SAND2009–6226
Unlimited Release
Printed October 2009

Crossing the Mesoscale No-Man's Land via Parallel Kinetic Monte Carlo

Steve Plimpton, Corbett Battaile, Mike Chandross, Liz Holm, Aidan Thompson,
Veena Tikare, Greg Wagner, Ed Webb, Xiaowang Zhou

Sandia National Laboratories
P.O. Box 5800, MS-1316
Albuquerque, NM 87185-1316
sjplimp@sandia.gov

Cristina Garcia Cardona

Computational Science Program, San Diego State University
5500 Campanile Drive
San Diego, CA 92182-1245
cgarcia@sciences.sdsu.edu

Alex Slepoy

Office of Nonproliferation Research & Development
U.S. Department of Energy, NNSA, NA-22
1000 Independence Ave., S.W.
Washington, D.C. 20585
alexander.slepoy@nnsa.doe.gov

Abstract

The kinetic Monte Carlo method and its variants are powerful tools for modeling materials at the mesoscale, meaning at length and time scales in between the atomic and continuum. We have completed a 3 year LDRD project with the goal of developing a parallel kinetic Monte Carlo capability and applying it to materials modeling problems of interest to Sandia. In this report we give an overview of the methods and algorithms developed, and describe our new open-source code called SPPARKS, for Stochastic Parallel PARticle Kinetic Simulator. We also highlight the development of several Monte Carlo models in SPPARKS for specific materials modeling applications, including grain growth, bubble formation, diffusion in nanoporous materials, defect formation in erbium hydrides, and surface growth and evolution.

Contents

1 Overview

Materials inherently interact with their environment at the atomic scale, e.g. via mechanical, chemical, or electrical means. However, their response is usually manifest and observed at the continuum scale. Kinetic Monte Carlo (KMC) is a powerful computational tool for spanning these length and time scales. Critical events, such as diffusive hops or reactions, are defined and rate equations can be specified in terms of external fields, such as electric potential or solute concentration, in order to capture the model's relevant physical underpinnings. Efficient KMC algorithms can then select events one after another with the correct probabilities and update the state of the system, without the need to follow detailed atomic motion and spend CPU time waiting for events to occur. However the classic KMC algorithm is inherently serial, and thus KMC has often been limited in the size of systems and timescale of phenomena it can model.

The goals of this LDRD project were three-fold:

- To develop new parallel KMC algorithms that remove this bottleneck for specific kinds of KMC models.
- To build a flexible code incorporating these algorithms to enable a suite of KMC models to be implemented.
- To use the new code to model several systems important for materials science applications at Sandia, at a scale that has not previously been possible.

The remainder of this report is structured as follows. In Section 2 the KMC method is described. In Section 3 our new Monte Carlo code SPPARKS and its basic functionality and design are discussed, focusing on the flexibility it offers for adding new models to the code. The heart of a KMC model is the algorithm for selecting events. In Section 4, three such algorithms are detailed, all of which are implemented as options in SPPARKS. In Section 5 the basic algorithms by which SPPARKS can run KMC models in parallel are highlighted, and some performance data is given for Potts grain growth models run on a large parallel machine. In Section 6 SPPARKS accuracy and performance is compared to that of a serial KMC code MESO. Since MESO was developed independently, this comparison serves as a "verification" of the SPPARKS implementation of the KMC model.

Section 7 is the largest section of the report. It contains 7 sub-sections, each of which describe a different physical model, its implementation in SPPARKS, and highlight simulation results which illustrate what the KMC method is capable of modeling. These applications are for abnormal grain growth in 7.1, nanoporous metals in 7.2, a solid-on-solid model for surface diffusion in 7.3, defect formation in erbium hydrides in 7.4, bubble formation in nuclear fuels in 7.5, the sintering of a nuclear fuel pin in 7.6, and thin film deposition and growth in 7.7.

A brief summary of future development still needed in SPPARKS as well as KMC modeling projects at Sandia that intend to use the code is given in Section 8.

Finally, we note that SPPARKS is available for download as an open-source code from this WWW site: www.sandia.gov/∼sjplimp/spparks.html. The web site includes HTML documentation as well as input scripts and pictures and movies of example calculations, some of which are discussed in this report.

2 Kinetic Monte Carlo

This section contains a paper that appeared in the journal "Computer Methods in Applied Mechanics and Engineering", titled "The kinetic Monte Carlo method: Foundation, implementation, and application". It was authored by Corbett C. Battaile and published in vol 197, pages 3386-3398, in 2008.

It provides a good overview of the kinetic Monte Carlo (KMC) method and its utility for materials simulations. Two additional recent papers which also give nice descriptions of KMC and related MC methods are [16, 83].

The entire text of the CMAME paper is included here, except for the appendices. We note that the simulations presented in this paper were not performed with SPPARKS (discussed next in Section 3), but with a serial KMC code developed independently by Corbett, prior to the start of our LDRD.

2.1 Abstract

The Kinetic Monte Carlo method provides a simple yet powerful and flexible tool for exercising the concerted action of fundamental, stochastic, physical mechanisms to create a model of the phenomena that they produce. This manuscript contains an overview of the theory behind the method, some simple examples to illustrate its implementation, and a technologically relevant application of the method to model the refinement of grains during polycrystalline thin film deposition. The objective is to provide an introduction to the method and its basics, and to present useful examples that might be followed to achieve its implementation for solving practical problems.

2.2 Introduction

Nature is rife with fundamental processes and elementary mechanisms that, when viewed at the appropriate scale, behave stochastically (or, at least, appear to). For example, the thermal vibrations of atoms in a potential energy "well" appear to be random. Therefore, the details of atomic diffusion are essentially stochastic, since diffusion is in some sense simply a result of the extreme vibrational excursions that push atoms from one energy well to another. And yet the collective behavior of large numbers of diffusing atoms can, on average, be quantified deterministically. When an atom is diffusing on an atomic lattice, its trajectory from jump to jump is random (appropriately weighted according to the energetic landscape that the atom "sees"), but Fick's second law of diffusion provides a deterministic equation relating the temporal composition gradient to the Laplacian (i.e. the divergence of the spatial gradient) of the average composition when many atoms are diffusing simultaneously. This leads to a second salient point, namely that statistically sufficient ensemble- or time-averaged behaviors of collective stochastic phenomena can be described deterministically.

In certain simple cases, the average (i.e. deterministic) behavior of fundamentally stochastic phenomena can be derived analytically, and closer examination is not required. Unfortunately, problems of technological and/or scientific interest are rarely so simple. Direct observation is sometimes possible, e.g. when composition analysis is performed to measure diffusion profiles, but this rarely provides insight into the nature or the details of the fundamental mechanisms that produced the observed behavior. A variety of theoretical and computational approaches exist for exercising fundamental processes in order to predict collective phenomena. For example, molecular dynamics (MD) is a particularly powerful technique that uses interatomic energy functionals to compute forces between atoms, and then solves Newton's equation of motion to predict the atom trajectories. However, these trajectories include (of course) atomic vibrations, and thus MD is usually not able to address phenomena that require treatment on much larger scales. The finite element method (FEM) lies at the other extreme and is invaluable for solving problems in solid mechanics, heat transport, and other fields, but its inherently continuum nature makes it ill-suited in situations where atomic or microstructural details are important.

The Monte Carlo (MC) method provides a relatively general approach for exploring the thermodynamic or kinetic behavior of sequences of arbitrary fundamental transitions. The approach offers considerable flexibility in the nature and specification of this fundamental process reflects, and the MC method can therefore

span potentially arbitrary spatial and temporal scales and is useful in an extremely wide range of applications, from the behavior of the stock market [39] to the formation of clusters of galaxies [53]. The Kinetic Monte Carlo (KMC) method is a variant specifically designed to assemble kinetic and procedural information about potentially arbitrary, fundamental processes and mechanisms, and to then examine stochastic sequences of them in order to model the temporal evolution of a complex system. Because these fundamental processes can be expressed on time and length scales that are much larger than those associated with atomic vibrations, the KMC approach can straightforwardly access regimes between MD and the FEM, e.g. when atomic-scale details are important but operate on a time scale beyond the reach of MD. In addition, the algorithmic foundation of KMC is stochastic by its very nature, making it well suited to describing the many process that behave likewise, e.g. atomic diffusion or chemical reaction. Elementary mechanisms that occur on very small scales can be condensed into coarser-scale processes that encompass the collective action of many sub-processes, e.g. KMC is used to model atomic diffusion without actually resolving the details of atomic vibration, or to simulation grain growth on time and length scales well in excess of those involved in the transport of atoms across a grain boundary.

This paper presents on overview of the basic underpinnings of KMC; a set of simple examples of the method's implementation in models of vacancy diffusion, grain growth, and film deposition; and a practical application of the method toward elucidating a potential mechanism for grain refinement during the deposition of polycrystalline thin films. The intent is to provide an introduction to the KMC method, and a basic overview of its utility and implementation.

2.3 Method

2.3.1 History

The Monte Carlo method as a means for solving scientific problems was first proposed in 1946 by Stan Ulam [79] in his correspondences with John von Neumann. Since then, it has been used to model a wide variety of phenomena, and has evolved into several well-established variants. In its earliest forms, the MC method was essentially an efficient means for numerically estimating complex integrals. In that context, it could be operated simply by computing the energies of a model system in randomly selected states, and weighting the likelihood of realizing each state according to the Boltzmann energy equation.

In 1953, Nicholas Metropolis et al. [55], while working with Ulam shortly after the method's inception [56], improved on this simple scheme by biasing the state's selection itself, and weighting each state equally. In this manner, the calculation wastes less time exploring configurations that are unlikely to exist. Bortz, Kalos, and Lebowitz [9] introduced yet another rephrasing of the MC method, and termed the new algorithm the N-Fold Way (NFW). The NFW algorithm chooses and implements random configurational transitions with unit probability (i.e. unconditionally), and the choices are biased according to each transition's likelihood (i.e. rate in a kinetic context) so that more likely transitions are chosen more often. The stochastic component of the NFW algorithm is also manifested in the time incrementation (which can thereby vary at each MC step). Independent discoveries of the NFW algorithm were presented by Gillespie [30], and more recently by Voter [82] (with a minor difference in the implementation of time incrementation).

These and other Kinetic Monte Carlo methods have been applied to a very wide range of applications, and the reader is referred to a selection of the numerous texts and review articles on this topic [4, 7, 17, 33, 38, 41, 43, 45, 47, 84].

2.3.2 Formulation

There are several other KMC variants in common use today, and the present paper focuses on a kinetic adaptation [5, 35, 82] of the NFW [9, 25, 30]. At its core, the kinetic version of Monte Carlo, i.e. the KMC method, stochastically explores sequences of transitions in the state of a model system, selecting links in the chain of events according to the rates of the transitions. In cases where the transitions are thermally activated, e.g. atomic diffusion, the transitions themselves are diffusional hops, and rates of the transitions are parameterized by activation energies, attempt frequencies, and temperature, usually via an Arrhenius equation. A KMC model must be parameterized by at least the rates of each type of transition, and the change (usually local) that each transition induces in the model system. For example, in the case of vacancy diffusion (by which most atomic self-diffusional processes are enabled), the diffusion rates are governed by an Arrhenius equation,

$$r_D = D_o exp(-\frac{Q}{kT}) \tag{1}$$

where r_D is the average rate of a vacancy hop, D_o is an exponential prefactor, Q is the activation energy for diffusion, k is Boltzmann's constant, and T is the temperature. These kinetic parameters can vary depending on the details of the local atomic configuration, but in the simplest case all the diffusion events are similar, and there is only one possible value for the rate even though individual hops can occur at various locations and in different directions. In order to specify these locations and directions precisely, the underlying lattice and the allowed types of diffusion must both be specified, e.g. hops to any of the six nearest-neighbor sites on a simple cubic lattice.

Even when only one type of event is possible, multiple realizations of it will generally be possible in any given state of the model system. For example, on a two-dimensional, simple-square lattice containing a single vacancy that can jump only to a nearest-neighbor lattice site, only one type of event is possible, but four realizations of it are available, one for each of the vacancy's four nearest atom neighbors.

The particular KMC variant addressed here maintains a "running tally" of all the possible individual realizations of the allowed event types. (These individual realizations are hereafter termed simply "events".) Each of the events has an associated rate, although their values might all be the same. The state of the model system is evolved by choosing one event stochastically, according to the events' rates, via

$$\sum_{i=1}^{m-1} r_i \leq \rho_1 \sum_{j=1}^{N} r_j < \sum_{k=1}^{m} r_k, \tag{2}$$

where i, j, and k are summation indices denoting the individual events, r_i is the rate of the event i, m is the site of the chosen event, N is the total number of possible events throughout the model system, and ρ_1 is a random number evenly distributed over the range $[0, 1)$. Selecting an event according to Eq 2 ensures that faster events have a greater probability of being chosen than do slower ones. Once an event is chosen, the model system is modified at the particular location corresponding to the specific event, according to the nature of the transition. For example, in the case of vacancy diffusion, the chosen event will correspond to a particular vacancy hopping in one of the available directions on the atomic lattice, and so the vacancy must be swapped with the atom in the destination lattice site. In the proximity of the event, the system will be modified (otherwise the event is trivial), and the running tally of events, i.e. the event list, must be modified accordingly. In some cases, this could even result in a change to the total number, N, of possible events. The simulation time is then advanced by

$$\Delta t = -\frac{ln(\rho_2)}{\sum_{j=1}^{N} r_j}, \tag{3}$$

where Δt is the time that has elapsed between the previous event, i.e. the one prior to m, and the event m itself; ρ_2 is a random number evenly distributed over the range (0,1); and N is the total number of possible events before event m has transpired. This procedure represents a single time step in a KMC model. At the conclusion of the time step, once the event list has been updated accordingly, the procedure is repeated to realize the next step. Note that, unlike in the Metropolis algorithm [55], an event is always realized at each and every time step, and the conditional event acceptance that's inherent in the Metropolis scheme is manifested by ρ_2 in Eq 3.

2.4 Implementation

The KMC algorithm described above is relatively straightforward in principle, and much of the work is involved in describing realistic model systems and kinetic parameters for input to the model. The procedural details of the latter are well beyond the scope of the present discussion, and the reader is referred to Appendix A (in the journal version of the paper) for a presentation of the method's implementation toward a trivial model system. In this section, three simple yet practical examples of such an implementation are presented to further clarify the process: vacancy diffusion, grain growth, and thin film deposition.

2.4.1 Vacancy Diffusion

Consider a rigid, simple-square lattice containing a random distribution of vacancy defects. Assume that a vacancy can diffuse to any of the four nearest non-vacant sites, and that the attempt frequency and activation energy are the same for all hops. Then the rate of a diffusional hop is

$$r_D = D_o exp(-\frac{Q}{kT}) \text{ if } \Delta E \leq 0; r_D = D_o exp(-\frac{Q+\Delta E}{kT}) \text{ if } \Delta E > 0, \tag{4}$$

where ΔE is the change in the system's energy due to the hop. If the energy of the model system is JB, where J is the bond energy and B is the total number of bonds, then $\Delta E = nJ$, where n is the change (either positive or negative) in the number solid-solid bonds due to the diffusion hop. Note that the bond energy, J, is generally negative, otherwise the material that occupies the lattice would not be thermodynamically stable as a solid phase.

For the starting system configuration in Fig 1a, i.e. a simple square lattice with periodic boundary conditions imposed in both directions, containing 128x128 sites with 10% vacancies, any pair of adjacent light (vacancy) and dark (solid) sites can exchange in a vacancy "hop." If the temperature and bond energy are chosen such that $kT = -J$, the activation energy for diffusion assumed to be $Q = 0$, and the attempt frequency taken as $D_o = 1$, then the average rate at which an lone vacancy hops to a nearby site is $r_1 = 1 * exp(0) = 1$ according to Eq 4. (Note that the diffusion of a lone vacancy does not change the system's energy.) Two lone vacancies create eight dangling atomic bonds total, whereas a pair of adjacent vacancies creates six. Therefore, not only is the rate of dissolution of a vacancy pair (i.e. the diffusion of one of the vacancies away from the other) equal to $r_2 = 1 * exp(-2) = 0.135335$ according to Eq 4, but also a vacancy pair is energetically favorable and will tend to persist, as indicated by the difference between the rates, r_1 and r_2. For this reason, the evolution of the system in Fig 1a, as depicted in Fig 1b-1f, results in the formation of vacancy clusters, i.e. pores, which coarsen until all that remains is a pair of large pores. (Given enough time, the smaller of the two pores would shrink and eventually disappear, while the larger would grow and persist.)

The images in Fig 2 contain a progression similar to that in Fig 1, except that the simulation temperature is $kT = -2J$. Notice that the vacancies again coarsen to form a large pore, and that because the temperature is higher, the large pore is surrounded by the thermodynamic equilibrium concentration of individual

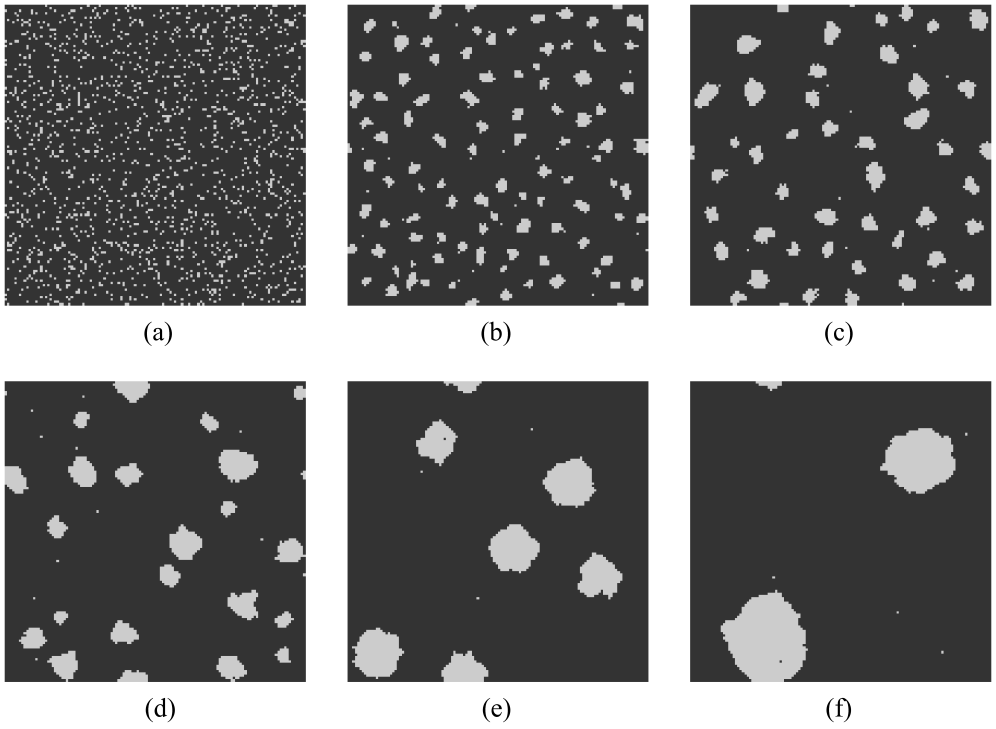


Figure 1: *Vacancy diffusion and pore formation at a simulation temperature of $kT = -J$, where J is the bond energy, at simulation times of a) 0 mcs, b) 1,000 mcs, c) 10,000 mcs, d) 100,000 mcs, e) 1,000,000 mcs, and f) 10,000,000 mcs. Light gray pixels are vacancy sites, and dark pixels are solid.*

vacancies and very small pores. Furthermore, the single, large pore contains the thermodynamic equilibrium concentration of solid particles in what essentially amounts to a gas phase. An intermediate case is achieved in Fig 3, where $kT = -3J/2$, and an extreme case in Fig 4 where $kT = -3J$ such that the equilibrium vacancy concentration is near the total and very little vacancy clustering occurs.

It is clear from Eq 4 that at higher temperatures, the rate of vacancy pair dissolution is higher whereas the rate of vacancy pair formation is unchanged (because it decreases the energy). Therefore, vacancies congregate more slowly at higher temperatures simply because their constituent vacancies are able to break away faster. This is not only the source of the relatively high concentrations of lone vacancies and small pores in Fig 2 and Fig 4, but also the reason that pore growth is faster at lower temperatures as shown in Fig 5. (Interestingly, although it is not shown here, the migration of large pores is faster at higher temperatures for exactly the same reason.) Note that this simple vacancy diffusion example is equivalent to the coarsening of any two phases, even when both are solid.

2.4.2 Grain Growth

Now consider another simple square lattice containing a random distribution of many different species ranging from 1 to 2,048, as in Fig 6a where each different species is shaded differently on a gray scale such that species 1 is black and 2048 is white. Assume that periodic boundary conditions are again imposed, and that any site can change its species (i.e. "flip") to assume the identity of one of its nearest neighbors. Also assume that the attempt frequency is the same regardless of the detailsails of the flip, that there is no activation energy for any flip, and that the bond energies between like and unlike species are $J = 0$ and

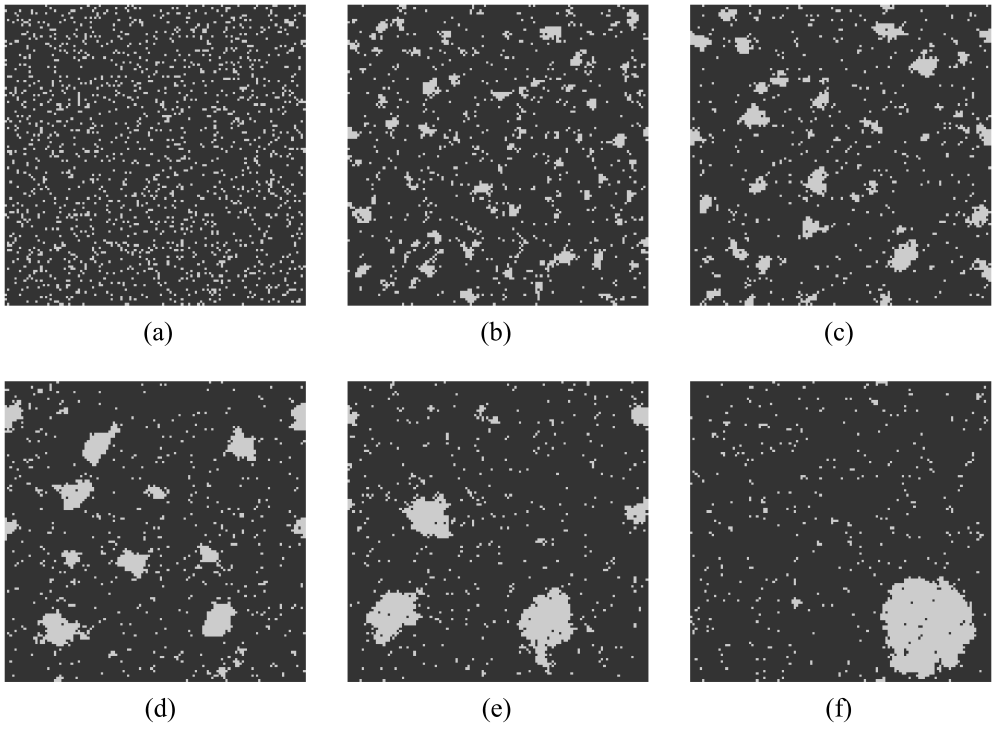


Figure 2: *Vacancy diffusion and pore formation at a simulation temperature of $kT = -2J$, where J is the bond energy, at simulation times of a) 0 mcs, b) 300 mcs, c) 2,000 mcs, d) 10,000 mcs, e) 60,000 mcs, and f) 800,000 mcs. Light gray pixels are vacancy sites, and dark pixels are solid.*

$J = kT$, respectively. In this case, it is clear that like bonds are preferred over unlike. Since $Q = 0$, the rate of flipping a lattice site to assume a neighbor's species is

$$r_G = A \text{ if } \Delta E \leq 0; r_G = Aexp(-\frac{Q + \Delta E}{kT}) \text{ if } \Delta E > 0, \tag{5}$$

where A is the activation energy, taken to be $A = 1$ for this example; and ΔE is the difference in the energies before and after the flip. As reflected in Fig 6b-6f, the system evolves toward a configuration containing only two species, i.e. a bicrystal. (Given sufficient time, one of the two grains in Fig 6f would disappear, leaving only one single crystal.) Clearly the size of the grains increases on average, i.e. the grains grow as evident in Fig 6, and the kinetics of grain growth are shown in Fig 7 where the average grain area increases approximately linearly in time (in the so-called scaling regime) according to $\bar{A} = 0.0037t^{1.15}$.

2.4.3 Film Deposition

Consider next a simple cubic lattice containing 128x128x128 sites, whose sites are empty, except along one face where the sites are filled by a solid substrate material. This is depicted in Fig 8a, where only the substrate is shown. Assume that periodic boundary conditions are imposed in the two directions parallel to the substrate's surface, and that any empty site that has at least one solid nearest-neighbor is a candidate for deposition, i.e. the transition of the empty site into a solid adatom species. Since this transition always implies the creation of new solid-solid bonds, it will always decrease the system's energy, and if the transition is barrierless then its rate is a constant representing the deposition flux. Assume further that an adatom can

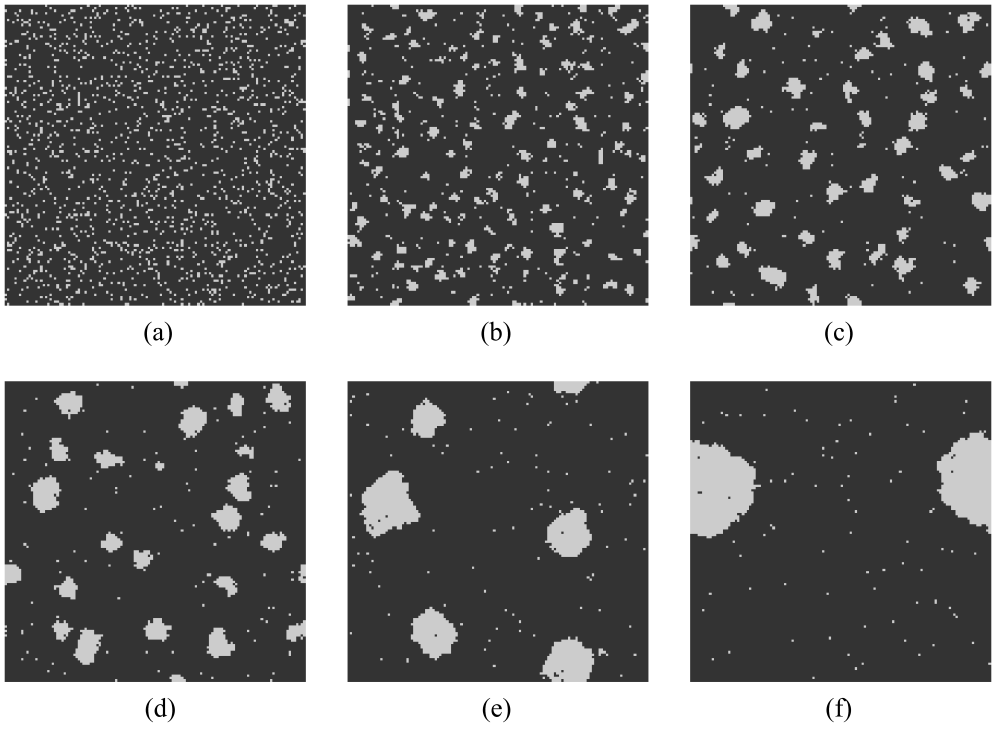


Figure 3: *Vacancy diffusion and pore formation at a simulation temperature of $kT = -3J/2$, where J is the bond energy, at simulation times of a) 0 mcs, b) 100 mcs, c) 2,000 mcs, d) 10,000 mcs, e) 200,000 mcs, and f) 5,000,000 mcs. Light gray pixels are vacancy sites, and dark pixels are solid.*

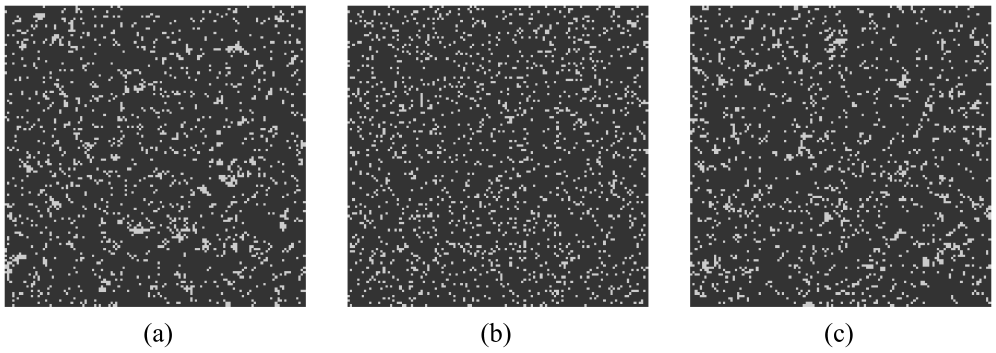


Figure 4: *Vacancy diffusion and pore formation at a simulation temperature of $kT = -3J$, where J is the bond energy, at simulation times of a) 0 mcs, b) 3,000 mcs, and c) 300,000 mcs. Light gray pixels are vacancy sites, and dark pixels are solid.*

diffuse into any empty nearest-neighbor site, as long as the adatom will have at least one solid-solid bond after it diffuses and does not leave any unbonded adatoms behind. The rate of adatom diffusion obeys Eq 4. Assume that the deposition rate is 0.001, the attempt frequency for adatom diffusion is $D_o = 1$, the adatom-adatom bond energy is $J = -2kT$, and the adatom-substrate bond energy is $J = -kT$. As evident in Fig 8b-8d, the early stages of deposition involve adatoms arriving at the substrate, diffusing along it, and

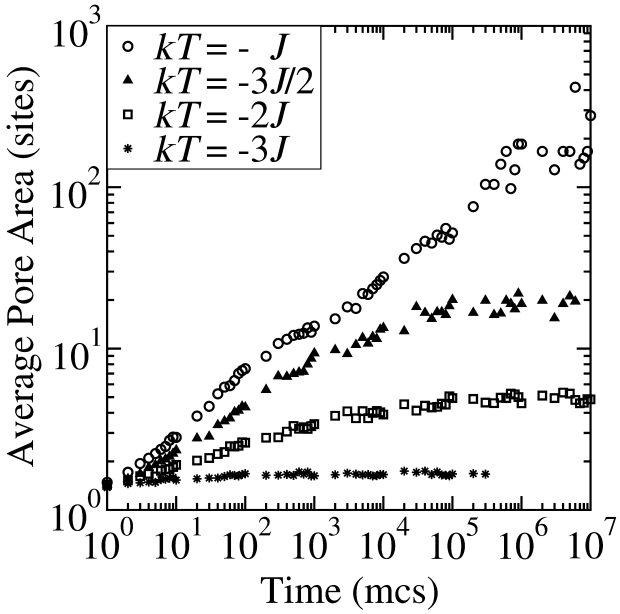


Figure 5: *Average pore area as a function of Monte Carlo time for the vacancy diffusion examples in Fig 1-4, at simulation temperatures of $kT = -J$, $-3J/2$, $-2J$, and $-3J$, where J is the bond energy.*

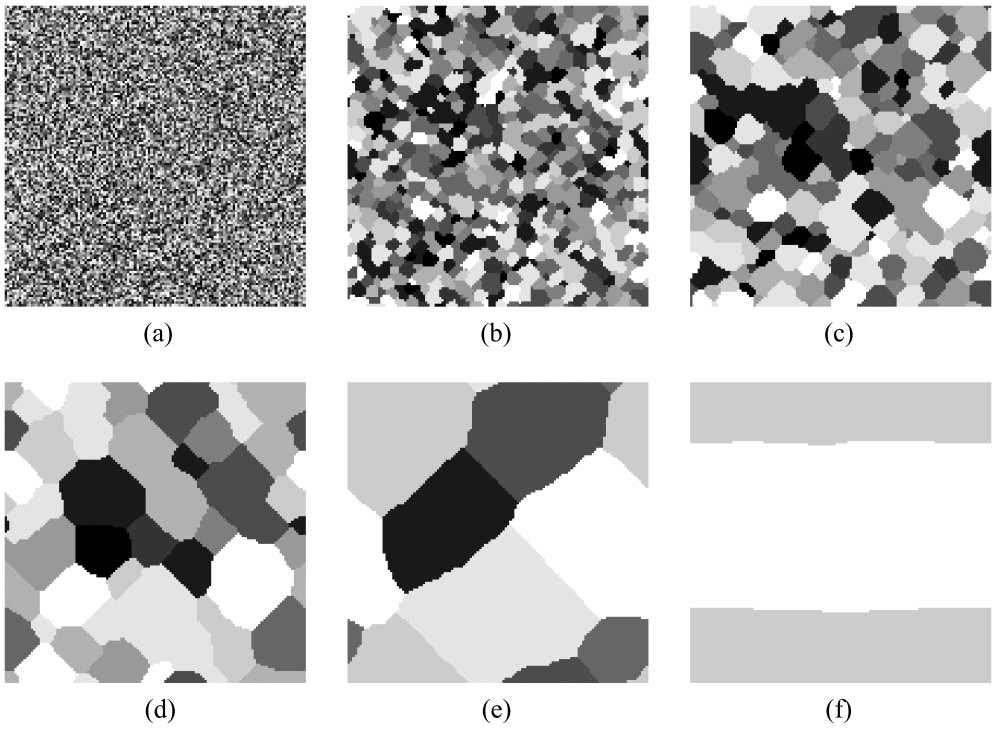


Figure 6: *Grain growth at simulation times of a) 0 mcs, b) 200,000 mcs, c) 600,000 mcs, d) 2,000,000 mcs, e) 6,000,000 mcs, and f) 2,000,000,000 mcs. Different gray levels represent different grains.*

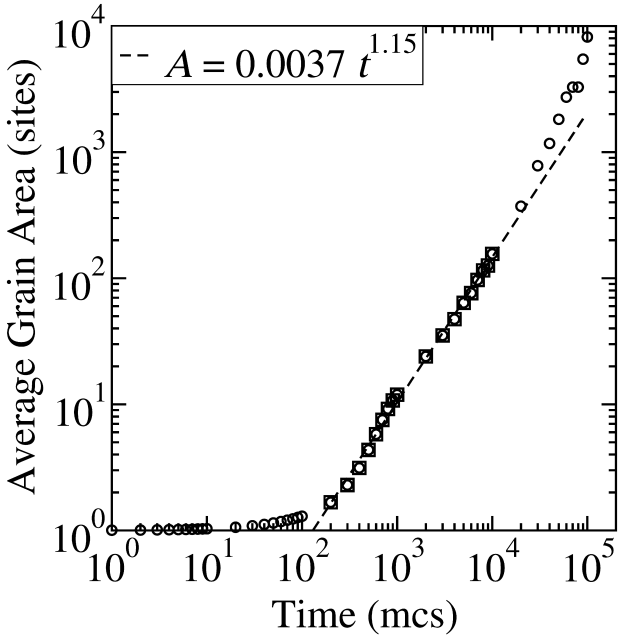


Figure 7: *Average grain area as a function of Monte Carlo time for the grain growth example in Fig 6. The dotted line is a function fit to the data in the range* t *= 200 - 10,000 mcs.*

clustering into islands to reduce the system's energy by maximizing the number of solid-solid bonds. The islands' aspect ratios are particularly high because an adatom's bond to another adatom is stronger than that to the substrate, such that adatoms are driven to mound (i.e. Stranski-Krastanov-like growth) rather then wet the substrate (i.e. layer-by-layer growth). Eventually the substrate becomes completely covered by adatoms, and subsequent adatom deposition and diffusion occurs on the existing and growing film. A signature of the adatom island morphology is retained during the early stages of the deposition process. The island evolution is plotted in Fig 9, which demonstrates that the island count increases quickly during the very earliest stages of deposition, and then peaks and decreases as the islands coalesce, until finally only one "island" (i.e. the continuous film) remains.

There are a number of obvious extensions to this example that could alter the growth behavior in a potentially favorable manner, e.g. implementing a more meaningfule atomic lattice, or enumerating multiple species of adatoms (as in the grain growth example) to model the deposition of a polycrystalline thin film. In the next section, these extensions and several others are applied to this basic film growth example in order to address a technologically relevant problem.

2.5 Application: Grain Refinement During Thin Film Deposition

More realistic applications of the KMC method are often much more complex than the examples provided above, largely because the fundamental mechanisms that conspire to produce technologically relevant processes are more varied and elaborate. For example, when electrodepositing metal films, it is not uncommon to introduce various chemical agents that modify the film's microstructure. In particular, because unaided electrodeposition often produces films with relatively large, columnar grains, so-called "grain refiners" are used to encourage the nucleation of new crystals and thereby produce smaller, more equiaxed grains [13]. In the present application, this process is modeled phenomenologically by considering thin film deposition

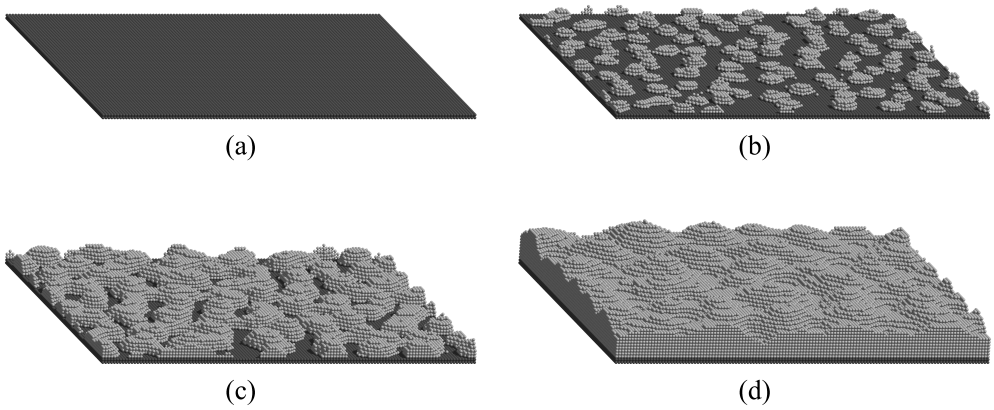


Figure 8: *Heteroepitaxy of adatoms, where the adatom-adatom bond is twice as strong as the adatom-substrate bond. Light gray spheres are adatoms, and dark gray spheres are substrate atoms.*

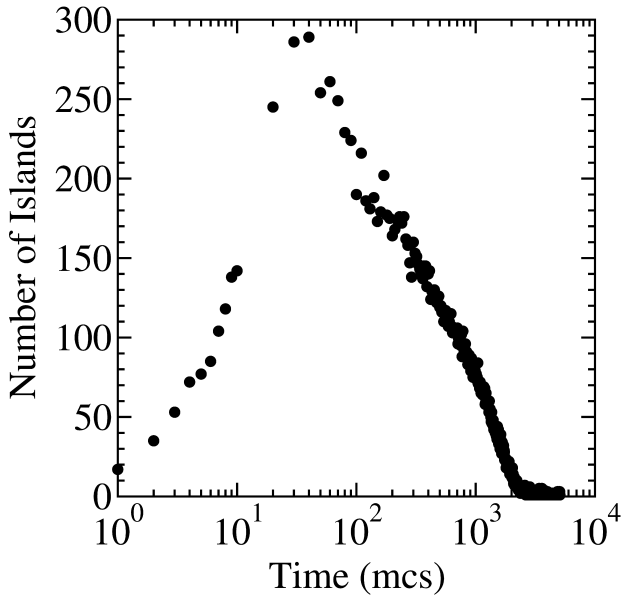


Figure 9: *The number of islands as a function of time for the film deposition example in Fig 8.*

(though not specifically electrodeposition) both with and without the codeposition of an agent that promotes the nucleation of new grains (i.e. secondary nucleation). The treatment of this agent, and of polycrystalline deposition in general, adds substantially more complexity to the idealized example provided above.

As in the simple example of film deposition described above, the computational domain in this case consists of a lattice of atomic sites that can assume one of several species: empty (i.e. vacuum), substrate, adatom, or grain refiner. However, in the present application the sites conform to a face-centered cubic lattice, rather than simple cubic, to more closely represent commonly-used metals like copper and nickel. Adatoms and grain refiners can deposit into any empty site that has at least two solid (i.e. substrate, adatom, or grain refiner) nearest-neighbors. Similarly, adatoms can diffuse into any empty site that has at least two solid nearest-neighbors. (Grain refiners are assumed to be immobile.) When a grain refiner deposits into

an empty site on the growing surface, it occupies the sites itself, as well as all the empty nearest-neighbors of the site. When an adatom deposits or diffuses, it is assigned a spin in the range 1 to 2,048 (as in the grain growth example), as follows. If the adatom deposits into a lattice site with no other adatom nearest-neighbors, it is assigned a random spin. If it diffuses into a lattice site with no other adatom or grain refiner nearest-neighbors, its spin remains unchanged. If it deposits or diffuses into a lattice site that has at least one other adatom nearest-neighbor (but no grain refiners see below), it is assigned the spin owned by the largest number of its nearest-neighbors. (In case of a tie, one the winners is chosen at random.) When an adatom diffuses into an empty site with at least one grain refiner nearest-neighbor, the adatom is assigned a random spin to represent the nucleation of a grain, and the new spin is retained until the adatom diffuses away from the refiner.

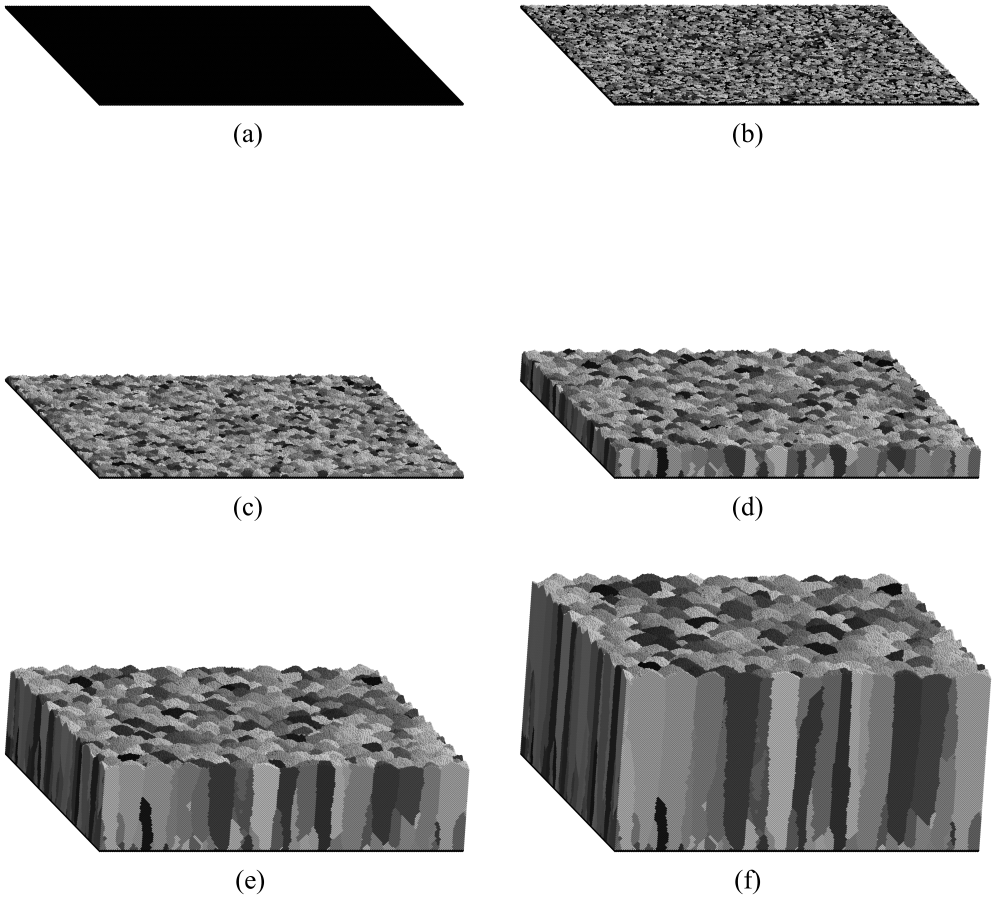


Figure 10: *Deposition of a polycrystalline film without a grain-refining agent, at simulation times of a) 0 sec, b) 50 sec, c) 300 sec, d) 2,600 sec, e) 7,800 sec, and f) 16,250 sec. Different gray levels represent different grains.*

Consider the FCC substrate in Fig 10a. Assume that $kT = 0.25eV$, the rate of adatom deposition is 0.01 1/sec, the attempt frequency of adatom diffusion is 10 1/sec, the bond energy between any two atoms is -0.8 eV, and the activation energy for adatom diffusion is 0.6 eV. (These parameters are based on data for nickel depositing onto copper, despite the rather high temperature.) If no grain refiners are introduced (i.e. their deposition flux is zero), then any and all grains in the film must be attached to the substrate, as depicted in Fig 10 which shows the deposition of approximately 16,0000,000 atoms. If, on the other hand, the rate of

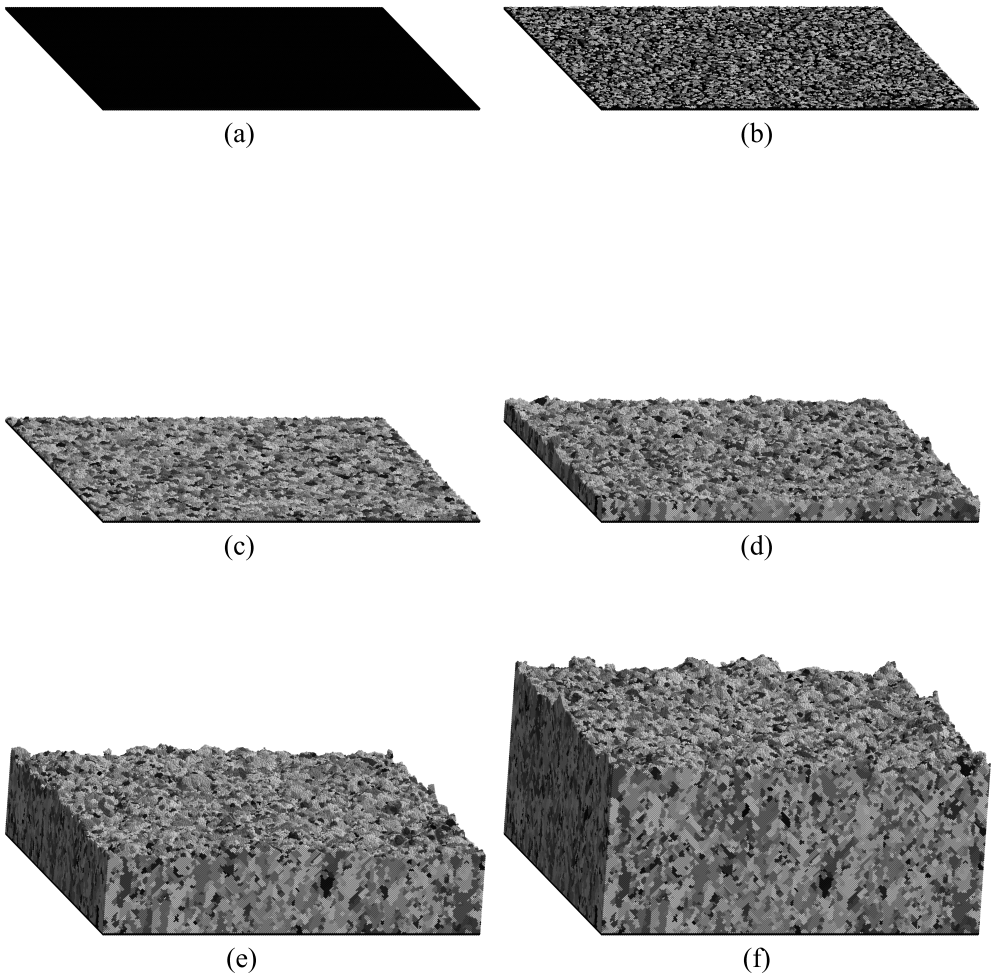


Figure 11: *Deposition of a polycrystalline film with a grain-refining agent, at simulation times of a) 0 sec, b) 30 sec, c) 150 sec, d) 1,500 sec, e) 6,000 sec, and f) 12,000 sec. Different gray levels represent different grains.*

grain refiner deposition is assumed to be 0.0001 1/sec, the film's microstructure develops as depicted in Fig 11. Clearly the relatively low-flux deposition of grain refiners servers to decrease the average grain size and to promote equiaxed, rather then columnar, grain structures. This same effect is observed experiments [13] of LIGA Ni electrpdeposition in nickel-sulfamate (without grain refiners) and Watts (with grain refiners) baths. Furthermore, these results suggest that grain-refining agents serve to promote secondary nucleation of new grains by attaching to the growing film and influencing the crystallographic orientation of material that deposits near them, and that this is a viable pathway to achieving the grain refinement observed experimentally [13] in electrodeposited Ni films.

2.6 Conclusion

The Kinetic Monte Carlo (KMC) method models the temporal evolution of a system by stochastically exploring sequences of fundamental transitions. The length and time scales accessible to the KMC method are limited only by the nature of these transitions. The basic approach is relatively simple to implement, yet

very complex problems can be addressed if the implementation is informed by the appropriate phenomenology. In addition, the method is amenable to a wide range of applications, including diffusion, microstructure evolution, and deposition. tion.

3 SPPARKS Simulator

As part of this project, a new parallel Monte Carlo code, called SPPARKS [75], for Stochastic Parallel PARticle Kinetic Simulator, was developed. It is an open-source code, freely available for download from the SPPARKS WWW site at www.sandia.gov/∼sjplimp/spparks.html. The site also contains documentation, example test problems, and pictures and movies of simulation output. SPPARKS is distributed under the terms of the GNU Public License (GPL), which means the code can be used or modified however a user wishes. The only restriction imposed by the GPL is on further distribution of the code; it must remain open source.

SPPARKS is written in C++. It can be run on single-processor desktop or laptop machine, but for many applications, can also be run in parallel. It will run on any parallel machine that compiles C++ and supports the MPI message-passing library. This includes distributed- or shared-memory machines.

SPPARKS was written as a "framework", with the goal of enabling new Monte Carlo models to be rapidly implemented. Key elements of a new model, including a definition of "events", their associated probabilities, and how an event changes the state of the system, must be defined and written as C++ code by the user. Other aspects of the simulation are handled by the framework. These include event selection according to kinetic Monte Carlo (KMC) or equilibrium (Metropolis) Monte Carlo rules, partitioning the problem across multiple processors, communicating information between processors, and output of snapshots of the system. The hope was that this code structure allows a user to focus solely on the unique attributes that define their model, and ignore other issues that are common to all models. It does mean, however, that the user must write new code to create a new "application". Unlike, for example, molecular dynamics simulators where the collection of commonly-used modeling options is well-established and a package such as LAMMPS [44] can implement them for the user, each user's Monte Carlo model typically has new features and thus requires new code. The mechanisms for adding a new application to SPPARKS are discussed below.

3.1 Options

SPPARKS defines 3 kinds of Monte Carlo models which can be implemented as "applications". The first two are motivated by materials modeling problems where events occur on a collection of spatial "sites" where the sites can be stationary points on a lattice (an on-lattice model) or particles which move freely in space (an off-lattice model). An example of an on-lattice application is Potts model grain growth. An example of an off-lattice application is atomistic energy minimization via Metropolis MC. The third kind of application is a general non-spatial model. SPPARKS currently has several on-lattice models, as described in Section 7. It has two off-lattice models (for atomistic energy relaxation and surface growth), and also two general models (for biochemical equations). The general models are included for completeness and as test drivers for KMC solvers, but cannot be run in parallel, since they have no spatial component.

For on-lattice applications, SPPARKS provides a variety of 2d and 3d lattice styles (square, triangular, cubic, fcc, bcc, diamond, etc), including a "random" style where lattice points are created at random locations in space and site-to-site connections are created within a user-specified cutoff distance. Arbitrary lattice topologies can also be read in from a file.

On-lattice or off-lattice spatial applications can be run in one of three Monte Carlo modes, assuming the application provides low-level methods that enable the particular mode; see Section 3.3 for more details.

General non-spatial applications can only be run in the first of these Monte Carlo modes. See reference [16] for an excellent description of these various MC variants.

The first mode is true kinetic Monte Carlo (KMC), also called non-equilibrium MC or rejection-free KMC or the N-fold way or the Gillespie Stochastic Simulation Algorithm. Each site defines zero or more events it can perform and associated probabilities. These can often be thought of as rates, associated with the crossing of an energy barrier. The set of probabilities is stored within SPPARKS by a KMC "solver" which selects the next event according to the KMC algorithm described in the papers of Sections 2 and 4. SPPARKS implements all 3 of the KMC solvers described in the $J.Chem.Phys.$ paper of Section 4: a linear-time, log-time, and constant-time solver. The cost of choosing the next event from N possible events, where N in this case is the number of sites, is $O(N)$, $O(logN)$, and $O(1)$ respectively, for the 3 solvers. After the event is selected, the application performs the event, and updates all affected probabilities, so that the next event can be selected properly.

The second mode is rejection KMC (rKMC), also called null-event MC or non-equilibrium MC. As with true KMC, each site defines a set of events if can perform with associated probabilities. It also defines a null-event with a probability that brings the summed probability of events for that site to a value that is the same for all sites. The event is "null" because if it is selected, nothing happens.

The advantage of rejection KMC over true KMC is simplicity. No list of probabilities for all sites need be maintained and no KMC solver is required to select events from a global list. Instead, a site is chosen randomly, and a second random number is used to select an event for that site, which may be the null-event. Once the event is performed, there is no need to update the probabilities of all affected sites. That calculation can be performed on a site-by-site basis once a site is selected.

The disadvantage of rejection KMC is that the aggregate probability of the null events for all sites may be large, and thus there can be a high probability of no event occurring at each iteration of the algorithm, which wastes CPU time. In particular, even if there are only a handful of high-rate events in the model, all sites must define a large-probability null-event to insure the per-site probability is equal for every site, which exacerbates the inefficiency. The trade-off between these effects and thus the relative speed of these two MC modes is model-dependent. An application may choose to implement one or both modes.

Both the true and rejection KMC modes track the dynamic evolution of the system in a time-accurate manner. This is because probabilities (rates) are computed and the event selection algorithm calculates a timestep for each event's occurrence.

A third Monte Carlo mode can also be implemented by an application, which is Metropolis Monte Carlo (MMC), also called equilibrium MC or barrier-free MC. As with rejection KMC, a site is chosen randomly as is a possible event for that site. No rate is assigned to the event; instead the energy change it induces in the system is computed and the event is accepted or rejected via the usual temperature-dependent Boltzmann factor.

The advantage of the Metropolis approach is that it offers great flexibility in defining and choosing events, since there is no requirement to compute an event's probability or to insure that the relative probability of two different attempted events occurring is correct. Moreover, unphysical events, such as swapping the species of two adjacent atoms or large-scale conformational changes or particle deletion/creation, can be performed. Likewise, the relative frequencies for considering different kinds of prospective events can be altered at will, so long as the constraint of "detailed balance" is observed, meaning that a) events are reversible and b) for any two states A and B joined by an event, the product of population A times the A-to-B event frequency should equal the product of population B times the B-to-A event frequency.

The disadvantage of Metropolis MC is that there is no accurate "time" associated with events, since rates are what infer time dependence. Instead the Metropolis algorithm will evolve the system from the initial state to a stationary distribution of states, corresponding to thermodynamic equilibrium at temperature T,

pressure P, etc. Often this distribution of states will be clustered around a local or global potential energy minimum.

3.2 Parallelism

A parallel KMC algorithm (or rKMC or MMC algorithm) allows for events to be performed simultaneously on multiple processors. As discussed in the following Section 5, strictly speaking, KMC is inherently a serial algorithm, in that selection of a single event depends, in principle, on the current set of probabilities for all events. Thus to enable parallelism, some approximation must be made. In SPPARKS, applications with spatial dependence (on-lattice or off-lattice) allow for this approximation. The physical justification is that an event in one location has minimal dependence on the state of the system far away, thus events can be decoupled spatially.

SPPARKS exploits this idea in two steps. First, the spatial domain is partitioned across processors, so that each processor owns the sites within its sub-domain and a small shell of surrounding ghost sites. Figure 12 illustrates this idea for a 2d simulation domain, split across a 5x4 grid of 20 processors. The dark lines represent processor boundaries.

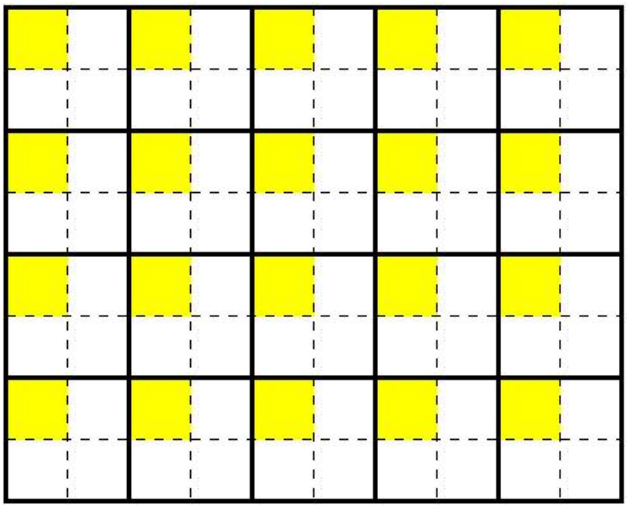


Figure 12: *2d domain partitioned for 20 processors (5x4) into sub-domains indicated by solid lines. Each processor sub-domain is further sub-divided into sectors (4 quadrants in 2d, 8 octants in 3d). Each processor can then perform events in its yellow sector without conflicting with other processors.*

Note, however, that a processor still cannot perform events independently within its sub-domain without coordinating with other processors. If two processors simultaneously performed events near a shared boundary they could execute in a conflicting manner or with incorrect probabilities. E.g. two atoms could hop to the same vacant site. The second step is to order events within a processor to avoid this conflict.

One method for doing this, only relevant for on-lattice applications and for the rejection KMC mode described above, is to "color" the lattice in a checkerboard fashion across the entire domain. The coloring is done in a manner that insures events on sites of the same color can be executed simultaneously without conflict. The number of required colors depends on the lattice and the application. When this option is selected in SPPARKS, a sweep over lattice sites consists of multiple stages, one per color. In between stages, processors must communicate boundary site information so that neighboring processors can update their ghost sites.

A second, more general method for ordering events within a processor, is to sub-divide the sub-domain into "sectors", typically 4 quadrants in 2d or 8 octants in 3d. The yellow squares in Figure 12 are sectors within each processor's sub-domain. The idea is that each processor can perform multiple events within its sector, without conflicting with events on other processors.

SPPARKS has several options for how to order events within a sector. For true KMC mode (see above), a KMC solver is created for each sector. All events within the sector are stored by the solver and selected one after the other via one of the KMC algorithms of Section 4. For rejection KMC or Metropolis mode, sites can be selected in random or consecutive order; see the SPPARKS "sweep" command. Coloring can also be applied within a sector to avoid dependency effects due to performing events sequentially on adjacent sites.

When using sectors, SPPARKS performs an outer loop over sectors, and an inner loop over events within the sector. When moving to the next sector, inter-processor communication must be performed to update boundary sites surrounding the sector. The amount of required information is smaller than in the coloring strategy discussed above, where a processor updated all its ghost sites. The pattern of communication is illustrated in Figure 13. The left side is one processor's portion of Figure 12. It needs to update sites in the dotted boundary region before performing events in the yellow quadrant. Each processor does this by sending its own sites to 3 neighboring processors (7 in 3d) and receiving ghost sites from a different 3 processors, as shown on the right side. The unfilled sites within the dotted square are the sector boundary sites which are already owned by the processor; they do not need to be communicated.

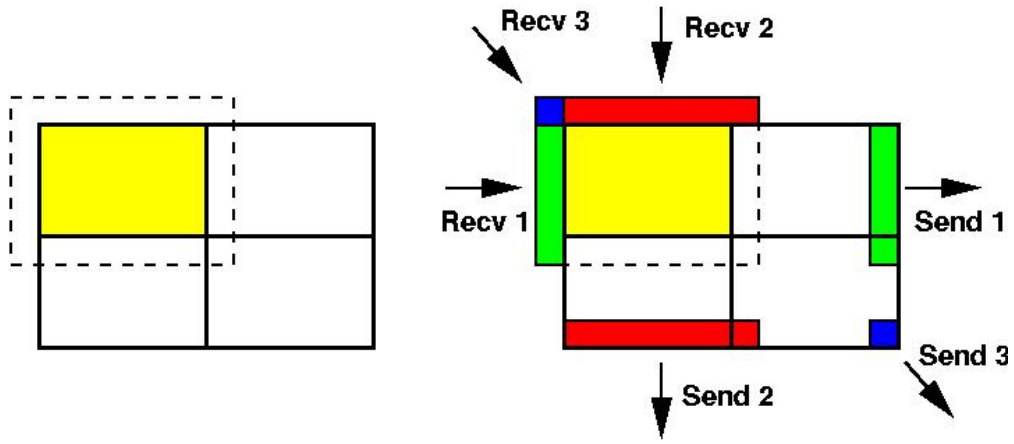


Figure 13: *(Left) One processor's 2d sub-domain with four quadrants. The yellow quadrant has a surrounding dotted box of sites it needs, some of them ghost sites owned by other processors. (Right) Sites the processor will send and receive in 3 exchanges (red, green, blue) in order to update the ghost sites surrounding its yellow quadrant.*

Note that while SPPARKS is performing events within a sector, the boundary region is effectively "frozen"; the state of those sites does not change. This is one source of the approximation error being made to enable parallelism. Additional sources of error are discussed in Section 5.

A related question is how many events should be computed in one sector, before moving to a new sector. SPPARKS controls this by a time threshold. The KMC clock advances, event by event. Once the threshold is reached, a new sector is selected, communication is performed, and the outer loop continues. The appropriate choice of time threshold is a trade-off between accuracy and parallel performance. The smaller the threshold, the less error accumulates due to frozen boundary sites. But more communication must be performed for the same overall simulation time. Conversely, the larger the threshold, the larger the error, but less communication is needed and thus parallel scalability is enhanced.

Section 5 discusses an automated strategy, implemented in SPPARKS, for adjusting this time threshold on-the-fly as a simulation runs, to balance these competing issues. Resulting parallel performance is also given for a large grain-growth Potts model.

Finally, we note that SPPARKS supports a second form of parallelism. The set of P allocated processors allocated can be used to perform a single simulation. But they can also be partitioned into M subsets, where each subset has P/M or any number of processors, so long as the total processor count sums to P. Each subset can then run an independent simulation simultaneously. This is managed by the input script, in which variables can be defined that assign different parameters to different simulations, or loop over a large set of input files. For example, with $M = 10$ subsets, a run could be launched to perform 500 simulations. Each of the 10 subsets starts a simulation. Whichever finishes first launches the 11th simulation, etc, until all 500 are finished. This is a useful technique for performing many independent runs to generate good statistics or to search a large parameter space.

3.3 Implementing New Models

SPPARKS is designed to allow new Monte Carlo models to the code. This is done by writing a C++ class that is derived from one of 3 parent classes: a general App class, an on-lattice AppLattice class, or an off-lattice AppOffLattice class. In each case the new child class inherits considerable functionality from the parent class. It needs to provide a set of methods that define events and their associated probabilities. Details are given in the on-line SPPARKS documentation. Here we illustrate with the list of methods needed for on-lattice applications:

- site_energy()
- site_event_rejection()
- site_propensity()
- site_event()
- input_app()

Only the first method is required. It computes the energy associated with a site. For on-lattice applications this is typically the Hamiltonian that describes the model. The site_event_rejection() method is required if rejection KMC or Metropolis MC is to be performed. The site_propensity() and site_event() methods are required if true KMC is to be performed. The former computes the probability for all events a site can perform. The latter performs an event and updates the propensity of all affected sites. All of these methods are invoked by the top-level iterators and solvers in SPPARKS when a simulation executes in a chosen mode (KMC, rKMC, MMC) with a given parallel strategy (coloring, sectoring) and/or sweeping option (random, raster, etc).

The last input_app() method is optional. It allows an application to define extra application-specific input commands that can be used in a SPPARKS input script. There are additional optional methods that can be defined to invoke operations needed when the application is created or at the beginning stages of a simulation.

Building SPPARKS with a new application is particularly simple. The user has created new "app_foo.cpp" and "app_foo.h" files that implement the new "Foo" application. Simply placing these files in the SPPARKS src directory, and typing "make machine" will include them in a new SPPARKS executable. This includes the ability to invoke the new application from an input script via a command like

```
app_style foo option1 option2 ...
```

Other new features can be added to SPPARKS in a similar fashion, by writing a child class that derives from a provided parent class and defines the appropriate methods. This includes "diagnostic" classes that allow add-on computations to be invoked during a simulation (e.g. for statistics gathering), "solver" classes that define KMC solvers, "pair" classes that define pairwise interatomic potentials for use with off-lattice models, and "command" classes that define new input script commands.

4 Solvers

The heart of the kinetic Monte Carlo method is the algorithm which selects each event from a list of events, based on their relative probabilities. The SPPARKS code implements 3 such algorithms, all of which are described in the following paper, which is included here in its entirety. This paper appeared in the "Journal of Chemical Physics", and was titled "A Constant-Time Kinetic Monte Carlo Algorithm for Simulation of Large Biochemical Reaction Networks". It's authors were Alexander Slepoy, Aidan P. Thompson, and Steven J. Plimpton. It was published in vol 128, page 205101, in 2008.

Although this article discusses the group-based algorithm in the context of biological networks, it can also be applied to materials modeling problems, such as those discussed in this report.

4.1 Abstract

The time evolution of species concentrations in biochemical reaction networks is often modeled using the stochastic simulation algorithm (SSA) [D. Gillespie, J. Phys. Chem, **81**, 2340 (1977)]. The computational cost of the original SSA scaled linearly with the number of reactions in the network. Gibson and Bruck developed a logarithmic scaling version of the SSA which uses a priority queue or binary tree for more efficient reaction selection. [M. Gibson and J. Bruck, J. Phys. Chem. A **104**, 1876 (2000)] More generally, this problem is one of dynamic discrete random variate generation which finds many uses in Kinetic Monte Carlo (KMC) and discrete event simulation. We present here a constant-time algorithm, whose cost is independent of the number of reactions, enabled by a slightly more complex underlying data structure. While applicable to kinetic Monte Carlo simulations in general, we describe the algorithm in the context of biochemical simulations and demonstrate its competitive performance on small- and medium-size networks, as well as its superior constant-time performance on very large networks, which are becoming necessary to represent the increasing complexity of biochemical data for pathways that mediate cell function.

4.2 Introduction

The metabolic, regulatory, and signaling pathways in biological cells are often represented by biochemical networks involving reactions between proteins, genes, and other molecular species. The response of such networks to perturbation is a ubiquitous modeling problem in computational biology. Simulations of the response track the time-dependent concentrations of individual species. Continuum versions of the models can be formulated as sets of coupled ordinary differential equations and integrated by standard methods. In cases where the concentrations of some species are small, stochastic effects impact the behavior of the system [1].

In 1976, Gillespie developed the stochastic simulation algorithm (SSA) to model these networks via Monte Carlo methods, in a way that correctly samples the dynamic probability distribution of possible reactions [19,29]. Derived from the chemical master equation, the algorithm evolves the system one reaction at a time, choosing the specific reaction to perform, advancing time by an appropriate interval, and updating

the probability distribution of future reactions to reflect the outcome of the selected reaction. The method is widely used to model biochemical networks (the original two papers have been cited over 1500 times), and to analyze the effects of stochasticity within the small reaction volumes of cells [11].

With rapid growth in experimental data characterizing biochemical interactions, researchers simulate ever-larger reaction networks [1, 8, 18, 23, 54, 64, 65, 68, 80, 87], where a network represents the interactions between biochemically reactive molecular species in and around the cell. The nodes of the network are biochemical species with concentrations; the edges are the relationships implied by the reactions, with edge weights corresponding to reaction rates. Currently, the largest protein interaction network [2] known to the authors contains 18,000 proteins and 44,000 interactions.

Note that each bound state of two or more molecules is typically counted as a separate "species" in formulations of these networks, though alternate methodologies have been proposed as discussed below [24, 50]. Thus, if the combinatorial richness of protein complex formation is included, network sizes can grow exponentially, since even a relatively small number of proteins which bind together can exhibit great variety in complexation and post-translational modification, with the enumeration of possible states for a single receptor complex reaching 10^6 to 10^8 states [37]. Estimates of average connectivity in such networks are as high as 38 interactions per protein [88]. The kinetic rate constants associated with biochemical reactions span many orders of magnitude. For example, fast enzymes operate at $\sim 10^5/sec$, while slow enzymes operate at $2/sec$ or slower [3, 48]. These characteristics define the size and other properties of the network which produces a probability distribution of reactions that must be sampled by the SSA in order to accurately model biochemical network dynamics.

The computational cost of the original SSA to perform a single reaction scaled as $O(N)$, i. e. linearly in N, the number of reactions in the network. Since the time increment per reaction also tends to shrink with increasing N, such scaling limited the size of networks that could be efficiently simulated. Gibson and Bruck [28] proposed an alternate implementation of the SSA which scales as $O(\log_2 N)$, enabling much larger networks to be modeled. Other optimizations have also been proposed, including pre-ranking the list of reaction probabilities from large to small [15], which allows a sequential search to outperform a logarithmic search for some probability distributions and small networks.

A variety of enhancements to the SSA have also been developed to enable its use for different problems. For example, much work has been done [14, 31, 32, 66, 67, 86] to address the issue of stiffness of the dynamical systems embodied in biological networks, and extend the timescale over which the SSA can be used, while bounding the errors induced by multi-timescale approximations. To address the issue of exponential growth in the number of possible reactions due to protein complexation, rule-based approaches have been developed [24, 50], which limit the number of reactions by generating new ones only as needed during a simulation as specific reactants are produced. Properties of the new reaction (e.g. its rate constant) are inferred from properties of its reactants, limiting the amount of information that must be stored. Depending on how reactions are selected, the computational cost in these approaches can depend on the number of molecules currently present in the simulation, rather than the number of possible reactions [20, 90]. Computationally, this can be a win when the number of molecules is smaller than a very large list of possible reactions.

In this paper, we do not address these enhancements directly. Rather we note that they are all built on top of some version of the SSA at their core. Thus improving the scalability of the SSA itself could benefit any of these approaches.

All of the algorithms discussed thus far solve the generalized problem of random variate generation (RVG) from a dynamic discrete probability distribution. The generated variate determines what "event" takes place in the next time increment. "Dynamic" means the distribution changes each time an event occurs. For biochemical networks, the "event" is a reaction, and the system is dynamic because the occurrence of a reaction changes the concentration of various species (its reactants and products) and hence the probabilities

for other reactions to occur at the next iteration. In this more general context, RVG is a well-studied problem. Devroye [21] provides a classification scheme for RVG and describes a rich compendium of algorithms, including all of the event-selection algorithms commonly used in SSA implementations. Efficient RVG is also a key kernel in discrete-event and kinetic Monte Carlo (KMC) simulators which model phenomena as diverse as factory scheduling (operations research) or grain growth and chemical vapor deposition (materials science). It is worth noting that the classic KMC algorithm for choosing events and the associated timestep, known as the n-fold way or BKL algorithm [10], is in fact equivalent to the SSA, though it was formulated independently.

In this paper, we adapt a particular RVG algorithm known as Composition and Rejection, which has been developed and enhanced in [26, 34, 63], and apply it to the SSA. It is well-suited to the simulation of large biochemical networks, because its scaling is $O(1)$, i. e. the computational cost to perform a reaction is constant, *independent* of N. This surprising result requires only two assumptions be met, both of which we argue in later sections, are reasonable for biochemical networks. The first is that the ratio of maximum to minimum probability for any two reactions is bounded. The second is that the average number of other reactions directly coupled to each reaction (products of one are reactants of others) does not grow continuously as large numbers of new reactions are added to the network.

In the remainder of the paper, we briefly describe the original SSA, the widely-used Gibson/Bruck enhanced algorithm, the new constant-time algorithm and its implementation details. We compare the computational cost of the logarithmic and constant-time algorithms on a synthetic test suite, showing the new algorithm to be competitive even for small networks, and to perform significantly faster as the network grows to large numbers of reactions.

4.3 Linear Time Algorithm

Consider a collection of molecules of different chemical species in a volume V. The initial count of molecules of each species i is n_i, so that molar concentrations are $c_i = n_i/(N_A V)$, where N_A is Avogadro's number. The species interact via a set of N chemical reactions. The *nth* reaction can be written in familiar form as

$$\sum_{i=1}^{r_n} R_i^n \xrightarrow{k_n} \sum_{i=1}^{p_n} P_i^n$$

where k_n is a reaction rate constant, R_i^n is a reactant molecule of a particular chemical species, and similarly for product molecules P_i^n. The number of reactants r_n on the left side of the equation can be limited to 0, 1, or 2 without loss of generality, while the number of products p_n can be 0, 1, or any number. The N reactions are "coupled" in the sense that products of each reaction can be reactants of others. The computational task is to evolve the species concentrations c_i over time, assuming the volume is a well-stirred reaction chamber, where each molecule is equally likely to encounter any other molecule.

The continuum formulation of this problem converts reactions to ordinary differential equations (ODEs) and integrate the set of coupled ODEs forward in time, where continuous concentrations are the variables of interest. As an alternative, Gillespie proposed the SSA, which treats individual molecules discretely, and showed it was rigorously equivalent to simulating the tome evolution of the chemical master equation formulated for the system of reacting molecules and, in the limit of large numbers of molecules, to the continuum formulation as well [19, 29].

In the SSA, the system evolves one reaction at a time, changing the counts of reactant and product molecules appropriately, and thus the associated species concentrations. Which reaction occurs next, and the time at which it occurs, are chosen using random numbers and probabilistic rules that ensure accurate

sampling. In the SSA, a "propensity" is computed for each reaction which is proportional to its probability of occurrence relative to other reactions. The propensity is $k_n N_A V$ for a zeroth-order reaction, $n_1 k_n$ for a first-order, and $n_1 n_2 k_n/(N_A V)$ for a second-order reaction, where n_1 and n_2 are the molecular counts of the reacting species, and the input k_n values have units of molarity/sec, 1/sec, and 1/molarity-sec for zeroeth-, first-, and second-order reactions respectively.

With these definitions, the "direct" SSA is outlined in Figure 14. A "first-reaction" version of the SSA was also discussed by Gillespie and shown to be exactly equivalent. Its scaling properties are the same as discussed here. Note that as presented in Figure 14, the following values must be pre-computed before the first iteration: p_i = the propensity for each reaction and p_s = the sum of p_i for all N reactions.

(1) Generate two random numbers r_1 and r_2
(2) $\Delta t = \frac{1}{p_s} \ln(\frac{1}{r_1})$
(3) Find the smallest m such that $r_2 p_s < \sum_{i=1}^{m} p_i$
(4) Perform *mth* reaction, incrementing reactant/product counts
(5) Compute propensity p_i of each reaction
(6) $p_s = \sum_{i=1}^{N} p_i$

Figure 14: *A single iteration of the original SSA, the Gillespie stochastic simulation algorithm, with $O(N)$ scaling in the number of reactions N.*

Two random numbers are used per iteration, each sampled from a uniform distribution bounded by 0 and 1. The first is used to compute a time increment in step (2). The second is used to pick a reaction in step (3). Conceptually, step (3) can be thought of in the following way. Each propensity represents a short line segment of length equal to p_i. If these segments are concatenated, the resulting long segment has length p_s. If a random point along this long segment is chosen, step (3) determines which short segment the point falls inside of. Step (4) updates molecular counts due to reaction m. Steps (5) and (6) compute new propensities resulting from changed molecular counts, in preparation for the next iteration.

As originally proposed by Gillespie, step (3) scales as $O(N)$ with the number of reactions N, using the following approach. Sum the N propensities in order from 1 to N, adding each in turn, continuing until the the p_m term causes the accumulating sum to exceed $r_2 p_s$. Step (4) scales as $O(1)$ since we assume each reaction has a small bounded number of products. As written, steps (5) and (6) also scale as $O(N)$.

In the nomenclature of random variate generation, step (3) "generates" a random variate from a dynamic discrete probability distribution (the set of propensities), and steps (5) and (6) "update" the distribution. Thus the scaling of the original Gillespie SSA is $O(N)$ (linear) in both its "generation" and "update" times. However, a simple enhancement improves the scaling of the "update" to $O(1)$ (constant). If we assume each chemical species occurs as a reactant in a small bounded number of reactions (a plausible assumption for biochemical networks), then the number of propensities that need to be updated in step (5) is also small and bounded, i. e. $O(1)$. The sum of step (6) can be similarly updated in $O(1)$ time, yielding an overall "update" scaling of $O(1)$.

The idea of only updating propensities for affected reactions was formalized as a "dependency graph" by Gibson and Bruck [28], though others may have implemented similarly efficient forms of steps (5) and (6) before this paper. Reactions are nodes of the graph representing the biochemical network and a directed edge from node i to j exists if a product of reaction i is a reactant of reaction j. Storing such a dependency graph enables a straightforward implementation of an $O(1)$ version of steps (5) and (6).

Another optimization of the direct method, proposed in Cao et. al. [15], is to pre-order the set of propensi-

ties so that large values appear at the beginning of the list. Then the accumulating sum in step (3) is likely to exceed $r_2 p_s$ quickly, yielding a small m. Cao et. al. argue this pre-ordering is feasible for some biochemical networks and can be re-computed periodically as concentrations change, leading to a faster algorithm.

We note that when N is small, the number of required updates (although fixed) can be close to N. Thus an algorithm whose update scaling is constant is a good choice even if its generation cost scales linearly. Hence the original SSA performed satisfactorily in a computational sense until the size of simulated biochemical networks grew larger. This motivated the algorithm of the next section.

4.4 Logarithmic Time Algorithm

The key advance of the Gibson/Bruck version of the SSA was to convert it from an algorithm with linear generation time and constant update time to one that is logarithmic in both generation and update time, thus enabling large networks to be simulated more efficiently [28]. The paper mainly focused on enhancements to the first-reaction version of the SSA yielding a "next-reaction" method, but enhancements to the direct SSA were also proposed. The resulting scaling is the same for both algorithms; we discuss the enhanced direct SSA here, which we refer to as SSA-GB.

The SSA-GB algorithm is outlined in Figure 15. It has the same sequence of steps as in Figure 14. Steps (3) and (6) now use a binary tree so that random variate generation and the update of the distribution scale more efficiently. Note that step (5) is now the $O(1)$ update discussed in the previous section using a dependency graph, as suggested by Gibson and Bruck.

(1) Generate two random numbers r_1 and r_2
(2) $\Delta t = \frac{1}{p_s} \ln(\frac{1}{r_1})$
(3) Search binary tree for smallest m such that $r_2 p_s < \sum_{i=1}^{m} p_i$
(4) Perform *mth* reaction, incrementing reactant/product counts
(5) Compute propensity p_i of affected reactions
(6) Percolate changed propensities up binary tree, yielding $p_s = \sum_{i=1}^{N} p_i$

Figure 15: *A single iteration of SSA-GB, the Gibson/Bruck stochastic simulation algorithm, with $O(log_2 N)$ scaling in the number of reactions N.*

A binary tree is used to store the set of N reaction propensities, assuming N is a power of 2. Each propensity is a "leaf" in the tree. Pairs of propensities (siblings) are summed to a parent value, stored at a "branch" location in the tree. Pairs of parent values are summed iteratively at the next level (grandparents) until a single "root" value results which is p_s, the sum of all N propensities. The resulting tree has $\log_2 N$ levels. Since $N - 1$ partial sums are stored, the entire tree can be stored in $2N$ memory locations. Generalization to a tree where N is not a power-of-two is straightforward, e. g. by padding the list of leaves with zeroes.

Step (3) can now be performed in a logarithmic number of operations, yielding an algorithm whose generation time scales as $O(\log_2 N)$. Begin at the root of the tree with a search value $s = r_2 p_s$. If s is less than the left child value p_l, branch to the left keeping s as the search value, else branch to the right using a new search value $s = s - p_l$. This operation is applied successively at each of the $\log_2 N$ levels of the tree, until a specific leaf is arrived at. This is the mth leaf of step (3).

Similarly, in step (6), the effect of each changed propensity on the summed p_s can be computed in a logarithmic number of operations. First, the appropriate leaf value is changed, then its parent value (changed value + sibling value) is re-computed. Then the grandparent value is re-computed and so forth until the root

value is re-computed. This operation is performed once for each changed propensity (the number of which is small and bounded); thus, the overall scaling of the update operation is also now logarithmic, i. e. $O(\log_2 N)$.

The overall logarithmic scaling of the Gibson/Bruck enhanced SSA algorithms (both next-reaction and SSA-GB) results in a large performance improvement over the linear time method for networks with even a few dozen reactions. Hence these algorithms are currently widely used in many biochemical network modeling codes.

4.5 Constant Time Algorithm

All the algorithms of the preceding sections are discussed (from a RVG perspective) in Devroye's compendium [21]. Methods linear in generation time and constant in update time (the original Gillespie SSA) are presented in chapter 3.2.3 (Inversion by Sequential Search). The Cao et. al. optimization [15] is described in the same chapter (Inversion by Sequential Search with Reorganization). Methods logarithmic in both generation and update time (SSA-GB) are discussed in Chapter 3.3.2 (Inversion by Binary Search). For these methods, Huffman trees are proposed to further reduce the generation time, but this does not change the fundamental logarithmic scaling behavior.

For better performance on networks with a very large number of reactions N, we turn to a class of methods called Composition and Rejection (CR) algorithms (chapter 2.4) that are constant in both generation and update time, i. e. their $O(1)$ scaling is independent of the number of reactions.

The rejection idea is illustrated in the left panel of Figure 16. Consider a set of N reaction propensities, listed along the x-axis. The y-axis height of each bar represents the propensity for that reaction. If we draw a rectangle that bounds the N vertical bars, then a valid algorithm for randomly choosing a reaction is as follows. Let the height of the bounding rectangle be p_{max}. Pick a uniform random integer i from 1 to N. Pick a second uniform random number r from 0 to p_{max}. If $p_i < r$, then reaction i is selected. If not, the selection is "rejected" and the algorithm is repeated. Thus, in the figure, point A would be rejected, while point B would select reaction 8. Effectively, this algorithm iterates until a point inside one of the bars is selected, using two random numbers at each iteration. Note that the cost of selecting a test point is independent of N. If the set of bars covers a high fraction of the bounding rectangle's area, the average rejection count per selection will also be small.

Now imagine the set of N propensities are first grouped by their propensity values, as illustrated in the right panel of Figure 16. In this case, 3 groups from p_{min} to p_{max} are used. The first group (reactions 5,6) contains propensities ranging from p_{min} to $2p_{min}$, the second group (1,3,7,9) from $2p_{min}$ to $4p_{min}$, and the third group (2,4,8) from $4p_{min}$ to p_{max}. The selection of a reaction can now be done efficiently via an algorithm composed of 2 stages (hence the "composition" aspect of the CR algorithm). The first stage selects a group. Let G be the number of groups. If the total propensity of all reactions in a group is p_g and the total propensity of all reactions is $p_s = \sum_{g=1}^{G} p_g$, then this requires one random number and a linear scan or binary search of the G values, as discussed in the preceding two sections. Once a group is selected, the reaction within the group is chosen via the rejection procedure, using a rectangle that bounds only the reactions in that group, as illustrated for the second group in the figure. The key point is that by choosing the group boundaries as cascading factors of 2 (p_{min}, $2 * p_{min}$, $4 * p_{min}$, etc), we have guaranteed the area covered by the bars of each group's reactions is greater than half the area of the group's bounding rectangle. Thus, on average, selection of each reaction will require less than two iterations of the "rejection" portion of the CR algorithm.

The two stages together constitute the "generation" portion of the CR algorithm. Its scaling is $O(1)$, independent of N, if the number of groups G is also independent of N. We now argue why this is a valid assumption for biochemical networks. Clearly, for a set of reactions, there is a p_{min} which can be computed from the propensity formulas for the zeroeth-, first- and second-order reactions, assuming only one molecule

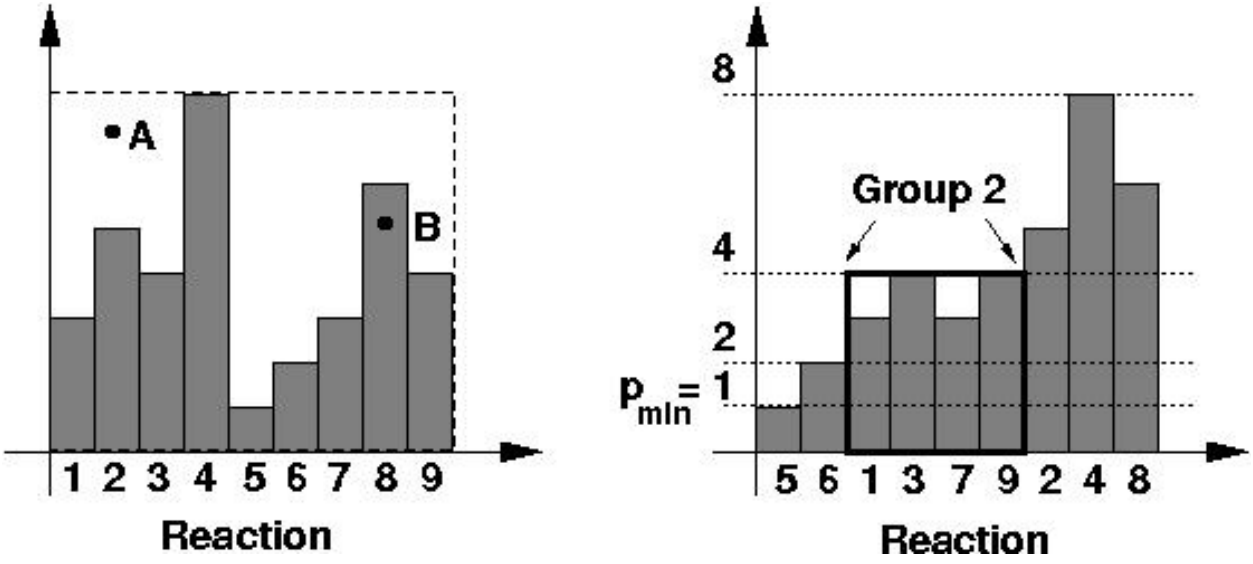


Figure 16: *Composition and Rejection algorithm for random variate generation. A reaction is selected from a set of reaction propensities (left) by picking random points (A,B) from a bounding rectangle until a point inside a vertical bar (B) is found. Grouping the propensities by their magnitude (right) makes rejected points less likely.*

of each reactant exists in the volume V. Note that reactions with propensity $p = 0$ need not be included as possible selections. Similarly, one can bound the maximum number of reactant molecules of any species by physical constraints or knowledge of the reaction network. For example, only so many molecules of a given species will be present in a cell. Thus the maximum propensity p_{max} for the set of reactions is also computable from the reaction rate constants k_i and the maximum molecular counts. The largest number of groups possible is then $G_{max} = \log_2(p_{max}/p_{min})$, though many fewer will likely be required when the SSA model executes. In a biochemical sense, adding new reactions to the model does not change G, assuming their rates are in the same range as those of previous reactions.

G can also be bounded by practical considerations. If p_{max} is computed, propensities below a chosen threshold value of p_{min} could be discarded, since statistically speaking, they will not occur frequently enough to impact the network dynamics. For example, if reactions with propensities a billion times smaller than p_{max} are discarded, then G is bounded at 30. Alternatively, p_{min} can be computed, and groups added on-the-fly as p_{max} grows during a simulation. If G grows too large, a larger p_{min} can be used for future simulations of the same network. In practice, G remains small (around 10-20) for networks we have modeled. We also note that even if G grows slowly with increasing N, this only affects the generation time for the algorithm, which scales as $\log_2(G)$ using a binary search of the group propensities. As shown in the next section, the generation cost is considerably smaller than the update cost, which is independent of the number of groups.

Using the CR algorithm for reaction selection leads to a constant time algorithm that is exactly equivalent to the original Gillespie SSA. We refer to this new algorithm as SSA-CR; it is outlined in Figure 17. Before the first iteration, reactions are assigned to groups, the summed propensities for each group p_g are computed, as is the total summed propensity $p_s = \sum_{g=1}^{G} p_g$.

Steps (3a) and (3b) are the reaction selection procedure outlined above. Step (3a) is the same as step (3) in either Figure 14 or 15, except that now the selection is from G groups instead of N reactions, thus

(1) Generate four random numbers r_1 thru r_4
(2) $\Delta t = \frac{1}{p_s} \ln(\frac{1}{r_1})$
(3a) Use r_2 to select a group of reactions (composition)
(3b) Use r_3 and r_4 to select reaction m within the group (rejection)
(4) Perform *mth* reaction, incrementing reactant/product counts
(5) Compute propensity p_i of affected reactions
(6) Assign affected reactions to new groups, yielding new p_g and $p_s = \sum_{i=1}^{G} p_g$

Figure 17: *A single iteration of SSA-CR, the Composition and Rejection stochastic simulation algorithm, with $O(1)$ scaling, independent of the number of reactions N.*

it scales as $O(G)$ or $O(\log_2 G)$. Step (3b) may require additional random numbers if rejection occurs, but this will happen less than half the time (on average), regardless of the distribution of reaction propensities. For randomly distributed propensity values, it will occur only one quarter of the time (75% area coverage by the vertical bars within a group). Note that a constant-time implementation of step (3b) requires that the *mth* reaction in a group can be accessed in a one-step operation. This is easily done by having each group maintain a linear list of its reactions, which can simply be integer indices from 1 to N.

Once a reaction has been performed in steps (4) and (5), the update portion of the SSA-CR algorithm is performed in step (6). The new propensity of each dependent reaction is compared to its old value. If the reaction stays in the same group, only the group sum p_g and total sum p_s need updating. If the group assignment has changed, the reaction is deleted from the old group and added to the new group, then p_g and p_s values for both groups are updated. Adding/deleting a reaction to/from a group is a constant-time operation. For addition, a new index is added to the end of the group list and the group size is incremented. For deletion, the reaction at the end of the group list replaces the deleted reaction and the group size is decremented. Thus the update portion of SSA-CR also scales as $O(1)$.

There are two requirements implicit in this scaling result. The first is that a dependent reaction can be located within its group in a one-step operation. This is easily done by having each of the N reactions store two integers: its current group assignment and its location within that group. The second is the assumption that the average number of dependencies per reaction does not grow continually larger as the number of reactions grows. Since extremely large networks have not been formulated, this assumption is hard to test empirically, but we note that any implementation of the SSA will suffer in performance if this is not the case, since the update time is necessarily proportional to the average number of dependencies.

Overall, in addition to propensities, the memory cost of the SSA-CR is three integers per reaction. This is similar to the SSA-GB memory cost for its binary tree of one additional floating point value per reaction.

Nothing in the preceding discussion requires that propensity boundaries between groups be chosen such that the ratio of upper and lower bounds for a group is $r = p_{upper}/p_{lower} = 2$. If $r < 2$, the cost of the rejection portion of the algorithm would decrease (less rejections) while the cost of the composition portion would increase (more groups). The converse would be true for $r > 2$. In either case, both portions would still be constant-time operations. A practical reason to use groups with $r = 2$, as illustrated in Figure 16, is that calculating which group a newly computed propensity value p falls into can be done in a single operation by calling a standard C math library function, namely frexp(p/p_{max},&gneg), which returns the negative of the group ID as the variable gneg.

4.6 Performance and Discussion

While the CR version of the Gillespie SSA has better theoretical performance than the Gibson/Bruck version, actual performance depends on pre-factors of the scaling terms and other implementation details. For SSA-CR, the pre-factors also depend on the number of groups G.

To test the algorithms for widely varying numbers of reactions N, we generated random reaction networks. We represented the network of reactions as an $1d$ array of N doubles representing reaction propensities and a $2d$ $N \times M$ array of connectivity for each of the propensities representing the dependency of the reactive species. For each of N reactions, initial propensities varying by a factor of a million (1.0e-6 to 1.0) were chosen randomly from an exponential distribution. For SSA-CR this created approximately 20 groups, with roughly equal numbers of reactions per group. Each reaction affected M other randomly-chosen reactions where M is a uniformly distributed integer from 1 to 30. Each time one reaction executed, the propensity of each of the M affected reactions was altered and the effect of the change on the overall probability distribution accounted for, before the next reaction was selected. Specifically, the new propensity of each of the M reactions was set to a uniform random value between 95% and 105% of its current propensity.

We created two versions of our test program. The first, which we call a high-memory version, stores a pre-computed random dependency graph, where the list of M affected reactions is generated in advance and stored for each of the N reactions. This requires 15 integers per reaction (on average), which limits the problem size that can be run for large N. So we also created a second, low-memory, version which generates M random dependencies on-the-fly, each time a reaction is selected. As before, M is a uniform random integer between 1 and 30. This second scheme could not be used for modeling an actual biochemical network, but allows the scaling of the SSA-GB and SSA-CR algorithms to be tested for much larger N.

Figure 18 shows timings for the high-memory version of the test program, which stores a dependency graph. Simulations of networks varying in size from $N = 100$ to $N = 100 * 2^{17} \approx 13.1$ million reactions, were run with the SSA-GB and SSA-CR algorithms outlined in Figures 15 and 17. The CPU time is in seconds for 1,000,000 iterations of each algorithm, i. e. 10^6 reactions are executed. The generation and update times for both algorithms are shown; the total time is simply the sum of generation and update for either algorithm. These timing tests were run on a single processor (core) of a Dell 690 desktop machine with two 2.66 GHz quad-core Xeon chips and 16 GB of memory. The two algorithms were implemented in C++, though simple C-style data structures and coding syntax were used for the key operations.

The generation time for both algorithms is roughly equal and nearly constant for networks of any size. Both algorithms are dominated by the cost of updating, since there are many dependencies per reaction. For $N < 100,000$ reactions, the logarithmic and constant scaling of the update time for the two algorithms are evident; logarithmic dependence is a sloped line on a log/linear plot. Around $N = 100,000$, both algorithms begin to run slower due to cache effects when the data structure (tree, groups) for storing propensities no longer fits in second-level cache.

For SSA-GB this is manifested by a logarithmic dependence with a steeper slope. For SSA-CR, the new slope is not as flat as for smaller problems. As we discuss below, this is not due to the algorithm, which still has $O(1)$ or constant-time scaling, but to memory-access issues for very large problem sizes.

Figure 19 shows timings with the low-memory version of the test program, where the reaction dependencies are generated on-the-fly, rather than stored. The update times are now somewhat slower than in Figure 18 (note the difference in vertical scale) due to the cost of generating dependencies each time a reaction executes. But we can now run networks up to size $N = 100 * 2^{21} \approx 210$ million reactions. The difference between logarithmic- and constant-time scaling is now more evident for very large N.

To address the apparent non-constant scaling of the update time for the SSA-CR algorithm for large N, we ran the same low-memory tests on a single processor (core) of a Cray XT3 with dual-core 2.4 GHz

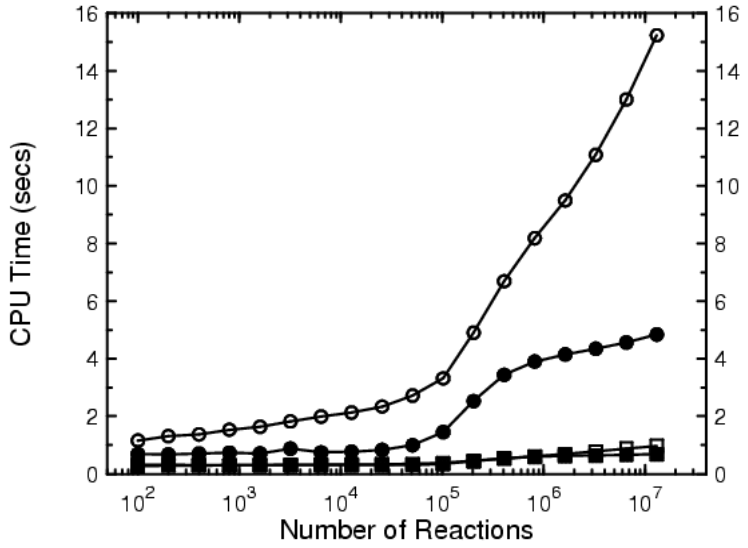


Figure 18: *CPU time in seconds for 1,000,000 iterations of the logarithmic-time Gibson/Bruck (open symbols) and constant-time Composition and Rejection (filled symbols) versions of the direct-method Gillespie stochastic simulation algorithm. Squares are generation times; circles are update times; total time is the sum of generation and update. This is the high-memory version of the test program which stores a reaction dependency graph.*

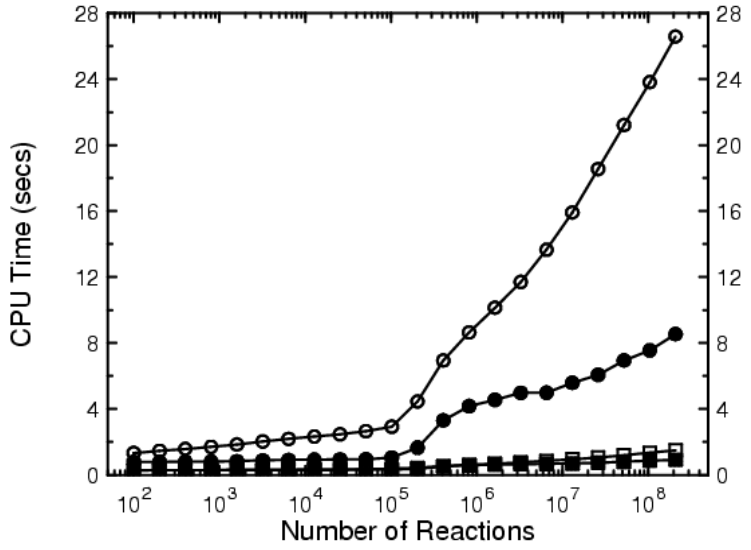


Figure 19: *CPU time in seconds for 1,000,000 iterations of the Gibson/Bruck and Composition and Rejection versions of the direct-method Gillespie stochastic simulation algorithm. The symbols have the same meaning as in Figure 18. This is the low-memory version of the test program which does not store a reaction dependency graph. These runs were performed on a Xeon processor.*

Opteron chips, each with 4 GB of memory. The operating system on the XT3 has the run-time option to

configure itself with either small memory pages (4 kB) or large pages (2 MB). On most Linux machines, including the Dell desktop machine of Figure 18, small memory pages are the default. Large-memory pages is an available option, but typically requires changes in the setup procedure for the OS and a reboot of the machine.

The results for runs with small and large pages on the Opteron processor are shown in Figure 20. The difference is only significant for the update times of the largest runs; elsewhere the dotted- and solid-line data overlay almost exactly. The update timings for small pages (dotted lines) are qualitatively the same as the Xeon timings in Figure 19, with a non-constant upturn for large N runs of the SSA-CR algorithm. Note that the vertical scales of Figures 19 and 20 are different; the last dotted-line data point for a SSA-GB update (open circles) is off the plot at 22.3 seconds.

The large-memory page timings illustrate the constant-time scaling of the SSA-CR algorithm (solid lines and symbols), up to $N = 100 * 2^{20} \approx 105$ million reactions. The large-memory pages improve the speed of both the SSA-GB and SSA-CR algorithms. The reason is that memory access to huge data sets (several gigabytes for the problems with largest N) in Linux is through a translation lookaside buffer (TLB) which is a list of page addresses. For small pages the size of the TLB becomes large for a huge data set, so that access to the TLB itself causes additional cache misses. For large pages, the TLB still fits in cache and overall memory access is more efficient.

Two other features of Figure 20 are interesting to note. First, the constant-time versus logarithmic scaling of even the less-costly generate time for the SSA-CR and SSA-GB algorithms is apparent. Second, the slow-down around $N = 50,000$ due to the size of the data set exceeding second-level cache happens for smaller problems than in Figure 19, due to a smaller cache on the Opteron (1 Mb) versus the Xeon (4 Mb).

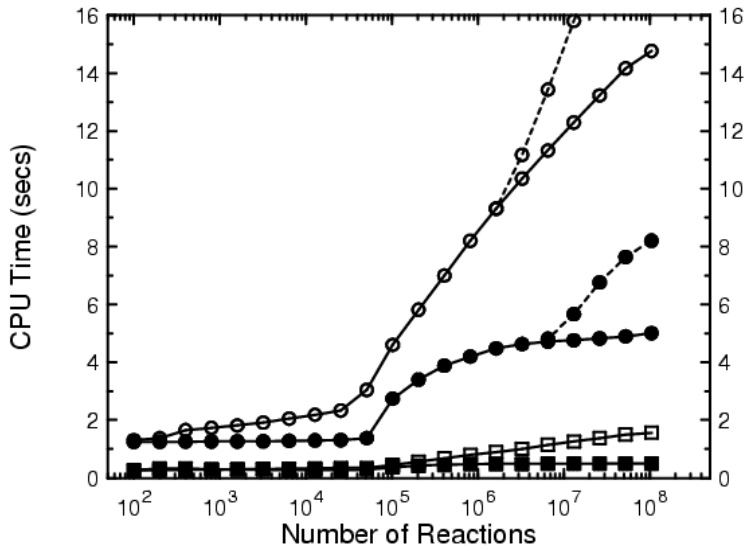


Figure 20: *Same timing runs as Figure 19 but on an Opteron processor. The dashed-line data points are for Linux configured with small-size memory pages; the solid-line data is for large-size memory pages. The latter evidences the constant-time scaling of the Composition and Rejection algorithm.*

Our main conclusion is not the fine details of the performance plots, since these may depend on specific processor attributes or optimized implementations of the algorithms. Rather we focus on the fact that the SSA-CR algorithm is competitive or faster than the SSA-GB algorithm across a large range of biochemical network sizes and exhibits the desired constant-time scaling behavior. We again emphasize that the CR

algorithm is applicable not only to simulations of biochemical networks, but can be used for efficient event selection in any large-scale kinetic Monte Carlo model whose event probabilities and inter-event dependencies meet the same assumptions discussed in this paper.

While biochemical networks with a million or more reactions are not common today, due to limited experimental or bioinformatic data, this is likely to change in the future. As this occurs, the use of $O(1)$ reaction-selection algorithms, such as the Composition and Rejection scheme described here, will become increasingly advantageous.

Acknowledgments

The authors thank Kevin Pedretti at Sandia for help in understanding the memory page-size issues discussed in the performance section. We thank one of the reviewers for calling our attention to the related recent work of Danos and Yang [20, 90].

We note that since this paper was submitted, a paper by Schulze was published [70], which discusses composition-rejection style algorithms in the kinetic Monte Carlo context, similar to the algorithm presented in section 4.5.

5 Parallel Algorithms and Performance

A key computational drawback to KMC is its inherently serial nature. In the standard algorithm, the probability distribution used to select events is a function of the global state of the system. As each event occurs, the probability distribution changes, and this affects which event will be selected next. Thus parallelization of KMC represents a significant algorithmic challenge. Two distinct solutions have been developed. The first is a hybrid approach in which KMC is used to choose events in the interior of the processor sub-domain, while rKMC is used to pick events in the boundary region [42, 52]. The second uses an iterative relaxation method to correct errors due to events at boundaries [22, 51] However, both of these methods suffer from poor scaling [72]. In addition, all of these schemes require specialized treatment of events occurring near processor boundaries which must be carefully worked out for each new application. Because of these shortcomings, we focused our efforts on developing an approximate, but more general, parallel algorithm. We required that the scheme provide accuracy, scalability and generality:

- Accuracy: demonstrate convergence to the standard KMC in some limit
- Efficiency: for a fixed number of events per processor, the simulation time should be independent of the number of processors
- Generality: the interaction between the KMC scheme and the application should be small

We identified the "synchronous sublattice" method of Shim and Amar [73] as a promising approach. The method is based on a previous parallel scheme developed by Heffelfinger and Lewitt [36] for off-lattice Metropolis Monte Carlo simulation. An extension of this scheme to a more complicated interaction potential (EAM) was also developed by Slepoy, and this formed the basis for the subsequent implementation in SPPARKS. We refer to the approach as "sectoring", as described in the previous Section 3. Here we discuss how the sectoring algorithm is controlled in SPPARKS to produce the desired accuracy and parallel efficiency.

We assume that a procedure exists for partitioning all the events amongst the processors. This is usually achieved by associating each event with one and only one particle or site, which in turn is associated with

one and only one processor, based on its position. In the same way, we can divide each processor's events into 2, 4 or 8 sectors as in Fig 12, corresponding to a bisection of the processor domain in each coordinate direction. This ensures a minimum spatial separation between events in corresponding sectors on adjacent processors. By having each processor perform events only in one sector at a time, the worst errors due to simultaneous events can be eliminated. The number of events that each processor performs can not be specified directly, because the total propensity of sectors on different processors are not equal, and these also change as events are performed. Instead, a threshold time t_{stop} is specified, after which all the processors update and proceed to the next sector.

Once t_{stop} is chosen, the basic parallel KMC algorithm in SPPARKS is as follows.

For each sector:

1. Update state of ghost sites adjacent to local sector (see Figure 13)
2. Update propensities of sites in local sector
3. Run KMC on sector until $t > t_{stop}$. The event that exceeds t_{stop} is not performed
4. Update state of local sites adjacent to ghost sectors

In the limit of $t_{stop}p_{sector} \ll 1$, most of the sectors will perform no events, and the few that are performed will be chosen with the correct probability. In this limit, the method looks like rKMC, with a very high rejection rate. Moreover, the ratio of computation to communication becomes vanishingly small. The question then is, can we find a region of "application space" where the approximate method exhibits good accuracy and good parallel efficiency. The coordinates of application space include some easily modified parameters such as system size and number of processors. It also includes some more complicated factors such as details of how the parallel communication is implemented, and how events at boundaries are handled for particular applications.

The question of accuracy is also quite complex. We have identified the following five distinct sources of error that are introduced by the approximate parallel scheme:

a) Events occur simultaneously on different processors.

b) Order of events depends on order in which sectors are visited.

c) Consecutive events occurring in the same sector are oversampled.

d) Consecutive events that straddle a sector boundary are undersampled.

e) Event probabilities affected by adjacent sectors that are both older and younger.

The first three effects are relatively weak. The reason is that their effect falls equally on all events. The last two are stronger, because they are concentrated at sector boundaries. To address the question posed above, we focused on the standard Potts model for grain growth. This is a well-characterized model, and is also representative of many lattice-based KMC applications. We used a 100x100x100 cubic lattice with 26 neighbors per site at zero temperature.

Figure 21 shows the evolution of grain size as a function of time using the exact (serial) KMC algorithm. Ten independent simulations were run, so ten points are plotted at each time, giving an indication of the statistical distribution of $\langle N(t) \rangle$. Time units are such that the maximum site propensity is 26. The initial state contained about 3 sites per grain and was obtained by running rKMC for 0.1 time units.

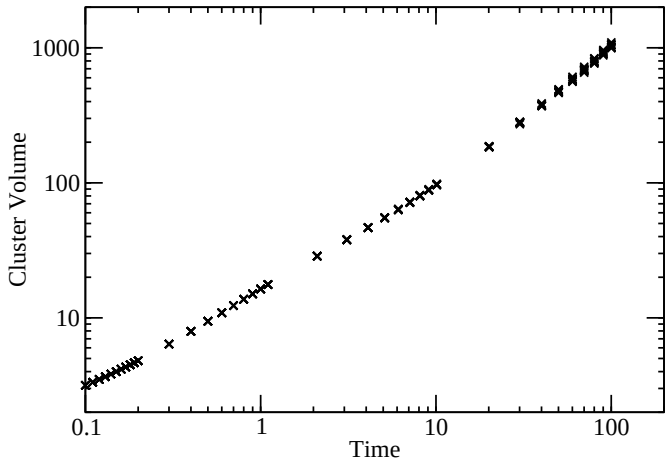


Figure 21: *Plot of average grain volume versus time from an exact KMC Potts model simulation. Ten points are plotted for each time, corresponding to ten independent simulations. See text for more details.*

In our initial implementation of sectoring, we followed Shim and Amar [73], by choosing a fixed value of t_{stop} throughout the simulation. We ran the same lattice on a 2x2x2 processor grid, so that the size of each sector was 25x25x25. In Figure 22 we compare the parallel algorithm for two different values of t_{stop}, along with the exact algorithm. We compare the quantity N/t, rather than N, as it varies less strongly with t. For $t_{stop} = 0.1$ grain size is slightly overestimated, relative to the exact algorithm, but the parallel efficiency is only 55%. Parallel efficiency was estimated as the ratio of compute time to compute plus communication time. By increasing t_{stop} to 5.0, the parallel efficiency is greatly improved, but the overestimation of grain size is now much more pronounced.

The apparent increase in grain size is caused by enhanced growth of grains near sector boundaries. This is an example of an error of type (e) above. The effect is illustrated in Figure 23 which shows a slice through the system at $t = 10$. The sector boundaries are indicated by the black lines. At sector boundaries, the grains are relatively large and protrude asymmetrically to one side or the other. A particularly good example of this is circled in red. In this case, because of the horizontal orientation of the sector boundary, the protrusion of the grains into the lower sector is reminiscent of pieces of grilled meat suspended on a spit. This "shish kebab" effect occurs because the upper sector was visited before the lower sector. When the lower sector is visited, it is initially younger than the upper sector and has smaller grains. The larger grains in the older sector tend to out-compete the smaller grains in the younger sector, resulting in the grains from the older sector protruding into the younger sector. The same effect occurs at all the sector boundaries, with a net increase in grain size. The magnitude of the effect increases with the size difference of grains in the old and new sectors.

We see from this example that it will be difficult to find a value of t_{stop} that provides good efficiency and good accuracy. This is because the rate of growth of the grains drops by many orders of magnitude during the simulation. A value of t_{stop} that works well at short times will be hopelessly inefficient at long times, while a value of t_{stop} that works well at large times will be hopelessly inaccurate at long times. The solution is to use the current state of the system to adaptively choose a good value of t_{stop}.

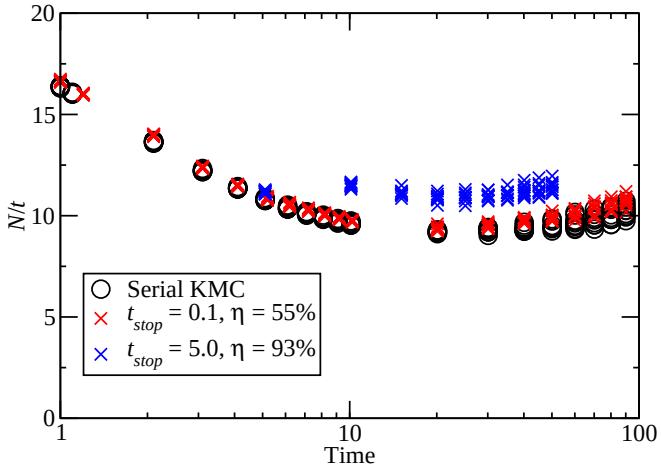


Figure 22: *Time evolution of average grain volume scaled by time t. Black circles are exact KMC results. Crosses are approximate parallel KMC results using fixed t_{stop} values of 0.1 (red) and 5.0 (blue).*

We consider first an idealized system consisting of a large number of identical events, each with propensity p. If we set $t_{stop} = n_{stop}/p$, we would expect to perform a fraction n_{stop} of the events in the sector, assuming that $n_{stop} \ll 1$. Even in cases where n_{stop} is not very small, it should still give a good estimate of the number of events performed. We exploit this relationship by defining p_s to be the total propensity of events in a sector, divided by the number of events in that sector. We explicitly exclude events that have zero propensity. At the start of each pass through the sectors we compute t_{stop} adaptively using

$$t_{stop} = n_{stop} p_{max}$$

where p_{max} is the maximum p_s across all processors and sectors, for p_s computed when each sector is visited on the previous pass. n_{stop} is an accuracy parameter that is specified at the start of the simulation. Because of the way that n_{stop} is defined, we expect that values less than one produce will give results comparable to exact KMC. This expectation is confirmed by the results in Figure 24. The system is the same as before. We see that for $n_{stop} = 1.25$ we obtain results that closely match the exact KMC algorithm. For $n_{stop} = 12.5$ we observe grains that are too large at short times, due to the previously mentioned "shish kebab" effect. In addition, we observe grains that are too small at large times. The latter effect corresponds to an error of type (d) above. As the grains and t_{stop} grow large, the tendency for them to be temporarily held up at sector boundaries increases, resulting in lower average grain size.

Parallel efficiency is much higher using the adaptive algorithm than using a fixed value of t_{stop}, for the same level of accuracy. The adaptive algorithm is both more efficient and more robust. At short times, when grains are small, and there are many active sites at grain boundaries, p_{max} is large and t_{stop} is small. At long times, the grains are large, most of the sites are in the inactive interior of grains, and so p_{max} becomes small and t_{stop} becomes large. As t_{stop} increases, the parallel efficiency improves.

All of the previous calculations were performed using only 8 processors. In order to truly test the scalability of the adaptive algorithm, we performed some benchmark calculations using large numbers of processors on Sandia's Red Storm supercomputer. The system was identical to the one described above, but larger sys-

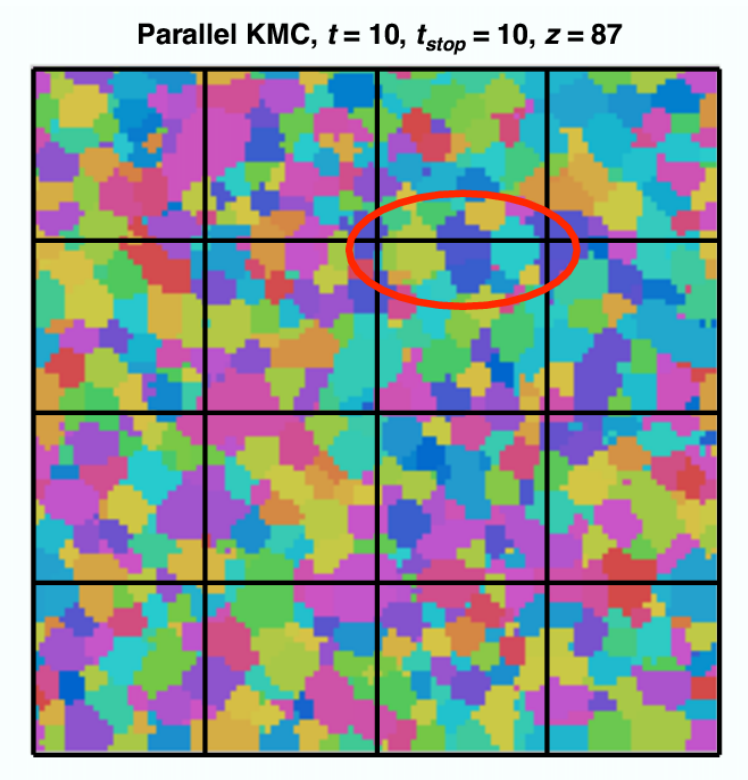


Figure 23: *Cross-section of snapshot from a 100x100x100 parallel KMC grain growth simulation using fixed* $t_{stop} = 10.0$*. Colors indicate spin values of sites. Black lines indicate sector boundaries. Red circle identifies a particularly prominent example of the "shish kebab" effect. See text for more details.*

tem sizes were used, and the simulations were run for a relatively short time of $t = 1.0$.

The strong scaling parallel efficiency is defined as follows:

$$ParEff = \frac{P * T(1)}{T(P)}$$

where $T(1)$ is the time required to run a calculation on one processor (using the sectoring algorithm), and $T(P)$ is the time required to run the problem on P processors. We studied systems consisting of 100^3, 150^3, 200^3, and 1000^3 sites. In all cases we used the adaptive algorithm with $n_{stop} = 1.25$. For the smallest system we also ran a higher accuracy run with $n_{stop} = 0.125$.

We ran the different systems on cubic grids of processors ranging from 1 to 3375 processors, and a non-cubic grid of 15x16x16 = 3840 processors. Figure 25 shows strong scaling parallel efficiency versus processor count for all the different cases. The surprising super-linear scaling is due to the use of the tree algorithm for selecting events in a sector. Unlike the constant-time algorithm described in the paper of Section 4, the computational cost per event of this algorithm increases logarithmically with the number of sites. In other words, the apparent super-linear scaling of the parallel algorithm is caused by the sub-linear scaling of the underlying serial algorithm. Because we were unable to run the 1000^3 system on 1 processor, we obtained $T(1)$ by fitting to the following equation

$$T(P)P = T(1) + mlog(P)$$

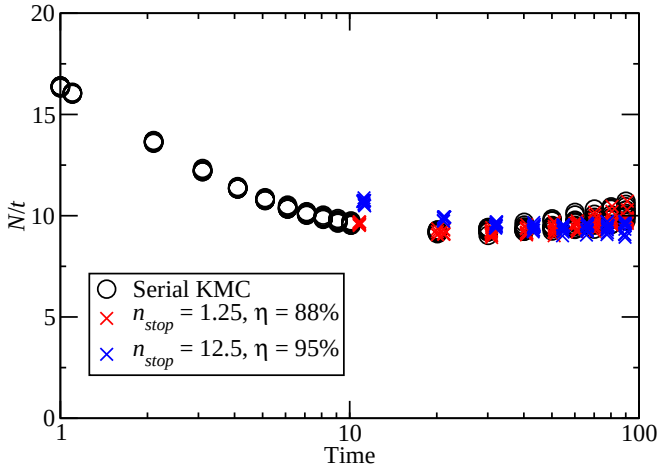


Figure 24: *Time evolution of average grain volume scaled by time t. Black circles are exact KMC results. Crosses are approximate parallel KMC results using n_{stop} values of 1.25 (red) and 12.5 (blue).*

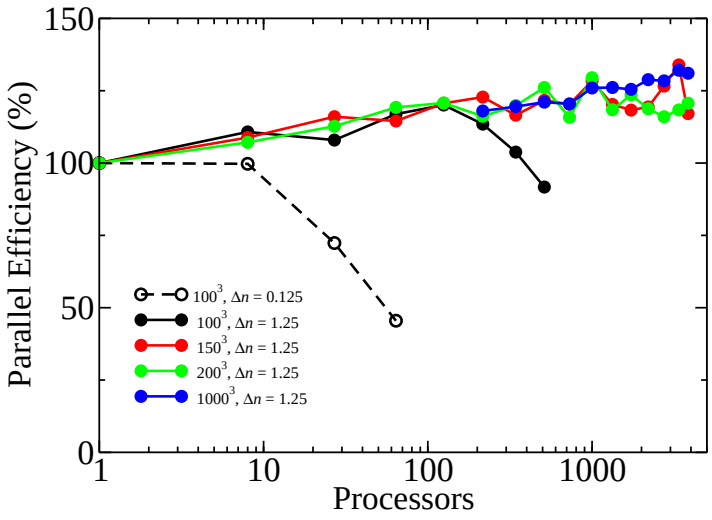


Figure 25: *Strong scaling parallel efficiency for approximate parallel KMC algorithm using n_{stop} values of 0.125 (open symbols) and 1.25 (filled symbols). The simulations used a grain growth Potts model on a 26 neighbor cubic lattice with 100^3 (black), 150^3 (red), 200^3 (green), and 1000^3 (blue) sites. All calculations were performed on Sandia's Red Storm supercomputer.*

As expected, the graph for 1000^3 matches those of the other systems. For the smallest system, the parallel efficiency decreases sharply for $P > 216$. Decreasing n_{stop} causes this breakdown in parallel efficiency to occur at a proportionately lower processor count. Simple scaling arguments would suggest that we also

see breakdown in 150^3 system for $P > 729$, but the parallel efficiency remained above 100% out to 3840 processors.

6 Verification

In order to test the speed and accuracy of the Potts model implementation in SPPARKS, a separate serial code was written from the ground up for testing purposes. This code, named MESO (for Microstructure Evolution Simulation and Optimization) was designed to replace a number of existing Sandia kinetic Monte Carlo codes that were not optimized for efficiency, and therefore not suitable for testing purposes.

MESO is coded entirely in ANSI-C and is completely portable across platforms, so it can be easily compared against SPPARKS on a variety of test machines. Lattice site searches in MESO are binary tree-based for speed, all memory is dynamically allocated, and the code is written modularly so that analysis can be performed in-code and easily added or removed. In addition to kinetic Monte Carlo, rejection Monte Carlo is implemented as an optional preconditioning step.

Figure 26 shows the results of timing runs of SPPARKS versus MESO as a function of the number of lattice sites. For the generic Potts model, timing in both codes is identical for lattice sizes up to 10^6 sites, which are 3d simulations of 100 sites on a side. Earlier versions of the SPPARKS code also included an option to use an explicit 3d data structure to store the lattice (lattice3d), and these timings are reported in this figure as well. The lattice3d option was found to be slower than the general lattice data structure and was removed from later versions of the code.

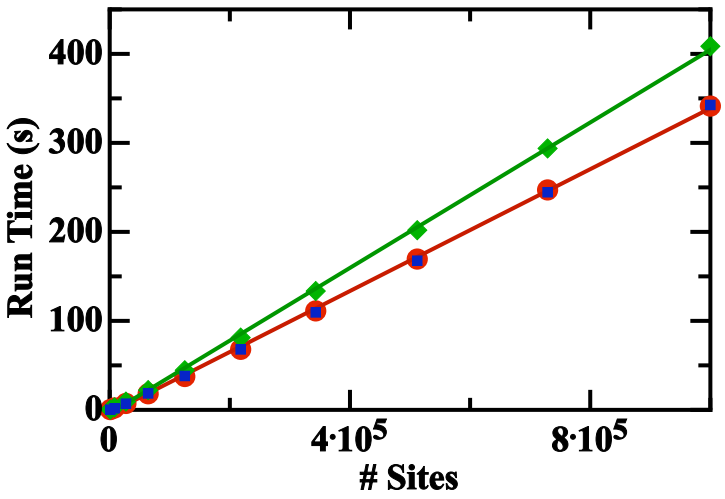


Figure 26: *Run time in seconds for 3d simulations as a function of total number of lattice sites. Data are shown for SPPARKS general lattice (red circles), SPPARKS lattice3d (green diamonds) and MESO (blue squares). Linear fits to the SPPARKS general lattice and SPPARKS lattice3d simulations are shown with red and green lines, respectively.*

The time-dependence of the total system energy provides a simple test of the accuracy of Potts model simulations. Figure 27 shows the total system energy at different times for the 100^3 site simulation in Figure 26 using MESO, SPPARKS on a single processor and SPPARKS on four processors. The energies produced by SPPARKS on different numbers of processors agree well with each other. While it was not possible to compare energies from SPPARKS and MESO at identical times, the evolution of energy generated by both

codes agree well.

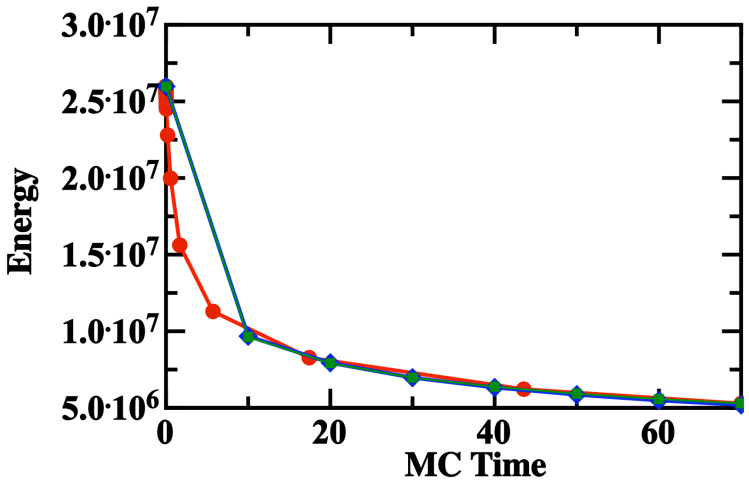


Figure 27: *A comparison of energy as a function of Monte Carlo time for a 3d grain growth simulation in the Potts model with* 100^3 *lattice sites MESO (red circles), SPPARKS on one processor (blue diamonds) and SPPARKS on four processors (green squares). Data are plotted as a function of Monte Carlo time with lines shown as a guide to the eye.*

The time-dependence of the average grain radius provides a stronger accuracy test of Potts model simulations. This comparison in shown in Figure 28 for the previous example. Again, this figure shows that grain radii produced by SPPARKS running on different numbers of processors agree well with each other and the evolution of radii generated by both SPPARKS and MESO agree well.

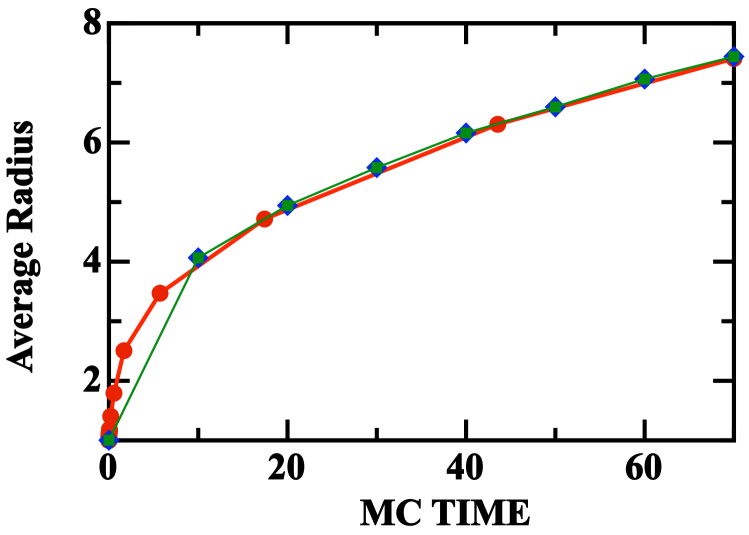


Figure 28: *Average grain radius for 3d grain growth from Potts model simulations with* 100^3 *lattice sites from MESO (red circles), SPPARKS on one processor (blue diamonds) and SPPARKS on four processors (green squares). Data are plotted as a function of Monte Carlo time with lines shown as a guide to the eye.*

7 Applications

This section contains seven sub-sections, each of which describe a different physical model, its implementation in SPPARKS as an on-lattice application, and highlight simulation results which illustrate what the method is capable of modeling. The seven applications are for abnormal grain growth in 7.1, nanoporous metals in 7.2, a solid-on-solid model for surface diffusion in 7.3, defect formation in erbium hydrides in 7.4, bubble formation in nuclear fuels in 7.5, the sintering of a nuclear fuel pin in 7.6, and thin film deposition and growth in 7.7.

7.1 Abnormal Grain Growth

Most metals and ceramics are polycrystalline; they are comprised of many individual crystallites, called grains, separated by internal interfaces, or grain boundaries. Because these grain boundaries contribute free energy due to broken atomic bonds, the equilibrium state does not include grain boundaries and is a single crystal.

When polycrystalline materials are annealed at sufficiently high temperatures, the grain boundaries will move and rearrange so as to increase the average grain size and decrease the grain boundary area per unit volume. In many cases, these rearrangements occur fairly uniformly; grains remain equiaxed and maintain a relatively small size distribution. This process is termed normal grain growth. However, in a variety of systems, the grain growth process is not uniform; a few, special grains grow very large at the expense of the other grains in the system. This process is called abnormal grain growth.

Abnormal grain growth is detrimental to many materials applications, causing premature failure of copper interconnect lines in integrated circuits and poor surface finish on aluminum automotive body panels, for example. However, it is desirable in some applications, where it improves magnetic permeability and loss properties in transformer steels and increases the toughness of nanocrystalline metals.

Abnormal grain growth is one phenomenon with many causes. It can occur in thin films due to surface energy anisotropy; in textured materials due to outlier grains with special boundary properties; in materials with a thin grain boundary phase due to variations in boundary structure; and in deformed materials to relieve stored strain energy. One of the most baffling instances of abnormal grain growth occurs in materials that contain stable, static particle dispersions, such as nickel-based superalloys. Conventional grain growth theory suggests that a stable particle dispersion should pin grain boundaries in positions that maximize their contact with the particles; when enough boundaries are pinned, grain growth cannot proceed. Nonetheless, abnormal grain growth is often observed in such materials and remained unexplained for many years.

In collaboration with Tod Hoffman and his adviser at Carnegie Mellon University, we performed the first simulations to shed some light upon this phenomenon. We began with an equiaxed microstructure with static particles deposited at random upon the grain boundaries. Boundary properties were isotropic and uniform, and we evolved the structures using the KMC Potts model for grain growth. For sufficiently high particle fractions, this system should be pinned and normal grain growth should not occur. Surprisingly, we found that in certain regimes of particle fraction and grain size, abnormal grain growth occurs, with one or a few abnormal grains consuming the entire structure. We termed this phenomenon particle-assisted abnormal grain growth.

We are working to develop a model for scaling the abnormal growth frequency with initial grain size and particle fraction. However, particle-assisted abnormal growth events are rare and stochastic. To acquire enough data for good statistics, we need to simulate large numbers of grains for long times under various conditions. There is a further challenge: Since the abnormally growing grain consumes the other grains in the system, one abnormal event can wipe out the possibility of other, later events, thus skewing the frequency data. Therefore, we want to run many independent MC Potts model simulations of moderately large size for

very long times. SPPARKS provides an ideal code platform for performing these simulations.

We developed a SPPARKS application that included both Potts model grain growth and the presence of inert pinning particles. The SPPARKS implementation also utilized an initialization routine that created equiaxed microstructures with particles decorating the grain boundaries. We verified the SPPARKS application against the serial code MESO and found excellent agreement, even in parallel mode, using the approximate algorithm of Section 5. We ran over 300 independent trials on both workstations and the Thunderbird cluster. Each trial was on a 300x300x300 system for at least a million MC steps (sweep over all lattice sites) at finite simulation temperature.

Typical results are shown in Figure 29. In a system with 5 volume percent pinning particles and initial grain radius of 10 sites, shown in Figure 29(a), the initial system is not fully pinned. There is sufficient driving force to move the boundaries away from their initial positions, but after some amount of normal grain growth, a particle-pinned structure is achieved. In contrast, in a system with 10 volume percent pinning particles and the same initial grain radius of 10 sites, shown in Figure 29(b), the initial system is strongly pinned. There is little microstructural evolution initially, but eventually a few grains break away from their particle clouds and begin to grow. Since the other grains in the system are still pinned, the growing grains can grow without competition, until they impinge upon each other; this is a particle-assisted abnormal growth event.

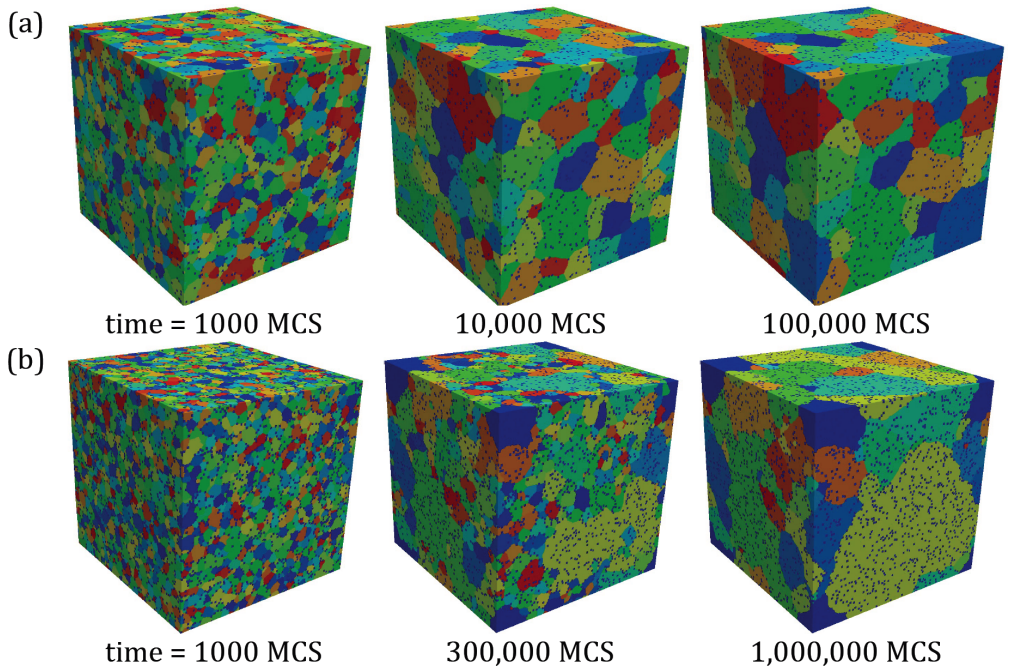


Figure 29: *(a) Normal grain growth in an equiaxed grain structure containing 5pinning particles by volume, initially located at grain boundaries. The initial structure is not fully pinned; after some normal grain growth, the system pins at a larger grain size. (b) Particle-assisted abnormal grain growth in an equiaxed grain structure containing 10boundaries. The initial structure is strongly pinned; after a long incubation time, a few grains grow abnormally and dominate the system at late times.*

We are still analyzing the data to develop a model, but we have already gotten more qualitative and quantitative insight into the particle-assisted abnormal grain growth phenomenon. In particular, we find that the abnormal grains are always among the largest initial grains, but most of the largest initial grains do not grow abnormally. By examining which grains grow and which do not, we hope to establish a criterion for

abnormal grain nucleation based on grain characteristics and environment.

7.2 Nanoporous Metals

7.2.1 Motivation and Description

Nanoporous materials have a range of applications based on their very large surface-to-volume ratios. At Sandia, nanoporous metals are being synthesized and developed for gas and energy storage applications. In particular, nanoporous palladium particles are being studied for use as a hydrogen storage medium, and nanoporous gold wires are being tested for use as electrodes in super-capacitors. As these complex materials age, especially under elevated temperatures, the evolution of their nanostructures with time can greatly affect material properties and performance. For example, nanoporous structures may coarsen or close off over time, leading to loss of accessible surface area and lower storage capacities. In this case, the coarsening is caused by surface diffusion that is driven (from the continuum viewpoint) by surface curvature gradients. Although this surface motion is driven by phenomena at the atomic scale, it occurs over a timescale too long to be simulated using classical molecular dynamics. Kinetic Monte Carlo modeling can allow simulations over longer times, capturing the evolution of these nanoscale structures.

A simple atomistic model that can qualitatively capture the coarsening due to surface diffusion has been implemented in SPPARKS. In this model, a lattice of sites represents the atomic configuration of a material. Each site is marked as either "vacant" or "occupied"; here we will let the variable ϕ_i represent the state of site i, with $\phi_i = 0$ denoting a vacant site and $\phi_i = 1$ an occupied site. The energy of an occupied site is assumed to be a function of its coordination number c_i:

$$E_i = \phi_i f(c_i) \tag{6a}$$

$$c_i = \sum_{j \in neigh(i)} \phi_j \tag{6b}$$

where $f(c)$ is a monotonically decreasing function.

One possible choice for $f(c)$ is a linear function that gives zero when an atom is surrounded by its full complement of neighbors:

$$f_{lin}(c) = Z - c \tag{7}$$

where Z is the maximum number of neighbors for a given lattice structure (e.g. $Z = 12$ for an FCC crystal). This choice of energy function gives a minimum system energy of zero when all sites are occupied (or when all sites are vacant).

Diffusion is described by Kawasaki dynamics, in which each event is the motion of an individual atom. A basic atom hop is defined as the motion of an atom to any vacant neighboring site. However, special consideration is given to the Schwoebel barrier, in which an atom diffuses up or down a ledge between atomic monolayers. To account for this case, atoms are allowed to hop to locations two sites away, provided that a vacant intermediate site exists. Prospective Schwoebel barrier events can be further limited by specifying a maximum coordination number for the originating site, and a minimum coordination number for the destination site.

In general, the rate of a possible atom hop event is computed using both the change in energy between initial and final states, along with a user-supplied energy barrier Q. Consider an event that takes the system from some configuration I to a new configuration J; define the energy change as $\Delta E_{I \to J} = E_J - E_I$,

where E_I and E_J are the energies of the entire system in the two configurations. The rate of this event is computed as:

$$P_{I\to J} = \begin{cases} \exp(-\frac{Q}{kT}) & \text{if } \Delta E_{I\to J} \leq 0 \\ \exp(-\frac{Q+\Delta E_{I\to J}}{kT}) & \text{if } \Delta E_{I\to J} > 0 \end{cases} \tag{8}$$

The energy barrier Q can be a constant for all events, or a function of the coordination numbers of the initial and destination sites (where coordination numbers are computed without counting the moving atom). In addition, different barriers can be prescribed for simple 1-site hops and for Schoebel barrier hops. The default value is $Q = 0$ for all events.

For the general case where the energy function $f(c)$ is a prescribed function of the coordination number, the energy change of the system for a hop from site i to site j must be computed by summing the energies of all sites whose energies are changed by the event, i.e. all sites that neighbor sites i and j. However, a simplification is possible when a linear energy function is used; in this case it can be shown that the energy change is given by:

$$\Delta E_{I\to J} = 2(E_{j,J} - E_{i,I}) \tag{9}$$

where $E_{i,I}$ is the site energy at site i at state I (before the event) and $E_{j,J}$ is the site energy at site j at state J (after the event).

7.2.2 Simulations

The diffusion application of SPPARKS has been used to study the formation and evolution of nanoporous structures. Typically, an initial condition is created in which random sites are filled according to a prescribed volume fraction. When the simulation is started, atoms quickly form a ligamented structure as the system evolves toward a lower energy configuration. As time progresses, the structure coarsens leading to larger ligament sizes and less surface area. Surface facets form for low energy surfaces such as $\{111\}$ planes; this facetting is more pronounced at lower temperatures. A snapshot of a typical atomic configuration is shown in Figure 30.

Simple arguments based on continuum models of surface diffusion predict that, for self-similar coarsening of nanoporous structures such as these, the characteristic length scale of the system raised to the fourth power should grow linearly with time. To test whether this scaling behavior is exhibitted by our KMC model, we choose the inverse of the surface-to-volume ratio as a measure of the characteristic length scale of the system, and plot the fourth power of this quantity against time. Figure 31 shows this curve for simulations at various temperatures. For all cases, above a certain length scale the curve appears linear, implying that the scaling law predicted by continuum models is obeyed.

7.3 Solid-on-Solid Model

7.3.1 Motivation and Description

The solid-on-solid (SOS) model is a useful simplified model of surface diffusion that captures many of the physical phenomena of diffusion, but is simple enough to be amenable to analysis [46]. In this model, the surface is described by a one- or two-dimensional lattice of sites. At each site an integer value represents the height of the surface at that site, so that collectively the heights of all the sites represent a surface profile with no overhangs or vacancies.

The energy of the system is given by the sum of all of the site energies:

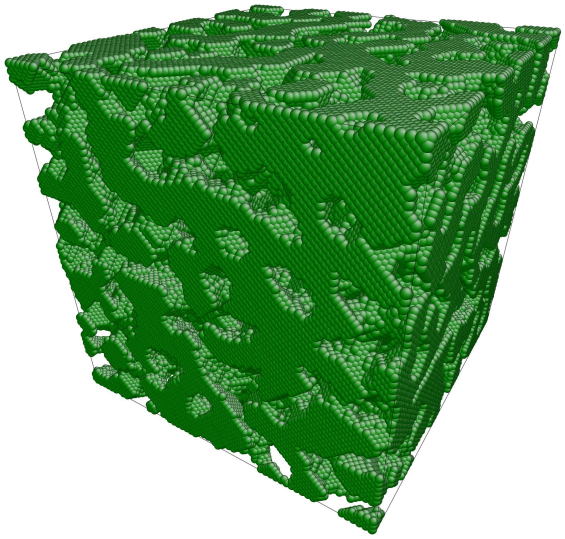


Figure 30: *Atomic configuration for a typical diffusion simulation. Simulation is an FCC lattice with 60 unit cells in each direction, with 50% solid fraction. Nondimensional temperature is T=1.0; nondimensional time is t=2500.*

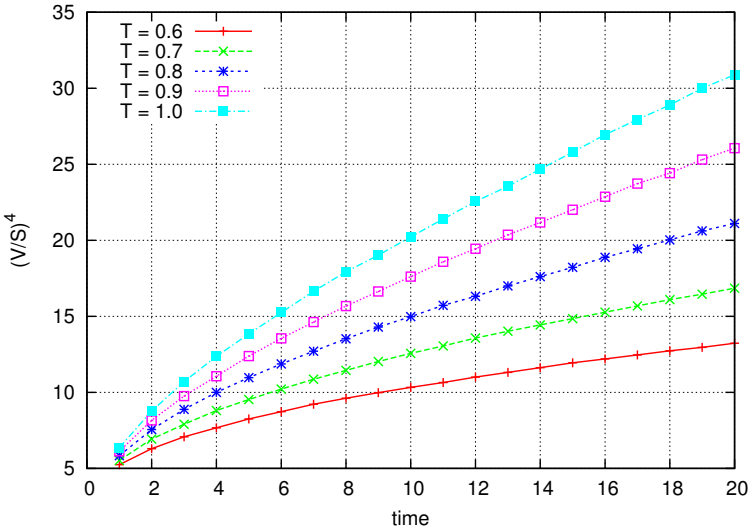


Figure 31: *Evolution of surface area with time for various non-dimensional temperatures. Surface-to-volume ratio to the -4^{th} power appears to show linear growth in time.*

$$E_{tot} = \sum_i E_i. \tag{10}$$

The energy of a given site i is a function of the height difference between site i and all of its neighbors:

$$E_i = \frac{1}{2} J \sum_{j \in neigh(i)} |h_i - h_j| \tag{11}$$

where J is a prescribed bond energy, h_i is the height at site i and $neigh(i)$ is set of sites neighboring site i. The energy in this model can therefore be thought of as resulting from the breaking of lateral bonds between atoms (so that a surface with constant height has zero energy).

Although the SPPARKS code can simulate this model for a range of lattice types, in our simulations we have concentrated on 1D periodic lattices. This system is often referred to in the literature as "1+1 dimensional", since in effect it models the 1D surface of a 2D solid. In this case the energy can be written more simply as:

$$E_{tot} = J \sum_{i=1}^{L} |h_i - h_{i-1}|. \tag{12}$$

where L is the total number of sites along the length of the system; sites $i = 0$ and $i = L$ are equivalent because of periodicity.

Surface diffusion is described by Kawasaki dynamics, i.e. by hopping of surface atoms to neighboring sites. We assume that only individual atoms hop as part of a single event, so that for any given configuration, there are $2L$ possible events: an atom from any site hopping to the left or to the right.

7.3.2 Simulations

To understand the dynamics of surface evolution described by this model, the decay of a sinusoidal surface has been studied using KMC. The initial condition for all simulations is:

$$h_i(t = 0) = \text{aint}(A_0 \sin(2\pi i/L)) \tag{13}$$

where A_0 is the initial amplitude and aint() is a function that rounds its argument to the next lower integer in magnitude. Figure 32 shows the initial condition corresponding to $L = 40$ and $A_0 = 5.5$.

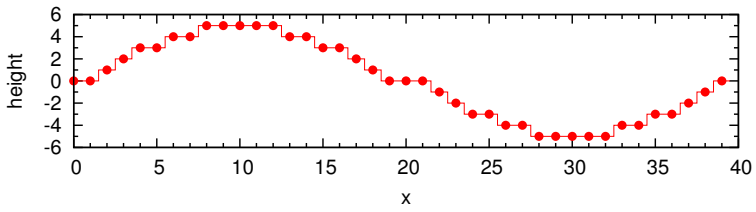


Figure 32: *Initial condition from equation (13) with* $L = 40$ *and* $A_0 = 5.5$.

Surface evolution data is computed by averaging over a large number of realizations using different random number seeds. This averaging operation will be denoted by an overbar, so that $\bar{h}_i$ represents the height at position i averaged over all realizations. Because this 1D problem runs very quickly, a large number of realizations can be run to obtain smooth data; 10^3 or 10^4 realizations are used to generate data in this work. The shape evolution is characterized by the amplitude of the surface profile, given by

$$A(t) = \frac{1}{2} \left[\max_i \bar{h}_i(t) - \min_i \bar{h}_i(t) \right]. \tag{14}$$

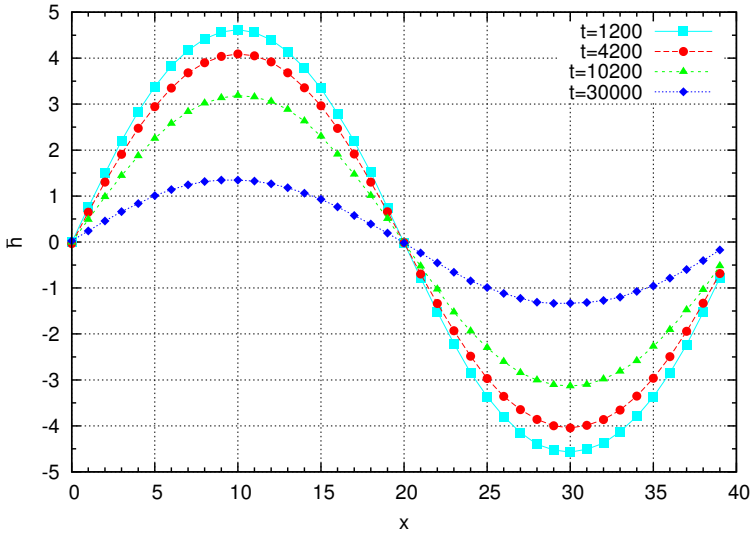


Figure 33: *Average profile* $\bar{h}(t)$ *over time for* $L = 40$, $A_0 = 5.5$, $T = 0.8$.

Figure 33 shows the average profile height $\bar{h}(t)$ for several different times for the case $L = 40$, $A_0 = 5.5$ at a temperature $T = 0.8$. The profile retains its sinusoidal shape. The time history of the amplitude for this case is plotted in Figures 34.

The theoretical continuum equation for surface diffusion [59], assuming a small-amplitude surface profile, is a fourth-order PDE:

$$\frac{\partial \bar{h}}{\partial t} = -B \frac{\partial^4 \bar{h}}{\partial x^4} \tag{15}$$

where B is a temperature-dependent surface mobility. Self-similar solutions to this equation on the periodic domain have the form

$$\bar{h}(x,t) = A_0 \sin\left(\frac{2\pi n x}{L}\right) \exp\left(-B\left(\frac{2\pi n}{L}\right)^4 t\right) \tag{16}$$

where n is an integer, so that sinusoidal profiles decay with a characteristic relaxation time τ given by

$$\tau = \frac{L^4}{B(2\pi n)^4} \tag{17}$$

Since relaxation time is strongly dependent on wavelength, shorter waves should decay much more quickly than longer ones, so that long-time dynamics is dominated by the longest mode in the system ($n = 1$). The relaxation time for the longest mode can be estimated by fitting the dependence of the amplitude on time to an exponential function. Figure 35 shows this fit for various system sizes L; the straight lines on this log-log plot indicate an exponential decay as predicted by the continuum theory in equation (16). The relaxation time τ for each system is given by the negative reciprocal of the slope of the line. Figure (36) shows the relation between τ and system size L (the wavelength of the longest mode) for various values of L on a log-log plot. The slope of the line is 4.12, showing good agreement with the continuum theory prediction of 4 (equation 17).

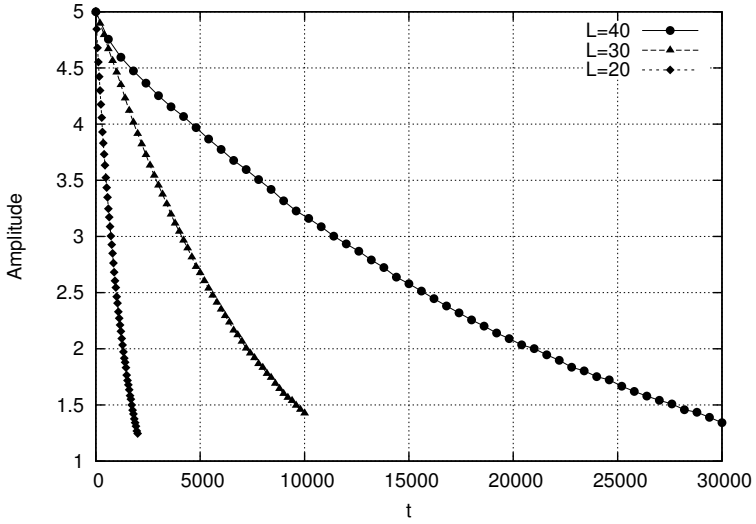


Figure 34: *Amplitude vs. time for various size systems with* $A_0 = 5.5$, $T = 0.8$.

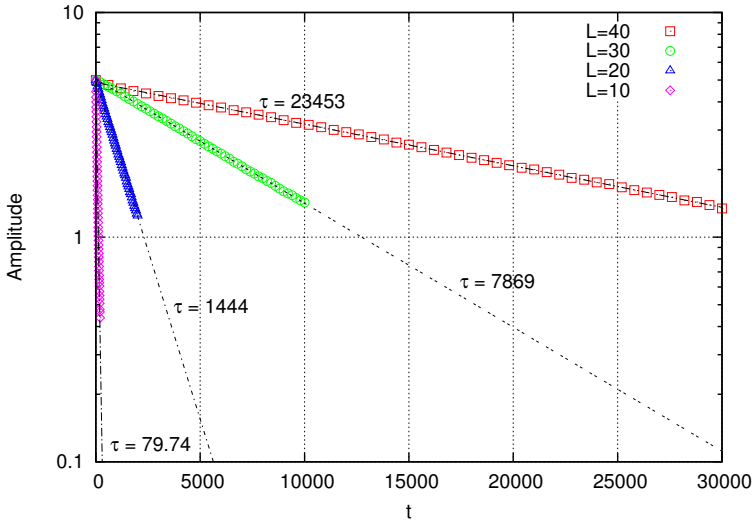


Figure 35: *Exponential fit of amplitude decay for various system sizes with* $A_0 = 5.5$, $T = 0.8$. *Relaxation time* τ *for each value of* L *is the negative reciprocal of the slope of the line on the log-log plot.*

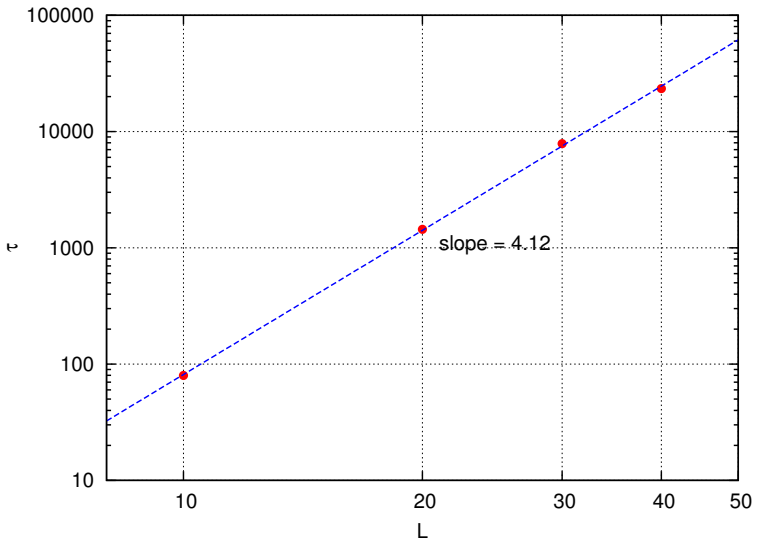


Figure 36: *Variation of τ with L for $A_0 = 5.5$, $T = 0.8$. The slope is very close to the continuum theory prediction of 4.0.*

7.4 Erbium hydrides for Neutron Generators

Metal hydrides (MH_2) are commonly used for neutron generation in many applications in industry, medicine, and national security, including petroleum exploration, explosives detection, toxic waste analysis, and boron neutron capture therapy. In the present work, we examine the mobility of gaseous species, i.e. tritium (T) and helium (^{3}He), on the instersitial lattices of FCC erbium (Er) in erbium tritide (ErT_2). Tritium is implanted into the erbium metal matrix, and resides primarily on the tetrahedral interstitial sites of the face-centered cubic (FCC) lattice. The tritium isotopes decay into helium with a half-life of approximately 12.3 years. Both the tritium and helium can diffuse through the FCC lattice on the tetrahedral and octahedral interstitial sites.

As tritium decays into helium, the helium gas clusters to form bubbles, often found to be preferentially oriented along $\{111\}$ planes. To investigate this phenomenon, we have utilized density functional theory via Sandia's SeqQuest electronic structure code, to calculate the energetics required to construct and parameterize a mechanism for the diffusion of tritium and helium in FCC erbium, as shown in Table 7.4. Recently, SeqQuest was used to estimate the binding energies of various clusters of vacant tetrahedral sites, and the implementation of defect binding into SPPARKS for simulating ErT_2, will be performed as part of an ASC project starting in 2010.

In order to demonstrate the ability of SPPARKS to perform large calculations of diffusion in ErT_2, we conducted a simulation of tritium diffusion on the tetrahedral and octahedral lattices using only the third and fourth reactions from Table 7.4 for simplicity, and seeding the tetrahedral interstitial lattice with 10% vacancies. A domain of 500x500x500 ErT_2 unit cells was created. A snapshot from this simulation is shown in Figure 37, where it is evident that in the absence of a mechanism for defect binding, the arrangement of tritium and vacancies on the lattice sites remains spatially random. In addition, the "one-way" diffusion of tritium from the tetrahedral to the octahedral lattice results in the accumulation of tritium on the latter. Nonetheless, this simuation is noteworthy because, assuming a lattice parameter of 5.1254 Å, it captures a simulation domain representing approximately 0.25 μm of material on each edge.

Table I: *Diffusion Mechanism in ErT_2.*

Reaction	Rate [1/ns] or Barrier [eV]
$T_{tet} \rightarrow {}^3He_{tet}$	1.78[1/ns]
$T_{oct} \rightarrow {}^3He_{oct}$	1.78[1/ns]
$T_{tet} + *_{tet} \rightarrow *_{tet} + T_{tet}$	0.98 [eV]
$T_{tet} + *_{oct} \rightarrow *_{tet} + T_{oct}$	1.89 [eV]
$T_{tet} + *_{oct} \leftarrow *_{tet} + T_{oct}$	0.68 [eV]
${}^3He_{tet} + *_{tet} \rightarrow *_{tet} + {}^3He_{tet}$	0.49 [eV]
${}^3He_{oct} + *_{oct} \rightarrow *_{oct} + {}^3He_{oct}$	1.49 [eV]
$T_{tet} + *_{oct} + T_{oct} \rightarrow T_{tet} + T_{oct} + *_{oct}$	0.62 [eV]
$T_{tet} + *_{oct} + {}^3He_{tet} \rightarrow {}^3He_{tet} + T_{oct} + *_{tet}$	1.31 [eV]
$T_{tet} + *_{oct} + {}^3He_{tet} \leftarrow {}^3He_{tet} + T_{oct} + *_{tet}$	0.16 [eV]
${}^3He_{tet} + T_{oct} + *_{oct} \rightarrow T_{tet} + *_{oct} + {}^3He_{oct}$	0.88 [eV]
${}^3He_{tet} + T_{oct} + *_{oct} \leftarrow T_{tet} + *_{oct} + {}^3He_{oct}$	0.16 [eV]

Note: Rates of the first two reactions are from the half-life of tritium, and barriers in the remaining reactions are from density functional theory [89]. Vacancies are denoted by an asterisk (*). The tetrahedral and octahedral interstitial lattices are denoted by subscripts tet and oct, respectively. The beta particle and antineutrino have been omitted from the products of the first two reactions for clarity.

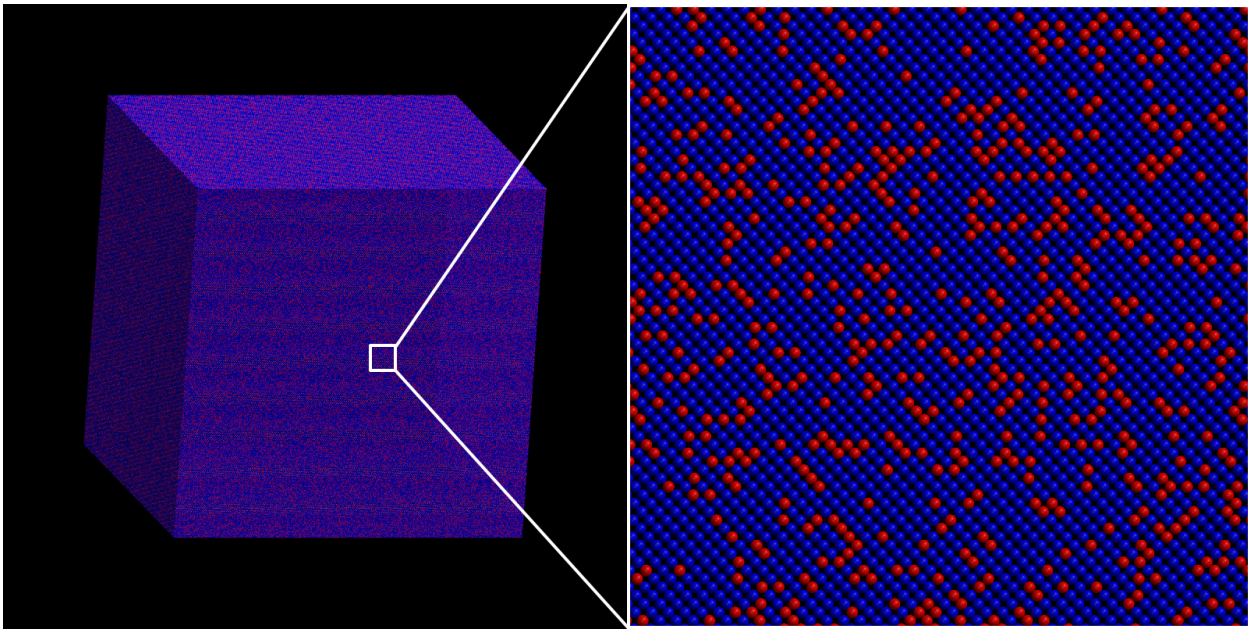


Figure 37: *Massively parallel simulation of tritium diffusion on two billion lattice sites of ErT_2. Only the outermost layers of octahedral interstitial sites are shown. Blue spheres are vacancies and red are tritium.*

7.5 Bubble formation in Nuclear Fuels

During operation in a nuclear fuel, approximately 0.3 atoms of Xe and Kr gas are generated per fission event. These atoms have very low solubility in the fuel so they diffuse in the grain structure and coalesce to form gas bubbles, both inter- and intra-granular bubbles. We developed a model in SPPARKS to simulate

the formation of gas atoms, their diffusion, nucleation to form bubbles (both in the crystalline structure of the grain and at grain boundaries), growth of the bubbles and redissolution of gas back in the crystalline structure due to fission fragment damage. The results of these simulations are presented here. Gas atom formation is simulated by introducing voxels designated as gas atoms with frequency that can be related to the frequency of fission events in a nuclear fuel. Diffusion of a gas atom in the fuel structure is simulated by random walk. Nucleation occurs at trapping sites, defects generated by irradiation, and at grain boundaries. Nucleated bubbles grow by gas precipitating on them. Another mechanism that is active in nuclear fuels is the redissolution of gas atoms from bubbles back into the fuel matrix. This occurs because fission fragments that pass near or through gas bubbles disorder the fuel matrix sufficiently to cause gas atoms to go back into solution in the fuel crystal.

The simulations to demonstrate this model was performed on a two-grain system of size 100 x100 x 200, with each grain being 100 x 100 x 100. Full periodic boundary conditions were use. The geometry is shown in Figure 38; each of the large colored features is a grain and blue features are trapping sites. The two grain boundaries are at the center plane between the grains and at the two free parallel to the center plane. The two edges are the same grain boundary.

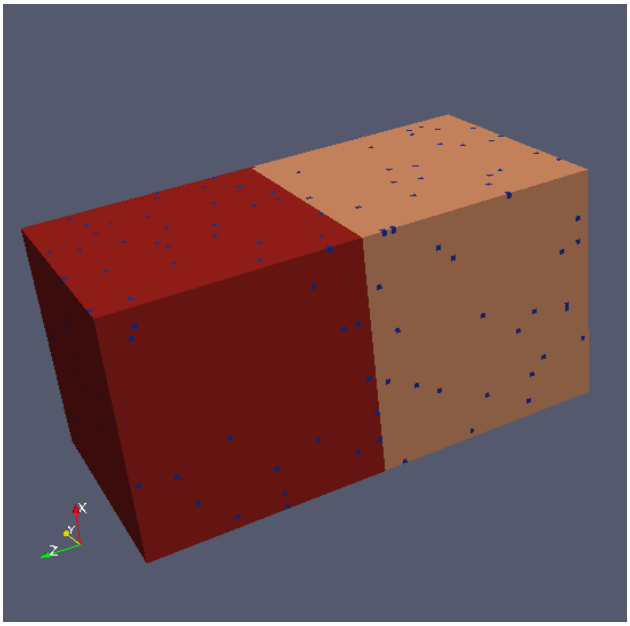


Figure 38: *The two-grained microstructure with full periodic conditions used for initial simulations.*

The SPPARKS model simulates gas atom generation, diffusion, coalescence, redissolution, bubble migration by surface diffusion and grain growth. The results are shown in Figure 39. Only bubbles and trapping sites are imaged, grain sites and gas atom sites are not. Due to periodic boundary conditions, the grain boundary at the ends of the simulation are contiguous, thus the visible plane at the near left edge is a cross-section at the grain boundary. Gas atoms are generated and diffuse to sites where they precipitate as intra- and inter-granular bubbles. The shape of the bubbles is changing by surface diffusion to obtain the minimum energy shape. There are small circular intragranular bubbles and larger lenticular bubbles at the grain boundaries.

While grain growth does not occur in this simulation, as there is no curvature to the grain boundary, the grain sites locally do change from one grain to another to accommodate the feature shapes locally.

To study the grain boundary bubbles, a modified two-grain geometry with large grain boundary face was

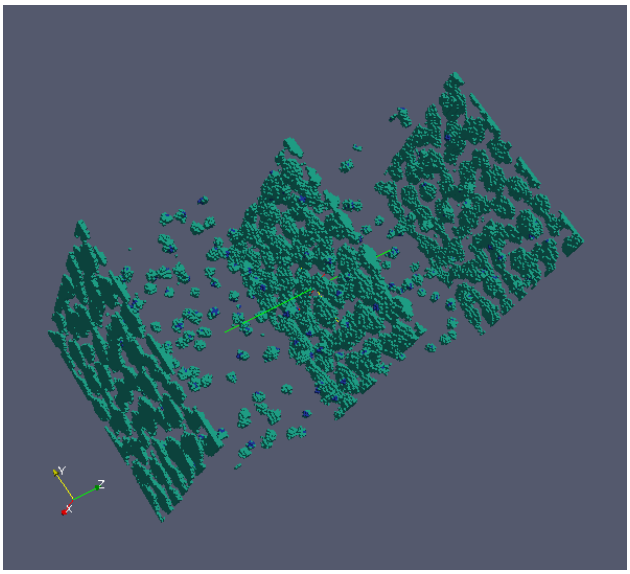


Figure 39: *The bubbles are the green features and the blue are trapping sites.*

used in the next set of simulations. The simulation size was 200 x 200 x 40. The green features in Figure 40 are bubbles at the grain boundary. The grains and the grain boundary are not imaged. Periodic boundary conditions mean that bubbles are continuous across the edge. The shape of the bubbles is shown in Figure 40 images of the grain boundary bubbles from different angles to show the bubble shapes. The bubbles are lenticular in shape as seen in real fuels. The intergranular bubble formation is shown in Figure 41. Initially small isolated bubbles form at the grain boundary. As more gas atoms were formed, diffused and precipitated at the grain boundary, the bubbles become larger and more connected. As their connectivity increases, the bubbles become more snaking in their shape. Finally, the bubbles become percolating. These bubble evolution results are in good agreement with that observed and reported in the Light Water Reactor, LWR, fuel literature. These simulations results are confirmed by determining the average bubble size as a function of fuel service time and plotted in Figure 42.

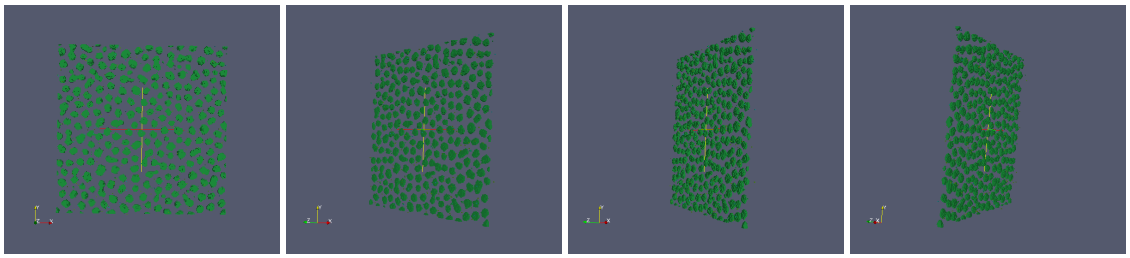


Figure 40: *The intergranular bubbles at the grain boundary have lenticular shapes.*

Initially, the bubbles nucleate and grow slowly due to gas being added to individual gas bubbles. In the later stage, bubbles grow by primarily by coalescence, although they do grow due to additional gas diffusing to the grain boundary as well. The percolation threshold is defined here as a single pore that touches all four edges of the grain boundary. One can see that bubble growth is rapid at and beyond the percolation threshold again due to rapid coalescence. This made very clear by Figure 43, a plot of the total volume fraction of bubble that percolates as a function of time. The volume fraction of the percolating bubble increases very

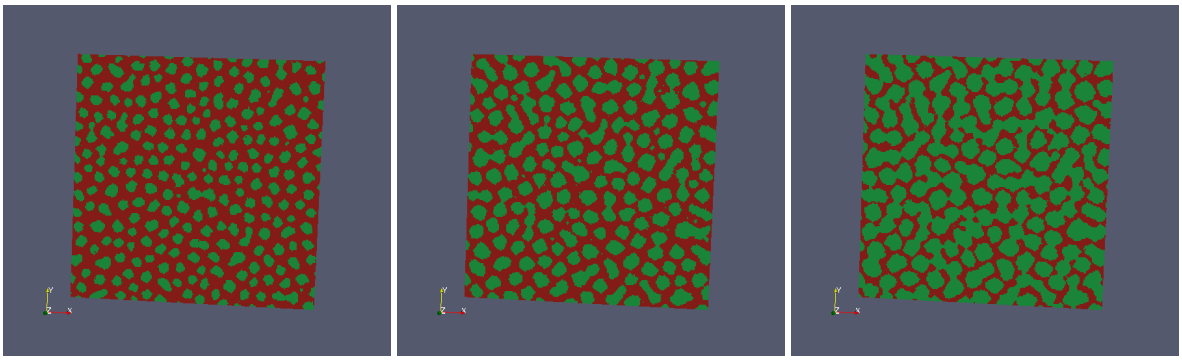


Figure 41: *The formation and growth of intergranular fission gas bubbles.*

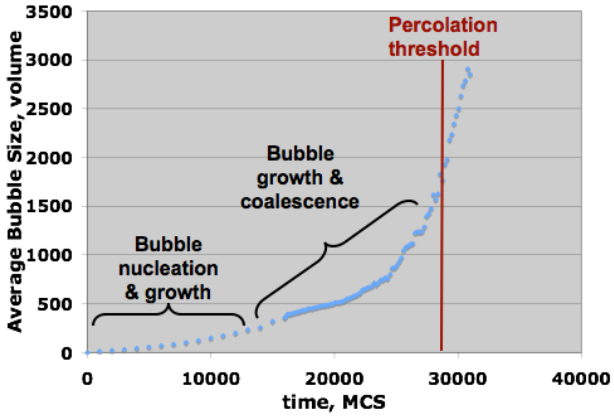


Figure 42: *Intergranular bubble growth during LWR fuel service.*

rapidly as soon as it becomes a percolating due to rapid coalescence by the surrounding bubbles. The other important parameter for LWR fuels is the area fraction of the grain boundary that is covered by the percolating bubble as it is an important engineering parameter in LWR fuels performance codes. In these simulations, the area fraction η of the grain boundary covered by the bubbles at percolation was found to be $\eta = 0.65$.

The same simulation with a polycrystalline geometry was repeated. The structure used is eight tetrakaidecdral grains with periodic boundary conditions so that they fill space as shown in Figure 44.

The polycrystalline structure on the left is the eight tetrahedral grains arranged so that they fill space. Each color corresponds to a grain, so the red colored feature in the corners are all a single grain and likewise for the other colors. The single grain shown on the right is the central grain and shows the shape of all the grains in the assembly. The same simulation as described above with gas atoms forming, diffusion, nucleating and growing was done in this geometry. Figure 45 shows the grain boundary bubble formation and growth. Only half the cube shown in Figure 45 is imaged, so that the intergranular bubbles can be seen clearly. The red bubbles on the left are all the bubble forming and growing and the blue ones on the right are only the percolating bubble. The results were very similar to that of the plane grain boundary. Initially, the intergranular bubbles nucleated and grew due to gas diffusing to the grain boundaries. Later the grain boundary bubbles grew primarily by coalescence until they started to percolate. And the volume fraction of

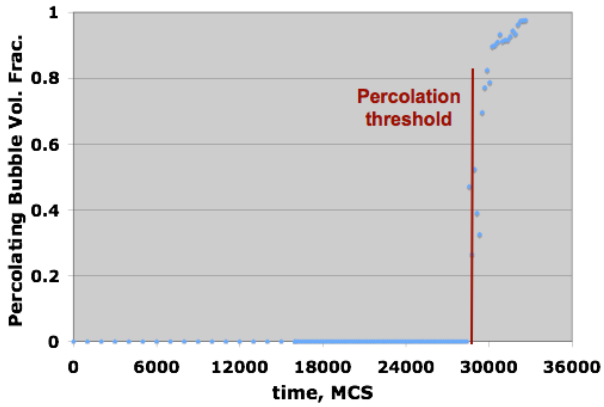


Figure 43: *The volume fraction of the percolating bubble. No bubble percolates until t= 28,000 MCS. Then there is a sudden and rapid rise in the volume fraction of the percolating bubble suggesting that almost all bubbles join up by coalescence.*

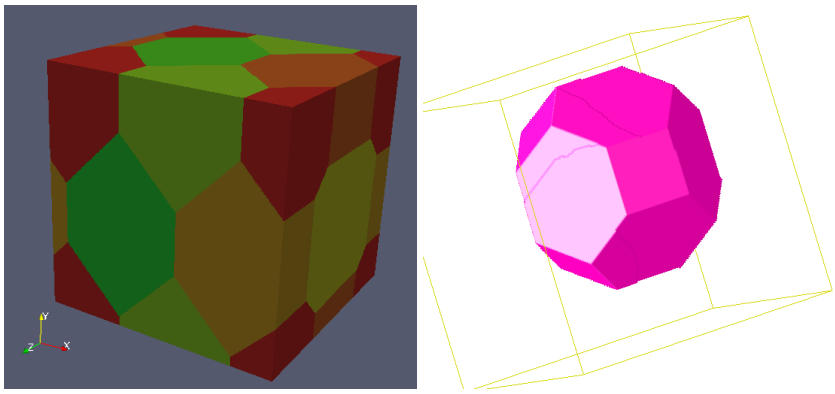


Figure 44: *The geometry for used to study grain boundary bubble formation and growth in a polycrystalline microstructure. Periodic boundary conditions are use so colored entities with the same color are the same grain and they are all the same size and shape of the tetrakaidecahedral grain shown on the right.*

the percolating bubble grew very rapidly upon percolation due to the other bubbles coalescing with it. The one big difference is the area fraction η of grain boundary covered with bubbles at the percolation threshold is lower than that for a plane grain boundary. For the polycrystalline microstructure it is $\eta = 0.53$ versus $\eta = 0.65$ for the planar grain boundary.

These results suggest that the percolation threshold may be dependent on the details of the microstructure. We had expected that bubbles would percolate along the edges before they percolate on the faces. The simulation results in Figure 45 do not support this. This will be investigated further to determine the percolation behavior of fission gas bubbles in polycrystalline microstructure in future work. k.

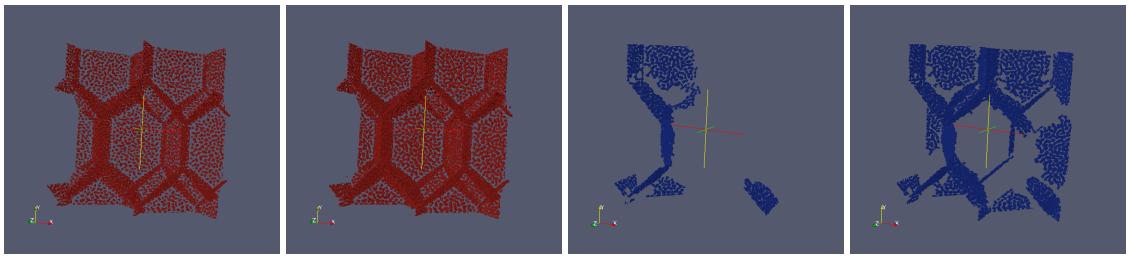


Figure 45: *Grain boundary fission gas bubbles in a polycrystalline microstructure for time t = 16,000, 24,000, 24,800 and 25,200 MCS. The red bubbles are all the bubbles and the blue bubble is the percolating bubble.*

7.6 Sintering for Nuclear Fuels Aging

This section contains a paper that will appear in the Sandia CSRI summer program proceedings for 2009. It's entitled "Parallel Monte Carlo Simulation of 3D Sintering" and was authored by Cristina Garcia-Carbona, Veena Tikare, and Steven J. Plimpton. Cristina was a summer student who visited Sandia as part of the CSRI program.

Like the previous section, the motivation for this work is from nuclear fuels modeling. This application, as implemented in SPPARKS, contains a novel parallel algorithm, developed by Cristina, for compacting the solid (removing vacancies), a key aspect for capturing the densification that results from the sintering process.

7.6.1 Abstract

A three-dimensional parallel implementation of a Monte Carlo model for the microstructure evolution during sintering is presented. The model accounts for the main phenomena occurring during sintering, including grain growth, pore migration and vacancy annihilation. The parallel implementation is based on the SPPARKS code and enables the simulation of systems with large number of particles. Several examples are shown and results are compared with a serial implementation as well as experimental data available.

7.6.2 Introduction

Sintering is a fabrication process where the fluid-like behavior of powders is exploited to build arbitrarily complex shapes. The consolidated piece is subjected to a firing process leading to strengthening and densification but is also accompanied by volume shrinkage. At the mesoscale level, sintering is the result of thermally activated adhesion processes which produce the bonding between particles and their coalescence [61]. The driving force for sintering is the reduction in surface free energy achieved by diffusional transport of material from the centers of the particles to the particle-particle neck [12].

One effective way of gaining insight into the process is to simulate the physical phenomena occurring during sintering. Different numerical simulation methods have been used, including finite element methods [74], finite difference methods [62], discrete element methods [40] and kinetic Monte Carlo models [60]. As the physico-chemical and mechanical properties of the final piece are determined by the structure of the sintered body there is an interest in studying the microstructural evolution of the material. Furthermore, the microstructural evolution provides the driving force for the deformation observed at the macroscopic level [12]. In this sense, the simplicity and versatility of kinetic Monte Carlo method makes it a sensible choice to investigate the microstructural evolution during sintering [78].

The kinetic Monte Carlo simulation of sintering has been developed in recent years and analysis for simple configurations in 2D [78] have been extended to estimation of macroscale parameters [60] as well as simulations of 3D complex configurations [76, 77]. This stochastic approach uses the Potts model to account for the local kinetics of the process, including grain growth, pore migration and vacancy diffusion and annihilation at grain boundaries. To obtain meaningful results a considerable number of particles must be considered. Nevertheless, simulation of microstructural evolution during sintering on the mesoscale with imaging of many particles sintering is challenging [77]. Therefore it is necessary to build techniques that enable a greater simulation space with capacity to include more particles.

This paper describes a parallel implementation of the kinetic Monte Carlo model for solid-state sintering of a three-dimensional powder compact. The implementation is built as an extension module for SPPARKS. SPPARKS is a kinetic Monte Carlo code designed to run efficiently on parallel computers using both rejection-free kinetic Monte Carlo and Metropolis Monte Carlo algorithms [75]. To verify the performance of the new parallel implementation, sintering for different initial configurations is simulated and results are compared with the serial version as well as some experimental data available. Results are encouraging in terms of the dynamics and the size of the problems that can be attempted.

The document is organized as follows. Section 7.6.3 describes the kinetic Monte Carlo model of sintering and the implementation under SPPARKS, and contrasts the differences between the serial and the parallel version. Next, Section 7.6.6 compares the results obtained with the serial and parallel versions for different initial configurations as well as different problem sizes. Finally, Section 7.6.7 draws some conclusions and suggests directions of future work.

7.6.3 Simulation

A powder compact is a porous medium composed by a phase of substance and a phase of voids (pores). The substance is formed by grains that are loosely in contact with each other. During sintering, the surface tension of the particles forces a mass movement that redistributes the substance and eventually leads to a reduction in the total porosity, i.e. densification with a corresponding reduction in the dimensions of the porous body. Briefly, the main mechanisms operating during sintering are [77]:

- Curvature driven grain growth in the presence of evolving porosity that inhibits grain growth by pinning.
- Pore migration by surface diffusion leading to pore shape evolution coarsening.
- Formation of vacancies, grain boundary diffusion of vacancies and vacancy annihilation leading to densification.

To simulate sintering for a three-dimensional powder compact using a Metropolis Monte Carlo model the controlling mechanisms of material flow under sintering must be emulated. Essentially three aspects are defined: a representation of the porous body, a set of events that can transform it and an energy function to drive the Metropolis algorithm used for the dynamic evolution of the kinetic Monte Carlo model.

To represent the material, a 3D cubic lattice structure (a grid) is overlaid in the simulation space and a neighborhood topology is prescribed. For each point in the grid a discrete state is assigned. Grain sites populating the lattice assume one of Q possible distinct states, the individual state is symbolized with q and the total number of states in the system is Q, thus: $q_{grain} \in \{1, 2, \ldots, Q\}$. The pore sites can assume only one state $q_{pore} = 0$. The neighborhood topology used is the 26 first neighbors in the cubic grid. In this model a vacancy is defined as a single, isolated pore site that does not have any other pore site in its neighborhood.

The events that transform the porous body correspond to the mechanisms operating during sintering. Consequently, grain growth is simulated by converting a grain site into the state of a neighboring grain chosen at random, pore migration is simulated by exchanging a pore site with a grain site, this last assuming the state of the neighboring grain site that results in the minimum possible energy for the grain site, and densification is simulated by producing vacancies in the grain boundaries and annihilating them by moving the vacancy to the surface of the material but in a way that conserves mass globally and moves the centers of mass of the grains adjacent to the site being annihilated closer together [12]. In addition, note than since this densification algorithm requires the definition of the surface of the material, periodic boundary conditions cannot be used.

The driving force for sintering is the reduction of the interfacial free energy [77]. To cast this condition in terms of the lattice configuration described, the energy of the system is given by the sum of all neighbor interaction energies of all sites:

$$E = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{n} \left(1 - \delta(q_i, q_j)\right) \tag{18}$$

where N is the total number of sites, n is the number of neighbors (26 first neighbors in a cubic grid), q_i is the state of the current site, q_j is the state of the j-th neighbor site and δ is the Kronecker delta with: $\delta(q_i = q_j) = 1$ and $\delta(q_i \neq q_j) = 0$. According to this energy definition, only unlike neighbors contribute to energy, i.e. only interfacial energy of the system is defined.

The dynamic evolution of the kinetic Monte Carlo model is driven by a reduction in (18), by means of a standard Metropolis algorithm. Therefore, every time an event is to be updated, a random number $R \in (0, 1)$ is generated and is compared with the probability P of accepting that change, if $R \leq P$ the change is accepted. The probability P is calculated as:

$$P = \begin{cases} \exp\left(\frac{-\Delta E}{k_B T}\right) & \text{for} \quad \Delta E > 0 \\ 1 & \text{for} \quad \Delta E \leq 0 \end{cases} \tag{19}$$

ΔE corresponds to the energy change, k_B is the Boltzmann constant and T is the simulation temperature, a measure of the thermal fluctuation in the system [78].

The parameterization of the model enables the consideration of different grain-pore mobility ratios, basically by varying the frequency in which the different events are attempted or using different values for T in (19). Similarly, the frequency of annihilation events f_a is adjusted inversely proportional to the average area of grain boundaries:

$$f_a \propto \frac{A_0}{A_{gb}} \tag{20}$$

In this equation A_0 stands for the average grain boundary area at the beginning of sintering and A_{gb} for the average grain boundary area at the current time. Time in the simulation is measured in terms of kinetic Monte Carlo steps, where one step corresponds to N attempted changes, being N the number of sites in the system. This simulation time is related linearly with the real sintering time.

In summary, the model described incorporates most of the characteristic phenomena in sintering: interfacial energy related to surface evolution, annihilation to introduce densification, sintering rate proportional to the pore surface area, sintering force inversely proportional to the grain size and change of dimension of the compact. Further details can be found in other references [12, 60, 78].

7.6.4 SPPARKS

This sub-section is omitted from this report, because its discussion of SPPARKS is well-covered in other sections of the report. See the original CSRI paper for details, if needed.

7.6.5 Serial vs. Parallel Implementation

During the Metropolis Monte Carlo used for simulate sintering, sites are chosen randomly while performing a batch of site-event rejections. These site-events correspond to the mechanisms operating in the sintering process that were described previously. Accordingly, once a local site is selected:

- If it corresponds to a grain site: a grain growth step is attempted. The state of the grain site is converted into the state of a neighboring grain site chosen at random.
- If it corresponds to a pore site: a pore migration step or a vacancy annihilation step is attempted. The pore migration step is simulated by exchanging a pore site with a grain site, this last assuming the state of the neighboring grain site that results in the possible minimum energy for the grain site. The vacancy annihilation step is computed by moving the vacancy to the surface of the material, while displacing the centers of mass of the grains adjacent to the site being annihilated closer together.

All these events are accepted or rejected according to (19), i.e. using a local criterion. Consequently, almost all the events can proceed independently in every processor [71]. The only exception is the annihilation step, because it requires the coordinate modification of sites along a path that, in general, is distributed along several processors. Hence, the main difference between the serial and the parallel implementation is the handling of the annihilation step.

In the serial version, if the vacancy annihilation is to be performed the center of mass of the adjacent grain is calculated, as well as the vacancy new position and the annihilation is performed immediately. The path of the annihilation starts in the vacancy current position, goes through the center of mass of the adjacent grain and continues in the same direction until arriving to the surface of the specimen (See Figure 46). Thus, the vacancy new position corresponds to the position of the last grain site in the annihilation path. At the same time, all the intermediate sites in the path are shifted one position in the direction toward the vacancy current position. In this way, mass is globally conserved, the centers of mass of the adjacent grains are moved closer together and the compact shrinks.

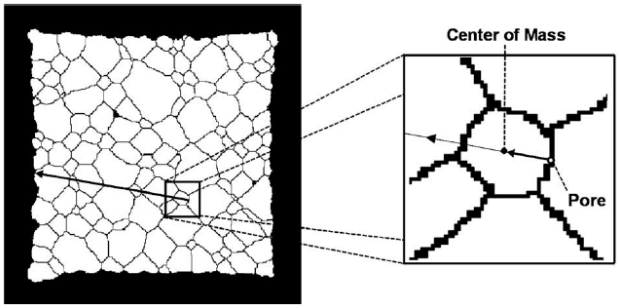


Figure 46: *2D schematic of vacancy annihilation. Black color denotes pores and grain boundaries; white denotes grains. Image taken from [12].*

In the parallel version, since the lattice is distributed spatially over several processors, long range communications are required to perform a vacancy annihilation step. Thus, instead of computing the annihilation

immediately it is registered in a list of pending annihilations to be updated at the end of the current batch of site-event rejections. At the end of the batch, the pending list of annihilations is processed normally: calculating the center of mass of the adjacent grain, the vacancy new position and the path of the annihilation. The annihilations in the list are performed one after the other. While performing the batch of site-event rejections, a flag is used to signal if the vacancy is pending annihilation and no further operations are allowed for the site. Nevertheless, it is possible that the actual processing of the annihilations in the list cause the shifting in vacancies pending annihilation. In those cases, as the vacancy is no longer there no annihilation is performed. The following description summarizes the processing of the list of pending annihilations.

a Each processor verifies that the local vacancies pending for annihilation are still vacancies and determines the adjacent grain.

b The list of states of the adjacent grains for all the annihilations pending is gathered in all processors.

c Each processor calculates the local center of mass of each one of the adjacent grains.

d The center of mass of all the adjacent grains is reduced (calculated) in all processors.

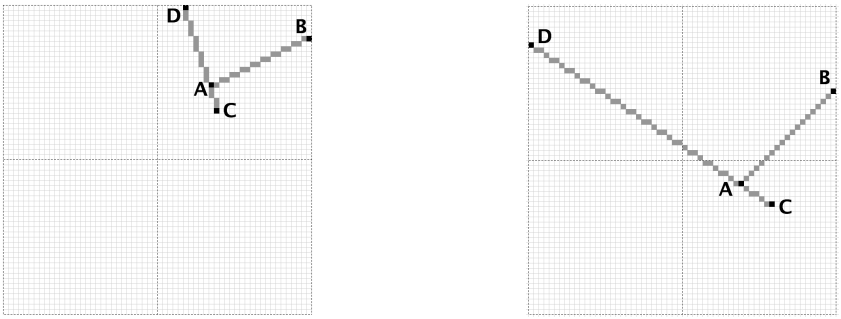
e Each processor determines the new position of the local vacancies pending.

f If the vacancy current position and the new position are in the same processor, the path of the annihilation is calculated and the sites traversed are updated. If current and new position are in different processors, the initial position, the direction of the annihilation path and the number of discrete steps to the final position are stored in a local list.

g A buffer is created and the annihilation paths that cross multiple processors are gathered in all processors.

h Each processor follows every annihilation path in the list, consequently any processor knows which sites in all the domain are being modified. When a processor owns part of the sites to be updated, it updates them. It also sends the state of the first local element to the previous processor in the path and waits to receive the update for the last local element from the next processor in the path.

Figures 47(a) and 47(b) demonstrate some of the conflicting cases occurring while updating the list of annihilations in the parallel implementation. These schematics, drawn for simplicity in 2D and without ghost cells, show a domain divided between four processors and two annihilation paths to be updated: path $\overrightarrow{AB}$ starting at the vacancy site marked A and ending at the border site (surface) marked B and path $\overrightarrow{CD}$ starting at the vacancy site marked C and ending at border site marked D. Annihilation paths are contained in one processor in Figure 47(a), while in Figure 47(b) they cross multiple processors. If these annihilations are ordered in the pending list in such a way that annihilation $\overrightarrow{AB}$ is first, then both can be performed. On the contrary, if annihilation $\overrightarrow{CD}$ is first in the list, then it is unlikely that annihilation $\overrightarrow{AB}$ takes place in the one-processor case and definitively it is not taking place in the multi-processors case. In the one-processor case, if after updating annihilation $\overrightarrow{CD}$ A is still a pore site, then updating for $\overrightarrow{AB}$ proceeds. However, if A is no longer a pore site then there is no point in processing it as a vacancy being annihilated. In the multi-processors case, even when all processors are following the path of annihilations (to know how and when to exchange border information) only the bottom-right processor knows the new state of site A, only it could verify if it is still a pore site. Thus, rather than incurring in the overhead of communicating the new state for site A, there is a local book-keeping that allows each processor to detect that the site has been changed and avoid computing an annihilation for a vacancy that could no longer be there.



(a) One-processor (b) Multi-processors

Figure 47: *Conflicting cases when updating the list of pending annihilations. Left: required updates contained in one processor. Right: required updates crossing multiple processors.*

7.6.6 Results

To verify the performance of the new parallel implementation, sintering simulations have been conducted for three different initial configurations. The first case studied corresponds to a random initialization and a lattice of: $200 \times 200 \times 50$. The second case is a close packing of spheres in a lattice of: $200 \times 200 \times 200$. The third case is based on a microstructure obtained from microtomographic imaging of a real powder compact of copper particles [77] and a lattice of: $403 \times 403 \times 95$. This section is dedicated to present the results computed with the serial as well as the parallel version. All images corresponding to 3D views and 2D slices are generated with ParaView, an open-source visualization application (http://www.paraview.org).

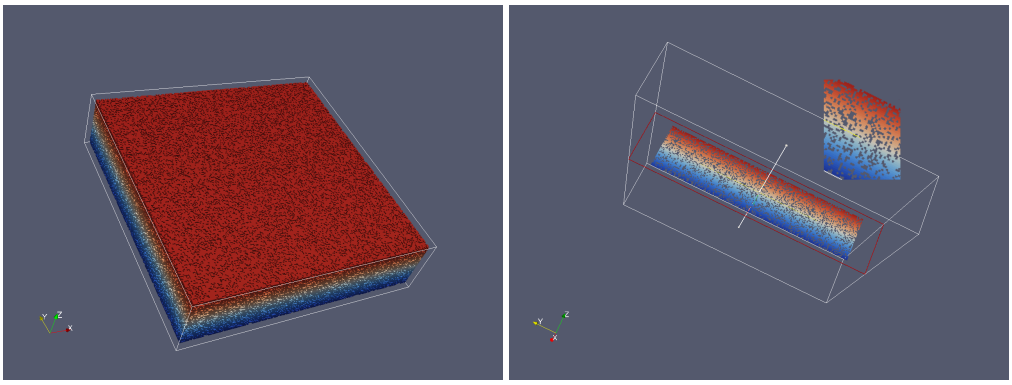
The parameters used to run the simulations are included in Table II. There, RFreq stands for the relative frequency for attempting the event, k_BT for the thermal factor to be applied in (19) and MCS Start for the Monte Carlo step to start the annihilation of vacancies. The parameters are kept almost equal, the only difference is in the first case where the microstructure has to be evolved, i.e grains should be available, before performing annihilation events. Thus, smaller factors of temperature for grain growth and pore migration are used to decrease the thermal fluctuation in the system while evolving the grain structure.

Table II: *Simulation Parameters*

	Grain Growth		Pore Migration		Annihilation		
Lattice Size	RFreq	k_BT	RFreq	k_BT	RFreq	k_BT	MCS Start
$200 \times 200 \times 50$	0.15	0.1	0.08	0.7	0.77	15.0	616
$200 \times 200 \times 200$	0.15	1.0	0.08	1.0	0.77	15.0	0
$403 \times 403 \times 95$	0.15	1.0	0.08	1.0	0.77	15.0	0

Figure 48(a) displays the initial configuration for the first case studied and Figure 48(a) contains a 2D view of the corresponding microstructure. For the initialization, each site inside the simulation space has been assigned with a positive number (grain site) or a zero (pore site) at random, to obtain a starting density of $\sim$70%. All the grain sites start with a different state, thus rather than a real grain structure the starting configuration is a collection of isolated one-site grains. Consequently, just grain growth and pore migration events are performed for a number of Monte Carlo steps until a rough grain structure is obtained and attempting vacancy annihilation makes sense. Figure 49 shows 2D slices of the resulting microstructure obtained for serial and parallel versions. Figure 50(a) compares the densification curves vs. Monte Carlo

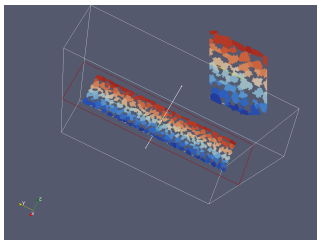
steps for both cases. Different initializations with density of 70% are generated and averaged results are reported. It can be noticed that the evolution is almost identical. Figure 50(b) plots the evolution of the grain size (radius) vs. Monte Carlo steps for both cases and again, there is a complete agreement between the two.



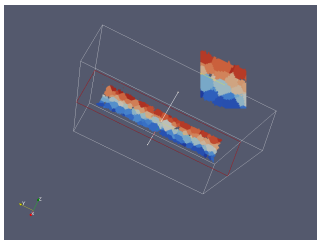
(a) 3D View - Initial State

(b) Slice - Initial State

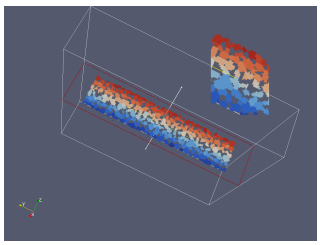
Figure 48: *Initial configuration - Random Initialization. Density: 70.0%*



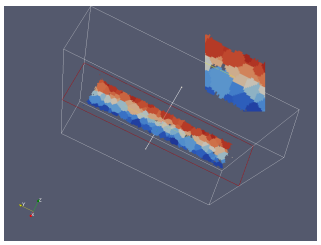
(a) Serial - Slice - 616 MCS



(b) Serial - Slice - 1231 MCS

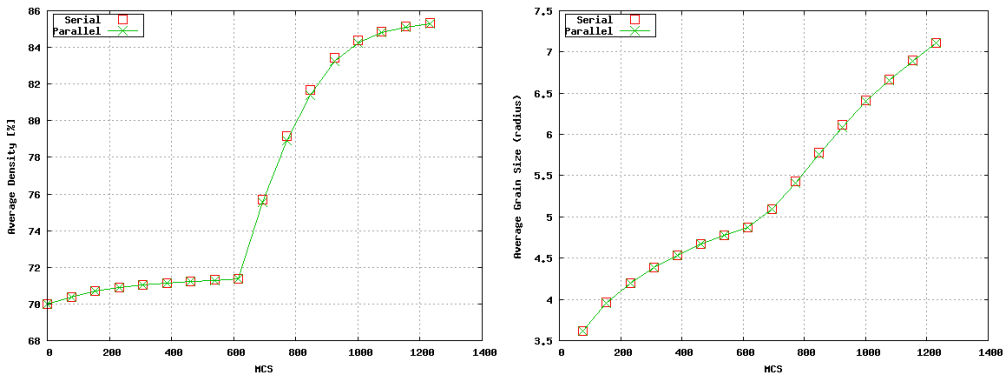


(c) Parallel - Slice - 616 MCS



(d) Parallel - Slice - 1231 MCS

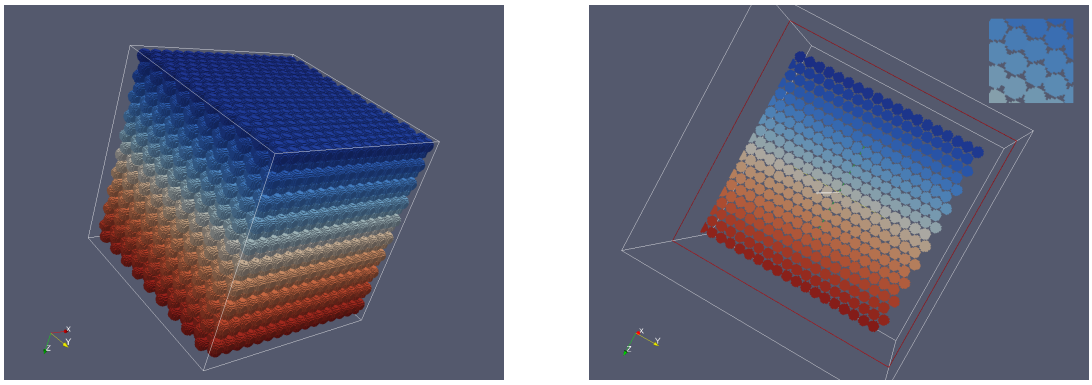
Figure 49: *Random Initialization - Initial Density: 70%. Microstructures at 616 and 1231 Monte Carlo Steps. Top: serial version, density: 71.3% and 85.3% respectively. Bottom: parallel version, density: 71.1% and 84.9% respecively*



(a) Density (b) Grain Size

Figure 50: *Evolution of Densification and Grain Size for random initialization*

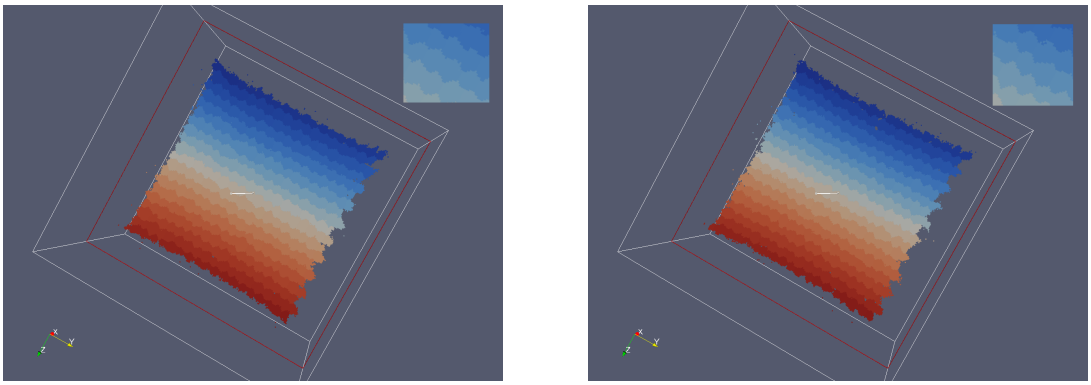
Figure 51 displays the initial configuration used for the close packing of spheres. Starting from an initial density of 73.1%, both versions reach a density of 94% around 924 Monte Carlo steps. The corresponding microstructures are included in Figure 52. Curves of evolution of densification and grain size (radius) vs. Monte Carlo steps for both cases can be encountered in Figure 53(a) and Figure 53(b), respectively. All the results obtained are similar.



(a) 3D View (b) Slice

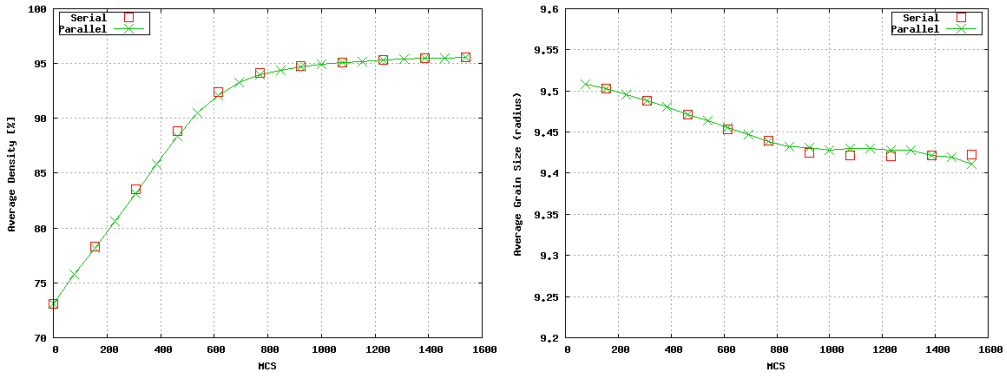
Figure 51: *Initial configuration - Close Packing of Spheres. Density: 73.1%*

Next, simulation results are compared to experimental data. An input image obtained from micrographic imaging of real powder compact of copper particles is pre-processed to extrapolate a grain structure into the original 3D structure as explained in [77]. The outcome, shown in Figure 54, is used as the starting microstructure for the simulation. Results for the parallel implementation can be found in Figure 55. Results of a serial implementation of the model originally published in [77] are included in Figure 56 to facilitate comparisons. Additionally, Figure 57(a) shows the densification curve obtained for the simulation and Figure 57(b) compares grain size (radius) distributions for serial and parallel versions as well as real data



(a) Serial - Slice (b) Parallel - Slice

Figure 52: *Sintering - Close Packing of Spheres: 924 Monte Carlo steps. Left: serial version, Density: 94.8%. Right: parallel version, density 94.7 %.*



(a) Density (b) Grain Size

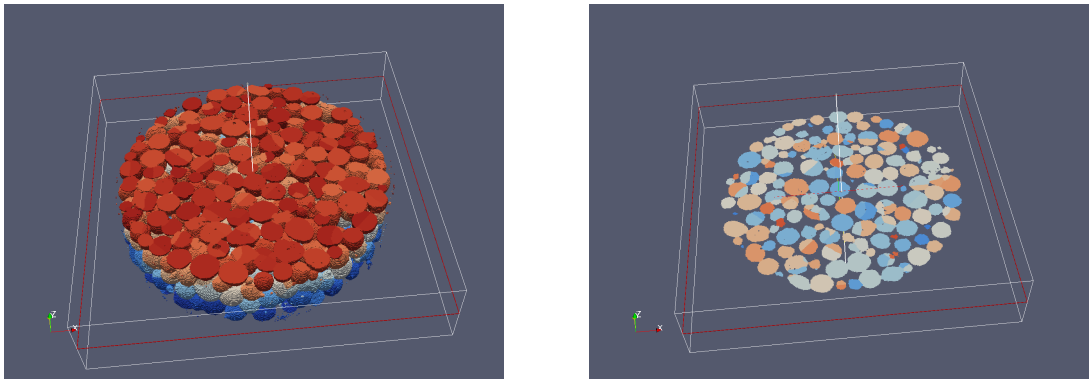
Figure 53: *Evolution of Densification and Grain Size for close packing of spheres*

calculated from the microtomographic images of Cu sintering (curves for serial and Cu sintering have been adapted from [77]). The distribution for the parallel version is computed at 82.9 % density, while distributions for serial and Cu sintering are calculated at 83.8% density. The parallel version seems to have a rate of grain growth similar to the real data, although it has more fine grains. Overall the microstructure evolution observed in the parallel version is compatible to that occurring in the real system.

Finally, Table III displays average times for execution in a cluster of 32 dual Intel Xeon processors, with 2.8 GHz and 4 GB RAM. The * indicates that the memory requirements have exceeded the available resources in one or several nodes and that the computation could not be done. The scaling in time is not ideal, ways to improve performance should be explored.

Table III: *Simulation Times in [sec]*

	Number of Processors						
Lattice Size	1	2	4	8	16	32	64
$200 \times 200 \times 50$	1731	2211	1138	570	349	262	314
$200 \times 200 \times 200$	*	*	15751	8260	4971	2519	1913
$403 \times 403 \times 95$	*	*	*	20660	10236	5674	3287



(a) 3D View

(b) Slice

Figure 54: *Microtomographic image with a grain structure extrapolated into the 3D structure. Density: 69.1%*

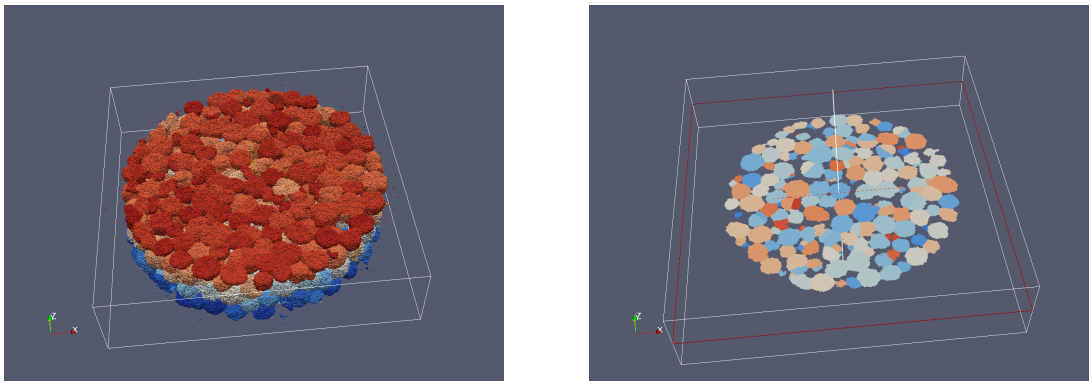
7.6.7 Conclusions

A parallel implementation of a Monte Carlo algorithm for 3D sintering simulation has been described. To verify its performance, different initial configurations were simulated and results were compared with the serial version, and when available, with experimental data. There is an agreement, in the statistical sense, between the expected (serial) and obtained (parallel) microstructure, as well as in the average grain size evolution and the densification curves. Results are encouraging in terms of the dynamics and the size of the problems that can be attempted. Nevertheless, future efforts must be directed toward the reduction of the overhead caused by the processing of the list of pending annihilations.

7.7 Thin Film Deposition and Growth

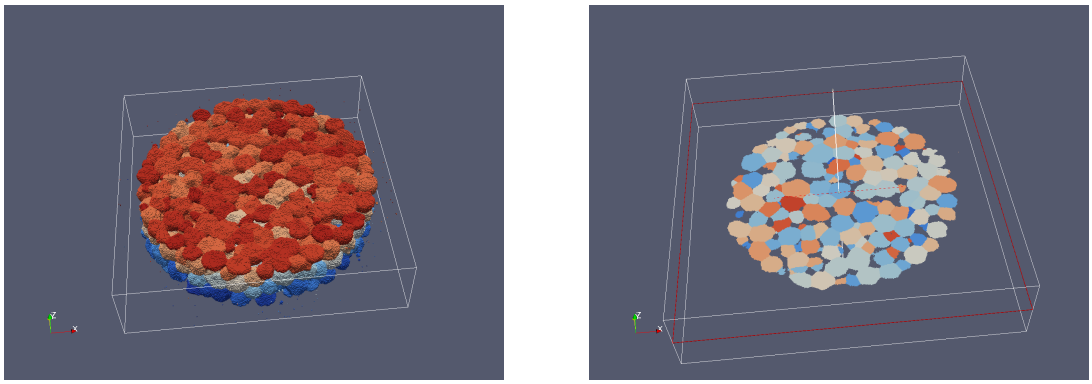
7.7.1 Introduction

Thin film applications emerge across a wide range of technology, from highly efficient solid state light sources to microelectronics and photonics applications to mechanical wear and/or thermal resistance. For many applications, a high degree of control is desired over film properties including surface/interface roughness, film texture, grain structure, and defect content. For example, when TiN is used as a diffusion barrier layer in semiconductor devices, $< 111 >$ texture is preferred. However, when TiN is used as a mechanical resistance coating, $< 100 >$ texture is desired. Controlling void content is critical to a number of thin film applications where voids created during deposition lead to shortened device lifetimes. Interface roughness is important to many photovoltaic, microelectronic, and optical thin film applications where multilayered



(a) 3D View - 1200 MCS

(b) Slice - 1200 MCS



(c) 3D View - 3600 MCS

(d) Slice - 3600 MCS

Figure 55: *Sintering - Parallel Implementation - From Microtomographic Image. Top: 1200 Monte Carlo steps. Density: 78.7%. Bottom: 3600 Monte Carlo steps. Density: 82.9%*

films are deposited. For example, when a thin layer of Ta is deposited on top of Cu as a diffusion barrier in semiconductor applications, a very chemically and structurally sharp interface is desired. These film characteristics can be controlled by altering deposition conditions but, for most systems, a one to one mapping between process variables and resultant film properties does not exist. Creating such a mapping via experiments is difficult and costly so models of thin film deposition bear great potential for increasing control over thin film engineering.

Analytical descriptions have been advanced of the thermodynamics and kinetics of thin film deposition and growth. Such models, while highly useful in describing some aspects of film growth, suffer from deficiencies inherently linked to not fully accounting for atomistic scale phenomena responsible for a given observed behavior. The same is true for continuum scale simulations of film growth. Atomic scale models of film growth reveal fundamental mechanisms important to determining observed behavior. They can be used to develop highly robust and accurate analytical descriptions, paving the way to optimized constitutive relations for continuum simulations of film growth. Models based on *ab initio* or even classical force fields, like molecular dynamics, suffer from being temporally too far removed from macroscopic deposition rates used in experiments. As an example, consider a MD simulation of film deposition onto a model substrate

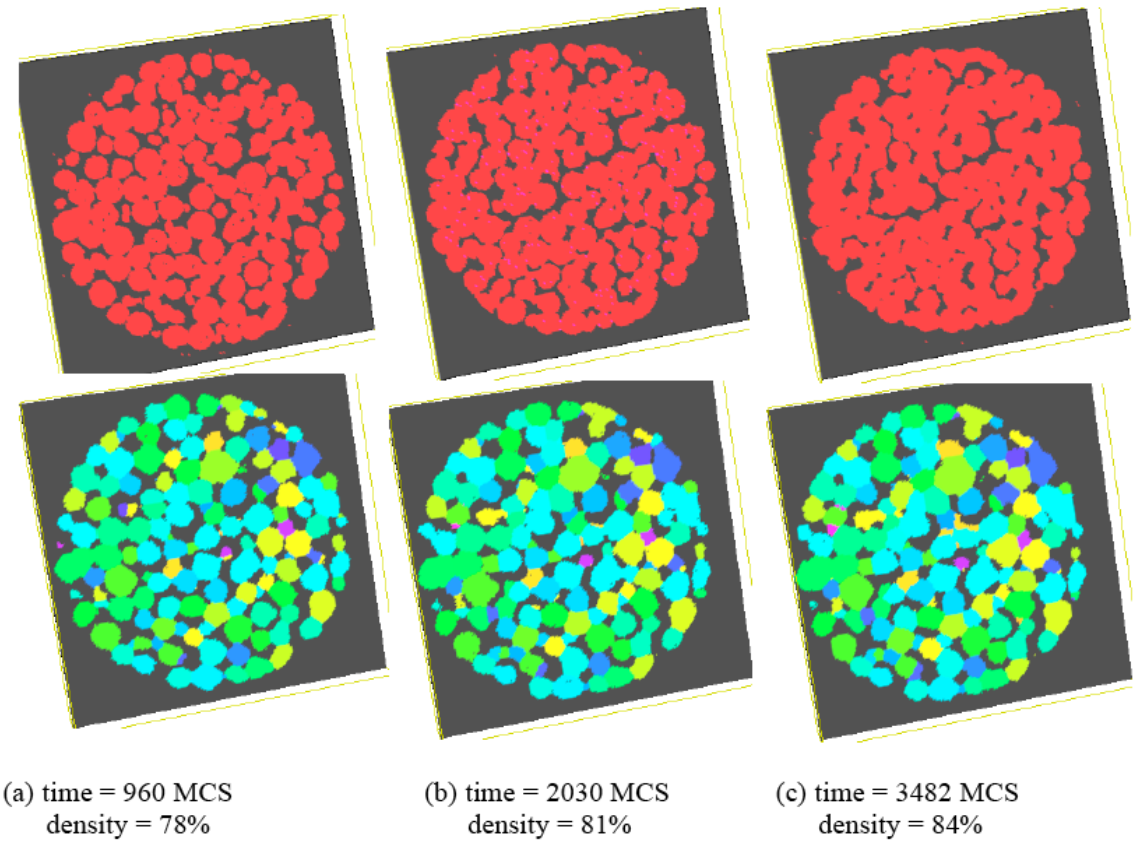
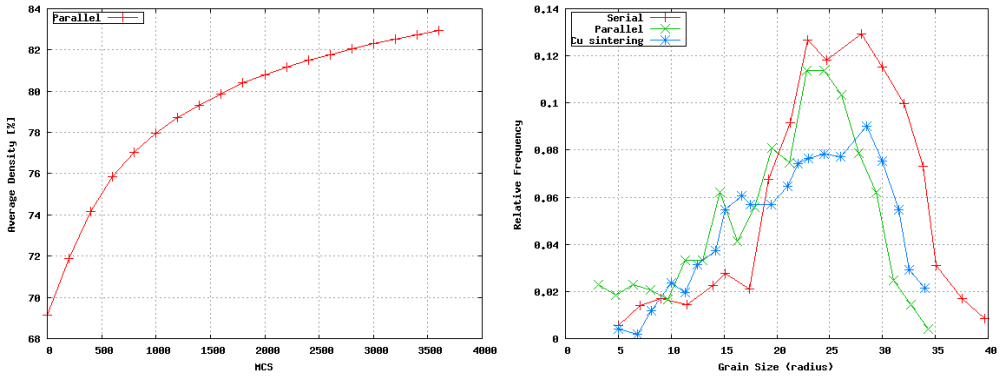


Figure 56: *Sintering - From Microtomographic Image. Top: Slices through the 3D microtomographic images. Bottom: Slices through the simulation - Serial Implementation. Image taken from [77].*

surface; using currently available, fairly standard computational resources, a reasonable deposition rate (film thickness per time) that can be modeled is of order 1 m/s [27]. This is many orders of magnitude greater than even the highest typically used experimental deposition rates (e.g. 10 nm/s in sputter or e-beam evaporation methods of metal film deposition). Because experiments use much slower deposition rates, a newly deposited atom has a relatively long time to diffuse on the growing surface before another atom is deposited. In MD simulations, the number of diffusive hops that occurs between each deposition event is significantly smaller than in experiment. This places great limitations on MD simulations for assessing film growth. Despite such limitations, a considerable body of work exists using this method to explore film growth at the atomic scale and many useful observations have been made. Nonetheless, an atomic scale model capable of examining film growth at realistic deposition rates bears great potential for improving the ability to engineer film structure and optimize the performance of devices dependent upon such films. The KMC method offers a solution in that models can be formulated with atomistic spatial resolution that are capable of accessing macroscale temporal regimes.

In this section we describe capabilities that have been built into SPPARKS for executing KMC models of film deposition and surface diffusion. These models can be classified as on lattice versus off lattice. On lattice models of atomic scale deposition and diffusion suffer from an inability to resolve, for instance, dislocation formation in the depositing crystal because this would entail atoms occupying non lattice positions. Nonetheless, this class of KMC models offer tremendous insight into connections between film morphology and film growth processing variables including temperature, angle of incidence, and deposition rate. Off



(a) Density

(b) Grain Size Distribution

Figure 57: *Left: densification for evolution of microstructure interpolated in microtomographic image. Right: comparison of grain size distribution (radius) for 82.9 % density (parallel) vs. 83.8% density (serial and Cu sintering adapted from [77]).*

lattice models are significantly more complex because finding candidate diffusion events, or hops, requires a robust algorithm capable of searching, in a given atom's neighborhood, for vacant sites amongst an arbitrary, continuous space ensemble of atoms. In addition, when a neighboring diffusion site is identified, the barrier for that hop in the given local environment must be computed, typically on the fly. In the following sections, details of a 2D and 3D on lattice model of film growth are presented along with representative results for both. This is followed by a description of off lattice KMC models for surface deposition. Off lattice model implementation in SPPARKS is an ongoing effort.

7.7.2 Motivation and Description

Much of the morphological behavior exhibited during thin film growth can be interpreted as a competition between diffusion and deposition. This can be thought of as the ratio of the number of diffusive events (or hops) to the number of deposition events; this ratio is determined by the deposition rate, the temperature, and the relevant energetics of diffusive hops on the given surface at the given state point. When the ratio of diffusive hops to deposition events is low, then atoms arrive at the surface and, relative to the deposition time, do not diffuse very far. In this regime, one expects a relatively rough surface because atoms do not diffuse to lower energy positions in the time scale of deposition; as such, material is deposited on - and essentially traps - higher energy, rougher atomic scale structures in the growing film. The opposite regime has significant diffusion occurring between deposition events and, in this case, a film structure is expected with lower energy atomic configurations. Atoms will minimize their number of missing bonds; this, in turn, means smoother film surfaces during growth. Such morphological transitions with changing growth parameters have been observed; indeed, a good deal of understanding exists about growth processing and film morphology as evidenced by the tremendous success of various technologies dependent upon controlling film growth. However, much is still unknown and a modeling tool capable of assisting experiments in understanding growth in a given system bears great potential for advancing film growth engineering science.

For a KMC model of growth, the two types of events are diffusive hops and new atom additions to the system to represent deposition. The diffusive hop part of this application is very similar to what was modeled for the nanoporous metal application discussed in a preceding section. Indeed, the diffusion code machinery

for the surface growth application discussed in this section is the same as was used for the nanoporous application. A difference between the nanoporous material application and surface growth applications is that one must properly implement the attempt frequency for the often disparate deposition event rate and diffusion event rates. The deposition rate is an input parameter. In order to scale the diffusion event rates properly relative to this, some temporal notion of diffusive time scale must be established.

This can be accomplished by assuming that the system is well described by Transition State Theory. This gives a rate law $w_i = \frac{kT}{h} exp(\frac{-\Delta F_i}{kT})$, where w_i is the rate for process i, kT is the product of the Boltzmann constant and temperature, h is Planck's constant, and ΔF_i is the change in free energy between the initial state and the saddle point state as a system traverses the reaction coordinate for molecular process i. The prefactor $\frac{kT}{h}$ can be loosely interpreted as the inverse of the time required for a particle with thermal energy kT to traverse the physical path associated with the reaction coordinate of process i. The second term reflects the probability that the energy distribution amongst ensemble degrees of freedom is such that there exists sufficient energy local to the event site for the particle to surmount the free energy barrier ΔF_i. Since $\Delta F_i = \Delta U_i - T\Delta S_i$, where ΔU_i and ΔS_i are the change in internal energy and entropy for process i, the rate law can be recast $w_i = w_{i0} exp(\frac{-\Delta U_i}{kT})$, where the prefactor is now an attempt frequency, $w_{i0} = \frac{kT}{h} exp(\frac{\Delta S_i}{k})$. The entropy term is typically small and estimating $w_{i0} \sim \frac{kT}{h}$ is reasonable. Detailed balance dictates that w_{i0} is the same for forward and reverse events; indeed, many KMC models of diffusion use a single value of for all diffusive events with a typical value being $w_{i0} \sim 5\ ps^{-1}$. One exception to this could be for a specific diffusive hop for which multiple mechanisms are identified. In this case, the lowest energy barrier mechanism is assumed for ΔU_i and additional mechanisms are absorbed into the $exp(\frac{\Delta S_i}{k})$ term in w_{i0}, leading perhaps to an altered value of w_{i0} for the given event.

In the current SPPARKS implementation, w_{i0} is assumed to be 1 for all diffusive hop events and the input deposition rate and total run time must be scaled accordingly to achieve the relative rate and total amount of deposited material desired. For models of experimentally used deposition rates, this introduces a somewhat cumbersome scaling factor of order 10^{12}; however, future code developments will permit users to specify event dependent attempt frequencies. This can be particularly important, for instance, for Schwoebel diffusive hops where multiple mechanisms for a given hop have been identified. As discussed above, this can manifest as an attempt frequency different from that of other diffusive hops.

7.7.3 On Lattice Simulation Results

We modeled 2D deposition using an on lattice model in SPPARKS; the lattice was triangular and the rate law used only energy barriers for diffusive hops. The barrier magnitudes selected were based on existing literature for a 2D model of Ni deposition that calculated barriers using an embedded atom method interatomic potential [91]. Note that the input energy barriers are not equal for forward and reverse events nor are there any simple function of the coordination number for the sites involved; this is a reflection of the multibody interatomic interaction prescribed by the embedded atom method. Events modeled included atoms diffusing via the Schwoebel-Ehrlich mechanism; there are referred to as Schwoebel hops. These were discussed previously but, briefly, they model a perhaps concerted diffusive event by which an adatom on a surface step effectively hops to a vacant site in the layer below the adatom, i.e. the layer forming the surface step. A concerted and a non-concerted mechanism have been proposed for this event. A different flavor of Schwoebel-Ehrlich barrier has also been identified; in this case, an adatom that is on top of an island terrace hops to an adsorption site on a side wall of the island. This mechanism has been identified as the first step necessary for an atom to diffuse down the side of the island toward ledge and kink sites that exist where the island meets a terrace plane [49]. Again, two mechanisms are proposed for this: one concerted and one not. In these Schwoebel hop cases, the final vacant site is two neighbors away from the initial site of the adatom; the current SPPARKS implementation only permits Schwoebel hops to occur from an initial site with a user

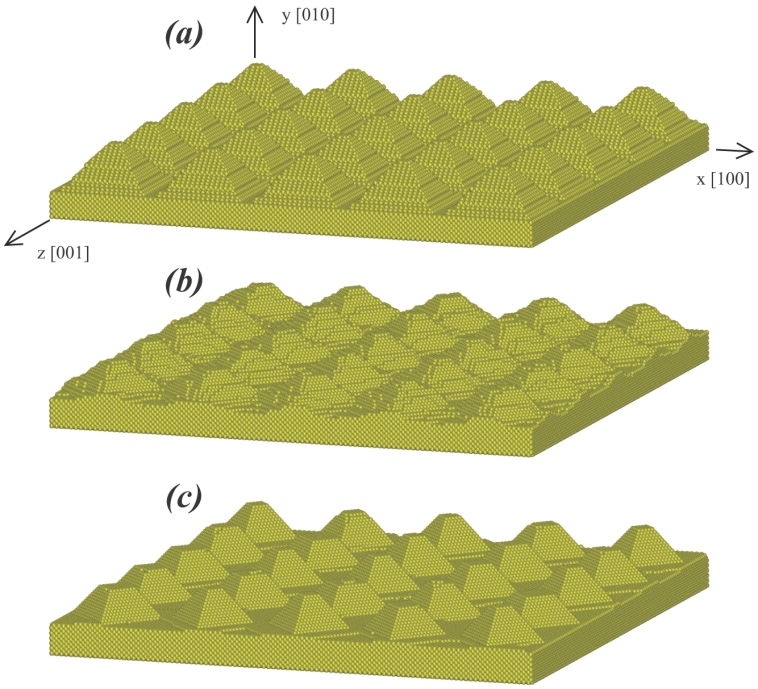


Figure 58: Evolution of a rough surface: (top) initial configuration with sinusoidal hillocks; (middle) after 10,000 secs at 500 K, no Schwoebel hops; and (bottom) after 10,000 secs at 500 K, with Schwoebel hops.

specified maximum coordination. The user also specifies a required minimum coordination for the final site.

To ensure that the KMC realistically simulate the surface diffusion processes including the Schwoebel jumps, test cases of evolution of a rough surface during annealing were first simulated. A three dimension FCC crystal was used. This crystal is aligned in the cubic orientation, i.e., x = [100], y = [010], and z = [001]. The surface is assumed to be in the y-direction, and the horizontal dimension has 100 fcc unit cells in the x-direction and 100 unit cells in the z-direction. An infinite surface was simulated using periodic boundary conditions in both the x- and z-directions, with a free surface boundary condition in the y-direction. To initialize a rough surface, sinoidal wave functions that included five periods over the length scale of the system was projected in both the x- and z-directions; atoms above the sinoidal functions were removed. The initial crystal thus created is shown in Figure 58(top).

To set up a simple, but physical surface diffusion problem, we assume that the energy of the system is a sum of the nearest neighbor bonding energy with a value of -0.2 eV/bond. A constant atom vibration frequency of 10^{12} /sec was used. A constant activation energy barrier parameter Q = 1.0 eV is superimposed upon the energy change during a nearest neighbor jump event to determine the real activation energy barrier for this event. Similarly, a constant activation energy barrier parameter Q_S = 1.0 eV is superimposed on the energy change during a Schwoebel jump event to determine the real activation energy barrier for the Schwoebel event. For the Schwoebel jumps permitted here, we further impose the constraint that the jumping atom must have no more than 6 nearest neighbors prior to the jump and at least 3 neighbors after the jump. Under these conditions, annealing simulations were performed at a temperature of 500 K with Schwoebel jumps both on and off. The configurations obtained after 10,000 sec. of annealing are shown in Figures 58(middle) and 58(bottom) for Schwoebel hops off and on respectively. Both Figures 58(middle) and 58(bottom) indicate that the unstable surface asperities seen in Figure 58(top) turned 45° to maximize the stable 111 type of surface. This is a good validation that the KMC simulations, at least

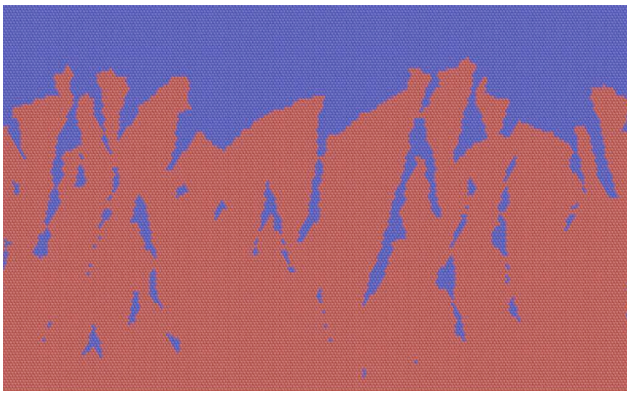


Figure 59: *Rendering of an atomistic KMC model of 2D Ni film deposition and diffusion, where Schwoebel hops are suppressed and T=250 K. Red sites are atoms and blue sites are vacancies; the total growth time is 2 s.*

qualitatively, correctly simulate the evolution. Comparing Figures 58(middle) and 58(bottom) indicates that without Schwoebel jobs, the system lacks the ability to anneal out terraces. But with Schwoebel hops, terraces were all annealed out and clear 111 and 100 facets developed. Such phenomena are exactly what is expected from Schwoebel hops, providing another strong validation of the code.

Deposition onto the surface was modeled by initially selecting a random position above the substrate (i.e. the growing film surface). In 2D(3D) simulations, the y(z) position of the incident atom is assigned to be the top of the simulation box and the x(x and y) coordinate(s) are randomly distributed across the simulation cell. A vector, dictated by the user supplied deposition angle, defines the incident atom's path toward the substrate. Lattice sites that are vacant and have a user supplied minimum coordination with occupied lattice sites are found that are also within some cutoff distance from the incident atom's trajectory. This cutoff distance is user specified; a larger value increases the chance that a deposition site will be found. The cutoff distance also permits one to investigate adatom steering effects. Once sites satisfying these criterion have been identified, the site in this set that is the shortest distance projected ald along the incident atom's trajectory is selected as the deposition site.

Both for validation purposes and to illustrate morphological film transitions that can be achieved with models implemented in SPPARKS, three deposition conditions were simulated using a 2D on lattice model. In all cases the deposition angle, defined as the angle between the incident atom's trajectory and the surface normal, was zero; the deposition rate was 10 nm/s (this is a value typically used in sputter deposition or e beam evaporation deposition of metals). The simulation cell dimension in x is 50 nm. In the first simulation, the temperature was relatively low ($T = 250$ K) and Schwoebel hops were suppressed. This can happen, for instance, due to an impurity effect or because some surfactant is used during growth. These conditions favor significant surface roughness and Fig. 59 shows this is what was obtained from SPPARKS. Red lattice sites signify atoms and blue lattice sites are vacant. Significant porosity and a rough surface, with vacancy channels extending preferentially along the growth direction, are observed. Perhaps one of the most notable features of the system depicted in Fig. 59 is the total simulated time of 2 s, corresponding to 20 nm of film thickness for the specified growth rate. This time of simulation is many orders of magnitude beyond what is obtainable from, for instance, standard MD methods and this illustrates why KMC models really permit one to cross the temporal no man's land between MD and continuum descriptions.

Two additional state points were modeled using the 2D on lattice model presented above. In both of these cases, Schwoebel hops were enabled; in the first case with Schwoebel hops active (Fig. 60), T=250

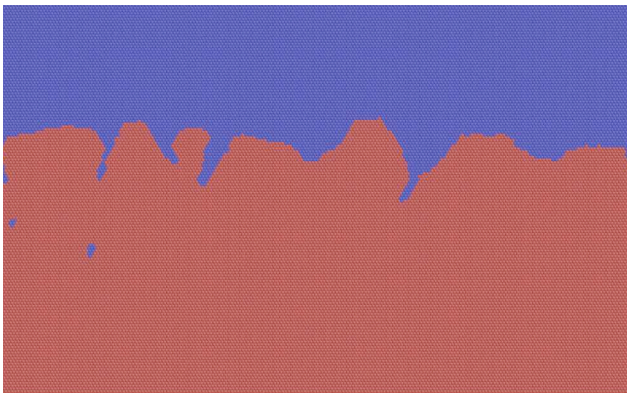


Figure 60: *Rendering of an atomistic KMC model of 2D Ni film deposition and diffusion, where Schwoebel hops are active and T=250 K. Red sites are atoms and blue sites are vacancies; the total growth time is 2 s.*

K while in the second case with Schwoebel hops (Fig. 61) T=600 K. The same amount of material (and time of deposition) was modeled in all three cases. It is clear from comparison of Figs. 59 and 60 that Schwoebel hop mechanisms of atomic transport - parameterized as they are here - lead to the evolution of smoother surfaces. Less porosity is also seen in Fig. 60 compared to Fig. 59, again reflective of a lower energy structure achieved during deposition. Elevating temperature lowers porosity even further and leads to a significantly smoother surface, as illustrated in Fig. 61. To illustrate the relationship between film morphology and the ratio of diffusion to deposition events, the ratio for the three systems studied - in order from no Schwoebel (T=250 K), to yes Schwoebel (T=250 K), to yes Schwoebel (T=600 K) - was 6.1, 3.1, and ~ 72000. The smooth surface obtained in the last system correlates with a very high ratio of diffusive to deposition events. The ordering of the first two numbers at first seems a bit counterintuitive since enabling Schwoebel hops actually lowers the total number of diffusive events (recall that the number of deposition events in all three cases is roughly equal). Indeed, this low T system exhibits ~ 6500 Schwoebel hops and, because of these, the number of regular diffusive hops is reduced by over half (~ 65000). This dramatic reduction in regular hops comes as a result of the surface smoothing generated by Schwoebel hops. These mechanisms lead to lower energy structures and, even though site energy is not defined in this model, the barriers assigned ensure it is less likely to jump from a more highly coordinated to a less highly coordinated site, compared to the opposite event. This is a profound illustration of the impact had on film growth morphology by the Schwoebel-Ehrlich mechanism. Note that these results show agreement with what was observed in the original literature presenting this model [91], providing some level of code implementation validation.

On lattice 3D models of deposition and diffusion require that a greater number of site coordination states (i.e. state changes) be addressed. For instance, in the triangular 2D lattice illustrated in results above, the number of nearest neighbors is six whereas in a face centered cubic (FCC) 3D lattice, the number of nearest neighbors is 12. The 2D model presented above was not sufficient to populate all the needed energy parameters for a 3D model so in the examples below, the material should simply be considered a model metal. In other words, the range of barriers are reasonable for a metallic material but we have not obtained these from any rigorous barrier finding method. For an atom to occupy a site, it had to have a minimum coordination number of 2; diffusion into sites with a coordination of up to 11 must be considered; however, common neighbor effects due to the geometry of the lattice reduce the number of coordination combinations that must be considered. Values were assigned to reflect physically reasonable trends (i.e. diffusing from a coordination of 9 to 3 has a significantly higher barrier than from 9 to 7); the barrier magnitudes were

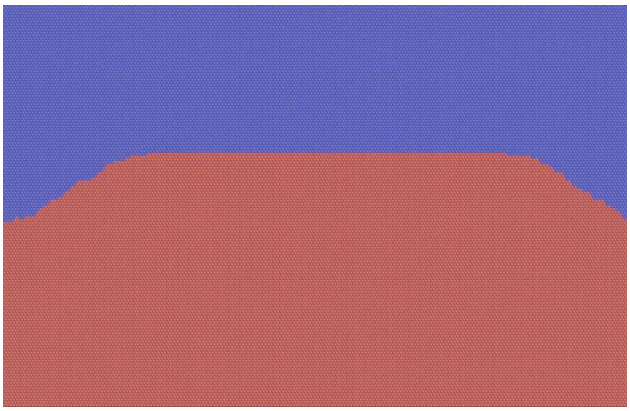


Figure 61: *Rendering of an atomistic KMC model of 2D Ni film deposition and diffusion, where Schwoebel hops are active and T=600 K. Red sites are atoms and blue sites are vacancies; the total growth time is 2 s.*

assigned to be within the range of values seen in literature addressing diffusion barriers in metals. A zero deposition angle was used and the FCC lattice was constructed with a lattice constant of 0.352 nm (this is the T=0 value for Ni but, again, this should be interpreted as a model metal). A 90 x 90 x 20 FCC lattice of sites was generated, corresponding to a simulation cell dimension of 31.7 nm x 31.7 nm x 7 nm; there are 4 sites per FCC unit cell such that there are 16200 sites per layer and 40 layers total in z (648000 sites total). A substrate was constructed by occupying the bottom 3 layers of sites (in z); deposition onto the (001) FCC surface was modeled.

To illustrate the significant effect Schwoebel hops have on morphology, one run is presented with them active at T=300 K in Fig. 62. Layer by layer growth is exhibited and a very smooth surface is obtained. Suppressing Schwoebel hops was also examined. At T=300 K, Fig. 63 shows this results in layer by layer growth but with significantly more surface roughness. In both these cases, the deposition rate modeled was 10 nm/s and the times shown are t = 0, 0.01 s, and 0.04 s. A reasonable test of the model implemented is to see if it can be driven into an island growth mode. This mode emerges experimentally when atoms are able to diffuse a significant amount between deposition events. It is also promoted when there exists less wettability of the deposited material on the substrate material. Because we have only one material type here, we cannot model the latter effect directly. However, by suppressing Schwoebel hops (particularly at elevated T), this gives a model where atoms on island terraces do not drop down a layer and effectively spread the terrace. This can be roughly interpreted as lower wettability. Figure 64 shows the case with no Schwoebel hops but for elevated T=400K. This increase in T activates more surface diffusive events. In addition, this simulation was run with a deposition rate of 0.1 nm/s (two orders of magnitude less than the first two simulations). Figure 64 shows this combination results in an island morphology; the times shown for this simulation are t=0, 1 s, and 4 s. Layered islands are observed; this type of structure is referred to in the experimental growth literature as a wedding cake morphology.

7.7.4 Off Lattice Surface Deposition Modeling

It is no exaggeration to say that the available literature decreases by over an order of magnitude when off lattice surface deposition KMC studies are compared to on lattice studies. Off lattice models permit atoms to occupy arbitrary positions in continuous space and, as such, the complexity inherent in their implementation increases markedly compared to on lattice models. While the underlying concepts of a KMC model are still applied (i.e. diffusion hops and deposition atom additions), these events are followed by some form of relax-

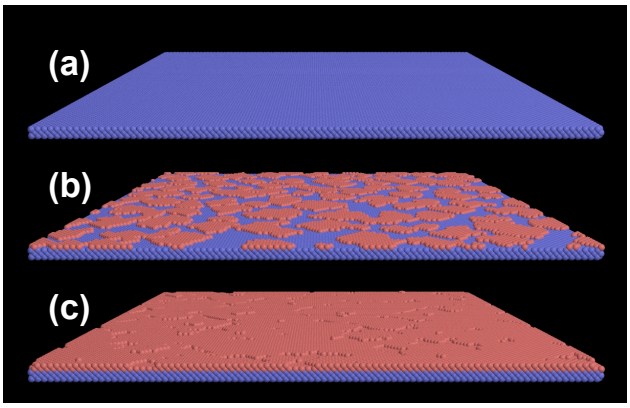


Figure 62: *Rendering of an atomistic KMC model of 3D metal film deposition and diffusion, where Schwoebel hops are active and T=300 K. Red sites are deposited atoms and blue sites are substrate atoms present at t=0; the growth times shown are (a) t=0, (b) t=0.01 s, and (c) t=0.04 s.*

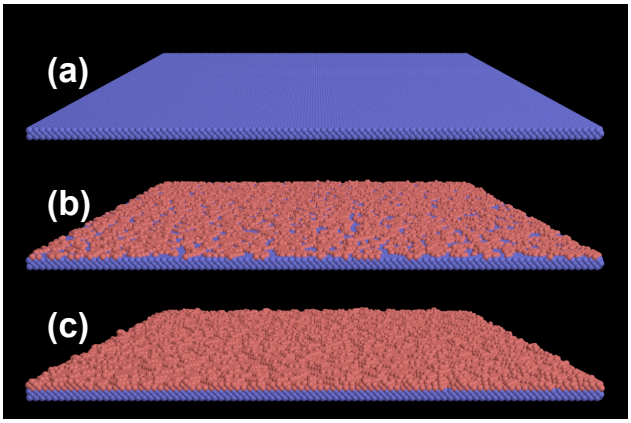


Figure 63: *Rendering of an atomistic KMC model of 3D metal film deposition and diffusion, where Schwoebel hops are not active and T=300 K. Red sites are deposited atoms and blue sites are substrate atoms present at t=0; the growth times shown are (a) t=0, (b) t=0.01 s, and (c) t=0.04 s.*

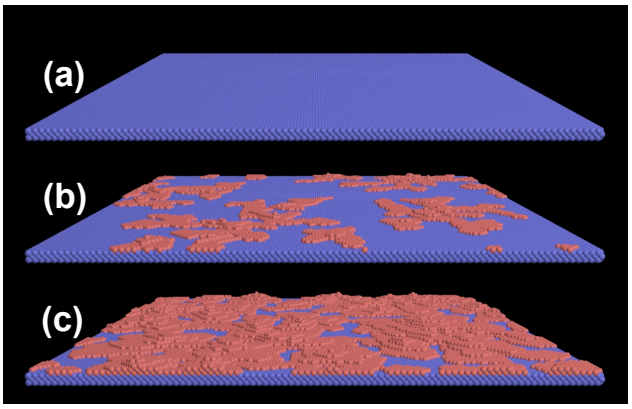


Figure 64: *Rendering of an atomistic KMC model of 3D metal film deposition and diffusion, where Schwoebel hops are not active and T=400 K. Red sites are deposited atoms and blue sites are substrate atoms present at t=0; the growth times shown are (a) t=0, (b) t=1 s, and (c) t=4 s.*

ation under the forces given by an assumed interatomic force field. This can be energy minimization or MD at the modeled temperature; relaxation can either be done local to the event or globally and a combination of the two is often utilized, with global relaxation done less frequently than local relaxation. Because no lattice is assumed, diffusive events can in theory connect any occupied point in space to any unoccupied point in space, within some reasonable diffusive hop distance. Ideally, one would utilize an algorithm that performs a search for local energy minimum sites throughout the volume of space so defined. However, this is computationally quite costly, perhaps sufficiently so to undo much of the time scale bridging achieved by using a KMC model. That said, in many relevant cases of film growth, crystalline structure, especially polycrystallinity, emerges. Thus, it may be possible to construct off lattice models that exploit some knowledge of relevant underlying lattice system(s). For the general amorphous case, it may be possible to use bond length and angle expectations to significantly reduce the volume of space that need be probed for potential vacant sites near an occupied site. Not surprisingly, most off lattice models of surface deposition and diffusion in the literature are 2D [6, 57, 58, 85] and existing 3D models are for a simple cubic lattice [69, 81].

In addition to the complexity in finding candidate final sites for a diffusive event, one must also compute the energy barrier (and/or energy change) on the fly for each event. The burden of this requirement can be lessened by storing encountered environments and reviewing them when new environments are being probed to determine potential events. However, in this case a highly efficient pattern recognition scheme is required and this introduces a separate level of complexity. Once an initial and final site for a diffusive event have been identified, a simpler approach is to displace the diffusing atom along a straight line between the two sites in small increments. At each position along the diffusion path, a local minimization is performed with the constraint that the diffusing atom can only relax its position by remaining in a plane perpendicular to the previous displacement vector. After each relaxation, the atom is displaced another increment and the process is repeated. By recording the position and energy of the atom during this process, a reaction path and, more relevantly, a barrier energy can be extracted. Once diffusive events and their energy data have been tabulated, an off lattice KMC simulation proceeds according to the same rate law presented previously. When an event is selected, it is performed and followed by some form of relaxation as discussed above. The simulation time is advanced and events are redefined local to the previous event. Deposition occurs in a fashion similar to what was done for the on lattice model except that finding potential sites along the incident atom's trajectory suffers from the same complications discussed above for site finding.

Despite these challenges, pursuing robust off lattice KMC models is an important activity to continue.

Such models, in principle, will permit us to resolve atomic scale defect formation during thin film growth at realistic growth rates. In the absence of an underlying assumed lattice, realistic models of heteroepitaxial growth can be implemented. Furthermore, while off lattice development in SPPARKS so far has been restricted to using a simple Lennard-Jones (12-6) interatomic potential for relaxing the system, in principle any interatomic potential scheme could be adopted for this phase of the simulation, permitting connection to a broad range of materials.

7.7.5 Conclusions

Capabilities built into SPPARKS enable both 2D and 3D modeling of film growth at the atomic scale via on lattice models; results show that this model captures well known morphology transitions with temperature and also with Schwoebel hops active versus inactive. Most importantly, models of growth into the seconds and even minutes regime can be formulated to establish more direct connection to experiment.

8 Summary

In this report, we have presented work done under the auspices of our LDRD to develop a parallel kinetic Monte Carlo modeling capability at Sandia. Our efforts focused on 3 areas, each of which has been highlighted in the report: creating a new code SPPARKS for parallel KMC modeling, developing novel serial and parallel KMC algorithms, and implementing materials modeling applications within the code.

As is evident from Section 7, the majority of the applications developed thus far have been on-lattice models. Though the SPPARKS framework supports them, we made less progress on developing off-lattice models, as discussed briefly in the preceeding Section 7.7. In on-going modeling projects that plan to use SPPARKS and its KMC applications, we hope to further develop various off-lattice algorithms for barrier identification and localized relaxation.

In two such projects, SPPARKS is being coupled to other continuum-based models and codes (either finite-element or meshless discrete element or equation-free methods) to perform multiscale calculations.

Additionally, a different LDRD project has plans to model Sn whisker growth. Whiskers have been observed to sometimes grow very rapidly from deposited Sn thin films; with the removal of Pb from components (including COTS parts), Sn whiskering has re-emerged as a potential failure mechanism. This is particularly important for high consequence applications where long microelectronics lifetimes are critical to performance. A model capable of describing whisker nucleation density and growth rate for varying film conditions would greatly improve reliability for a collection of applications. Results indicate that stress gradient driven atomic diffusion is responsible for whisker growth. We intend to use SPPARKS to better understand how grain structure, grain size, and stress gradient magnitude control the kinetics of this growth process.

9 Acknowledgements

Funding for this project came from the Laboratory Directed Research and Development program at Sandia National Laboratories.

References

[1] A. Arkin, J. Ross, and H. H. McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage λ-infected *e*scherichia coli cells. *Genetics*, 149:1633–1648, 1998.

[2] D. A. Bader and K. Madduri. A graph-theoretic analysis of the human protein-interaction network using multi-core parallel algorithm. In *IEEE International Parallel and Distributed Processing Symposium*, 2007.

[3] S. K. Banerjee, I. Kregar, V. Turk, and J. A. Rupley. Lysozyme-catalyzed reaction of the n-acetylglucosamine hexasaccharide. dependence of rate on ph. *J. Biol. Chem.*, 248(13):4786–4792, 1973.

[4] C. C. Battaile. Monte carlo methods for simulating thin film deposition. In S. Yip, editor, *Handbook of Materials Modeling*, pages 2363–2377. Springer, 2005.

[5] C. C. Battaile, D. J. Srolovitz, and J. E. Butler. A kinetic monte carlo method for the atomistic simulation of chemical vapor deposition: Application to diamond. *J. Appl. Phys.*, 82:6293–6300, 1997.

[6] M. Biehl, M. Ahr, W. Kinzel, and F. Much. Kinetic monte carlo simulations of heteroepitaxial growth. *Thin Solid Films*, 428:52–55, 2003.

[7] K. Binder. *Monte Carlo Methods in Statistical Physics*. Springer-Verlag, 1986.

[8] W. J. Blake, M. Kaern, C. R. Cantor, and J. J. Collins. Noise in eukaryotic gene expression. *Nature*, 422(6932):633–637, April 2003.

[9] A. B. Bortz, M. H. Kalos, and J. L. Lebowitz. A new algorithm for monte carlo simulation of ising spin systems. *J. Comp. Phys.*, 17:10–18, 1975.

[10] A. B. Bortz, M. H. Kalos, and J. L. Lebowitz. New algorithm for monte carlo simulation of ising spin systems. *J. Comput. Phys.*, 17(1):10–18, 1975.

[11] I. Bose, R. Karmakar, and S. Roy. Stochastic gene expression in a single cell. *Science*, 297(5584):1183–1186, August 2002.

[12] M. Braginsky, V. Tikare, and E. Olevsky. Numerical simulation of solid state sintering. *International Journal of Solids and Structures*, 42:621–636, 2005.

[13] T. E. Buchheit, D. A. LaVan, J. R. Michael, T. R. Christenson, and S. D. Leith. Microstructural and mechanical properties investigation of electrodeposited and annealed liga nickel structures. *Met. Trans. A*, 33:539–554, 2002.

[14] Y. Cao, D. Gillespie, and L. Petzold. Multiscale stochastic simulation algorithm with stochastic partial equilibrium assumption for chemically reacting systems. *J. Comput. Phys.*, 206(2):395–411, 2005.

[15] Y. Cao, H. Li, and L. Petzold. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. *Journal of Chemical Physics*, 121((9)):4059–4067, 2004.

[16] A. Chatterjee and D. G. Vlachos. An overview of spatial microscopic and accelerated kinetic monte carlo methods. *J Computer-Aided Material Design*, 14:253–308, 2007.

[17] A. A. Chernov and J. Lewis. Computer model of crystallization of binary systems - kinetic phase transitions. *J. Phys. Chem. Solids*, 28:2185–2198, 1967.

[18] F. Ciocchetta, J. Hillston, M. Kos, and D. Tollervey. Modelling yeast pre-rrna processing. In *Computational Methods in Systems Biology*, pages 32–47, 2007.

[19] T. Gillespie D. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81(25):2340–2361, 1977.

[20] V. Danos, J. Feret, W. Fontana, and J. Krivine. Scalable modelling of biological pathways. In Z. Shao, editor, *Proceedings of APLAS 2007*, volume 4807, pages 139–157, 2007.

[21] L. Devroye. *Non-uniform random variate generation*. Springer-Verlag, New York, 1986.

[22] S. G. Eick, A. G. Greenberg, B. D. Lubachevsky, and A Weiss. *ACM Trans. Model. Comput. Simul.*, 3:287, 1993.

[23] M. B. Elowitz, A. J. Levine, E. D. Siggia, and P. S. Swain. Stochastic Gene Expression in a Single Cell. *Science*, 297(5584):1183–1186, 2002.

[24] J. R. Faeder, M. L. Blinov, B. Goldstein, and W. S. Hlavacek. Rule-based modeling of biochemical networks: Research articles. *Complexity*, 10(4):22–41, 2005.

[25] K. A. Fichthorn and W. H. Weinberg. Theoretical foundations of dynamical monte carlo simulations. *J. Chem. Phys.*, 95:1090–1096, 1991.

[26] B. L. Fox. Generating markov-chain transitions quickly: I. *Operations Research Society of America Journal on Computing*, 2(2):126–135, 1990.

[27] M. F. Francis, M. N. Neurock, X. W. Zhou, J. J. Quan, H. N. G. Wadley, and E. B. Webb III. Atomic assembly of Cu/Ta multilayers: Surface roughness, grain structure, misfit dislocations, and amorphization. *J. Appl. Phys.*, 104:034310, 2008.

[28] M. A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Chem. Physics*, 104:1876–1889, 2000.

[29] D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Chem. Physics*, 22:403–434, 1976.

[30] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81:2340–2361, 1977.

[31] D. T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics*, 115(4):1716–1733, 2001.

[32] D. T. Gillespie. Stochastic simulation of chemical kinetics. *Annual Review of Physical Chemistry*, 58(1):35–57, 2007.

[33] G. H. Gilmer, H. Huang, and C. Roland. Thin film deposition: Fundamentals and modeling. *Comp. Mat. Sci.*, 12:354–380, 1998.

[34] T. Hagerup, K. Mehlhorn, and I. Munro. Maintaining discrete probability distributions optimally. *Lecture Notes In Computer Science*, 700:253, 1993.

[35] G. N. Hassold and E. A. Holm. A fast serial algorithm for the finite temperature quenched potts model. *Computers in Physics*, 7:97–107, 1993.

[36] G. Heffelfinger and M. E. Lewitt. A comparison between two massively parallel algorithms for monte carlo computer simulation: An investigation in the grand canonical ensemble. *Journal of Computational Chemistry*, 17(2):250–265, 1996.

[37] W. S. Hlavacek, J. R. Faeder, M. L. Blinov, R. G. Posner, M. Hucka, and W. Fontana. Rules for modeling signal-transduction systems. *Science:STKE*, 2006(344):ref 6, 2006.

[38] M. Itoh. Atomic-scale homoepitaxial growth simulations of reconstructed III-V surfaces. *Prog. Surf. Sci.*, 66:53–153, 2001.

[39] P. Jackel. *Monte Carlo Methods in Finance*. Wiley, 2002.

[40] A. Jagota and G.W. Scherer. Viscosities and sintering rates of a two-dimensional granular composite. *J. Am. Ceram. Soc.*, 12:3123–3135, 1993.

[41] T. Kawamura. Monte carlo simulation of thin-film growth on Si surfaces. *Prog. Surf. Sci.*, 44:67–99, 1993.

[42] G. Korniss, M.A. Novotny, and P.A. Rikvold. *J. Comput. Phys.*, 153:488, 1999.

[43] M. Kotrla. Numerical simulations in the theory of crystal growth. *Comp. Phys. Comm.*, 97:82–100, 1996.

[44] LAMMPS. Web site is http://lammps.sandia.gov.

[45] J. Lapujoulade. The roughening of metal surfaces. *Surf. Sci. Reports*, 20:195–249, 1994.

[46] H.J. Leamy, G.H. Gilmer, and K.A. Jackson. Statistical thermodynamics of clean surfaces. In J.M. Blakely, editor, *Surface Physics of Materials*, volume 1, pages 139–157. Academic Press, New York, 1975.

[47] A. C. Levi and M. Kotrla. Theory and simulation of crystal growth. *J. Phys. Condens. Matter*, 9:299–344, 1997.

[48] S. Lindskog. Structure and mechanism of carbonic anhydrase. *Pharmacology and Therapeutics*, 74:1–20(20), 1997.

[49] S. J. Liu, E. G. Wang, C. H. Woo, and H. Huang. Three-dimensional Schwoebel-Ehrlich barrier. *J. Comp. Aid. Mat. Des.*, 7:195–201, 2001.

[50] L. Lok and R. Brent. Automatic generation of cellular reaction networks with Moleculizer 1.0. *Nature Biotechnology*, 23(1):131–136, January 2005.

[51] B Lubachevsky and A Weiss. Synchronous relaxation for parallel Ising spin simulations. In *15th Workshop on Parallel and Distributed Simulation*, pages 185–192, 2001.

[52] B. D. Lubachevsky. *J. Comput. Phys.*, 75:103, 1988.

[53] H. T. Macgillivray and R. J. Dodd. Monte carlo simulations of galaxy systems. *Astrophysics and Space Science*, 105:331–337, 1984.

[54] H. H. McAdams and A. Arkin. Stochastic mechanisms in gene expression. *Proc. Natl. Acad. Sci. USA*, 94:814–819, 1997.

[55] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1092, 1953.

[56] N. Metropolis and S. Ulam. The monte carlo method. *J. Amer. Stat. Assoc.*, 44:335–341, 1949.

[57] F. Much, M. Ahr, M. Biehl, and W. Kinzel. A kinetic monte carlo method for the simulation of heteroepitaxial growth. *Comp. Phys. Comm.*, 147:226–229, 2002.

[58] F. Much and M. Biehl. Simulation of wetting-layer and island formation in heteroepitaxial growth. *Europhys. Lett.*, 63:14–20, 2003.

[59] W. W. Mullins. Theory of thermal grooving. *Journal of Applied Physics*, 28:333–339, 1957.

[60] E. Olevsky, V. Tikare, and T. Garino. Multi-scale study of sintering: A review. *J. Am. Ceram. Soc.*, 89(6):1914–1922, 2006.

[61] E. A. Olevsky. Theory of sintering: from discrete to continuum. *Material Science and Engineering*, R23:41–100, 1998.

[62] P.M. Raj, A. Odulena, and W. R. Cannon. Anisotropic shrinkage in particle-oriented systems – numerical simulation and experimental studies. *Acta Mater.*, 50(6):2559–2570, 2002.

[63] S. Rajasekaran and K. W. Ross. Fast algorithms for generating discrete random variates with changing distributions. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 3(1):1–19, jan 1993.

[64] C. V. Rao and A. P. Arkin. Control motifs in intracellular regulatory networks. *Annual Review of Biomedical Engineering*, 3:391–419, 2000.

[65] J. M. Raser and E. K. O'Shea. Control of Stochasticity in Eukaryotic Gene Expression. *Science*, 304(5678):1811–1814, 2004.

[66] M. Rathinam, L. R. Petzold, Y. Cao, and D. T. Gillespie. Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. *The Journal of Chemical Physics*, 119(24):12784–12794, 2003.

[67] A. Samant and D. G. Vlachos. Overcoming stiffness in stochastic simulation stemming from partial equilibrium: A multiscale monte carlo algorithm. *The Journal of Chemical Physics*, 123(14):144114, 2005.

[68] M. Samoilov, S. Plyasunov, and A. P. Arkin. Stochastic amplification and signaling in enzymatic futile cycles through noise-induced bistability with oscillations. *Proceedings of the National Academy of Sciences*, 102(7):2310–2315, 2005.

[69] M. Schroeder and D. E. Wolf. Diffusion on strained surfaces. *Surf. Sci.*, 375:129–140, 1997.

[70] T. P. Schulze. Efficient kinetic monte carlo simulation. *J. Comput. Phys.*, 227:2455–2462, 2008.

[71] Y. Shim and J. G. Amar. Hybrid asynchronous algorithm for parallel kinetic monte carlo simulations of thin film growth. *Journal of Computational Physics*, 212:305–317, 2006.

[72] Y Shim and JG Amar. Rigorous synchronous relaxation algorithm for parallel kinetic Monte Carlo simulations of thin film growth. *Phys. Rev. B*, 71(11), MAR 2005.

[73] Y Shim and JG Amar. Semirigorous synchronous sublattice algorithm for parallel kinetic Monte Carlo simulations of thin film growth. *Phys. Rev. B*, 71(12), MAR 2005.

[74] K. Shinagawa. Finite element simulation of sintering process. *JSME Int. J. Ser. A*, 39(4):565–572, 1996.

[75] SPPARKS. Web site is http://www.cs.sandia.gov/∼sjplimp/spparks.html.

[76] V. Tikare. 3d numerical simulation of solid state sintering. 2008. submitted.

[77] V. Tikare, M. Braginsky, D. Bouvard, and A. Vagnon. An experimental validation of a 3d kinetic, monte carlo model for microstructural evolution during sintering. *CIMTEC*, pages 1–8, 2009.

[78] V. Tikare, M. Braginsky, and E. Olevsky. Numerical simulation of solid-state sintering: I. sintering of three particles. *J. Am. Ceram. Soc.*, 86(1):49–53, 2003.

[79] S. Ulam, R. D. Richtmyer, and J. von Neumann. Statistical methods in neutron diffusion. Technical Report LAMS-551, Los Alamos National Laboratory, 1947.

[80] C. Versari and N. Busi. Stochastic simulation of biological systems with dynamical compartment structure. In *Computational Methods in Systems Biology*, pages 80–95, 2007.

[81] T. Volkmann, F. Much, M. Biehl, and M. Kotrla. Interplay of strain relaxation and chemically induced diffusion barriers: Nanostructure formation in 2d alloys. *Surf. Sci.*, 586:157–173, 2005.

[82] A. F. Voter. Clasically exact overlayer dynamics: diffusion of rhodium clusters on Rh(100). *Phys. Rev. B*, 34:6819–6829, 1986.

[83] A. F. Voter. Introduction to the kinetic monte carlo method. In K. E. Sickafus, E. A. Kotomin, and B. P. Uberuaga, editors, *Radiation Effects in Solids*, pages 1–24. Springer, NATO Publishing Unit, Dordrecht, The Netherlands, 2008.

[84] H. N. G. Wadley, A. X. Zhou, R. A. Johnson, and M. Neurock. Mechanisms, models, and methods of vapor deposition. *Prog. Mat. Sci.*, 46:329–377, 2001.

[85] M. Walther, M. Biehl, and W. Kinzel. Formation and consequences of misfit dislocations in heteroepitaxial growth. *Phys. Stat. Sol. C*, 4:3210–3220, 2007.

[86] E. Weinan, D. Liu, and E. Vanden-Eijnden. Nested stochastic simulation algorithm for chemical kinetic systems with disparate rates. *J Chem Phys*, 123(19):194107, 2005.

[87] L. S. Weinberger, J. C. Burnett, J. E. Toettcher, A. P. Arkin, and D. V. Schaffer. Stochastic gene expression in a lentiviral positive-feedback loop: HIV-1 tat fluctuations drive phenotypic diversity. *Cell*, 122(2):169–182, July 2005.

[88] T. Wilhelm, H.-P. Nasheuer, and S. Huang. Physical and functional modularity of the protein network in yeast. *Mol Cell Proteomics*, 2(5):292–298, 2003.

[89] R. R. Wixom, J. F. Browning, C. S. Snow, P. A. Schultz, and D. R. Jennison. First principles site occupation and migration of hydrogen, helium, and oxygen in beta-phase erbium hydride. *Journal of Applied Physics*, 103:123708, 2008.

[90] J. Yang, M. I. Monine, J. R. Faeder, and W. S. Hlavacek. Kinetic monte carlo method for rule-based modeling of biochemical networks, Dec 2007.

[91] Y. G. Yang, R. A. Johnson, and H. N. G. Wadley. A monte carlo simulation of the physical vapor deposition of nickel. *Acta Mater.*, 45:1455–1468, 1997.

Distribution

3	MS	1316	Steve Plimpton, 1416
1		0889	Corbett Battaile, 1814
1		0889	Mike Chandross, 1814
1		1411	Liz Holm, 1814
1		1322	Aidan Thompson, 1435
1		0747	Veena Tikare, 6774
1		9409	Greg Wagner, 8365
1		1411	Ed Webb, 1814
1		9404	Xiaowang Zhou, 8246
1		1322	John Aidun, 1435
1		1411	Allen Roach, 1814
1		0899	Technical Library, 9536 (electronic copy)