second neighbor shells (Figure 47). Primary neighbor interactions are weighted more strongly than secondary neighbor interactions to account for the decrease in bond strength over distances. The adsorption and desorption rates are consistent with the rates described in the single particle model. The initial gas phase composition, described by the fraction of type A atoms, is specified by the user along with the rate of adsorption. Desorption rates do not vary with the molecular type.

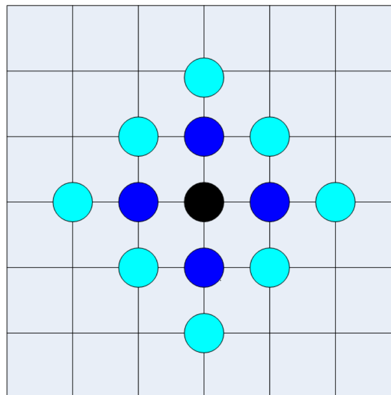


Figure 47: The particle of reference is colored black. Primary neighbors in dark blue and secondary neighbors in light blue.

The major difference introduced by adding another component is a change in the rate of migration. While the bonding strength of a type A-A bond was considered to be the same as for a type B-B bond, $E_n^A = E_n^B$, interactions between unlike species, A-B or B-A, have one third of the bond strength of a type A-A or B-B bond. This leads to an increase in rate of migration of atoms of the same type towards one another. The rate of migration is given explicitly by:

$$r_{mig}^{j} = \frac{k_o^m}{4}\,\mathbf{exp}\left(\frac{-E_S^j - 0.5(\Delta n_{jj} + \frac{1}{3}\Delta n_{ij})E_n^j}{kT}\right) \quad [10]$$

where j refers to the atom type (A or B) at the site of interest, the pre-exponential term k_o^m is a diffusion frequency factor, E_s^j is the diffusion surface energy barrier, Δn_{jj} is the change in the number of nearest neighbors of the same type of particle, Δn_{ij} is the change in the number of nearest neighbors of a different type of particle, E_n^j is the neighbor bond energy, k is the Boltzmann constant, and T is the temperature of the substrate. In order to determine the change in the nearest neighbors of similar and different particle types, the simulation weighs nearest neighbor bonding at the current site and the destination site according to whether the interaction is between a primary or a secondary neighbor. This is computed explicitly by:

$$n_{ij} = \sum_{x=1}^{l}\sum_{y=-x}^{x} S^{ij} * \left(\mathbf{1} - \frac{\sqrt{x^2+y^2}}{l}\right) \quad \text{and} \quad n_{jj} = \sum_{x=1}^{l}\sum_{y=-x}^{x} S^{jj} * \left(\mathbf{1} - \frac{\sqrt{x^2+y^2}}{l}\right) \quad [11]$$

where x is the horizontal position in which the atom will migrate to, y is the vertical position in which the atom will migrate to, l is the maximum length of interaction which is specified to always be two since the simulation only examines primary and secondary neighbors, and S^{ij} and S^{jj} are factors incorporated to include or exclude terms in the sum based on whether or not the neighbor under observation is a like or unlike particle. S^{ij} is equal to one if the neighbor under observation is an unlike particle and is equal to 0 if the neighbor under observation is a like

particle. S^{jj} is the reverse. This factor makes certain that the counting and weighting of like nearest neighbors only includes particles of the correct type.

The introduction of a second component into the system requires that the type of particle that is bound on the surface and the behaviors of different types of particles are accounted for. For simplicity, in the following figures, type A particles have been colored red and type B particles blue. As the concentration of component A in the gas phase increases the concentration on the surface increases as well.

In order to measure the degree of aggregation of these types of atoms a new system parameter was introduced. The order parameter, which measures the distribution of square aggregates of either type A or type B atoms, can be computed by the following equation, where g_i is the degree of order of size i.

$$order\ parameter = \frac{\sum_{i=1}^{gmax} i \times g_i}{\sum_{i=1}^{gmax} g_i} \quad [12]$$

G^{max} is the largest square aggregate found during runtime when analyzing a surface (Figure 48). The order parameter is as a system measurement is only calculated for the minority species. Higher values for the order parameter correspond to larger aggregates of like atoms. The lowest value for the order parameter is equal to one, meaning that every atom is exclusively surrounded by an unlike atoms. Theoretically, at a 50% concentration with an equal distribution of type A and B atoms, the order parameter for both atoms is equal.

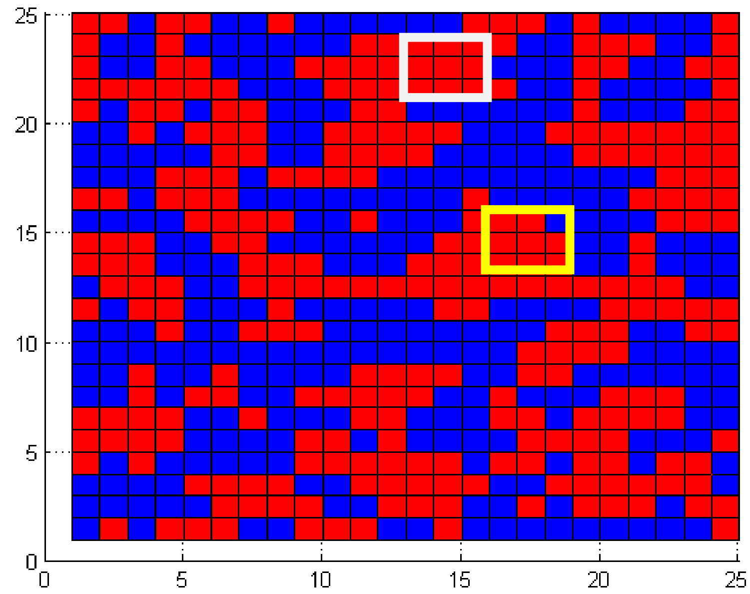


Figure 48: Order parameter size determination. The white box would count as an order parameter of 3 while the yellow box would not count as an order parameter of 3.

7 Results of Multiple Component Thin Film Growth

7.1 Effect of Lattice Size

As in the single component thin film growth, a large enough lattice size is necessary to fully capture the surface morphology of the substrate without resulting in excessively expensive simulations. For the multiple component simulation, a lattice size of 25x25 was determined to be sufficient to capture the full spectrum of the surface properties without a large constraint on the simulation. Note that the binary model is more computationally demanding than the single component one and therefore a smaller system was chosen. Smaller lattices are unable to properly model the surface features that arise during binary deposition.

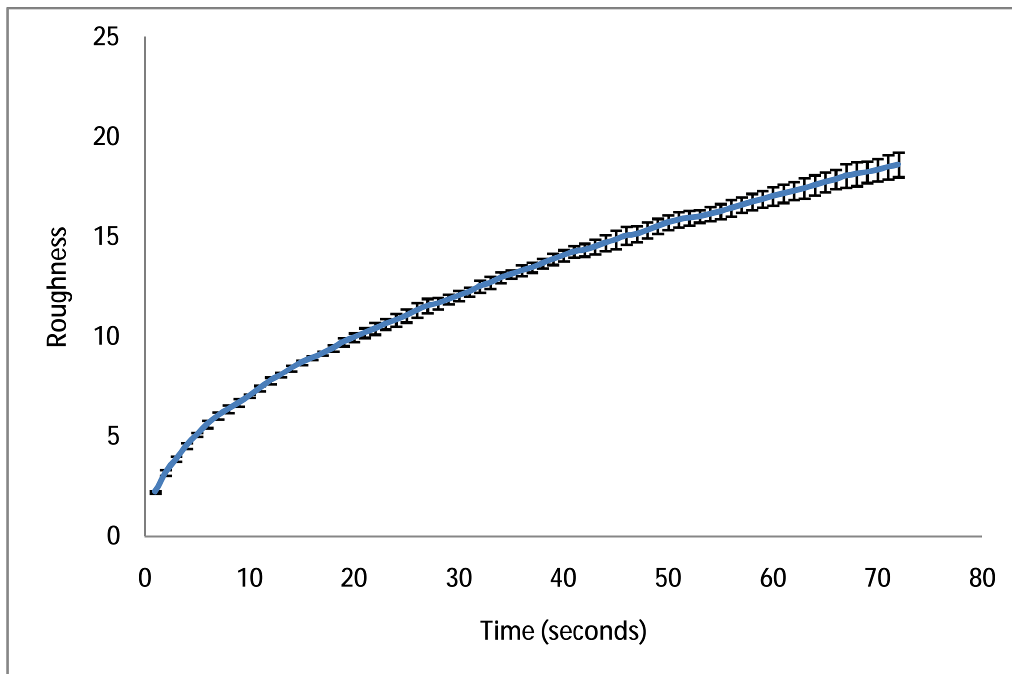


Figure 49: Roughness versus time for multiple component simulation run on a 25x25 lattice. The standard deviation at each time point is small enough so that this size lattice fully captures the surface morphology of the thin film.

7.2 Effect of Temperature

The temperature of the substrate in a multi-component system affects the roughness, growth rate and molecular structure of the thin film. Different temperatures, gas phase fluxes and compositions lead to significant changes in the system evolution. At lower temperatures the system is diffusion limited and the atoms, which adsorb to the surface randomly in a ratio equal to that of their ratio in the gas phase, are well mixed on the surface. At higher temperatures, when the atoms have sufficient energy to diffuse around the surface, they start to become phase separated (Figure 51). As phase separation occurs, the measured order parameter increases (Figure 50).

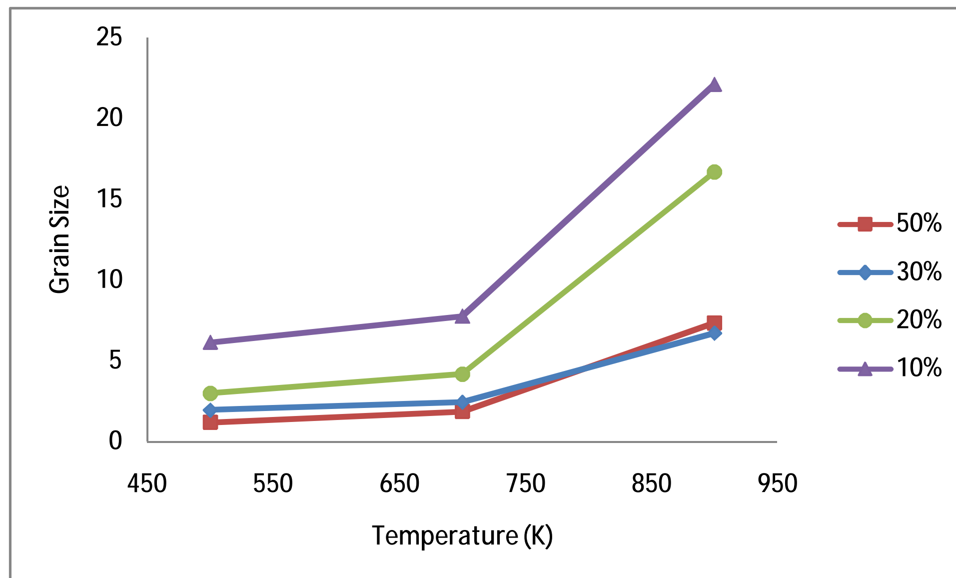
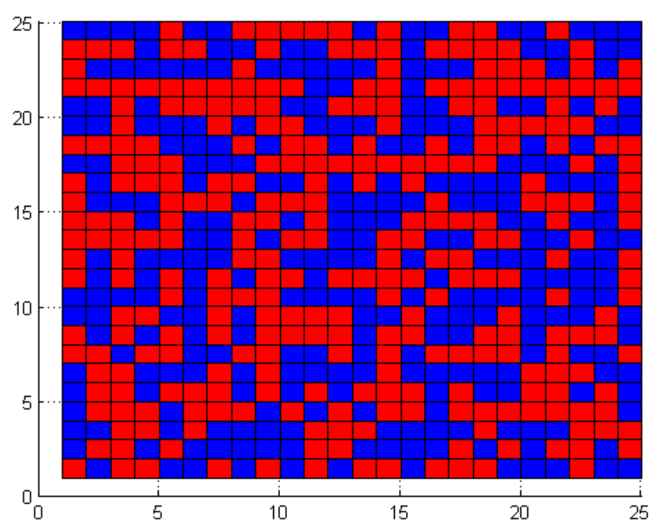
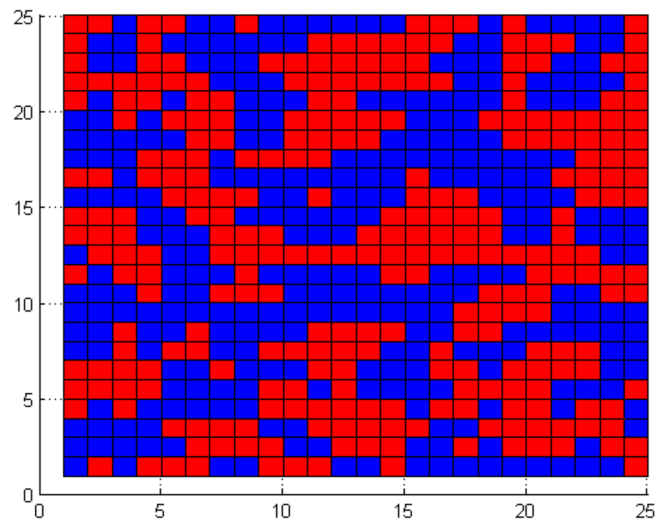


Figure 50: Effect of temperature on order parameter at constant adsorption rate, 1 ML/site-sec for different gas phase compositions of type A atoms.





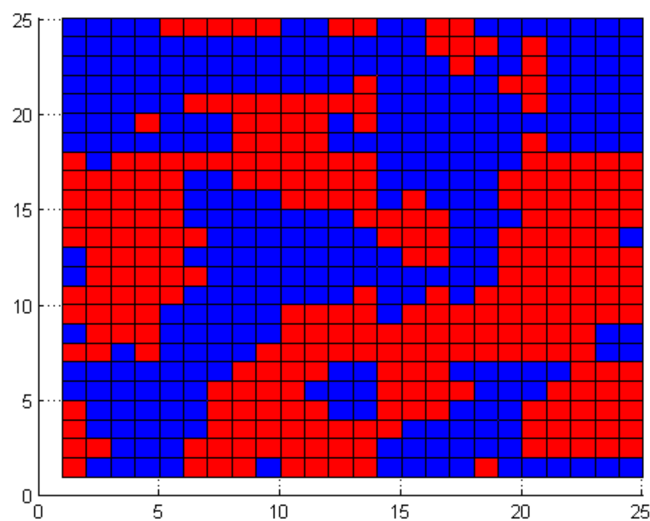


Figure 51: Increasing phase separation for constant flux and composition at different temperatures. A) 500K B) 700K C) 900K

7.3 Effect of Flux

The gas phase flux determines the rate of which particles adsorb onto the surface of the thin film. If the rate of adsorption is much greater than the rate of diffusion, there is even mixing of atoms on the surface as the rate of adsorption depends only on the gas phase composition. At temperatures where there is a moderate rate of diffusion, different rates of adsorption at a constant temperature and gas phase composition lead to different surface morphologies (Figure 52).

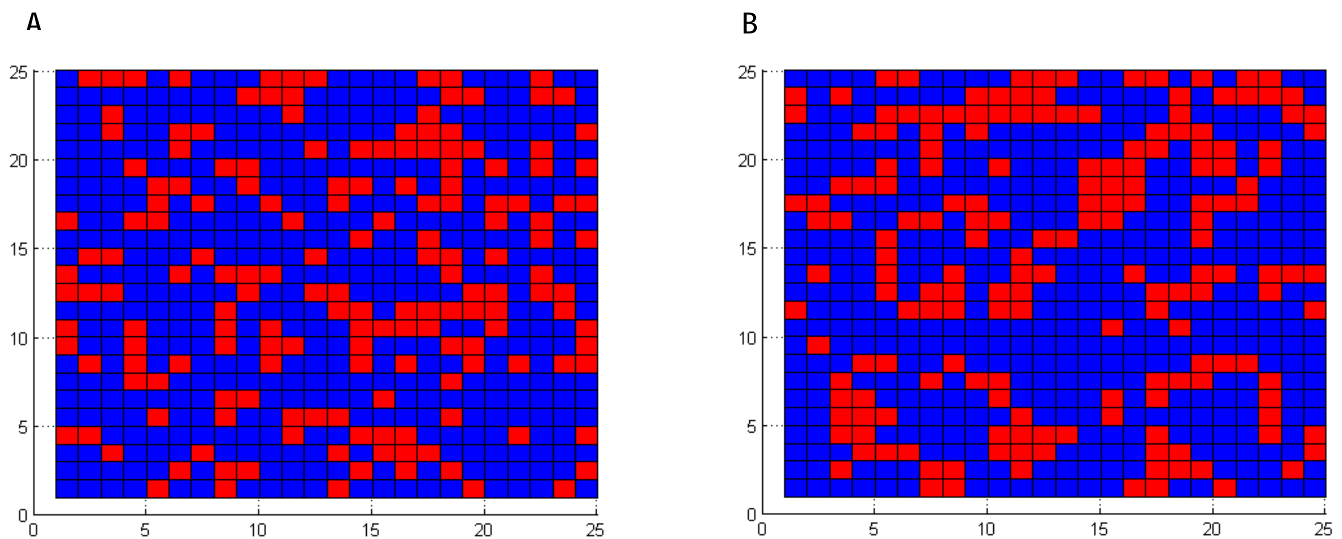


Figure 52: Increasing phase separation at constant temperature, 700K, gas phase composition, 30% A, and different fluxes. A) 1 ML/site-sec B) 9ML/site-sec

7.4 Effect of Gas Phase Composition

Gas phase composition also plays a significant role in the order parameter. As the percentage of type A particles in the gas phase increases, the order parameter at a given temperature and adsorption rate also increases (Figure 53). As stated previously, type A particles have been colored red and type B particles blue.

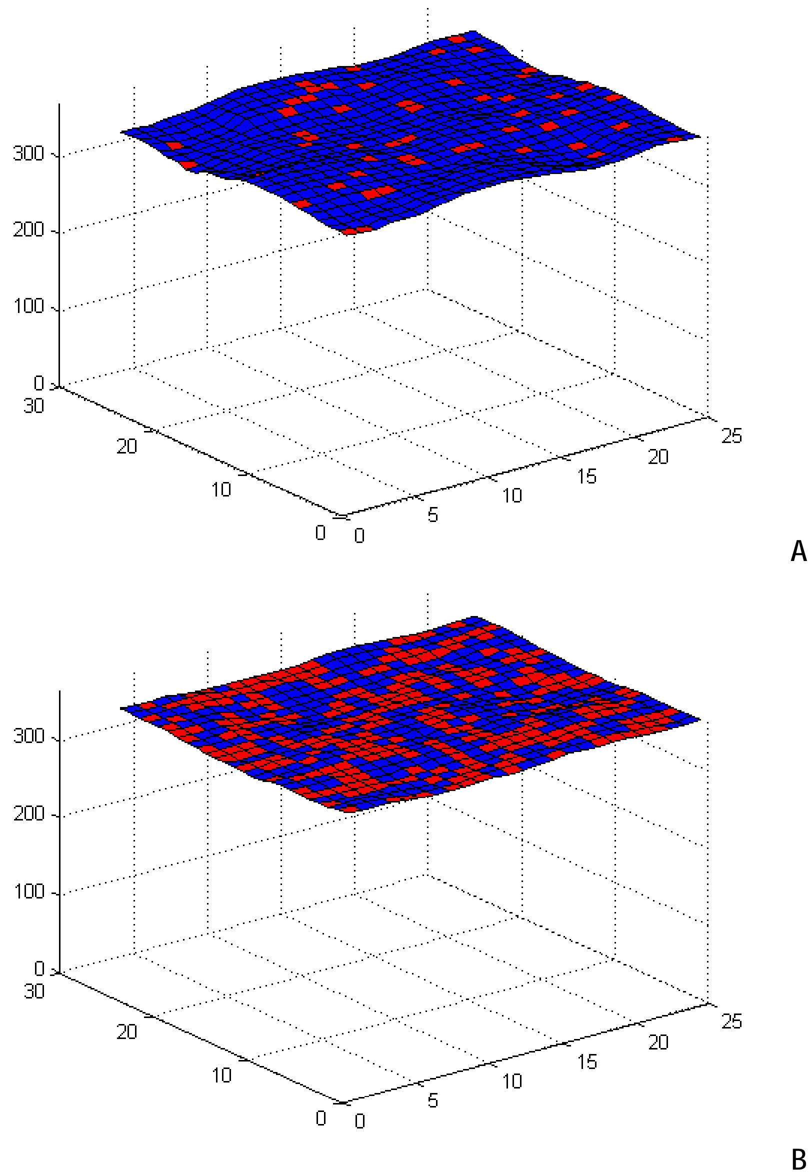


Figure 53: Effect of gas phase composition on the order parameter at constant temperature, 700K, and flux, 7 ML/site-sec. A) 10% B) 50%

8 Multiple Component Control of Thin Film Growth

8.1 Control Goals

In order to control a multi-component system the dependent system parameters discussed in section 4.1 have been redefined as roughness, growth rate and order parameter. As discussed previously, the controller goals include minimizing roughness and maximizing growth rate. To ensure that thin film properties are uniform it is required that the film have a uniform distribution of type A and type B atoms. With these control goals in mind, the general deposition goal of depositing a 100 nm film was analyzed.

8.2 Optimal Profile

Optimal profiles using procedures discussed previously were found for different gas phase compositions. Simulations were run at 10%, 20%, 30%, 40% and 50% of type A atoms in the gas phase at constant temperatures and fluxes. The objective function was adjusted for the new control parameters and has the following general equation:

$$OF = \frac{A\times(t_f)}{(t_{max}-t_{min})} + \frac{B\times(r_f)}{(r_{max}-r_{min})} + \frac{C\times(g_f)}{(g_{max}-g_{min})} \qquad [13]$$

Each parameter, time (t), roughness (r), and order parameter (g) is converted into a fraction based on the minimum and maximum values observed in the data set. The weighting factors A, B, and C were chosen to be 0.1, 0.4 and 0.5 respectively. These parameters place a heavy emphasis on the uniformity of the thin film and the final roughness.

The optimal profile is found at the point where the objective function is minimized in a given set of data. Simulations were performed with constant adsorption rates (1, 3, 5, 7, and

9ML/site-sec) at constant temperatures (500, 600, 700, 800 and 900 K). As seen in the surface plots, the minimization of the three different parameters, time, roughness, and order parameter, require different operating conditions that are not directly correlated. To minimize time, high adsorption rates are necessary, while the temperature has no effect. Roughness is minimized at low adsorption rates and high temperatures. The order parameter is minimized at low temperatures and high adsorption rates.

The additional parameter of changing gas phase compositions also affects the optimal profile. One optimal profile was generated for each gas phase composition. At lower percentages of type A atoms, order parameters are naturally larger and the control of order parameter is not as strict. The optimal profiles are shown below in Figure 54 through Figure 59.

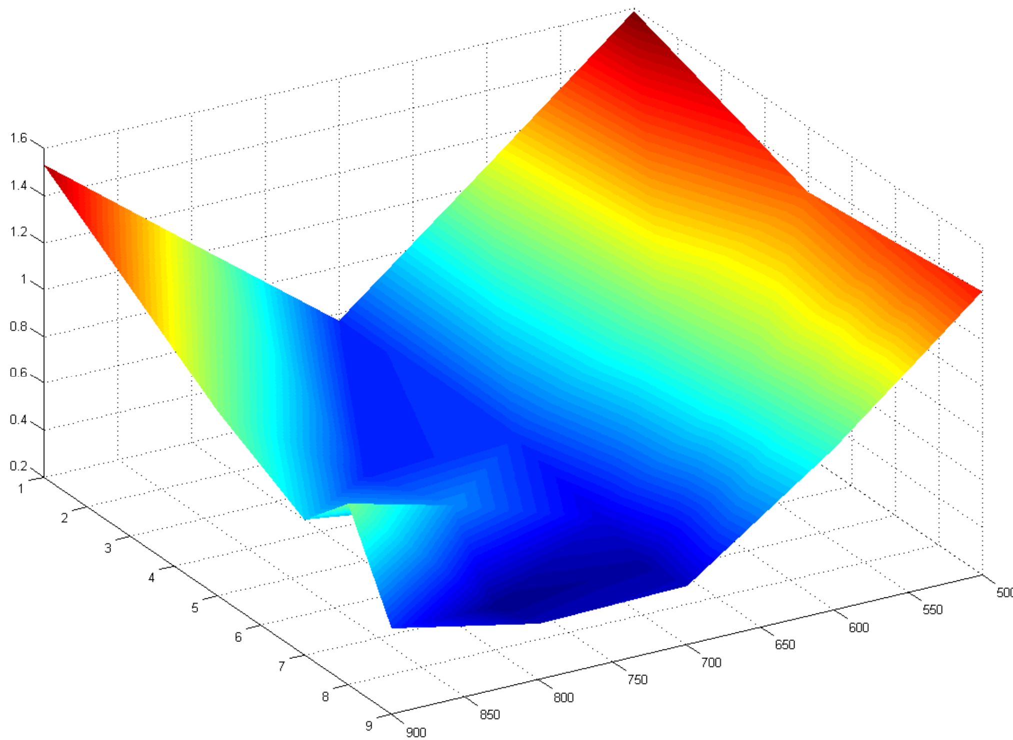


Figure 54: Objective function surface for 10% A in the gas phase.

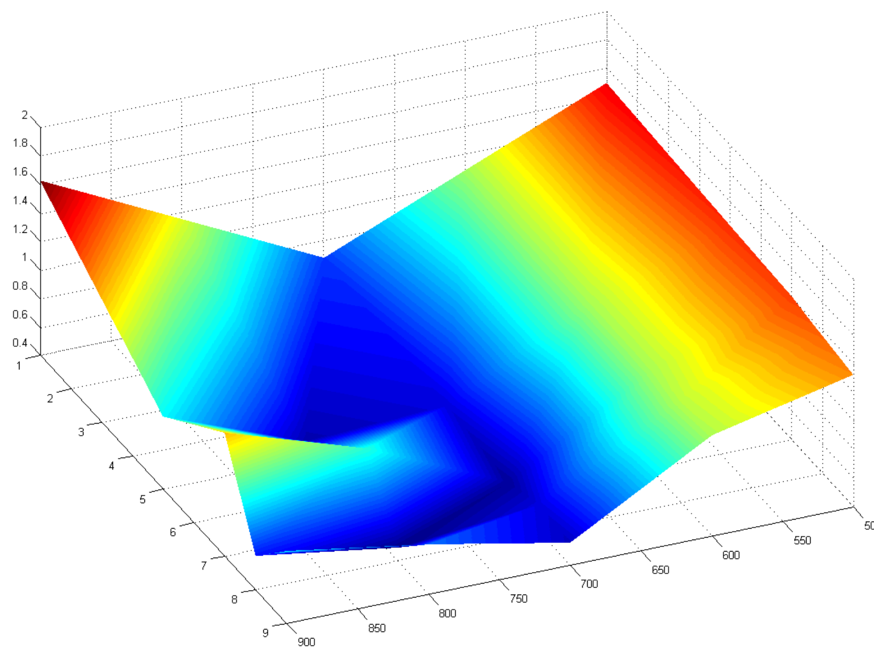


Figure 55: Objective function surface for 20% A in the gas phase.

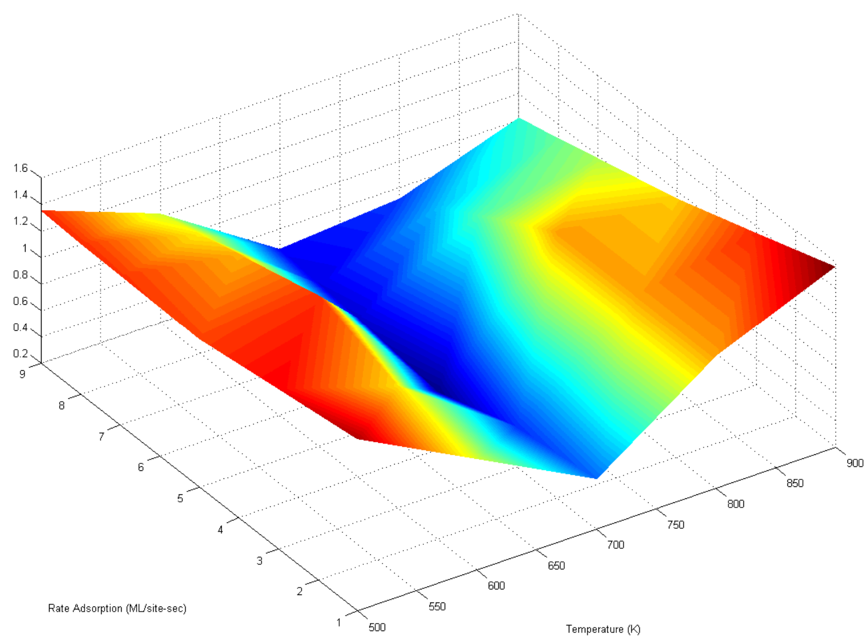


Figure 56: Objective function surface for 30% A in the gas phase.

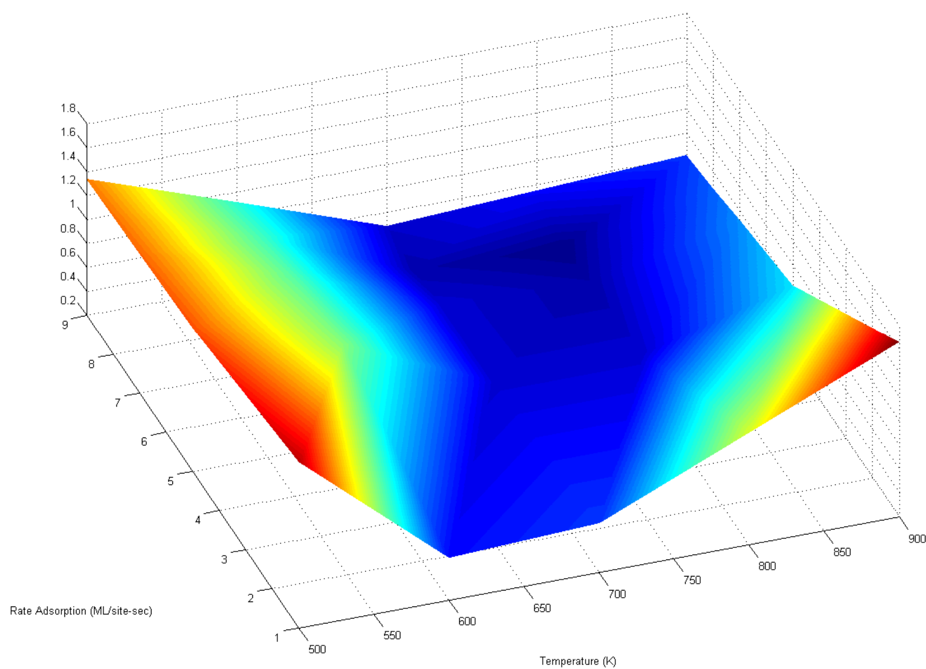


Figure 58: Objective function surface for 35% A in the gas phase.

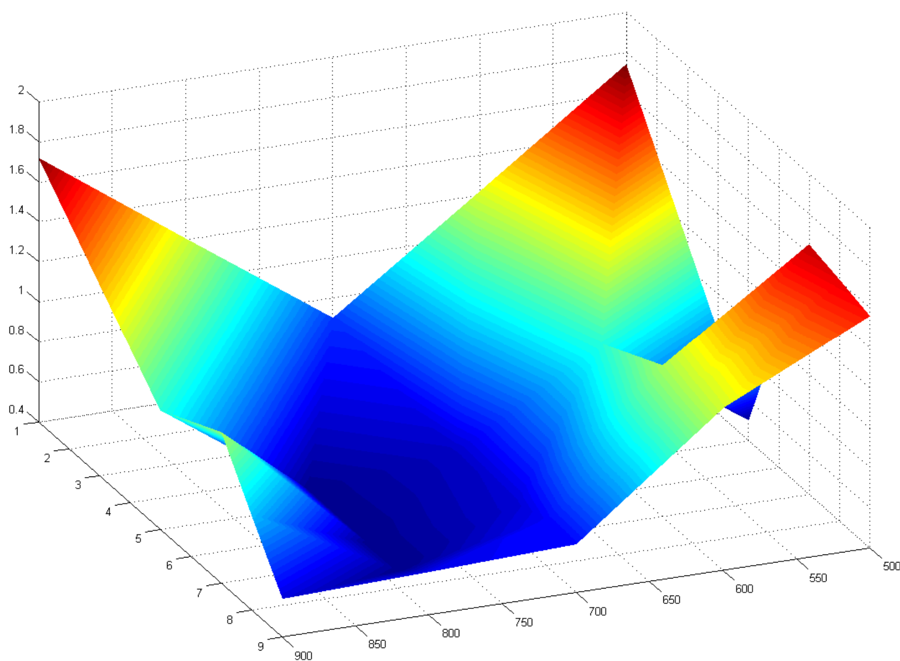


Figure 57: Objective function surface for 40% A in the gas phase.

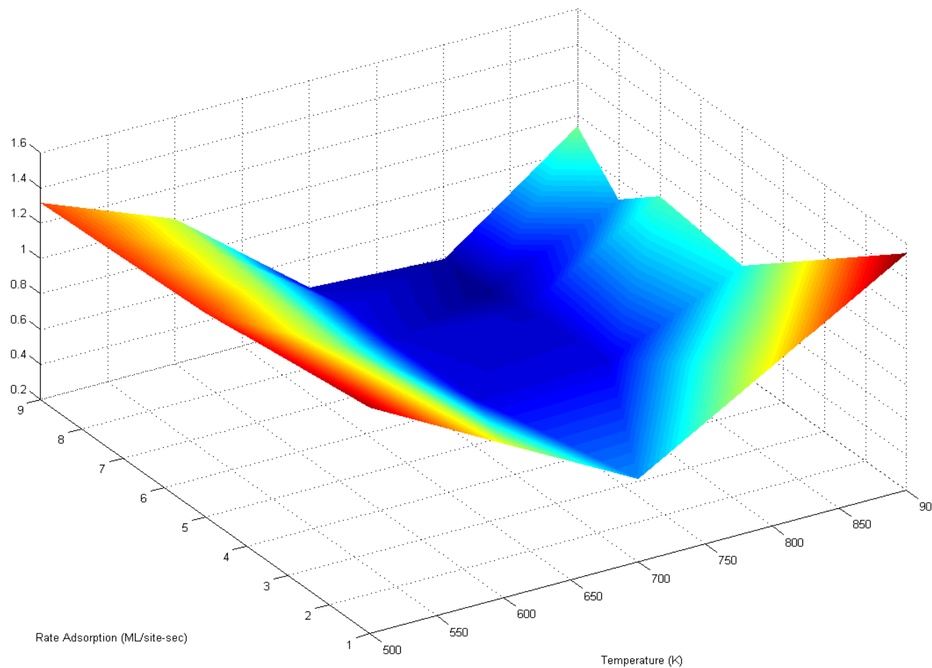


Figure 59: Objective function surface for 50% A in the gas phase.

8.3 Control Strategy

The controller strategy used for the multi-component model was similar to the one discussed for a single component controller. The optimal profile is dependent on the fraction of type A atoms in the gas phase and the dependant variables sampled are the roughness, growth rate and order parameter. The temperature and flux are used to control the surface morphologies. A flow diagram of the simulation is shown in Figure 60.

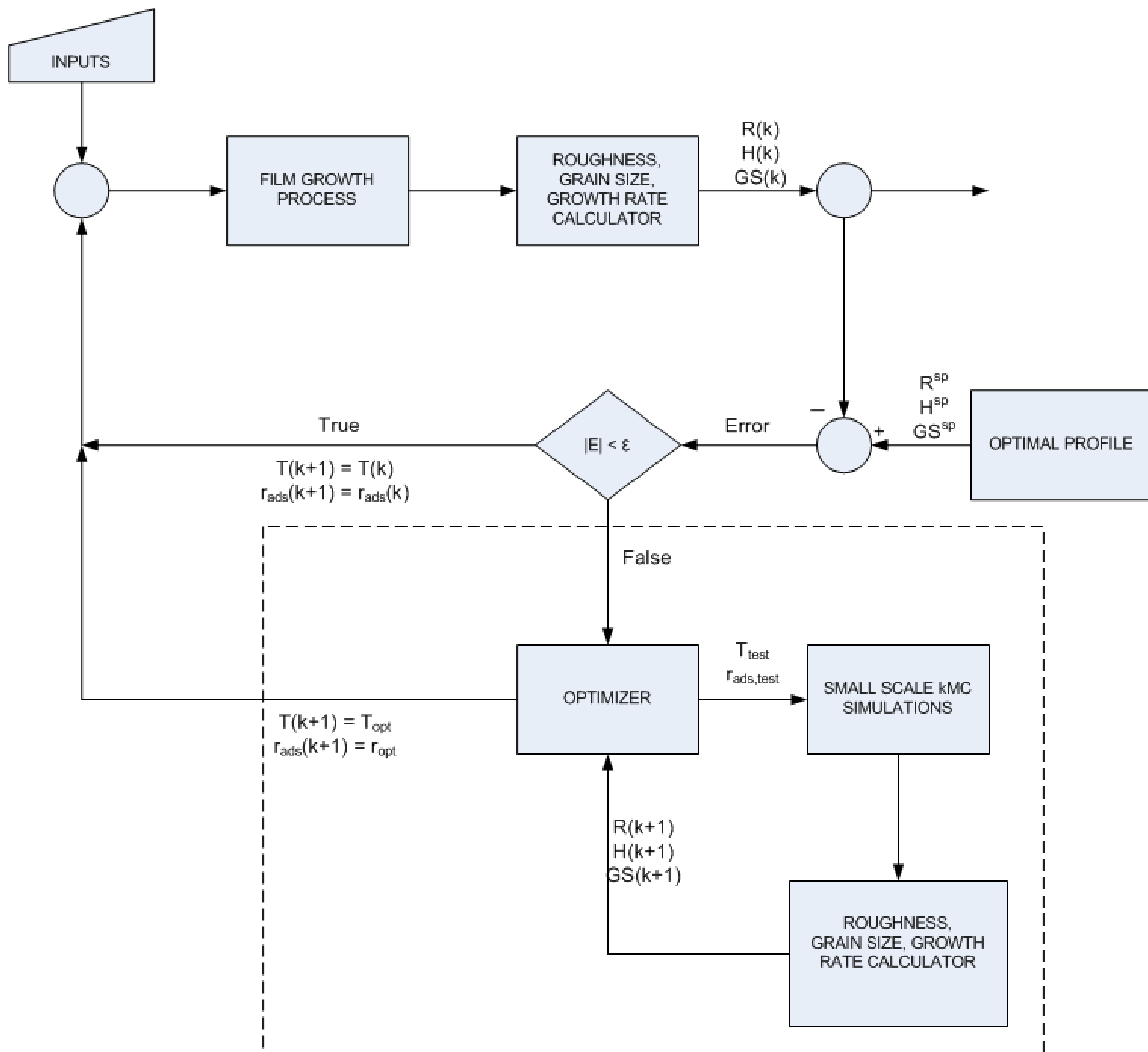


Figure 60: Controller block diagram.

9 Results of Multiple Component Control of Thin Film Growth

Based on the optimal profiles found in section 8.2 the following optimal operating parameters, shown in Table 5, were found. Each profile was rerun five times and average values of the expected roughness, height and order parameter were used as the reference trajectory in the controller. It should be repeated that these optimal profiles were run off of a flat surface starting with an initial height of zero. The recorded temperature and adsorption rates were kept constant through the course of the simulation.

Percent A	Adsorption Rate (ML/site-sec)	Temperature (K)
10	8	800
20	8	800
30	5	700
35	8	800
40	8	800
50	8	800

Table 5: Optimal conditions for different percentages of A in the gas phase.

To start controller testing certain operating parameters were specified. All tests were conducted with 30% A in the gas phase and upon each initial surface a 100 nm film was deposited. To examine the system's response, the appropriate optimal profile corresponding to a 30% composition of A, was used. This profile is equivalent to an optimal adsorption rate of 5 ML/site-sec, a surface at 700K and an average order parameter of 2.28. The development of the roughness and height profiles are seen in Figure 61 and Figure 62 in light blue.

To examine the controller behavior in response to a small deviation in the initial roughness a surface starting with random height at every lattice point from zero to ten was used. The overall roughness of this surface was 3 ML with an initial average height of 5 ML. Figure 61

and Figure 62 show that although the controller was able to reach to optimal profile, there were significant fluctuations in the surface roughness. By examining the temperature and flux profiles this controller followed, the cause of these fluctuations can be seen (Figure 63 and Figure 64).

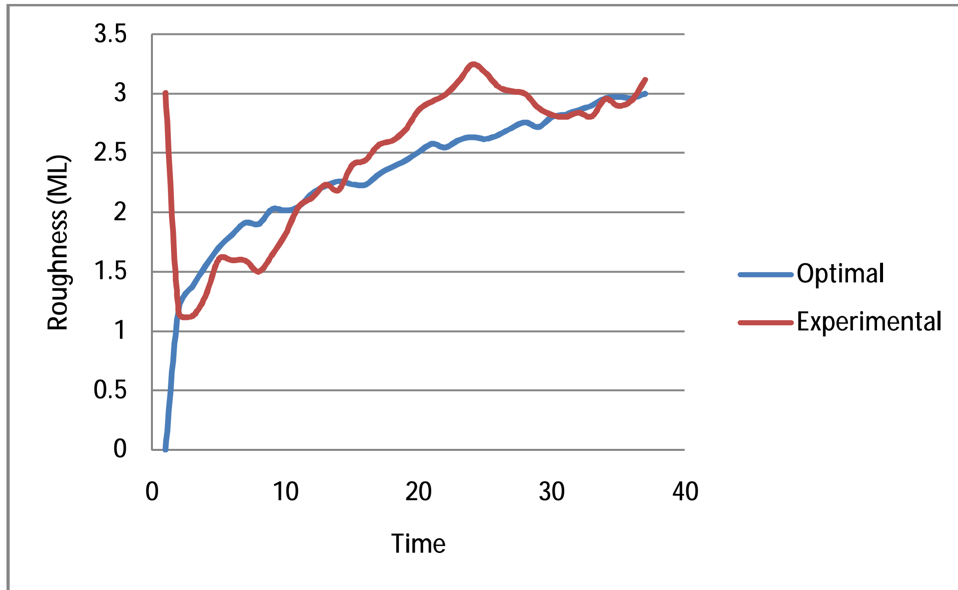


Figure 61: Roughness profile for an initial surface of random height between 1 and 10.

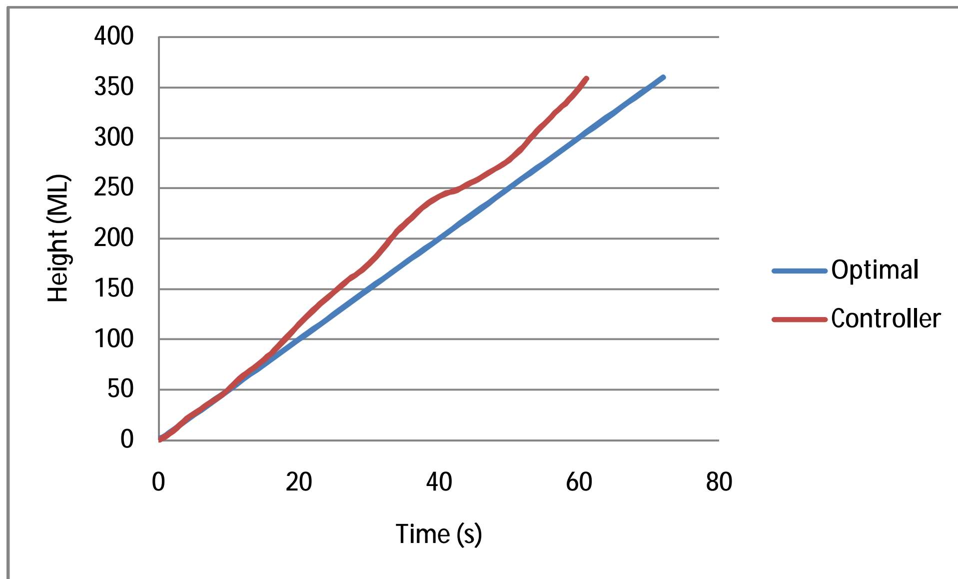


Figure 62: Height profile for an initial surface of random height between 1 and 10.

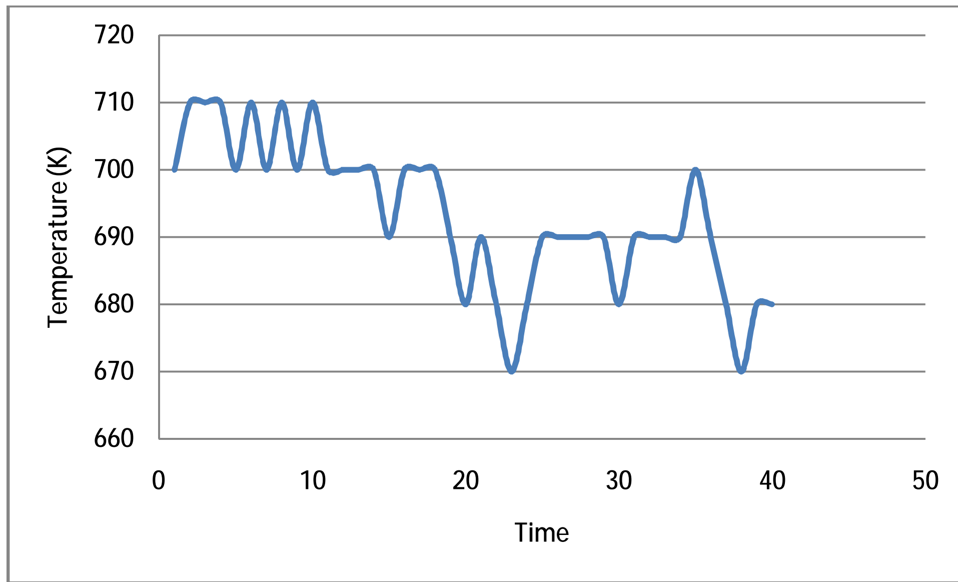


Figure 63: Temperature profile for a surface starting with a random height between 0 and 10.

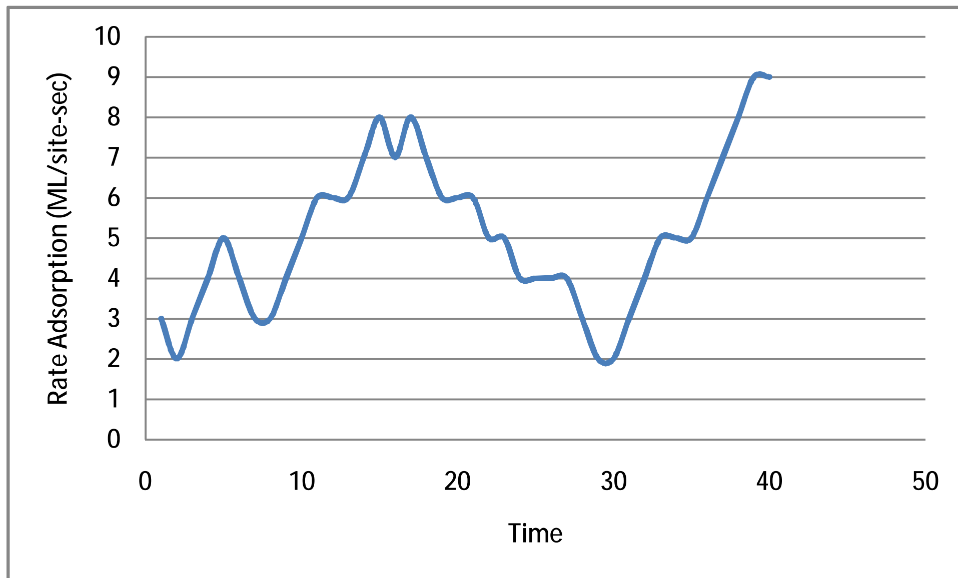


Figure 64: Adsorption rate profile for a surface starting with a random height between 0 and 10.

The roughness of the surface is a parameter that is very sensitive to temperature. In order to control the roughness within the allowed error bound of .05, the temperature fluctuations were very frequent. These temperature fluctuations not only changed the roughness, but also the order parameter which resulted in compensatory fluctuations in the gas flux. If this error bound were greater, it is possible that the fluctuations would be reduced. In other words, control is more difficult for a binary system in which compositional distribution must also be considered.

To further test the controller, another initial surface with a greater roughness deviation was used. This hill and valley surface is similar to that shown in Figure 35. The surface has an average height of zero monolayers, with a minimum height of negative ten monolayers and a maximum of ten monolayers. Its average roughness was calculated to be 10 monolayers. The surface reaches its periodic maximum height every five lattice points. This periodicity allows the controller to accurately use a 10 x 10 grid as a test input into the optimizer. This disturbance is in the roughness only as the initial height is at zero, which is the same as for the optimal profile.

The initial composition of the surface, all type A atoms, is not considered to be a disturbance because the order parameter is dependent on temperature and the rate of adsorption, but not on the order parameter of the thin film below the surface. In other words, after the growth of the first monolayer, the order parameter will be a result only of the operating conditions of the controller and not of the surface that this monolayer was deposited on.

As can be seen in Figure 65 below, the controller caused the roughness profile of the surface to approach that of the optimal profile. The final objective function values were calculated to be .26 for the controlled and 1.24 for the uncontrolled run. Without control, the roughness increases from its starting value of 10 ML to above 20 ML. With the controller on, the roughness is quickly decreased and levels out below 5 ML, but it does not reach the optimal

profile. This is due to the undesired increase in order parameter at higher temperatures. Since the order parameter and roughness are about equally weighted the controller had to balance decreasing the temperature to decrease the order parameter and increasing the temperature to minimize the roughness. Furthermore, in order to decrease the order parameter the rate of adsorption was increased, which caused the controlled to build to 100 nm before the optimal profile. It is possible that if the film were grown with a controller more sensitive towards roughness the thin film roughness profile would have matched the optimal roughness profile.

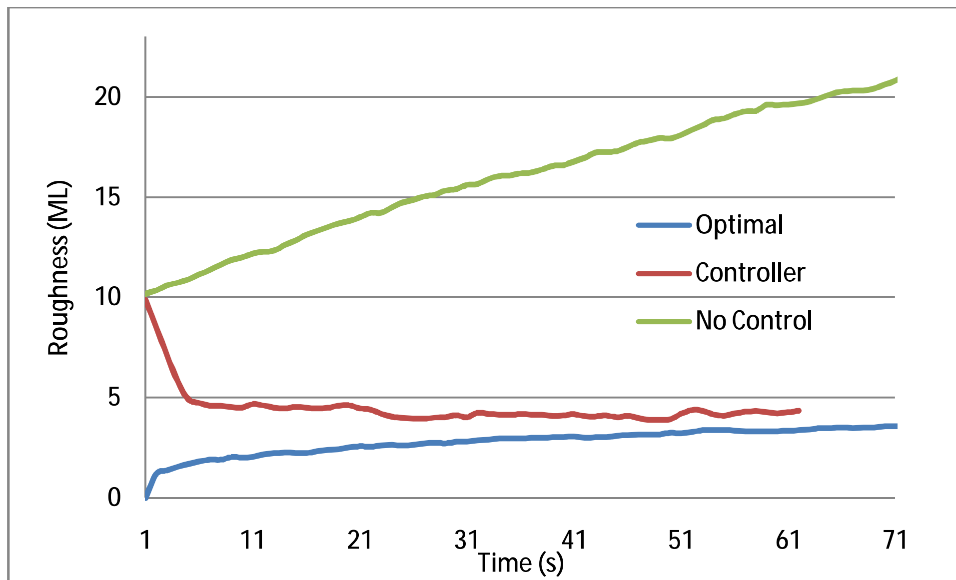


Figure 65: Roughness profile for a hill/valley surface with and without control.

To show the evolution of these surfaces visually Figure 66 shows the growth of the surface with control over time and Figure 67 the difference between the final surfaces with and without a controller. With the controller the final surface is much smoother, but with larger order parameters. The hills and valleys, which are initially distinct, start to fade in the controlled surface and are flattened out, while on the uncontrolled surface they remain. Since particles

adsorb onto the surface uniformly the uncontrolled surface still shows the hill and valley macrostructure with additional height disturbances on a lattice by lattice size scale due to these adsorption events.

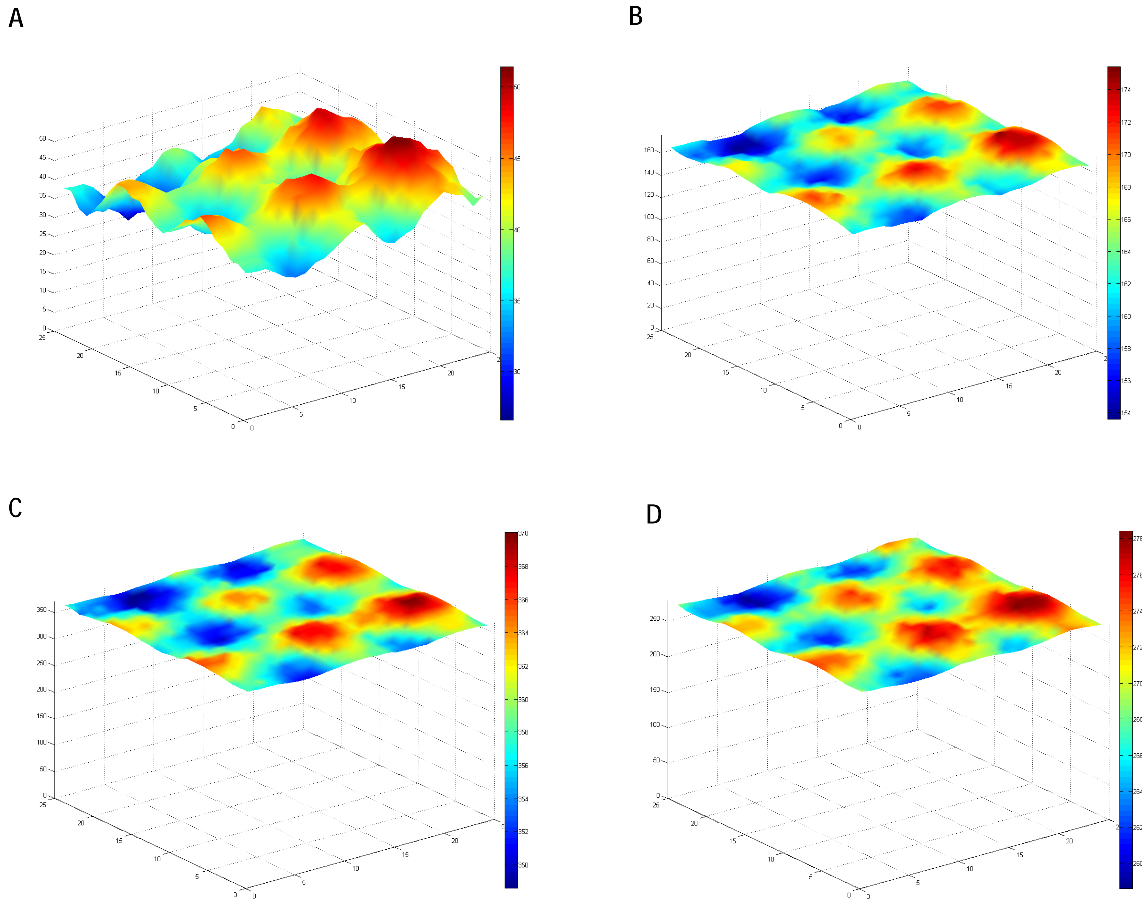


Figure 66: Evolution of surface roughness with control. A) 10sec, B) 30 sec, C) 50 sec, D) 63 sec.

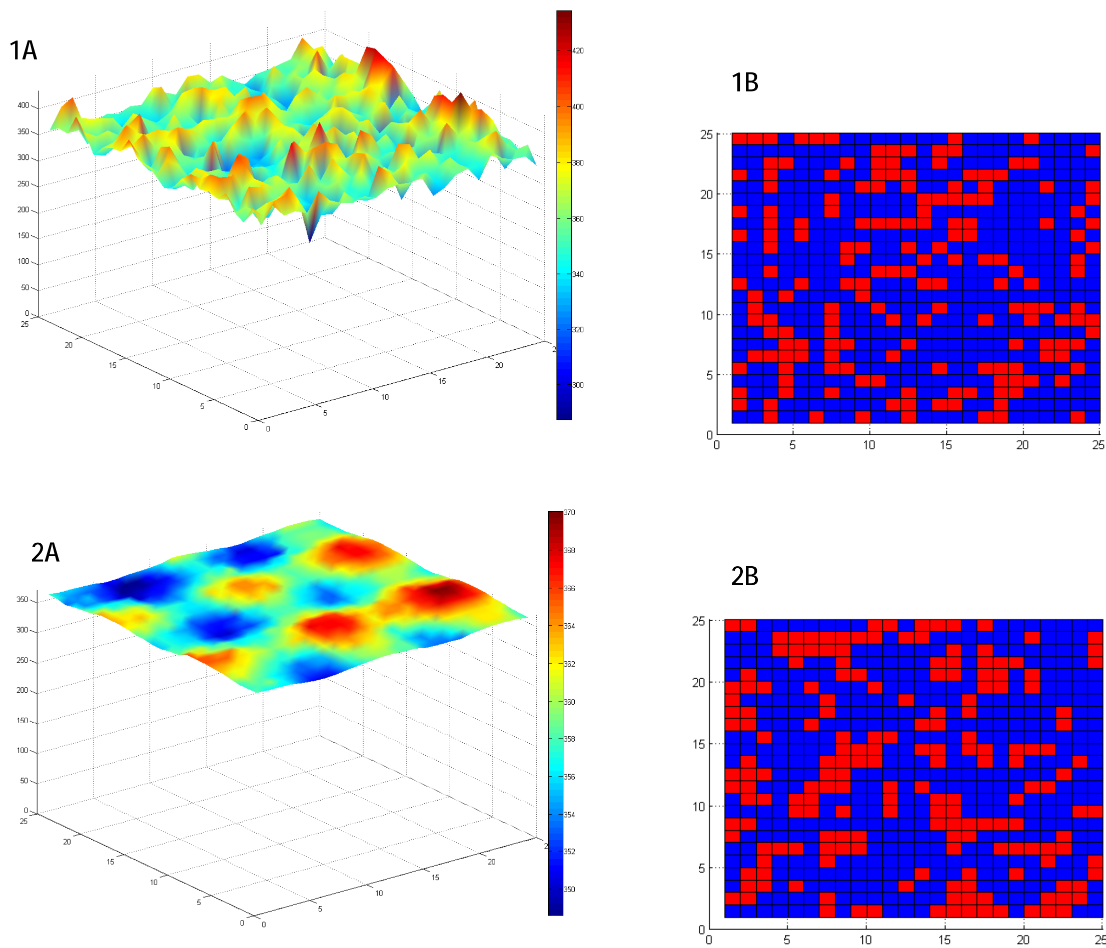


Figure 67: (A) Hill/valley surface contour and (B) particle distribution images: 1: without controller action, 2: with controller action.

It should be noted that the control of the hill and valley surface also fluctuated in its temperature and gas flux. While the controller is effective, it can be improved to recognize larger surface morphologies and moderate the temperature and flux response at small time scales. Since the roughness and order parameter are both very sensitive to temperature, an optimal profile in which their importance in the objective function is almost equally weighted is hard to control. If one of these parameters were selected, the system would be easier to control.

The tests performed on surface roughness disturbances show that the proposed controller can drive the system towards a desired trajectory. However there are frequent fluctuations in the manipulated variables. In order to not change the system temperature and gas flux as often the allowed controller error can either be increased or a new method of control which predicts into the future for a longer time span can be implemented. While the second option will give better control of the system it also requires additional computation time and will slow the controller.

Multi-component system control is more difficult than single component control due to the additional degrees of freedom introduced into the system. The gas phase flux, the substrate temperature and the gas phase composition can all be treated as manipulated variables. Furthermore, the model must be able to accurately predict the response of the surface both morphologically and compositionally. Tight control on a system where two parameters, the roughness and order parameter, are both strongly and inversely correlated to a manipulated variable, temperature, is difficult. While the controller proposed in this report treats the gas phase composition as a constant, the control mechanism can be expanded to treat this as a manipulated variable. The measurement of surface roughness can also be expanded to recognize macroscopic features of the thin film surface and to control their growth. With improvements in computer process speed technology, the proposed controller mechanism can be improved for tighter, more reliable, and cost effective control.

10 Financial Analysis

10.1 Semiconductors Market Overview

Any complete financial plan for a potential start-up company must begin with an investigation and understanding of the current marketplace. The proposed product has the potential to improve efficiency, quality, and throughput for chipmakers in the semiconductors industry. As a whole, the semiconductors industry grew by 2.7% to $238 billion globally and is expected to climb to the $300 billion range within the next five years (Semiconductors 2009). However, due to the current economic climate, immediate forecasts indicate a possible leveling, or shrinkage, of the overall market over the next two years. This software for model predictive control of thin film deposition would be marketed primarily to companies that perform thin film deposition in the production of wafers for microelectronics and other devices. Therefore the start-up company can be considered a part of the semiconductor equipment industry, with potential clients in the semiconductor production industry.

The semiconductors industry would not be possible without the technological development of advanced equipment to produce increasingly powerful and novel chip solutions. Unfortunately the semiconductor equipment industry has recently experienced a sharp decline, falling from $42.8 billion to $30.9 billion, a decrease of nearly 30% (Semiconductor Equipment 2009). Historically, both the semiconductors and semiconductor equipment industry have been more volatile than the general economy, therefore it is difficult to accurately assess and forecast the direction of the industry.

As a small company providing software solutions for the development and control of high quality thin films, there is the potential to penetrate the global market since selling software

does not have the physical hurdles of shipping and installation in most industries. However, other difficulties such as providing customer service in other regions require significant costs; therefore it is appropriate to limit the desired market to North America. The global breakdown of the industry is given in the following figure, Figure 68, adapted from the Semiconductor Equipment industry report. Based on the current breakdown, the start-up company would be entering a $5.5 - $6.5 billion industry in North America.

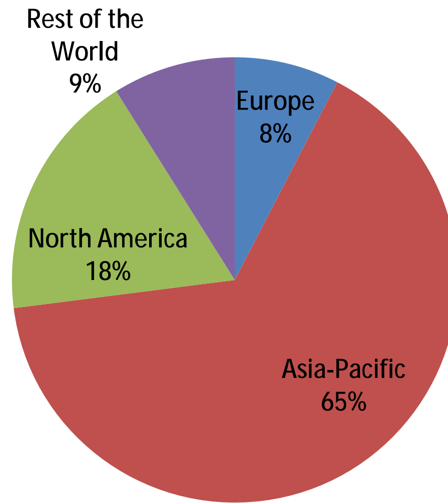


Figure 68: Global Semiconductor Equipment market segmentation.

10.2 Competitive Environment

For a start-up company to be successful breaking into a well established, yet rapidly developing industry, it must have a deep understanding of the operations of current industry powerhouses and companies that will comprise the primary competition. Major chip producers, such as Intel and Samsung, generally purchase equipment from a variety of companies. However, there are significant advantages to buying from a single company that provides complete solutions from front-end to back-end chip processing. Any emerging company faces the challenge of competing with larger, well established, industry leaders such as Applied Materials and KLA-Tencor. Therefore, a successful entrant to the marketplace must provide a product that is unique and effective. The general rule of thumb in the industry is that a product must provide at least 25% value added to a company before the company will consider a purchase. An appropriate goal for a start-up company in this industry is to penetrate the market with a product that provides a competitive advantage and ultimately become bought-out by a larger firm. Four companies, Applied Materials, Tokyo Electron, ASML Holding NV and KLA-Tencor Corporation, together hold nearly 80% of the market. Although all are global companies to some extent, Applied Materials and KLA-Tencor will be analyzed since they are based in the United States. Smaller companies that provide specific software solutions, such as STR Group, Inc and Synopsys, Inc, are also analyzed to encompass all the potential competitors.

Applied Materials offers systems that cover every step of chip fabrication, from front-end wafer processing, to material deposition, to back-end wafer metrology and cleaning. Applied Materials also offers systems for both factory scale and small tool scale software control. The company has shown an interest in expanding their software offerings and expertise, having bought Brooks Automation's Software division for $125 million in 2006. Information regarding

the specific transfer of assets was not publicly available. As a physical equipment provider, Applied Materials has a trained support staff of 2,450 engineers to cover 22,000 installed systems globally. This information can be used to estimate the specific market for a software control system product.

$$\begin{aligned} &\mathit{market\ size} = \\ &= \frac{\mathbf{22{,}000}\ \mathit{installed\ systems}(\mathit{Applied\ Materials}) \times \mathbf{16\%}\ \mathit{North\ America\ sales}}{\mathbf{20\%}\ \mathit{market\ share}} \\ &= \mathbf{17{,}600}\ \mathit{total\ systems\ installed\ in\ North\ America} \end{aligned}$$

The market size calculated based on Applied Materials data will be used as an estimate in later sections.

KLA-Tencor is very similar to Applied Materials in that it provides equipment and services to a broad area of the semiconductor production industry. One specific product that KLA-Tencor markets to both research institutions and industrial companies is PROLITH lithography modeling software. While this software would not be a direct competitor of software for control of a deposition process, its development mimics a potential path. PROLITH was developed as part of a PhD thesis and resulted in a small start-up company. The software provided a novel system for companies to research various lithography strategies by simulation before having to invest in capital equipment to perform the procedure, greatly reducing research and development costs. This advantage allowed the company to become profitable in the semiconductor equipment industry without having a widespread selection of products and services, and the company was acquired by KLA-Tencor after six years. Currently, the software is provided for free for academic research use (stripped down version) and for $18,000 per year per computer license for commercial use. The selling point remains the software's ability to

perform standalone simulations and integrate with other hardware solutions, saving time and money in research and development.

STR Group, Inc is a small company based in Russia that provides software and consulting services centered on the modeling of crystal growth and devices. Of specific interest to the proposed potential business is their Virtual Reactor-CVD software. The tool models all aspects of a deposition reactor in detail, and its capabilities include gas phase particle transport, growth kinetics, heat and mass transport analysis in various flow situations, as well has versatility to be adapted to different reactor geometries in three dimensions. Virtual Reactor-CVD is sold to both academic and commercial clients. Based on the configuration the license price ranges from $23,460 to $40,595 per year per computer, with a 65% discount if the software is used for non-profit ventures.

Synopsys, Inc already offers KMC process simulation software for semiconductor applications. The software is once again marketed to academic and commercial customers. Through the Synopsys University Program, institutions can purchase a two year subscription of 50 seat licenses for $3,000. The costs are significantly more to use the software for commercial purposes. Synopsys has three different strategies to sell their software to industry. One option is a Technology Subscription License (TSL). TSL's have a predetermined, finite term, as defined by the sales contract. During the length of the contract, the customer has the right to receive any new updates to the software and maintenance and customer service. Another option for the customer is a Term License contract. Like TSL's, the software is activated for a finite term. However, Term Licenses do not include software updates. Maintenance can be provided at an additional cost, usually calculated as a percentage of the initial license fee. Lastly, Perpetual Licenses allow the customer to continue using the software product forever, as long as

maintenance and support is renewed. Under this option, there are no rights to updates for the customer. Due to the all encompassing nature of the contract, TSL's are the most attractive method to market and sell the proposed software controller product. In the business model that follows, TSL's will be the method of sales. As the business grows it may become worthwhile to reexamine other sales options.

Analysis of the current marketplace reveals there is a potential for entry to the semiconductor equipment industry by a small start-up company that provides a specific solution that will give a significant benefit to its clients. Despite the volatility of the industry, small companies can penetrate the market significantly and become taken over by larger industry leaders. Additionally, semiconductor producers are willing to pay tens of thousands of dollars per license for a useful simulation software system. This information will help justify and make additional assumptions in later sections.

10.3 Business Model

With the semiconductor device, semiconductor equipment, and software market segment analyzed, it is now important to build a financial framework to evaluate the viability and potential for the proposed model predictive control software solution. Several assumptions are made in the following analysis. The potential market was judged based on data from Applied Materials, presented in the previous section. The time horizon for the start-up company was taken as five years, after which the company would hope to be bought out by a larger firm. The first year after founding the company was assumed to be devoted entirely to research and development to fine tune and prepare the product for sales beginning in the second year. The pricing of the product was analyzed using cost plus pricing and target return pricing. At this stage in development it is difficult to make accurate evaluations of customer needs and preferences for use in value based pricing. Using various potential growth models, the price required for an internal rate of return of 20% was found. A 3% adjustment for inflation was used for all recurring costs and revenues.

The various costs the company anticipates to incur were broken down as follows:

Cost Breakdown

Bare Module/Capital Costs		*Unit Price*	*Units*	*Quantity*		*Total*
Clean Room (Class 1000 25% chase)	$	400.00	/sq ft	1600	$	640,000.00
Deposition Reactor	$	290,510.00	/reactor	1	$	290,510.00
Fume Hood	$	2,000.00	/ hood	5	$	10,000.00
Glove Box	$	8,410.00	/ box	2	$	16,820.00
Computers	$	2,000.00	/ pc	9	$	18,000.00
AFM	$	100,000.00	/ m'scope	1	$	100,000.00
Profilometer	$	15,000.00	/ unit	1	$	15,000.00
Refractometer	$	15,000.00	/ unit	1	$	15,000.00
					$	1,105,330.00

Operating Cost		*Unit Price*	*Units*	*Quantity*		*Total*
Office Space	$	20.00	/sq ft/yr	5000	$	100,000.00
Operating Cost (office supl + util)	$	2,000.00	/yr	1	$	2,000.00
MATLAB licenses	$	2,000.00	/seat/yr	3	$	6,000.00
Reactor 1% of purch cost	$	2,905.10	/yr	1	$	2,905.10
Clean room, 2% of purch cost	$	12,800.00	/yr	1	$	12,800.00
Cost of Sales (10% of sales)		20%	variable/yr			
					$	123,705.10

Labor Cost		*Unit Price*	*Units*	*Quantity*		*Total*
Software Developer	$	80,000.00	/person/yr	3	$	240,000.00
Lab Coordinator	$	80,000.00	/person/yr	1	$	80,000.00
Lab Technician	$	60,000.00	/person/yr	2	$	120,000.00
Marketing	$	80,000.00	/person/yr	1	$	80,000.00
Implementation/ Cust. Service	$	60,000.00	/person/yr	3	$	180,000.00
CEO	$	150,000.00	/person/yr	1	$	150,000.00
VP-Legal	$	150,000.00	/person/yr	1	$	150,000.00
				9 people	$	1,000,000.00

Recurring Fixed Costs	$	1,123,705.10
Capital Costs	$	1,105,330.00
Initial Investment	$	2,229,035.10

Although the final product is a piece of software to be implemented in conjunction with a control system of a deposition reactor, laboratory equipment is still necessary for in house testing, calibration, and analysis of the software, making up the majority of the bare module costs. A clean room is necessary to perform most laboratory procedures involving delicate wafer surfaces. A Class 1000 bay and chase clean room provides for decreased operational costs without compromising quality compared to a ballroom clean room. Despite the steep cost per square foot of clean room, a 1600 foot clean room installed initially gives the company room and flexibility to increase its wet lab research and development abilities. The most important piece of equipment is the actual thin film deposition reactor. The price quoted is from Cambridge NanoTech, Inc and includes installation costs, a three year warranty, and associated equipment to operate the reactor. This system is flexible to study various thin films. To complete the laboratory, several post processing analysis tools are required. These bare module costs will be considered depreciable capital. Considering the rapid technological innovation in the semiconductors industry and a five year horizon, straight line depreciation will be taken as 20% of total depreciable capital. Quotes for the bare module costs can be found in Appendix B.1-3.

Due to the time horizon of five years, corporate space will be leased rather than purchased. Currently, the prevailing market value for office space is $20 per square foot in Philadelphia, PA. To develop the software, MATLAB licenses are necessary. Operating costs of the clean room and reactor are estimated conservatively and are not trivial. Cost of sales was also conservatively estimated at 20% of sales.

The start-up company will require a full time staff of 12 employees, split up into four divisions with three employees each. One division will be exclusively devoted to developing the control software. Another division will be responsible for performing all necessary experimental

duties in the clean room and wet lab space. The management division will include a marketing director, chief executive officer, and legal advisor. A customer support division will be required to implement the product on site, train end users, and handle customer service queries. Depending on the growth of the company, staff may need to be added to any or all of the four divisions.

Revenues were calculated based on three sales growth models. An aggressive model uses an initial market penetration of 1% in year two with 100% sales growth in subsequent years. A more moderate model calls for the same initial success, followed by only 50% sales growth. Lastly, a conservative sales estimate estimates an initial penetration of 0.5% of the market and steady increase of sales of 0.5% of the market each year thereafter. The sales breakdown is presented in the following tables. The sales price was calculated based on the cash flow summary presented next.

Sales Breakdown

Year		Total Market Units	% of Market	Units Sold		Price
Sales - Aggressive						
	1	17600	0%	0	$	-
	2	17600	1%	176	$	7,150.00
	3	17600	2%	352	$	7,364.50
	4	17600	4%	704	$	7,585.44
	5	17600	8%	1408	$	7,813.00
Sales - Moderate						
	1	17600	0	0	$	-
	2	17600	1%	176	$	11,803.00
	3	17600	1.50%	264	$	12,157.09
	4	17600	2.25%	396	$	12,521.80
	5	17600	3.38%	594	$	12,897.46
Sales- Conservative						
	1	17600	0	0	$	-
	2	17600	0.50%	88	$	19,313.00
	3	17600	1%	176	$	19,892.39
	4	17600	1.50%	264	$	20,489.16
	5	17600	2%	352	$	21,103.84

Internal rate of return (IRR) is defined as the discount rate at which the net present value (NPV) of all future cash flows reduces to zero. IRR is a profitability measure used in industry to evaluate potential projects. Typically, an IRR in the range of 15-30% is reasonable in the semiconductor industry. Therefore, the price required for an IRR of 20% for each growth model was calculated. The cash flow summary for the moderate growth model follows. The analogous summary for the aggressive and conservative models can be found in Appendix B.4-5.

Cash Flow Summary

Year	% Market	Sales	Capital Costs	Fixed Costs	Cost of Sales	Depreciation Allowance
1	0	-	(1,105,330.00)	(1,123,705.10)	-	-
2	1%	2,077,328.00	-	(1,157,416.25)	(415,465.60)	(221,066.00)
3	1.50%	3,209,471.76	-	(1,192,138.74)	(641,894.35)	(221,066.00)
4	2.25%	4,958,633.87	-	(1,227,902.90)	(991,726.77)	(221,066.00)
5	3.38%	7,661,089.33	-	(1,264,739.99)	(1,532,217.87)	(221,066.00)

Taxable Income	Income Tax Costs	Net Earnings	Annual Cash Flow	Interest Adjusted
	-	-	(2,229,035.10)	$ (2,229,035.10)
283,380.15	(184,197.10)	99,183.05	320,249.05	$ 266,874.21
1,154,372.67	(750,342.23)	404,030.43	625,096.43	$ 434,094.75
2,517,938.19	(1,636,659.83)	881,278.37	1,102,344.37	$ 637,930.77
4,643,065.47	(3,017,992.56)	1,625,072.92	1,846,138.92	$ 890,306.19
			NPV:	$ 170.81

Tax Rate =	35%
Interest Rate =	20%

Based on the different growth models, the price per unit installation ranges from $7,150 to $11,803 to $19,313 in order for an IRR of 20%. According to current companies providing different software solutions, industry is willing to pay this order of magnitude cost for a control system that will provide increased throughput and efficiency to their processes.

11 Conclusions

Despite the current economic climate the semiconductor industry is still expected to grow. An analysis of the current marketplace revealed that although a few firms dominate the market, there is potential for a small start-up company to be competitive with a novel product. The presented control system can improve quality and throughput for chip producers and is likely to be of great value in the industry. There is evidence that companies are willing to pay upwards of $40,000 for related software systems. Depending on the growth of sales of this product, charging between $10,000 and $20,000 per license per year for this software will result in an IRR of 20% for the five year time horizon.

The proposed software can both evolve with new technology and be adapted to different types of reaction systems. The software aims to maximize accuracy while minimizing computation time to improve its efficiency. Depending on the type of control required the model can be adapted to maximize output or accuracy. As computing power continues to improve, even more rigorous and more detailed experiments and simulations can be performed. By modifying the energy barriers in the event rate equations the software simulations can be used to study many different chemical depositions and interactions.

As the computation time decreases due to emergent computer processing technologies the controlling mechanisms can be improved. By calculating the roughness based on each lattice point height, the overall morphology of the system is lost. An improved control scheme would calculate roughness with different levels of resolution, make a conclusion on the morphology of the surface, and act appropriately based on those conclusions. The model could also predict the reaction of the system to a change in the manipulated variables over a greater time scale and so

more quickly identify large morphological changes. By adding considerations for different roughness resolutions and for greater predictive accuracy, the flexibility and usefulness of this system will only be improved.

The general control scheme presented proves that this method of model predictive control is effective in directing systems that have experienced disturbances back to the desired profile. With the proposed product consumers will have tighter control over chip product surfaces, will more easily be able to meet manufacturing specifications and will increase the throughput of their existing facilities.

12 Acknowledgements

The authors would like to acknowledge the advice and guidance provided by Professor Leonard A. Fabiano, Dr. Warren Seider, and Dr. Talid R. Sinno. Professor Fabiano has lent his expertise on various product design issues. Dr. Seider has provided great assistance in implementing effective control strategies, and has also provided the fundamental skills required to complete a successful product design through our fall design course. As our faculty advisor, Dr. Sinno has been invaluable in discussing the science behind the simulation process and in developing an effective strategy to tackle the various control issues. Additionally, Matthew Flamm worked extensively to provide us with the core code, without which this project would not be possible. The design consultants have also devoted many hours providing useful suggestions and advice from an industrial point of view.

13 Bibliography

Azevedo, G. et al. “Effect of hydrogenation on the optical and structural properties of GaAs thin films prepared by rf-magnetron sputtering.” *Nuc. Inst. Meth.*, **238**, 329-333 (2005).

Carpick, RW and Salmeron, M. “Scratching the Surface: Fundamental Investigations of Tribology with Atomic Force Microscope.” *Chem. Rev.*, **4**, 1163-1194 (1997).

Fichthorn, K.A. and W.H.Weinberg. “Theoretical Foundations of Dynamic Monte Carlo Simulations.” *J. Chem. Phys.*, **95** 1090 (1991).

Granneman, E. H. “Thin Films in the Integrated Circuit Industry: Requirements and Deposition Methods.” *Thin Solid Films*, **228**, 1 (1993).

Kang, H. C., and W. H. Weinberg. “Dynamic Monte Carlo Simulations of Surface-Rate Processes.” *Acc. Chem. Res.* **25**, 253 (1992).

Lou, Y., Christofides, P.D. “Feedback control of surface roughness using stochastic PDEs.” *AIChE Journal* **49**, 2099-2113.

Nanostellar, Inc: www.nanostellar.com/multiscale.htm

Ni, D. and Christofides, P.D. “Dynamics and control of thin film surface microstructure in a complex deposition process.” *Chem. Eng. Sci.*, **60**, 1603-1617 (2005).

"Semiconductor Equipment Industry Profile: Global." *Semiconductor Equipment Industry Profile: Global* (March 2009): 1.*Business Source Premier*, EBSCO*host* (accessed April 12, 2009).

"Semiconductors Industry Profile: Global." *Semiconductors Industry Profile: Global* (December 2008): 1. *Business Source Premier*, EBSCO*host* (accessed April 12, 2009).

Shitara et al. “Step-density variations and reflection high-energy electron-diffraction intensity oscillations during epitaxial growth on vicinal GaAs.” *Phys. Rev.* **B 46**, 6815 - 6824 (1992).

Van Kampen, N. G. “Stochastic Processes in Physics and Chemistry.” North-Holland, Amsterdam (1992).

Appendix A Fundamental Equations

A.1 Master Equation

The probability that the surface is in a given configuration, α, is given by:

$$\frac{dP_\alpha}{dt} = \sum_\beta (W_{\alpha\beta} P_\beta - W_{\beta\alpha} P_\alpha)$$

where P_α is that probability and $W_{\alpha\beta}$ is the transition probability rate of the surface going from a configuration α to β .

A.2 Arrhenius Relationship

The hopping and desorption probabilities are modeled by:

$$k = k_o exp\left(-\frac{E}{k_B T}\right)$$

where k_o is the vibrational frequency of a surface atom, T is the substrate temperature, E is the energy barrier to hopping and k_B is Boltzmann's constant. And k_o at high temperatures can be found from:

$$k_o = \frac{2k_B T}{h}$$

where h is Planck's constant.

Appendix B Financial Details

B.1 Reactor Quote

Selections used marked in **bold.**

Provided by Jill S. Becker, Ph.D. | Founder | Cambridge NanoTech Inc.

System Prices

Savannah S100 – Standard 4 inch reactor	$118,000
Savannah S200 – Standard 8 inch reactor	$145,000
Savannah S300 – Standard 12 inch reactor	**$185,000**

Option Prices

ALD Reactor Chamber:

-	Savannah S100 Dome Lid with Cassette	$10,000
-	Savannah S200 Dome Lid with Cassette	$13,000
-	**Savannah S300 Dome Lid with Cassette**	**$18,000**
-	**MBraun Glovebox Interface**	**$1,850**
-	**Vapor Trap /w heating jacket**	**$8,500**
-	Vapor Trap w/o heating jacket	$7,500

Gas delivery:

-	**Additional Precursor Line Kit (4 additional lines max)**	**$8,500**
-	Low vapor pressure boost, each	$9,000
-	Liquid delivery system	$30,000
-	Ozone generator	$10,000
-	**Rapid exchange 50 cc precursor cylinder + manual valve**	**$3,000**

High vacuum reactor pumping:

-	Alcatel 2005I pump B-prepped (Fomblin)	$2,850
-	**Edwards XDS10 dry pump**	**$6,250**

Installation, Extended Warranty and Support:

-	**On-site 2-day Installation, Training and ALD Seminar**	**$7,500**
-	**Second year warranty and support (% of hardware cost)**	**6.5%**
-	**Second and third year warranty and support (% of hardware cost)**	**10%**

TOTAL COST $290,510.00

B.2 Clean Room Quote

Products & Services | Projects | Industry Online Resources | Careers

Cleanrooms Home

Cleanroom Cost Calculator

City:
Philadelphia, PA

Cleanroom Class:

- ( ) 10 (ISO 4)
- ( ) 100 (ISO 5)
- (•) 1000 (ISO 6)

Cleanroom Arrangement

- ( ) Ballroom
- (•) Bay (filtered) & Chase (unrated).

Percent of total area to be Chase (10 to 50%):
10 25

Cleanroom Area (1000 to 10,000 SF. For Bay & Chase enter total area): 1600
SF *(click on box)*

Air Return

- (•) Low Sidewall
- ( ) Raised Access Floor

Gown-Up Room

- (•) None
- ( ) Add to estimate

TOTAL COST: $539,300 Cost/SF: $337

Costs are in US dollars (2001).

Copyright © 2001, Industrial Design & Construction (IDC)

Search
Legal
Contact
IDC News

IDC, 2020 SW Fourth Avenue, Portland, Oregon, 97201, USA, info@www.idc.ch2m.com, tel: 503.224.6040
Copyright 1997-2002, CH2M HILL Industrial Design & Construction Inc. (IDC)

Cost increased to $400/SF for inflation.

B.3 Glove Box Quote

Cole-Parmer Catalog > Glove Boxes > Labconco Glove Boxes and Balance Enclosures > Labconco PRECISE Controlled Atmosphere Glove Boxes

Labconco® PRECISE™ Controlled Atmosphere Glove Boxes - Product Detail

(2 of 4) [Previous | Next]

Labconco® PRECISE™ Controlled Atmosphere Glove Boxes

click to enlarge

EW-34762-05
Controlled Atmosphere Glove Box, 230V

Qty: 2
Add to Cart

$8410.00 / each (USD) Available in 30 days.

Product Rating

(0 Ratings)

Write a Review

- **Contamination-free work environment at an affordable price**
- Leak-tight environment for for work with contamination-sensitive products
- Chemical-resistant work surface resisits spills

Specifications

Dimensions	52.7"W x 31.6"D x 40"H
Side door dimensions	11" Diameter x 12"L
Power	230V, 50Hz
Chamber size	33 1/2"W x 27 1/2"D x 25"H

B.4 Aggressive Growth Cash Flow Summary

Year	% of Market	Sales	Capital Costs	Fixed Costs	Cost of Sales	Depreciation Allowance
1	0%	-	(1,105,330.00)	(1,123,705.10)	-	-
2	1%	1,258,400.00	-	(1,157,416.25)	(251,680.00)	(221,066.00)
3	2%	2,592,304.00	-	(1,192,138.74)	(518,460.80)	(221,066.00)
4	4%	5,340,146.24	-	(1,227,902.90)	(1,068,029.25)	(221,066.00)
5	8%	11,000,701.25	-	(1,264,739.99)	(2,200,140.25)	(221,066.00)

Taxable Income	Income Tax Costs	Net Earnings	Annual Cash Flow	Interest Adjusted
	-	$ -	(2,229,035.10)	$ (2,229,035.10)
(371,762.25)	-	$ (371,762.25)	(150,696.25)	$ (125,580.21)
660,638.46	(429,415.00)	$ 231,223.46	452,289.46	$ 314,089.90
2,823,148.09	(1,835,046.26)	$ 988,101.83	1,209,167.83	$ 699,749.90
7,314,755.01	(4,754,590.76)	$ 2,560,164.25	2,781,230.25	$ 1,341,256.87
			NPV:	$ 481.37

Tax Rate = 35%
Interest Rate = 20%

B.5 Conservative Growth Cash Flow Summary

Year	% Market	Sales	Capital Costs	Fixed Costs	Cost of Sales	Depreciation Allowance
1	0	-	(1,105,330.00)	(1,123,705.10)	-	-
2	1%	1,699,544.00	-	(1,157,416.25)	(339,908.80)	(221,066.00)
3	1.00%	3,501,060.64	-	(1,192,138.74)	(700,212.13)	(221,066.00)
4	1.50%	5,409,138.69	-	(1,227,902.90)	(1,081,827.74)	(221,066.00)
5	2.00%	7,428,550.47	-	(1,264,739.99)	(1,485,710.09)	(221,066.00)

Taxable Income	Income Tax Costs	Net Earnings	Annual Cash Flow	Interest Adjusted
	-	$ -	(2,229,035.10)	$ (2,229,035.10)
(18,847.05)	-	$ (18,847.05)	202,218.95	$ 168,515.79
1,387,643.77	(901,968.45)	$ 485,675.32	706,741.32	$ 490,792.58
2,878,342.05	(1,870,922.33)	$ 1,007,419.72	1,228,485.72	$ 710,929.23
4,457,034.38	(2,897,072.35)	$ 1,559,962.03	1,781,028.03	$ 858,906.27
			NPV:	$ 108.77

Tax Rate = 35%
Interest Rate = 20%

Appendix C MATLAB Code

C.1 Single Component Code

Get Rate Hop

```
function [new_rate]=get_rate_hop(site,hopdir,time,nsitesrow,nsitescol,...
   nsites_max,height,sitecol,siterow,nnlist, T)
%...set rate of hopping from a particular site in a particular
%.....direction(hopdir)

%hmk
count=0;
for i=1:4
    if(height(nnlist(i,site))>= height(site))
        %if it has neighbors
       count=count+1;
    end
end



%diff surface energy barrier = 1.58 eV
%neighbor bond energy = .27 eV
%k=8.617343E-5 eV/K
rate_hop = 10^13/4;
%preexponential factor
new_rate=rate_hop*exp((-1.58+.5*(-count)*.27)/(8.617343*10^(-5)*T));
```

Get Rate Desorption

```
%#eml
function [new_rate]=get_rate_desorption(site,time,...
    nsitesrow,nsitescol,nsites_max,height,...
    sitecol,siterow,nnlist, T)
%...set rate of desorption at a particular site

%HMK
count=0;
for i=1:4
    if(height(nnlist(i,site))>= height(site))
        %if it has neighbors
        count=count+1;
    end
end

%desorp surface energy barrier = 2.32 eV
%neighbor bond energy = .27 eV
%k=8.617343E-5 eV/K
```

```
rate_desorption = 10^13;
%same as hopping pre-exponential
new_rate= rate_desorption*exp(-(2+count*.27)/(8.617343*10^(-5)*T));
```

Get Rate

```
%#eml
function [rate,wait_time,head,tail,forward]=get_rate(site,time,...
    levelorder,nsitesrow,nsitescol,nsites_max,max_level,...
    null_event,height,sitecol,siterow,nnlist,rate,wait_time,...
    head,tail,forward, T, rate_adsorption)
%...for a specific site, update all of the possible events at that site.
%.....site - site to update
for hopdir=1:4
    %...for all hopping directions, first get position for specific event
    rate_position=(site-1)*6+hopdir;
    %...if neighboring site in that direction is lower, get hop rate, else
    %.....set to 0
    if(height(nnlist(hopdir,site))<height(site)+1)
        if (height(site)-height(nnlist(hopdir,site))<6)
            [new_rate]=get_rate_hop(site,hopdir,time,...
                nsitesrow,nsitescol,nsites_max,height,...
                sitecol,siterow,nnlist, T);
        else
        [new_rate]=get_rate_desorption(site,time,...
            nsitesrow,nsitescol,nsites_max,height,...
            sitecol,siterow,nnlist, T);
        end

    else
        new_rate=0.0;
    end
    %...update queue with new rate
    [rate,wait_time,head,tail,forward]=update_list(...
        time,rate,wait_time,max_level,levelorder,head,tail,forward,...
        null_event,new_rate,rate_position);

end
%...do the same with adsorption events (can always occur)
hopdir=5;
rate_position=(site-1)*6+hopdir;
[new_rate]=get_rate_adsorption(site,time,...
    nsitesrow,nsitescol,nsites_max,height,...
    sitecol,siterow,nnlist, rate_adsorption);
[rate,wait_time,head,tail,forward]=update_list(...
        time,rate,wait_time,max_level,levelorder,head,tail,forward,...
        null_event,new_rate,rate_position);
%...do the same with desorption events (can always occur)
hopdir=6;
rate_position=(site-1)*6+hopdir;
[new_rate]=get_rate_desorption(site,time,...
    nsitesrow,nsitescol,nsites_max,height,...
```

```
    sitecol,siterow,nnlist, T);
[rate,wait_time,head,tail,forward]=update_list(...
        time,rate,wait_time,max_level,levelorder,head,tail,forward,...
        null_event,new_rate,rate_position);
```

Draw Surface

```
function [grid] = draw(nsitescol,nsites_max,height)

grid=zeros(nsitescol,nsitescol);

for i=1:nsites_max;
    x=mod(i,nsitescol);
    if(x==0);
        x=nsitescol;
    end
    y=(i-x)/nsitescol +1;
    grid(x,y)=height(i);
end

picture=surf(grid);
zlim([0,max(height)]);
shading interp;
colorbar;
```

C.2 Single Component Controller Code

Single Component Controller

```
function [Tmatrix, Rmatrix, Adsmatrix, Growthrate,finaltime]= ...
   SP_Controller(T,rate_adsorption)

%nsitesrow  - number of sites in each row
%nsitescol  - number of sites in each column (see initiate_colrow)
nsitesrow=100;
nsitescol=100;
nsites_max=nsitesrow*nsitescol;

%Starts with an input surface
height=zeros(nsitescol*nsitesrow,1);
[height] = initiate_height(nsites_max,height);
startheight = mean(height);

%Initiate matrices
q=0;
heighttemp(1) = startheight;
Rmatrix(1) = std(height);
Effmatrix(1) = 0;
Adsmatrix = [];
Growthrate(1) = rate_adsorption;

if (startheight >= 357)
    fprintf('%s %e \n','Height greater than desired height:', startheight);
end

%Find corresponding position on Optimal Profile
[time] = find_optimal(startheight);
t=time;

%Controller Parameters
error= .1;

while (t<1000)
    t=t+1;
    q=q+1;

    %check to see if controller action is required
    [sheight, srough, seff, sT, srate_ads] = get_optimal(t);
    [sum, fix] = calc_sum(Rmatrix(q), srough, heighttemp(q), sheight,
       Effmatrix(q), seff);
    if (sum < error)
        control = 0;
    else
        control = 1;
    end

    %temporary to save time, if on curve will complete curve
    if (control == 0 && T == sT && srate_ads == rate_adsorpton)
        %end program
```

```
        %set output variables
        break
    end

    if (control == 1)
        %roughness bigger factor
        [sheight, srough, seff, sT, srate_ads] = get_optimal(t+1);
        if (fix == 1)
            T= T+10;
            tempheight = [];
            %choose 100 different heights to input into the model
            for i=1:10;
                number = uint32(9999*rand())+1;
                %bounds if number outside range
                if number > nsites_max - 99;
                    number = nsites_max - 4*99;
                end
                for n=number:number+99;
                    tempheight(end+1) = height(n);
                end
                %start cycle
                [theight, troughness,tgrowthrate,tcountads,tcountdes]=...
                    main_kmc_code(1,1, tempheight, 10, 10, T, …
                    rate_adsorption);
                %caution, if number of moves < 10 will fail
                ttroughness(i) = troughness;
                ttheight(i) = theight;
            end
            troughness = mean(ttroughness);
            theight = mean(ttheight);
            [sum] = calc_sum(troughness, srough, theight, sheight, tcountdes,
               seff);
            Optimalcheck(1) = sum;

            %%%%%%Check Down%%%%%%
            T= T-20;
            for i=1:10;
                %restart
                [theight, troughness,tgrowthrate,tcountads,tcountdes]=...
                    main_kmc_code(1,1, tempheight, 10, 10, T, ...
                       rate_adsorption);
                %caution, if number of moves < 10 will fail
                ttroughness(i) = troughness;
                ttheight(i) = theight;
            end
            troughness = mean(ttroughness);
            theight = mean(ttheight);
            [sum] = calc_sum(troughness, srough, theight, sheight, tcountdes,
                  seff);
            Optimalcheck(2) = sum;
            if (Optimalcheck(1) < Optimalcheck (2))
                T=T+20;
            end
        end

        %growthrate bigger factor
```

```
        if (fix == 2)
            rate_adsorption = rate_adsorption + 1;
            tempheight = [];
            %choose 100 different heights to input into the model
            for i=1:10;
                number = uint32(9999*rand())+1;
                %bounds if number outside range
                if number > nsites_max - 99;
                    number = nsites_max - 4*99;
                end
                for n=number:number+99;
                    tempheight(end+1) = height(n);
                end
                %start cycle
                [theight, troughness,tgrowthrate,tcountads,tcountdes]=...
                    main_kmc_code(1,1, tempheight, 10, 10, T, ...
                       rate_adsorption);
                %caution, if number of moves < 10 will fail
                ttroughness(i) = troughness;
                ttheight(i) = theight;
            end
            troughness = mean(ttroughness);
            theight = mean(ttheight);
            [sum] = calc_sum(troughness, srough, theight, sheight, tcountdes,
                    seff);
            Optimalcheck(1) = sum;
            %%%%%%Check Down%%%%%%
            rate_adsorption = rate_adsorption -2;
            for i=1:10;
                %restart
                [theight, troughness,tgrowthrate,tcountads,tcountdes]=...
                    main_kmc_code(1,1, tempheight, 10, 10, T, ...
                       rate_adsorption);
                %caution, if number of moves < 10 will fail
                ttroughness(i) = troughness;
                ttheight(i) = theight;
            end
            troughness = mean(ttroughness);
            theight = mean(ttheight);
            [sum] = calc_sum(troughness, srough, theight, sheight, tcountdes,
                    seff);
            Optimalcheck(2) = sum;
            if (Optimalcheck(1) < Optimalcheck (2))
                rate_adsorption=rate_adsorption+2;
            end
        end
    end

    %input into 100x100 matrix
    [height, roughness,growthrate,countads,countdes]=...
        main_kmc_code(1,1, height, nsitesrow, nsitescol, T, rate_adsorption);

    if mean(height) >357
        finaltime = t;
        t=1001;
    end
```

```
    Tmatrix(q) = T;
    Effmatrix(q)=countdes;
    Rmatrix = [Rmatrix roughness];
    Adsmatrix = [Adsmatrix rate_adsorption];
    heighttemp(q+1) = mean(height);
    Growthrate(q) = heightemp(q+1)-heightemp(q);

end
draw(nsitescol,nsites_max,height)
save ('')
end
```

Get Optimal Profile

```
function [sheight, srough, seff, sT, srate_ads] = get_optimal(t);



optads =
[7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,9,9,9,9,9,9,9,9,9,9,9,
9,9,9,9,9,9,9,9,9];
optT =
[600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,
600,600,600,600,600,600,600,600,590,580,570,560,550,540,530,520,510,500,500,5
00,500,500,500,500,500,500,500,500];
optheight=
[0,6.95960000000000,13.9640000000000,20.9648000000000,27.9914000000000,34.986
5000000000,41.9982000000000,48.9740000000000,55.9486000000000,62.889000000000
0,69.8974000000000,76.8901000000000,83.9238000000000,90.9535000000000,97.9560
000000000,104.905500000000,111.917900000000,118.915200000000,125.926300000000
,132.957500000000,139.969400000000,146.952900000000,153.917700000000,160.8845
00000000,167.905600000000,174.870000000000,181.872000000000,190.853000000000,
199.846000000000,208.891000000000,217.889700000000,226.873900000000,235.85520
0000000,244.834600000000,253.846800000000,262.829200000000,271.854400000000,2
80.865100000000,289.871300000000,298.846800000000,307.885700000000,316.912400
000000,325.926500000000,334.912600000000,343.842600000000,352.893800000000,36
1.917200000000];
optrough=[0,2.62125947300000,3.70114748100000,4.50593897900000,5.212432357000
00,5.82339106900000,6.32894985600000,6.78306634500000,7.24039156200000,7.7238
0143200000,8.12862109200000,8.54382814400000,8.94060721100000,9.2798175510000
0,9.66132811800000,10.0270677300000,10.3238886500000,10.6544736900000,10.9649
608000000,11.2641999900000,11.5553970300000,11.8276863800000,12.0808815800000
,12.3611031700000,12.5958001200000,12.8853822100000,13.1533793500000,13.50019
27400000,13.8436856800000,14.1512184700000,14.4533822300000,14.7887317500000,
15.1063735700000,15.3738147200000,15.6510069400000,15.8957356000000,16.197268
8400000,16.5053579800000,16.7504087000000,16.9796451900000,17.1999052600000,1
7.4165237200000,17.6756137000000,17.9413265500000,18.2558671000000,18.5362982
900000,18.8155883700000];
opteff=[0,0.326086957000000,0.326086957000000,0.326086957000000,0.32608695700
0000,0.326086957000000,0.326086957000000,0.326086957000000,0.326086957000000,
```

```
0.326086957000000,0.326086957000000,0.326086957000000,0.326086957000000,0.326
086957000000,0.326086957000000,0.326086957000000,0.326086957000000,0.32608695
7000000,0.326086957000000,0.326086957000000,0.326086957000000,0.3260869570000
00,0.326086957000000,0.326086957000000,0.326086957000000,0.326086957000000,0.
326086957000000,0.326086957000000,0.326086957000000,0.326086957000000,0.32608
6957000000,0.326086957000000,0.326086957000000,0.326086957000000,0.3260869570
00000,0.326086957000000,0.326086957000000,0.326086957000000,0.326086957000000
,0.326086957000000,0.326086957000000,0.326086957000000,0.326086957000000,0.32
6086957000000,0.326086957000000,0.326086957000000,0.326086957000000];

seff=opteff(t);
srough=optrough(t);
sheight=optheight(t);
sT=optT(t);
srate_ads=optads(t);
```

C.3 Multiple Component Code

Main KMC

```
%#eml
function
[particle_matrix,height_matrix,Roughness,Growthrate,Grain,Height,finaltime]=m
ain_kmc_code(timef,timesteps,T,rate_adsorption,fraction_A)
%*******************MAIN KMC CODE****************************************
%surface deposition with adsorption, desorption and hopping
%2D lattice with height function
%
%timef - final system time
%timesteps - # of time steps for output
%nsitesrow - dimension of lattice
%nsitescol - dimension of lattice
%
%
%The following are simple ways to set dyanmics (will need to be changed
%     according to the system of interest)
%
%   rate_hop        - rate of hopping
%   rate_adsorption - rate of adsorption
%   rate_desorption - rate of desorption
%************************************************************************
%
%...first initiate main system variables
%nsitesrow          - number of sites in each row
%nsitescol          - number of sites in each column (see initiate_colrow)
%max_level_particle - maximum number of atoms for storage at each site
%max_level          - number of levels for skiplist
%levelorder         - probability of gaining a level for skiplist
%***the skiplist variables should not need to be changed unless
%   max_events>10^8***
nsitesrow=25;
nsitescol=25;
max_level_particle=50;
max_level=8;
levelorder=0.1;
y=1;
%
eml.extrinsic('fprintf');%used only for compiled matlab output (probably not
useful though)
%
%generates different seed each run time
rand('twister',sum(100*clock))
countads=0;
countdes=0;

%nsites_max - total number of sites
%max_events - currently there are 6 possible events (4 hopping directions,
%                adsorption and desorption)
%null_event - needed for skiplist
nsites_max=nsitesrow*nsitescol;
```

```
max_events=nsites_max*6;
null_event=max_events+1;
particle_matrix = zeros(timesteps,nsites_max);
height_matrix = zeros(timesteps,nsites_max);
%
%height - height of surface above site #
height=zeros(nsitescol*nsitesrow,1);
%
%particle_list - stores atom_types for each site up to max_level_particle
%                    Initialized to all type 1 atoms.
particle_list=ones(nsitescol*nsitesrow,max_level_particle);
%
%sitecol - number of sites up the column
sitecol=zeros(nsites_max,1);
%
%siterow - number of sites down the row (see initiate_colrow)
siterow=zeros(nsites_max,1);
%
%nnlist - stores the nearest neighbor sites (see initiate_nnlist)
nnlist=zeros(4,nsites_max);
%
%rate - stores the rate for all possible events
rate=zeros(null_event,1);
%
%wait_time - stores the waiting_time for all possible events (also the key
%                for the skiplist
wait_time=zeros(null_event,1);
%
%initialize time
time=0.0;
%
%head - stores position of the min waiting time
%tail - stores position of the max waiting time (not used here, but needed
for
%             skiplist functions)
head=null_event;
tail=null_event;
%
%...main skiplist array, all pointers start with null_event
forward=ones(max_level,null_event)*null_event;
%
%...count keeps track of number of steps
%
count=0;
%
%...timebin partitions the time for outputting with timesteps
%...counter_time_bin counts number of outputs
%...write_flag is used to break loop for outputting
%
timebin=timef/timesteps;
counter_time_step=1;
write_flag=true;
%
%...print variables to command window
%
fprintf('%s %e \n','    Temperature: ', T);
fprintf('%s %e \n','Adsorption Rate: ', rate_adsorption);
```

```
%fprintf('%s %e \n','Desorption Rate: ', rate_desorption);
%
%...initiate column/row numbering (not strictly needed see function)
%
[sitecol,siterow]=initiate_colrow(...
    nsitescol,nsites_max,...
    siterow,sitecol);
%
%...initiate nnlist (not strictly needed see function)
%
[nnlist]=initiate_nnlist(...
    nsitesrow,nsitescol,nsites_max,...
    siterow,sitecol,nnlist);
%
%...set initial height profile
%
[height]=initiate_height(...
    nsites_max,...
    height);
height_in=height;
%
%Initiate Outputs
Height(1) = mean(height_in);
Growthrate(1) = rate_adsorption;
[grain_size] = get_grainsize(25,25, 625, particle_list, 2);
Grain(y) = grain_size;
%
%
%...initiate rates of all possible events
%
[rate,wait_time,head,tail,forward]=initiate_rate(...
    time,rate_adsorption,levelorder,...
    nsitesrow,nsitescol,nsites_max,max_level,null_event,...
    siterow,sitecol,nnlist,...
    height,particle_list,rate,wait_time,...
    head,tail,forward, T);
%
%...MAIN LOOP (until final system time is reached)
%
while(time<=timef)
    %
    save ('');
    fprintf('\n %s %10f %s %e %s %10f',...
       'moves:',count,' time:',time,' #:',counter_time_step-1)
    %
    while(write_flag)
        %
        count=count+1;
        %...Execute event with minimum waiting_time

[time,rate,wait_time,head,tail,forward,height,particle_list,countads,countdes
     ]=event(...
            rate_adsorption,fraction_A,levelorder,...
            nsitesrow,nsitescol,nsites_max,max_level,null_event,...
            siterow,sitecol,nnlist,...
            height,particle_list,rate,wait_time,...
            head,tail,forward, T, countads,countdes);
```

```
        %
        if(time>=timebin*counter_time_step)
            write_flag=false;
        end
        %

    end
    %
    counter_time_step=counter_time_step+1;
    write_flag=true;
    %
    if mean(height) >357
        finaltime = time;
        time=timef +1;
    end

    %HMK
    %Code for outputs
    y=y+1;
    Roughness(y) = std(height(1:end-1));
    Height(y) = mean(height);
    [grain_size] = get_grainsize(25,25, 625, particle_list, 2);
    Grain(y) = grain_size;
    Growthrate(y) = Height(y)-Height(y-1);
    for i=1:nsites_max
        particle_matrix(y,i) = particle_list(i);
    end
    for i=1:nsites_max
        height_matrix(y,i) = height(i);
    end
end
%
fprintf('\n %s %10f %s %e %s %10f \n',...
       'moves:',count,' time:',time,' #:',counter_time_step-1)
save ('');
draw(nsitescol,nsites_max,particle_list,height);
```

Get Rate Desorption

```
%#eml
function [new_rate]=get_rate_desorption(site,time,...
    nsitesrow,nsitescol,nsites_max,height,particle_list,...
    sitecol,siterow,nnlist,T)
%...set rate of desorption at a particular site

%HMK
count=0;
for i=1:4
    if(height(nnlist(i,site))>= height(site))
```

```
        %if it has neighbors
        count=count+1;
    end
end

%desorp surface energy barrier = 2.0 eV
%neighbor bond energy = .2eV
%k=8.617343E-5 eV/K
rate_desorption = 1;
new_rate= rate_desorption*exp(-(2.0+count*.2)/(8.617343*10^(-5)*T));
```

Get Rate Hop

```
%#eml
function [new_rate]=get_rate_hop(site,hopdir,time,...
    nsitesrow,nsitescol,nsites_max,height,particle_list,sitecol...
    ,siterow,nnlist,T)
%...set rate of hopping from a particular site in a particular
%.....direction(hopdir)

%HMK
%AB interactions less strong than AA or BB interactions
%Only will look in following directions:
%       2
%  2,3  2  2,4
%       .
%3  3...X...4  4
%       .
%  1,3  1  1,4
%       1

%secondary particles have a weighted bond attraction that is = 1- sqrt(2)/2
number = 1-sqrt(2)/2;



sitetype = particle_list(site); %type of particle at the site
siteheight = height(site); %height of site
countself=0;
countother=0;
countself2=0;
countother2=0;

%current site
for i=1:4
    %primary neighbors
    nsitetype = particle_list(nnlist(i,site)); %type of neighbor particle
    nheight = height(nnlist(i,site)); %height of neighbor
    if(nsitetype == sitetype && nheight >= siteheight)
        countself = countself+1;
```

```
    else if (nsitetype ~= sitetype && nheight >= siteheight)
            countother= countother+1;
        end
    end
    %2ndary neighbors
    nsitetype2 = particle_list(nnlist(i,nnlist(i,site))); %type of 2ndary
particle
    nheight2 = height(nnlist(i,nnlist(i,site))); %height of 2ndary particle
    if(nsitetype2 == sitetype && nheight2 >= siteheight)
        countself2 = countself2 +number;
    else if (nsitetype2 ~= sitetype && nheight2 >= siteheight)
            countother2 = countother2 +number;
        end
    end
    if (i==1 || i==2)
        nsitetype23 = particle_list(nnlist(3,nnlist(i,site)));
            %type of 2ndary particle
        nsitetype24 = particle_list(nnlist(4,nnlist(i,site)));
            %type of 2ndary particle
        nheight23 = height(nnlist(3,nnlist(i,site)));
            %height of 2ndary particle
        nheight24 = height(nnlist(4,nnlist(i,site)));
            %height of 2ndary particle
        if (nsitetype23 == sitetype && nheight23 >= siteheight)
            countself = countself+number;
        else if (nsitetype23 ~= sitetype && nheight23 >= siteheight)
                countother2 = countother2 +number;
            end
        end
        if (nsitetype24 == sitetype && nheight24 >= siteheight)
            countself = countself+number;
        else if (nsitetype24 ~= sitetype && nheight24 >= siteheight)
                countother2 = countother2 +number;
            end
        end
    end
end

count = countself + .333*countother;

fcountself=-1;
fcountother=0;
fcountself2=0;
fcountother2=0;
%-1 because will count itself as a neighbor of the site it will move to
% if it will have primary neighbors

%future neighbors
for i=1:4
    %if it will have primary neighbors
    fnsitetype = particle_list(nnlist(i,nnlist(hopdir,site))); %future
neighbor type
    fnheight = height(nnlist(i,nnlist(hopdir,site))); %height of future
neighbor
    if(fnsitetype==sitetype && fnheight>= siteheight)
        fcountself = fcountself+1;
```

```
    else if (fnsitetype ~= sitetype && fnheight>= siteheight)
            fcountother= fcountother+1;
        end
    end
    %if it will have secondary neighbors
    fnsitetype2 = particle_list(nnlist(i,nnlist(i,nnlist(hopdir,site))));
            % future type of 2ndary particle
    fnheight2 = height(nnlist(i,nnlist(i,nnlist(hopdir,site))));
            %future 2ndary particle height
    if (fnsitetype2 == sitetype && fnheight2 >= siteheight)
        fcountself2 = fcountself2 +number;
    else if (fnsitetype2 ~= sitetype && fnheight2 >= siteheight)
            fcountother2 = fcountother2 +number;
        end
    end
    if (i==1 || i==2)
        fnsitetype23 =
particle_list(nnlist(3,nnlist(i,nnlist(hopdir,site)))); %type of 2ndary
particle
        fnsitetype24 =
particle_list(nnlist(4,nnlist(i,nnlist(hopdir,site)))); %type of 2ndary
particle
        fnheight23 = height(nnlist(3,nnlist(i,nnlist(hopdir,site))));
            %height of 2ndary particle
        fnheight24 = height(nnlist(4,nnlist(i,nnlist(hopdir,site))));
            %height of 2ndary particle
        if (nsitetype23 == sitetype && nheight23 >= siteheight)
            fcountself = fcountself+number;
        else if (nsitetype23 ~= sitetype && nheight23 >= siteheight)
                fcountother2 = fcountother2 +number;
            end
        end
        if (fnsitetype24 == sitetype && fnheight24 >= siteheight)
            fcountself = fcountself+number;
        else if (fnsitetype24 ~= sitetype && fnheight24 >= siteheight)
                fcountother2 = fcountother2 +number;
            end
        end
    end
end

fcount=fcountself + .333*fcountother;

%diff surface energy barrier = 1.58 eV
%neighbor bond energy = .2eV
%k=8.617343E-5 eV/K
rate_hop = 10^13;
%preexponential factor

new_rate=rate_hop/4*exp((-1.58+.5*(fcount-count)*.2)/(8.617343*10^(-5)*T));
```

Get Order parameter

```
function [grain_size, particle_type] = get_grainsize(nsitescol, nsitesrow,
nsites_max, particle_list, particle_type);
%PB
%This function inputs the dimensions of the lattice, the number of lattice
%points, the current height vector, and the size to measure roughness (i.e.
%rough_size = 2 means take 2X2 subsets of the lattice, average the height
%in the mini-lattices and take the variance of those.
%mean_type = zeros(1,nsites_max);

particle_list = particle_list(1:nsites_max+1);
particle_list = particle_list';

for j = 1:25;
    %list_type = zeros(1,j*j);
    count = 0;
    %num = j;
    for i=1:nsites_max;
        %Establish counters,
        i_new = i;    %used to checkperiodicity
        k = 0;
        %num = 0;
        list_type_counter = 1;
        truefalse = 1;
        i_new = mod(i_new,nsitesrow);   %converting the position in actual
height matrix to position in column 1, same row

        if (i_new == 0)
            i_new = nsitesrow;
        end

        if (i_new + j - 1 > nsitesrow)  %if when you go down the column you
will fall off, come back to top of col
            numbeforewrap = nsitesrow - i_new+1 ;  %number of sites that work
before wrapping
        else
            numbeforewrap = j;         %if you don't wrap around
        end

        %Get starting top row
        for k=0:j-1
            startingpoint = i +nsitesrow*k;
            %start at k = 0 for column where i is located, move over a column
            as many times as size requires
            if (startingpoint>nsites_max)
                startingpoint = startingpoint-nsites_max;
            %column periodicity- go back to beginning if it goes over end
            end
            for (h=0:(j-1))
                if (h>=numbeforewrap)
                  %if you're sampling a site that has wrapped around
                    wrap = nsitesrow;
                else
                    wrap = 0;
```

```
                end

            list_type(list_type_counter) = particle_list(startingpoint+h-
            wrap);   %get the height at each location
            if (list_type(list_type_counter) ~= particle_type)
                truefalse = 0;
            end
            list_type_counter = list_type_counter+1;
            end
        end
        list_type;
        diff = mean(list_type) - particle_type;
        if (truefalse == 1)
            count = count+1;

        end
    end
    histogram(j) = count;
    %length(list_type)
    if (histogram(j) ==0)
        histogram = histogram(1:length(histogram)-1);
        break
    end
end
%histogram
%length(histogram)
totgrains = sum(histogram);
for (i = 1:length(histogram))
   histogram(i) = histogram(i)*i*i;
        %if(histogram(i) <0)
        %    histogram(i) = 0;
        %end

end
grain_size = sum(histogram)/totgrains;      %calculate grainness by variance
in the set
```

Draw Surface/Particle Type Distribution

```
function [height_grid,particle_list_grid,color_grid] =
draw_new(nsitescol,nsites_max,particle_list,height);

%Converts height and particle list vectors into matrices
for i=1:nsites_max;
    x=mod(i,nsitescol);
    if(x==0);
        x=nsitescol;
    end
    y=(i-x)/nsitescol +1;
```

```
    height_grid(x,y)=height(i);
    particle_list_grid(x,y)=particle_list(i);
end

%Makes a color grid with red if particle == 1, blue otherwise
for i=1:nsitescol;
    for j=1:nsitescol;
        if (particle_list_grid(i,j) == 1)
            color_grid(i,j,1)=1;
            color_grid(i,j,2)=0;
            color_grid(i,j,3)=0;
        else
            color_grid(i,j,1)=0;
            color_grid(i,j,2)=0;
            color_grid(i,j,3)=1;
        end
    end
end

%Makes contour picture
picture=surf(height_grid);
zlim([0,max(height)]);
colorbar;
shading interp;
saveas(picture,'contour.fig');
saveas(picture, 'contour.jpg');
close all;
%Makes particle type picture
part_pic = surf(height_grid,color_grid);
zlim([0,max(height)]);
%shading interp;
saveas(part_pic,'part_type.fig');
saveas(part_pic, 'part_type.jpg');
%close all;
```

C.4 Multiple Component Controller Code

Main Controller Code

```
function
[Temperature,Adsorption,Roughness,Growthrate,Grain,Error,Hmatrix,Pmatrix,fina
ltime]=MP_Controller(fraction_A)

%nsitesrow  - number of sites in each row
%nsitescol  - number of sites in each column (see initiate_colrow)
nsitesrow=25;
nsitescol=25;
nsites_max=nsitesrow*nsitescol;
max_level_particle=50;

%particle_list - stores atom_types for each site up to max_level_particle
%Initialized to all type 1 atoms.
particle_list=ones(nsitescol*nsitesrow,max_level_particle);

%Starts with an input surface
height=zeros(nsitescol*nsitesrow,1);
[height] = initiate_height(nsites_max,height);

%Initiate matrices
q=1;
if (fraction_A <= .5)
    parttype=2;
else
    parttype=1;
end
Height(1) = mean(height);
Roughness = std(height);
[grain_start] = get_grainsize(nsitesrow, nsitescol, nsites_max,
particle_list, parttype);
Grain = grain_start;
Error = [];
Hmatrix = height';
for i=1:nsites_max
       Pmatrix(1,i) = particle_list(i);
end

if (Height(1) >= 357)
    fprintf('%s %e \n','Height greater than desired height:', Height(1));
end

%Controller Parameters
t=0;
error= .05;
[height_grid,particle_grid,grid_size] =
get_gridMP(nsitescol,nsitesrow,nsites_max,height,particle_list);
[T, rate_adsorption] = get_IC(height_grid,
particle_grid,grid_size,fraction_A,parttype);
```

```
%Initiate Matrices
Temperature = T;
Adsorption = rate_adsorption;
Growthrate = rate_adsorption;

while (t<1000)
    t=t+1;
    %check to see if controller action is required
    [sheight, srough, sgrain, sT, srate_ads] = get_optimal(t,fraction_A);
    [sum] = calc_sum(Roughness(q), srough, Height(q), sheight, Grain(q),
            sgrain, fraction_A);
    Error(t) = sum;
    if (sum < error)
        control = 0;
    else
        control = 1;
    end

    %temporary to save time, if on curve will complete curve
    if (control == 0 && T == sT && srate_ads == rate_adsorpton)
        %end program
        %set output variables
        break
    elseif (control ==0)
        if (T> sT)
            T = T-10;
        elseif (T < sT)
            T = T+10;
        end
        if (rate_adsorption > srate_ads)
            rate_adsorption = rate_adsorption-1;
        elseif (rate_adsorption < srate_ads)
            rate_adsorption = rate_adsorption+1;
        end
    end
    %Minimums
    if (T<310)
        T=310;
    end
    if (rate_adsorption <2)
        rate_adsorption = 2;
    end
    %
    if (control == 1)
        [sheight, srough, sgrain] = get_optimal(t+1,fraction_A);
        [height_grid,particle_grid,grid_size] =
            get_gridMP(nsitescol,nsitesrow,nsites_max,height,particle_list);

        %TEST 1
        %choose 100 different heights to input into the model
        for i=1:10;
            %start cycle
            [tparticle_list, theight, troughness]=...
                main_kmc_code(1,1, height_grid(i,1:end), 10, 10, T,
                  rate_adsorption,fraction_A,particle_grid(i*grid_size*grid_s
            ize-grid_size*grid_size+1:i*grid_size*grid_size,1:end));
```

```
    ttroughness(i) = troughness;
    ttheight(i) = mean(theight);
end
troughness = mean(ttroughness);
theight = mean(ttheight);
[tgrain_size] = get_grainsize(10,10, 100, tparticle_list, parttype);
[sum] = calc_sum(troughness, srough, theight, sheight, tgrain_size,
    sgrain,fraction_A);
Optimalcheck(1) = sum;

%TEST 2
T= T+10;
%choose 100 different heights to input into the model
for i=1:10;
    [tparticle_list, theight, troughness]=...
        main_kmc_code(1,1, height_grid(i,1:end), 10, 10, T,
          rate_adsorption,fraction_A,particle_grid(i*grid_size*grid_s
    ize-grid_size*grid_size+1:i*grid_size*grid_size,1:end));
    ttroughness(i) = troughness;
    ttheight(i) = mean(theight);
end
troughness = mean(ttroughness);
theight = mean(ttheight);
[tgrain_size] = get_grainsize(10,10, 100, tparticle_list, parttype);
[sum] = calc_sum(troughness, srough, theight, sheight, tgrain_size,
    sgrain,fraction_A);
Optimalcheck(2) = sum;

%TEST 3
T= T-20;
for i=1:10;
    [tparticle_list, theight, troughness]=...
        main_kmc_code(1,1, height_grid(i,1:end), 10, 10, T,
          rate_adsorption,fraction_A,particle_grid(i*grid_size*grid_s
    ize-grid_size*grid_size+1:i*grid_size*grid_size,1:end));
    ttroughness(i) = troughness;
    ttheight(i) = mean(theight);
end
troughness = mean(ttroughness);
theight = mean(ttheight);
[tgrain_size] = get_grainsize(10,10, 100, tparticle_list, parttype);
[sum] = calc_sum(troughness, srough, theight, sheight, tgrain_size,
    sgrain,fraction_A);
Optimalcheck(3) = sum;

%TEST 4
rate_adsorption = rate_adsorption + 1;
for i=1:10;
    [tparticle_list, theight, troughness]=...
        main_kmc_code(1,1, height_grid(i,1:end), 10, 10, T,
          rate_adsorption,fraction_A,particle_grid(i*grid_size*grid_s
    ize-grid_size*grid_size+1:i*grid_size*grid_size,1:end));
    ttroughness(i) = troughness;
    ttheight(i) = mean(theight);
end
troughness = mean(ttroughness);
```

```
theight = mean(ttheight);
[tgrain_size] = get_grainsize(10,10, 100, tparticle_list, parttype);
[sum] = calc_sum(troughness, srough, theight, sheight, tgrain_size,
    sgrain,fraction_A);
Optimalcheck(4) = sum;

%TEST 5
rate_adsorption = rate_adsorption - 2;
for i=1:10;
    [tparticle_list, theight, troughness]=...
        main_kmc_code(1,1, height_grid(i,1:end), 10, 10, T,
          rate_adsorption,fraction_A,particle_grid(i*grid_size*grid_s
    ize-grid_size*grid_size+1:i*grid_size*grid_size,1:end));
    ttroughness(i) = troughness;
    ttheight(i) = mean(theight);
end
troughness = mean(ttroughness);
theight = mean(ttheight);
[tgrain_size] = get_grainsize(10,10, 100, tparticle_list, parttype);
[sum] = calc_sum(troughness, srough, theight, sheight, tgrain_size,
    sgrain,fraction_A);
Optimalcheck(5) = sum;

%TEST 6
T = T + 10;
for i=1:10;
    [tparticle_list, theight, troughness]=...
        main_kmc_code(1,1, height_grid(i,1:end), 10, 10, T,
          rate_adsorption,fraction_A,particle_grid(i*grid_size*grid_s
    ize-grid_size*grid_size+1:i*grid_size*grid_size,1:end));
    ttroughness(i) = troughness;
    ttheight(i) = mean(theight);
end
troughness = mean(ttroughness);
theight = mean(ttheight);
[tgrain_size] = get_grainsize(10,10, 100, tparticle_list, parttype);
[sum] = calc_sum(troughness, srough, theight, sheight, tgrain_size,
    sgrain,fraction_A);
Optimalcheck(6) = sum;

%TEST 7
T = T + 10;
for i=1:10;
    [tparticle_list, theight, troughness]=...
        main_kmc_code(1,1, height_grid(i,1:end), 10, 10, T,
          rate_adsorption,fraction_A,particle_grid(i*grid_size*grid_s
    ize-grid_size*grid_size+1:i*grid_size*grid_size,1:end));
    ttroughness(i) = troughness;
    ttheight(i) = mean(theight);
end
troughness = mean(ttroughness);
theight = mean(ttheight);
[tgrain_size] = get_grainsize(10,10, 100, tparticle_list, parttype);
[sum] = calc_sum(troughness, srough, theight, sheight, tgrain_size,
    sgrain,fraction_A);
Optimalcheck(7) = sum;
```

```
    %TEST 8
    rate_adsorption = rate_adsorption + 2;
    for i=1:10;
        [tparticle_list, theight, troughness]=...
            main_kmc_code(1,1, height_grid(i,1:end), 10, 10, T,
              rate_adsorption,fraction_A,particle_grid(i*grid_size*grid_s
        ize-grid_size*grid_size+1:i*grid_size*grid_size,1:end));
        ttroughness(i) = troughness;
        ttheight(i) = mean(theight);
    end
    troughness = mean(ttroughness);
    theight = mean(ttheight);
    [tgrain_size] = get_grainsize(10,10, 100, tparticle_list, parttype);
    [sum] = calc_sum(troughness, srough, theight, sheight, tgrain_size,
        sgrain,fraction_A);
    Optimalcheck(8) = sum;



    %TEST 9
    T = T - 10;
    for i=1:10;
        [tparticle_list, theight, troughness]=...
            main_kmc_code(1,1, height_grid(i,1:end), 10, 10, T,
              rate_adsorption,fraction_A,particle_grid(i*grid_size*grid_s
        ize-grid_size*grid_size+1:i*grid_size*grid_size,1:end));
        ttroughness(i) = troughness;
        ttheight(i) = mean(theight);
    end
    troughness = mean(ttroughness);
    theight = mean(ttheight);
    [tgrain_size] = get_grainsize(10,10, 100, tparticle_list, parttype);
    [sum] = calc_sum(troughness, srough, theight, sheight, tgrain_size,
        sgrain,fraction_A);
    Optimalcheck(9) = sum;

    index = find(Optimalcheck == min(Optimalcheck));
    if (index < 4)
        rate_adsorption = rate_adsorption  - 1;
    end
    if (index == 5 || index == 6 || index == 7)
        rate_adsorption = rate_adsorption  - 2;
    end
    if (index == 3 || index == 4 || index == 5)
       T = T - 10;
    end
    if (index ==2 || index == 7 || index == 8)
        T = T + 10;
    end

end

%input into 25x25 matrix
[particle_list, height, roughness]=...
```

```
        main_kmc_code(1,1, height, nsitesrow, nsitescol, T,
            rate_adsorption,fraction_A,particle_list);

    if mean(height) >357
        finaltime = t;
        t=1001;
    end

    q=q+1;
    for i=1:nsites_max
        Hmatrix(q+1,i) = height(i);
    end
    for i=1:nsites_max
       Pmatrix(q,i) = particle_list(i);
    end
    [grain_size] = get_grainsize(nsitesrow, nsitescol, nsites_max,
            particle_list, parttype);
    Temperature(q) = T;
    Grain(q)=grain_size;
    Roughness(q) = roughness;
    Adsorption(q) = rate_adsorption;
    Height(q) = mean(height);
    Growthrate(q) = Height(q)-Height(q-1);
    save ('')
end

draw(nsitescol,nsites_max,particle_list,height);
save ('')
end
```

Get Initial Conditions

```
function [T, rate_adsorption] =
get_IC(height_grid,particle_grid,grid_size,fraction_A,parttype)

[sheight, srough, sgrain, sT, srate_ads] = get_optimal(1,fraction_A);

%TEST 1
%choose 100 different heights to input into the model
T = 500;
rate_adsorption = 3;
for i=1:10;
    [tparticle_list, theight, troughness]=...
        main_kmc_code(1,1, height_grid(i,1:end), 10, 10, T,
rate_adsorption,fraction_A,particle_grid(i*grid_size*grid_size-
grid_size*grid_size+1:i*grid_size*grid_size,1:end));
    ttroughness(i) = troughness;
    ttheight(i) = mean(theight);
end
troughness = mean(ttroughness);
```

```
theight = mean(ttheight);
[tgrain_size] = get_grainsize(10,10, 100, tparticle_list, parttype);
[sum] = calc_sum(troughness, srough, theight, sheight, tgrain_size,
sgrain,fraction_A);
Optimalcheck(1) = sum;

%TEST 2
T = 500;
rate_adsorption = 5;
for i=1:10;
    [tparticle_list, theight, troughness]=...
        main_kmc_code(1,1, height_grid(i,1:end), 10, 10, T,
rate_adsorption,fraction_A,particle_grid(i*grid_size*grid_size-
grid_size*grid_size+1:i*grid_size*grid_size,1:end));
    ttroughness(i) = troughness;
    ttheight(i) = mean(theight);
end
troughness = mean(ttroughness);
theight = mean(ttheight);
[tgrain_size] = get_grainsize(10,10, 100, tparticle_list, parttype);
[sum] = calc_sum(troughness, srough, theight, sheight, tgrain_size,
sgrain,fraction_A);
Optimalcheck(2) = sum;

%TEST 3
T = 500;
rate_adsorption = 7;
for i=1:10;
    [tparticle_list, theight, troughness]=...
        main_kmc_code(1,1, height_grid(i,1:end), 10, 10, T,
rate_adsorption,fraction_A,particle_grid(i*grid_size*grid_size-
grid_size*grid_size+1:i*grid_size*grid_size,1:end));
    ttroughness(i) = troughness;
    ttheight(i) = mean(theight);
end
troughness = mean(ttroughness);
theight = mean(ttheight);
[tgrain_size] = get_grainsize(10,10, 100, tparticle_list, parttype);
[sum] = calc_sum(troughness, srough, theight, sheight, tgrain_size,
sgrain,fraction_A);
Optimalcheck(3) = sum;

%TEST 4
T = 600;
rate_adsorption = 3;
for i=1:10;
    [tparticle_list, theight, troughness]=...
        main_kmc_code(1,1, height_grid(i,1:end), 10, 10, T,
rate_adsorption,fraction_A,particle_grid(i*grid_size*grid_size-
grid_size*grid_size+1:i*grid_size*grid_size,1:end));
    ttroughness(i) = troughness;
    ttheight(i) = mean(theight);
end
troughness = mean(ttroughness);
theight = mean(ttheight);
[tgrain_size] = get_grainsize(10,10, 100, tparticle_list, parttype);
```

```
[sum] = calc_sum(troughness, srough, theight, sheight, tgrain_size,
sgrain,fraction_A);
Optimalcheck(4) = sum;

%TEST 5
T = 600;
rate_adsorption = 5;
for i=1:10;
    [tparticle_list, theight, troughness]=...
        main_kmc_code(1,1, height_grid(i,1:end), 10, 10, T,
rate_adsorption,fraction_A,particle_grid(i*grid_size*grid_size-
grid_size*grid_size+1:i*grid_size*grid_size,1:end));
    ttroughness(i) = troughness;
    ttheight(i) = mean(theight);
end
troughness = mean(ttroughness);
theight = mean(ttheight);
[tgrain_size] = get_grainsize(10,10, 100, tparticle_list, parttype);
[sum] = calc_sum(troughness, srough, theight, sheight, tgrain_size,
sgrain,fraction_A);
Optimalcheck(5) = sum;

%TEST 6
T = 600;
rate_adsorption = 7;
for i=1:10;
    [tparticle_list, theight, troughness]=...
        main_kmc_code(1,1, height_grid(i,1:end), 10, 10, T,
rate_adsorption,fraction_A,particle_grid(i*grid_size*grid_size-
grid_size*grid_size+1:i*grid_size*grid_size,1:end));
    ttroughness(i) = troughness;
    ttheight(i) = mean(theight);
end
troughness = mean(ttroughness);
theight = mean(ttheight);
[tgrain_size] = get_grainsize(10,10, 100, tparticle_list, parttype);
[sum] = calc_sum(troughness, srough, theight, sheight, tgrain_size,
sgrain,fraction_A);
Optimalcheck(6) = sum;

%TEST 7
T = 700;
rate_adsorption = 3;
for i=1:10;
    [particle_list, theight, troughness]=...
        main_kmc_code(1,1, height_grid(i,1:end), 10, 10, T,
rate_adsorption,fraction_A,particle_grid(i*grid_size*grid_size-
grid_size*grid_size+1:i*grid_size*grid_size,1:end));
    ttroughness(i) = troughness;
    ttheight(i) = mean(theight);
end
troughness = mean(ttroughness);
theight = mean(ttheight);
[tgrain_size] = get_grainsize(10,10, 100, tparticle_list, parttype);
[sum] = calc_sum(troughness, srough, theight, sheight, tgrain_size,
sgrain,fraction_A);
```

```
Optimalcheck(7) = sum;

%TEST 8
T = 700;
rate_adsorption = 5;
for i=1:10;
    [tparticle_list, theight, troughness]=...
        main_kmc_code(1,1, height_grid(i,1:end), 10, 10, T,
rate_adsorption,fraction_A,particle_grid(i*grid_size*grid_size-
grid_size*grid_size+1:i*grid_size*grid_size,1:end));
    ttroughness(i) = troughness;
    ttheight(i) = mean(theight);
end
troughness = mean(ttroughness);
theight = mean(ttheight);
[tgrain_size] = get_grainsize(10,10, 100, tparticle_list, parttype);
[sum] = calc_sum(troughness, srough, theight, sheight, tgrain_size,
sgrain,fraction_A);
Optimalcheck(8) = sum;

%TEST 9
T = 700;
rate_adsorption = 7;
for i=1:10;
    [tparticle_list, theight, troughness]=...
        main_kmc_code(1,1, height_grid(i,1:end), 10, 10, T,
rate_adsorption,fraction_A,particle_grid(i*grid_size*grid_size-
grid_size*grid_size+1:i*grid_size*grid_size,1:end));
    ttroughness(i) = troughness;
    ttheight(i) = mean(theight);
end
troughness = mean(ttroughness);
theight = mean(ttheight);
[tgrain_size] = get_grainsize(10,10, 100, tparticle_list, parttype);
[sum] = calc_sum(troughness, srough, theight, sheight, tgrain_size,
sgrain,fraction_A);
Optimalcheck(9) = sum;


index = find(Optimalcheck == min(Optimalcheck));

if (index < 4)
    T = 500;
elseif (index <7)
    T = 600;
else
    T = 700;
end

if (mod(index,3) ==1)
    rate_adsorption = 3;
elseif (mod(index,3)==2)
    rate_adsorption = 5;
else
    rate_adsorption = 7;
end
```

Get Optimal Profile

```
function [sheight, srough, sgrain, sT, srate_ads] = get_optimal(t,fraction_A)

if(fraction_A == .1 || fraction_A == .9)
    optads = 8;
    optT= 800;
    optgrain= 8.694646422;
      optheight = [0  8.0088  15.9428 23.9696 31.8608 39.9104 47.9048 55.8732
      63.8632 71.7888 79.798  87.7828 95.7876 103.7664    111.7664
      119.8044    127.7928    135.8656    143.8784    151.9256    159.9944
      167.9004    176.0012    183.9276    191.9156    199.8788    207.8296
      215.7864    223.774 231.728 239.7396    247.7268    255.6816
      263.6636    271.618 279.6232    287.65  295.7736    303.7692
      311.7496    319.6756    327.6276    335.6236    343.5748    351.6536
      359.6336];
    optrough =[0    0.890437477 1.024142022 1.141548463 1.162676204
    1.239559116 1.290669198 1.297866314 1.31107656  1.288290967 1.336087659
    1.366970034 1.376462975 1.309515879 1.348255686 1.325907725 1.338463675
    1.333502878 1.306278227 1.306775101 1.31976435  1.341270372 1.383519423
    1.401931669 1.333903002 1.346072901 1.368052727 1.369001435 1.419992368
    1.395525221 1.444813635 1.444087262 1.478427467 1.47284383  1.424577381
    1.441629191 1.403417492 1.450848187 1.444113975 1.482530164 1.56713143
    1.473726933 1.456310477 1.49575777  1.479630202 1.514649083];

end
if(fraction_A == .2|| fraction_A == .8)
    optads = 3;
    optT=3 ;
    optgrain= 3;
t =
[0,6.95960000000000,13.9640000000000,20.9648000000000,27.9914000000000,34.986
5000000000,41.9982000000000,48.9740000000000,55.9486000000000,62.889000000000
0,69.8974000000000,76.8901000000000,83.9238000000000,90.9535000000000,97.9560
000000000,104.905500000000,111.917900000000,118.915200000000,125.926300000000
,132.957500000000,139.969400000000,146.952900000000,153.917700000000,160.8845
00000000,167.905600000000,174.870000000000,181.872000000000,190.853000000000,
199.846000000000,208.891000000000,217.889700000000,226.873900000000,235.85520
0000000,244.834600000000,253.846800000000,262.829200000000,271.854400000000,2
80.865100000000,289.871300000000,298.846800000000,307.885700000000,316.912400
000000,325.926500000000,334.912600000000,343.842600000000,352.893800000000,36
1.917200000000];
    optrough
=[0,2.62125947300000,3.70114748100000,4.50593897900000,5.21243235700000,5.823
39106900000,6.32894985600000,6.78306634500000,7.24039156200000,7.723801432000
00,8.12862109200000,8.54382814400000,8.94060721100000,9.27981755100000,9.6613
2811800000,10.0270677300000,10.3238886500000,10.6544736900000,10.964960800000
0,11.2641999900000,11.5553970300000,11.8276863800000,12.0808815800000,12.3611
031700000,12.5958001200000,12.8853822100000,13.1533793500000,13.5001927400000
,13.8436856800000,14.1512184700000,14.4533822300000,14.7887317500000,15.10637
35700000,15.3738147200000,15.6510069400000,15.8957356000000,16.1972688400000,
```

```
16.5053579800000,16.7504087000000,16.9796451900000,17.1999052600000,17.416523
7200000,17.6756137000000,17.9413265500000,18.2558671000000,18.5362982900000,1
8.8155883700000];

end
if(fraction_A == .3 || fraction_A == .7)
    optads = 5;
    optT= 700;
    optgrain= 2.280975904;
    optheight = [1.078901808    4.920266667 9.971733333 14.98773333
20.07173333 25.03333333 30.05626667 35.04453333 40.11333333 45.11413333
50.15413333 55.1128 60.1064 65.05706667 70.08026667 75.02986667 80.0888
85.1184 90.07253333 95.1176 100.084 105.0432    110.044 115.0309333
120.0226667 125.1005333 130.1234667 135.1085333 140.0498667 145.0754667
150.0797333 155.1245333 160.056 165.0362667 170.0050667 175.0365333
180.0733333 185.0434667 190.0317333 195.0122667 199.9829333 204.9696
210.0168    215.0210667 220.0242667 225.0752    230.0909333 235.0824
240.0682667 245.0472    250.1344    255.1546667 260.1677333 265.0986667
270.1032    275.0725333 280.0309333 285.0821333 290.1336    295.1650667
300.1645333 305.228 310.2128    315.2354667 320.2576    325.2189333
330.1178667 335.12  340.1213333 345.1002667 350.0776    355.0624
360.0648];
    optrough =[0    1.193207176 1.374035651 1.547492475 1.700010662
1.811286824 1.912338176 1.900701048 2.026567168 2.014924765 2.048660014
2.157518435 2.220943697 2.261755399 2.235196222 2.231146288 2.320312252
2.379038341 2.433875316 2.506549916 2.575521464 2.546293639 2.606702489
2.632976133 2.617523657 2.651743069 2.710593486 2.757708599 2.717365757
2.801548027 2.822024484 2.860550296 2.898477937 2.958757005 2.970929075
2.962413588 2.997768092 2.994615021 3.02863191  3.044031823 3.06352815
3.012043021 3.019159833 3.03553169  3.097826673 3.126214088 3.149129378
3.153888697 3.168010296 3.246245556 3.21948907  3.2824264   3.374950911
3.38121563  3.389503509 3.344052161 3.321315185 3.307259292 3.331311529
3.319145031 3.356795082 3.371012442 3.411747913 3.487319279 3.474263661
3.513568666 3.491020857 3.496851069 3.511979934 3.557804357 3.571761402
3.55023316  3.545971325];
end
if(fraction_A == .4 || fraction_A == .6)
    optads = 3;
    optT= 3;
    optgrain= 3;
    optheight =
[0,6.95960000000000,13.9640000000000,20.9648000000000,27.9914000000000,34.986
5000000000,41.9982000000000,48.9740000000000,55.9486000000000,62.889000000000
0,69.8974000000000,76.8901000000000,83.9238000000000,90.9535000000000,97.9560
000000000,104.905500000000,111.917900000000,118.915200000000,125.926300000000
,132.957500000000,139.969400000000,146.952900000000,153.917700000000,160.8845
00000000,167.905600000000,174.870000000000,181.872000000000,190.853000000000,
199.846000000000,208.891000000000,217.889700000000,226.873900000000,235.85520
0000000,244.834600000000,253.846800000000,262.829200000000,271.854400000000,2
80.865100000000,289.871300000000,298.846800000000,307.885700000000,316.912400
000000,325.926500000000,334.912600000000,343.842600000000,352.893800000000,36
1.917200000000];
    optrough
=[0,2.62125947300000,3.70114748100000,4.50593897900000,5.21243235700000,5.823
39106900000,6.32894985600000,6.78306634500000,7.24039156200000,7.723801432000
00,8.12862109200000,8.54382814400000,8.94060721100000,9.27981755100000,9.6613
2811800000,10.0270677300000,10.3238886500000,10.6544736900000,10.964960800000
```

```
0,11.2641999900000,11.5553970300000,11.8276863800000,12.0808815800000,12.3611
031700000,12.5958001200000,12.8853822100000,13.1533793500000,13.5001927400000
,13.8436856800000,14.1512184700000,14.4533822300000,14.7887317500000,15.10637
35700000,15.3738147200000,15.6510069400000,15.8957356000000,16.1972688400000,
16.5053579800000,16.7504087000000,16.9796451900000,17.1999052600000,17.416523
7200000,17.6756137000000,17.9413265500000,18.2558671000000,18.5362982900000,1
8.8155883700000];
end
if(fraction_A == .5)
    optads = 8;
    optT= 800;
    optgrain= 2.167423587;
    optheight = [0  7.93984 15.98624    23.99936    31.93568    39.94304
47.91328    55.93024    63.97248    72.0048 80.06496    88.06528    96.00832
103.97952   111.90656   119.84736   127.82752   135.77504   143.7632
151.76224   159.63136   167.6016    175.60288   183.51296   191.52032
199.512 207.46144   215.43008   223.49344   231.41056   239.33664   247.36768
255.2736    263.20704   271.24096   279.2032    287.28032   295.22624
303.27008   311.30304   319.37056   327.36224   335.34496   343.37664
351.29472   359.3824];
    optrough =[0.156976503  0.719548377 0.809012554 0.800801981 0.797764012
0.80586232  0.817896691 0.838572493 0.836778171 0.820798856 0.847516908
0.848643349 0.884001038 0.868333084 0.847930638 0.819743693 0.834350354
0.832066713 0.838220664 0.840757059 0.884167628 0.911582631 0.904446945
0.898412991 0.923928376 0.884621976 0.870290991 0.916370528 0.932771505
0.872618731 0.828274292 0.867951495 0.893192988 0.825795199 0.856614169
0.840464719 0.906566104 0.914239666 0.862388555 0.885581156 0.858943069
0.823066323 0.751229803 0.819130019 0.791802418 0.704322116];
end

sheight = optheight(t);
srough = optrough(t);
sgrain = optgrain;
sT = optT;
srate_ads = optads;
```

Calculate Error

```
function [sum] = calc_sum(rough, srough, height, sheight, grain,
sgrain,fraction_A)

 if(fraction_A == .1 || fraction_A==.9)
     if(grain > sgrain + 0.756019029)
         sgrain = sgrain - 0.756019029;
     elseif (grain < sgrain - 0.756019029)
         sgrain = sgrain + 0.756019029;
     end
 end
  if(fraction_A == .3 || fraction_A==.7)
     if(grain > sgrain + 0.137408295)
         sgrain = sgrain - 0.137408295;
     elseif (grain < sgrain - 0.137408295)
```

```
         sgrain = sgrain + 0.137408295;
     end
  end
   if(fraction_A == .5)
     if(grain > sgrain + 0.102079518)
         sgrain = sgrain - 0.102079518;
     elseif (grain < sgrain - 0.102079518)
         sgrain = sgrain + 0.102079518;
     end
 end


 if (srough ==0)
     dA =1;
 else
     dA= srough;
 end


 if (sheight == 0)
     dB = 1;
 else
     dB = sheight;
 end

 if (sgrain == 0)
     dC = 1;
 else
     dC = sgrain;
 end


sum = abs(rough-srough)/dA + abs(height-sheight)/dB + abs(grain - sgrain)/dC;
```

Get Controller Lattice Patches

```
function [height_grid,particle_grid, grid_size] = get_gridMP(nsitescol,
nsitesrow, nsites_max, height,particle_list)
%This function inputs the dimensions of the lattice, the number of lattice
%points, the current height vector, and the particle list

grid_size = 10;

for i=1:10;
    %Pick a random lattice point (top left corner of grid)
    i_new = uint32((nsites_max-1)*rand())+1;

    %Establish temporary matrices
    list_height = 0;
```

```
    list_height_counter = 1;
    list_particle = 0;

    test = mod(i_new,nsitesrow);   %to check vertical periodicity

    if (test == 0)
        test = nsitesrow;
    end
    if (test + grid_size - 1 > nsitesrow)
      %if when you go down the column you will fall off, come back to top
        numbeforewrap = nsitesrow - test+1 ;
      %number of sites that work before wrapping
    else
        numbeforewrap = grid_size;         %if you don't wrap around
    end

    %Get starting top row
    for k=0:grid_size-1
        startingpoint = i_new +nsitesrow*k;
      %start at k = 0 for column where i is located, move over a column as
            many times as size requires
        if (startingpoint>nsites_max)
            startingpoint = startingpoint-nsites_max;
      %horizontal periodicity- go back to beginning if it goes over end
        end
        for (j=0:grid_size-1)
            if (j>=numbeforewrap)
            %if you're sampling a site that has wrapped around
                wrap = nsitesrow;
            else
                wrap = 0;
            end

        list_height(list_height_counter) = height(startingpoint+j-wrap);
            %get the height at each location
        list_particle(list_height_counter, 1:50) =
            particle_list(startingpoint+j-wrap,1:50);
            %record the particle list at each location
        list_height_counter = list_height_counter+1;
        end
    end

    %Enter into grid info
    height_grid(i, 1:grid_size*grid_size) = list_height;
    for h=0:99
        particle_grid(i*100-99+h, 1:50) = list_particle(h+1,1:50);
    end
end
```

Main Multiple Component KMC

```
%#eml
function [particle_list, height,
roughness]=main_kmc_code(timef,timesteps,height,nsitesrow,nsitescol,T,rate_ad
sorption,fraction_A,particle_list)

%******************MAIN KMC CODE**************************************
%surface deposition with adsorption, desorption and hopping
%2D lattice with height function
%
%timef - final system time
%timesteps - # of time steps for output
%nsitesrow - dimension of lattice
%nsitescol - dimension of lattice
%
%
%The following are simple ways to set dyanmics (will need to be changed
%     according to the system of interest)
%
%   rate_hop         - rate of hopping
%   rate_adsorption - rate of adsorption
%   rate_desorption - rate of desorption
%********************************************************************
%
%...first initiate main system variables
%nsitesrow          - number of sites in each row
%nsitescol          - number of sites in each column (see initiate_colrow)
%max_level_particle - maximum number of atoms for storage at each site
%max_level          - number of levels for skiplist
%levelorder         - probability of gaining a level for skiplist
%***the skiplist variables should not need to be changed unless
%   max_events>10^8***
max_level=8;
levelorder=0.1;
y=0;
%
eml.extrinsic('fprintf');%used only for compiled matlab output (probably not
useful though)
%
%generates different seed each run time
rand('twister',sum(100*clock))
countads=0;
countdes=0;

%nsites_max - total number of sites
%max_events - currently there are 6 possible events (4 hopping directions,
%               adsorption and desorption)
%null_event - needed for skiplist
nsites_max=nsitesrow*nsitescol;
max_events=nsites_max*6;
```

```
null_event=max_events+1;
%
%height - height of surface above site #
%height=zeros(nsitescol*nsitesrow,1);
%
%
%sitecol - number of sites up the column
sitecol=zeros(nsites_max,1);
%
%siterow - number of sites down the row (see initiate_colrow)
siterow=zeros(nsites_max,1);
%
%nnlist - stores the nearest neighbor sites (see initiate_nnlist)
nnlist=zeros(4,nsites_max);
%
%rate - stores the rate for all possible events
rate=zeros(null_event,1);
%
%wait_time - stores the waiting_time for all possible events (also the key
%               for the skiplist
wait_time=zeros(null_event,1);
%
%initialize time
time=0.0;
%
%head - stores position of the min waiting time
%tail - stores position of the max waiting time (not used here, but needed
for
%             skiplist functions)
head=null_event;
tail=null_event;
%
%...main skiplist array, all pointers start with null_event
forward=ones(max_level,null_event)*null_event;
%
%...count keeps track of number of steps
%
count=0;
%
%...timebin partitions the time for outputting with timesteps
%...counter_time_bin counts number of outputs
%...write_flag is used to break loop for outputting
%
timebin=timef/timesteps;
counter_time_step=1;
write_flag=true;
%
%...print variables to command window
%
fprintf('%s %e \n','    Temperature: ', T);
fprintf('%s %e \n','Adsorption Rate: ', rate_adsorption);
%fprintf('%s %e \n','Desorption Rate: ', rate_desorption);
%
%...initiate column/row numbering (not strictly needed see function)
%
[sitecol,siterow]=initiate_colrow(...
    nsitescol,nsites_max,...
```

```
    siterow,sitecol);
%
%...initiate nnlist (not strictly needed see function)
%
[nnlist]=initiate_nnlist(...
    nsitesrow,nsitescol,nsites_max,...
    siterow,sitecol,nnlist);
%
%...set initial height profile
%
height_in=height;
%
%...initiate rates of all possible events
%
[rate,wait_time,head,tail,forward]=initiate_rate(...
    time,rate_adsorption,levelorder,...
    nsitesrow,nsitescol,nsites_max,max_level,null_event,...
    siterow,sitecol,nnlist,...
    height,particle_list,rate,wait_time,...
    head,tail,forward, T);
%
%...MAIN LOOP (until final system time is reached)
%
while(time<=timef)
    %
    fprintf('\n %s %10f %s %e %s %10f',...
       'moves:',count,' time:',time,' #:',counter_time_step-1)
    %
    while(write_flag)
        %
        count=count+1;
        %...Execute event with minimum waiting_time

[time,rate,wait_time,head,tail,forward,height,particle_list,countads,countdes
]=event(...
            rate_adsorption,fraction_A,levelorder,...
            nsitesrow,nsitescol,nsites_max,max_level,null_event,...
            siterow,sitecol,nnlist,...
            height,particle_list,rate,wait_time,...
            head,tail,forward, T, countads,countdes);

        %
        if(time>=timebin*counter_time_step)
            write_flag=false;
        end
        %

    end
    %
    counter_time_step=counter_time_step+1;
    write_flag=true;
    %
    roughness = std(height(1:end-1));
end
%
fprintf('\n %s %10f %s %e %s %10f \n',...
```

```
        'moves:',count,' time:',time,' #:',counter_time_step-1)
```

C.5 Initiate Surfaces

Initiate Height Hills/Valleys

```
function [height] = initiate_height_humps(nsites_max, height)
height_base = 0;
for i=1:nsites_max
    height(i)=height_base;
end
for i =1:nsites_max
    height(i) = height(i) +10*sin(floor((i+99)/100)*pi/20)-
10*sin(mod(i,100)*pi/20);
end
```