SANDIA REPORT

SAND2011-6659
Unlimited Release
Printed September 2011

Accelerated Molecular Dynamics and Equation-Free Methods for Simulating Diffusion in Solids

Gregory J. Wagner, Jie Deng, Lindsay C. Erickson, Steven J. Plimpton, Aidan P. Thompson, Xiaowang Zhou, Jonathan A. Zimmerman, W. Michael Brown

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2011-6659
Unlimited Release
Printed September 2011

Accelerated Molecular Dynamics and Equation-Free Methods for Simulating Diffusion in Solids

Gregory J. Wagner, Jie Deng, Lindsay C. Erickson, Steven J. Plimpton,
Aidan P. Thompson, Xiaowang Zhou, Jonathan A. Zimmerman
Sandia National Laboratories
Livermore, CA 94551
gjwagne@sandia.gov

W. Michael Brown
Oak Ridge National Laboratories
Oak Ridge, TN 37831
brownw@ornl.gov

Abstract

Many of the most important and hardest-to-solve problems related to the synthesis, performance, and aging of materials involve diffusion through the material or along surfaces and interfaces. These diffusion processes are driven by motions at the atomic scale, but traditional atomistic simulation methods such as molecular dynamics are limited to very short timescales on the order of the atomic vibration period (less than a picosecond), while macroscale diffusion takes place over timescales many orders of magnitude larger. We have completed an LDRD project with the goal of developing and implementing new simulation tools to overcome this timescale problem. In particular, we have focused on two main classes of methods: accelerated molecular dynamics methods that seek to extend the timescale attainable in atomistic simulations, and so-called "equation-free" methods that combine a fine scale atomistic description of a system with a slower, coarse scale description in order to project the system forward over long times.

Acknowledgments

We are very grateful for helpful discussions with several individuals throughout the course of this project, especially Art Voter of Los Alamos National Laboratory, Iannis Kevrekidis of Princeton University, and Dongbin Xiu of Princeton University.

Contents

List of Figures

List of Tables

Chapter 1

Overview

Many of the most important and hardest-to-solve problems related to the synthesis, performance, and aging of materials involve diffusion through the material or along surfaces and interfaces. Since accurate interatomic potentials exist to describe many materials at the atom scale, these problems would seem to be candidates for modeling via molecular dynamics (MD) simulation. However, MD requires integration of timescales on the order of atomic vibrations (picoseconds), while many processes of interest in solids, including surface or bulk diffusion, occur on much larger timescales. This scale discrepancy renders MD simulation of many of these important problems intractable. A number of simulation methods have been put forward to remedy this difficulty. Notable among these are kinetic Monte Carlo (KMC) and temperature-accelerated dynamics (TAD), both of which follow individual events on the atomic scale (e.g. atom hopping) that drive the evolution of the material. These methods help to extend the achievable timescales, but have their own limitations on attainable timescales and number of atoms in the simulation. To make further progress, we can note that the quantities most of interest to a modeler, such as concentrations and structural feature sizes, can be described at a much coarser level than the atomic scale. One method that takes advantage of this fact is the so-called "equation-free" approach, which uses microscale computations as a set of numerical experiments from which can be distilled macroscale information, such as time derivatives of coarse scale variables.

Our project has had two major goals. The first was to implement and study extended-timescale molecular dynamics methods. The two methods on which we have focused are Parallel Replica Dynamics (PRD) and Temperature-Accelerated Dynamics (TAD), two separate approaches to finding transition events between low-energy system states. The second goal was to develop extended timescale methods for surface and bulk diffusion in solids using equation-free projective integration (EFPI). The keys to this method are the identification of a set of coarse scale variables that parameterize the system, and the development of interscale operators that map between the coarse and fine descriptions.

The results of our project can be summarized in four major areas of achievement:

1. **We have added parallelized implementations of accelerated MD methods to LAMMPS.** In particular, the Parallel Replica Dynamics (PRD) and Temperature Accelerated Dynamics (TAD) methods, have been implemented, along with Nudged Elastic Band (NEB), a method for computing activation energies between states. In-

novations have been made in the parallelization of these schemes, and clear and concise user command lines have been added, example problems have also been added to the LAMMPS release. These methods will impact a range of research projects in the design and analysis of complex materials, including NW and energy applications. By including these methods in the LAMMPS software release, we have made them available throughout Sandia the external research community.

2. **We have demonstrated the use of the Equation-Free Projective Integration method for accelerated simulations of solid surface evolution.** This is the first application of the method to the evolution of solid surface profiles, a phenomenon which is extremely important in understanding the aging of nano-structured materials. Simulation speed-ups of 20× have been demonstrated, with the possibility of further improvement. As part of our research, we have shed light on the types of coarse variables that must be used to parameterize the slow dynamics of evolving solid systems.

3. **We have developed a new, generalized method for initializing fine-scale systems based on the principle of maximum entropy.** This work was motivated by our previous difficulty with generating systems that are both consistent with a given coarse scale description, but also obey the proper dynamics in time. The method that we have developed allows all but the slow coarse scale variables to come into equilibrium in the sense that maximizes entropy; we have shown that properly doing this allows us to reproduce the correct dynamics of the system. This method has use not just in our EFPI algorithm but in MD, atomistic-to-continuum, and any method in which it is useful to generate the fine scale representation of a system based on a coarse scale description.

4. **We have developed new interscale operators for bulk diffusion processes, and implemented them in LAMMPS.** These operators can be used to distill a set of atomic positions into a spatially varying continuum concentration field (the "restrict" operation), and in the other direction, to randomly generate a set of atomic positions consistent with a given continuum concentration field (the "lift" operation). These newly developed operators and the insights we have gained from their development will impact other atomistic-to-continuum projects that rely on dual fine and coarse descriptions of complex atomic arrangements.

The rest of this report goes into more depth on each of these topics. In Chapter 2 we summarize extended-timescale MD methods, describe their LAMMPS implementation, and show some examples. In Chapter 3 we provide an introduction to the Equation-Free method and show results for the simulations of surface diffusion in 2D; this section is in large part a reproduction of a journal article based on our project published in 2010 in the International Journal for Multiscale Computational Engineering [64]. Chapter 4 describes some recent developments we have made to the interscale operators used for the problem of surface diffusion in 3D to overcome some of the shortcomings of our earlier 2D work; in particular, present our new method for generating fine scale realizations based on the

maximum entropy method. Finally, in Chapter 5 we describe the development and testing of interscale operators used in the application of EFPI to bulk vacancy diffusion in a lattice.

Chapter 2

New Accelerated-Time Algorithms for LAMMPS

2.1 Motivation

Standard molecular dynamics (MD) is limited in the length and time scales it can simulate. If atoms are modeled explicitly (as opposed to coarse-grained approaches), then the fundamental length scale for atom-atom separations is ångstroms, and required timesteps are on the femtosecond scale. This means a million-atom simulation run for a million timesteps, a moderate-sized calculation, using an interatomic potential such as the embedded atom method (EAM), requires a few hours on a hundred or so processors. Yet, in the case of bulk Ni, it only models a nanosecond of elapsed time for a cube of material 215 ångstroms on a side.

These limitations are exacerbated if more realistic interatomic potentials are used, such as modified EAM (MEAM) or reactive potentials like ReaxFF, since the CPU cost per atom per timestep goes up by factors of 10 to 100 to 1000. In the context of this LDRD, these time and length scales are a bottleneck to effective multiscale coupling since they force the "fine" scale to be very small compared to the desired observable continuum "coarse" scale, and they require more computational resources be spent on modeling the fine scale systems.

Because of these limitations, there has been considerable work within the MD community in the last decade to devise "accelerated" time methodologies that enable much longer time scales to be modeled. In solid-state systems the most attractive of these methods for the kinds of defect- and diffusion-driven problems considered in this project are the collection of methods devised by Art Voter at LANL. Two in particular, the parallel replica dynamics (PRD) method [62] and the temperature-accelerated dynamics (TAD) method [55], seemed good candidates in this project for extending the timescale of the fine-scale models. Both methods are designed to accelerate the task of finding rare events, such as a diffusive hop (either by a single atom or in coordinated fashion by several atoms), so that computations focus on the trajectories of interesting events and less on the uninteresting vibrations of atoms about their lattice sites. Additionally, part of the accelerated effect for both methods is achieved through parallelism, which made them good candidates for implementation in Sandia's parallel molecular dynamics package LAMMPS [38].

As explained below, both methods invoke multiple replicas (copies) of the system being simulated and run them simultaneously on different processors. LAMMPS is designed to allow this because the user can specify that the set of P processors allocated for use by LAMMPS be split into M partitions of one or more processors. Each partition can run an independent simulation, or in the case of these accelerated methods, M replicas can be run in a loosely coordinated fashion, communicating information between replicas as needed. This even works if the user desires to run more replicas than there are physical processors (e.g. on his desktop machine), since the MPI message-passing library, which LAMMPS uses for distributed-memory parallelism, supports virtual processors. For example, a 20-replica model can be run on a quad-core processor (4 physical cores) without any modifications by the user to his input script.

When the TAD method identifies an energy barrier that the system has transitioned across, the height of the barrier must be calculated. This can be done using the nudged-elastic band (NEB) method, which itself is a multi-replica method. NEB is also useful on its own for calculating the trajectory of a system across a barrier between two minimum-energy states. Thus we implemented NEB as a third multi-replica method in LAMMPS, usable either as a stand-alone command, or invoked internally by the TAD method.

We note that implementing these 3 methods in LAMMPS has several benefits beyond its utility for this LDRD:

- Since LAMMPS is an open-source, widely-used tool in the computational materials science community, it makes the methods available to a broad user base. This often helps us find bugs and leads to new features being added to the methods, either LAMMPS developers in response to user requests, or by the users themselves.

- It allows the methods to be used in conjunction with the wide variety of existing boundary and constraint conditions in LAMMPS, and with the many interatomic potentials available in LAMMPS [39]. In the latter case, this means that accelerated-time simulations can now be performed for a wider variety of materials.

- It gives other LAMMPS users the option to use the accelerated methods as part of their own multiscale efforts, e.g. by coupling LAMMPS to other coarse-grained methods such as mesoscale kinetic Monte Carlo simulators. We already have users attempting to do this.

In the subsequent sections we give a brief overview of each method (PRD, NEB, TAD), some details of the LAMMPS implementation, and illustrate what they can model.

2.2 Parallel replica dynamics (PRD)

2.2.1 Standard PRD Algorithm

PRD is a method devised by Art Voter [62] for performing accelerated dynamics on systems that undergo infrequent events that obey first-order kinetics. Because the events are infrequent, multiple copies of the system can be simulated independently, until the first event occurs in any of the replicas. This gives an effective enhancement in the timescale spanned by the multiple simulations, and can achieve a parallel speed-up nearly linear in the number of replicas used.

The steps for performing a PRD simulation, as described in the paper by Voter [62] are as follows. The global clock referred to is the effective time simulated by the ensemble of replicas.

1. Starting from an initial or current configuration of the system, replicate the configuration M times.
2. Perform an energy minimization to generate a reference configuration in each replica. This reference state will later be used for transition checks.
3. In each replica, randomize velocities to eliminate correlations between replicas. For example, in [62] Voter presents the example problem of diffusion of a surface vacancy on the Cu(100) surface at a temperature of 500 K (the system contains 5 layers of 18 atoms per layer except for the top layer, which consists of 17 atoms). For this step, atomic momenta are randomized every 1.0 ps over a total run time of 10 ps. Transitions can be accepted or prevented during this stage, depending on whether or not this run time is included in the global clock.
4. Each replica runs a standard NVT simulation (using a Langevin or other thermostat) and checks for transition events every Δt_{block} of integration time. This check typically consists of performing a conjugate gradient (CG) or steepest descent (SD) minimization of the system energy and comparing characteristic variables of the system between the current minimized configuration and the reference configuration. Characteristics may include system energy, center of mass, the sum or max of atomic displacements above a threshold value, average value of atomic coordination number, etc. If no event has occurred on any replica, all replicas continue running dynamics from their respective pre-minimized states. When one replica detects an event, all replicas stop, and proceed to the next step.
5. The global clock is advanced by $t_{\text{sum}} = M \times n \times \Delta t_{\text{block}}$, where n is the number of checks performed by the replica that detected an event. Since events are rare, this is the main source of parallel speed-up provided by the PRD algorithm.
6. Subsequent events correlated to the original event must be detected and allowed to occur before a new state is assigned to all replicas. The replica that detected the event

does this by running additional dynamics until a correlation time $\Delta t_{\rm corr}$ elapses with no events. Note that $\Delta t_{\rm corr}$ should be larger than the correlation time $\tau_{\rm corr}$ characterizing the vibrational dynamics of the system within a local energy basin. Also note that dynamics in this step is running on a single replica, hence the time spent here detracts from the overall parallel scalability.

7. The global clock is advanced by the total dynamics time run on the single replica in the preceding step ($\Delta t_{\rm corr}$ or greater if correlated events occurred) and the final state of this replica becomes the current configuration that will be shared with all replicas in step (1). Steps 1-6 are then repeated.

2.2.2 Details of LAMMPS implementation

The PRD algorithm of Section 2.2.1 was implemented in LAMMPS as a new "prd" input script command. Mike Brown (formerly at Sandia, now at ORNL) did much of the coding and testing. Similar to NEB and TAD calculations, PRD within LAMMPS runs as a multi-replica simulation.

This means that M replicas are simulated simultaneously, each on one or more processors. This is invoked in a LAMMPS simulation, by using the "-partition" command-line switch to allocate M partitions of processors for the simulation. For example, the following commands

```
mpirun -np 10 lmp_linux -partition 10x1 -in in.prd
mpirun -np 20 lmp_linux -partition 10x2 -in in.prd
```

would run a PRD calculation (specified in the in.prd input script) on 10 or 20 processors, using 10 replicas in each case. The first command assigns one processor per replica; the second command assigns two.

Note that LAMMPS allows the use of multiple processors per replica. If the system is large, this can be an effective way to use many processors, e.g. 1000 processors, using 100 replicas with 10 processors each. For PRD, LAMMPS insures that when one replica sends its current state (all the atom coordinates) to other replicas that each processor receives the atom coordinates it needs. This communication operation accounts for the fact that there is no guarantee that the Pth processor in the sending replica owns the same subset of atoms as the Pth processor in a different replica, since the two replicas may have diverged over time, with atoms migrating to different processors within each replica.

Also note that it is not possible to define multiple partitions per processor with LAMMPS. However this is not a restriction, even when performing a PRD calculation on a desktop machine with one or a handful of cores. This is because the mpirun command allows an MPI program, such as LAMMPS, to run on more virtual processors than there are physical processors. In the PRD context, this means any number of replicas M can be run on P processors, even if $M > P$ or $P = 1$.

The syntax for the new "prd" command is as follows:

```
prd N t_event n_dephase t_dephase t_correlate compute-ID seed keyword value ...
```

where

- N = # of timesteps to run (not including dephasing/quenching)
- t_event = timestep interval between event checks
- n_dephase = number of velocity randomizations to perform in each dephase run
- t_dephase = number of timesteps to run dynamics after each velocity randomization during dephase
- t_correlate = number of timesteps within which 2 consecutive events are considered to be correlated
- compute-ID = ID of the compute used for event detection
- seed = random # seed (positive integer)

The user also specifies a "compute" command in the input script with the compute-ID referred to. This is the piece of code that checks if the current state of a replica is sufficiently different than a previous state, so that an "event" is considered to have occurred. The interface for this style of compute is defined so that users can easily add their own definitions of events. A prototypical "compute event/displace" command is provided where a threshold distance is defined. Two minimum energy states are compared (the current and reference states). An event is considered to have occurred if any atom has displaced a distance further than the threshold, i.e. a diffusive hop has taken place. As discussed in step (4) of Section 2.2.1 other criteria for an event could be encoded as well. For example, another group at Sandia is writing their own "compute" that defines an "event" peculiar to grain boundary re-configuration, in hopes of using PRD to track grain boundary evolution at longer timescales.

The implementation of the PRD algorithm in LAMMPS follows the 7 steps outlined in Section 2.2.1. N is the total number of timesteps to run on each replica. The dephasing portion of the PRD loop is step (3) of Section 2.2.1 and is run in *n_dephase* segments, each of *t_dephase* timesteps.

The search-for-events step (4) is then run independently on each replica, checking for events every *t_event* steps. Once an event is found the correlation check of step (6) is run on one replica until *t_correlate* steps have elapsed without an event occurrence.

The specified random number seed is used to generate random velocities within each replica, as part of the dephasing procedure. Additional keyword/value pairs of arguments

can be specified to adjust the tolerance for energy minimizations, the target temperature for dephasing, and options for how the random velocities are generated.

Four kinds of output can be generated during a PRD calculation: (1) event statistics, (2) thermodynamic output by each replica, (3) dump files, and (4) restart files.

The statistics printed to the screen and main log file each time an event occurs are the timestep, CPU time, global clock, event number, a correlation flag, the number of coincident events, and the replica number which observed the event.

The timestep is the usual LAMMPS timestep, except that time does not advance during dephasing or quenches, but only during dynamics. Note that there are two kinds of dynamics in the PRD algorithm. The first is when all replicas are performing independent dynamics. The second is when correlated events are being searched for, and only one replica is running dynamics.

The CPU time is the total processor time since the start of the PRD run.

The global clock is the same as the timestep except that it advances by M steps every timestep during the first kind of dynamics when the M replicas are running independently. The global clock thus represents the real time that effectively elapses during a PRD simulation of N steps on M replicas. If most of the PRD run is spent in this stage, searching for infrequent events, then the clock will advance nearly $N * M$ steps.

The event number is a counter that increments with each event, whether it is uncorrelated or correlated.

The correlation flag will be 0 when an uncorrelated event occurs in step (4) of Section 2.2.1. The flag will be 1 when a correlated event occurs during step (6) Section 2.2.1, i.e. when only one replica is running dynamics.

When more than one replica detects an event at the end of step(4) of the algorithm, then one of them is chosen at random. The number of coincident events is the number of replicas that detected an event. Normally, this value should be 1. If it is often greater than 1, then either the number of replicas is too large, or t_event is too large.

The replica number is the ID of the replica (from 0 to M-1) that found the event.

Any dump files defined in the LAMMPS input script, will be written to during a PRD run at timesteps corresponding to both uncorrelated and correlated events. This means the the requested dump frequency in the dump command is ignored. There will be a single dump file (per dump command) created for all partitions.

The atom coordinates of the dump snapshot are those of the minimum energy configuration resulting from quenching following a transition event. The timesteps written into the dump files correspond to the timestep at which the event occurred. A dump snapshot corresponding to the initial minimum state used for event detection is also written to the dump file at the beginning of each PRD run. Thus the sequence of snapshots in the dump

file represent successive events for the system.

If the restart command is used, a single restart file for all the replicas is generated, which allows a PRD run to be continued by a new input script. When an input script reads a restart file from a previous PRD run, the new script can be run on a different number of replicas or processors.

2.2.3 Example PRD calculation

We have validated the LAMMPS implementation of PRD by direct comparison with trajectories generated using standard MD at a high temperature. The chosen test problem was the diffusion of a single-site vacancy in a bulk silicon crystal (4x4x4x8 atoms), as illustrated in Figure 2.1. A conventional NVT MD simulation at 2000 K was run for 100 ps using a 1 fs timestep. This temperature was chosen to be as high as possible, in order to maximize the vacancy hopping rate while avoiding melting the crystal, thus enabling a large number of vacancy hops to be sampled. In addition, because the time between hopping events is very short (about 0.5 ps), this provides a stringent test of the approximations invoked by the PRD method. The positions of the atoms adjacent to the vacancy were printed out every 100 timesteps. These atoms were identified by relaxing all the atom positions to the local minimum and then searching for atoms with fewer than 4 neighbors within a distance of 2.835 Å. The instantaneous position of the vacancy was taken to be the centroid of these atoms. The PRD method was then run on 1 replica for 1000 ps. Running on a single replica is actually less efficient than standard MD, due to the PRD overhead, and so would not be useful for a real calculation. Nonetheless, it does test the validity of the PRD implementation, as it exercises the various steps in the PRD algorithm (dephasing, event checking, correlated event checking) . The PRD parameter values used were $t\_event = 100$, $n\_dephase = 10$, $t\_dephase = 10$, and $t\_corr = 100$. The compute event/displace threshold displacement was 0.5 Å. Figure 2.2 shows a comparison of the mean square displacement of the vacancy versus time for both methods. In both cases, the three statistically independent and equivalent x, y, and z displacement components are plotted separately, indicating the level of statistical variation for each sample point. The PRD results lie within the expected range of the MD results at all times. The scatter in the PRD is less than that for MD, because roughly ten times more events were sampled using PRD.

In order to evaluate the parallel efficiency of the method, we ran PRD simulations on different processor counts at 300 K. At this low temperature, the time between hops is large, and PRD is expected to exhibit high accuracy and parallel efficiency. The PRD values used were the same as before, except $t\_event = 1000$. PRD simulations were run for 10^4 steps (10 ps) on Sandia's Red Sky capacity compute cluster. Processor counts ranged from 1 to 1024, and one replica was assigned to each processor i.e. $M = P$. Parallel efficiency was defined to be the CPU time required to run 10^4 steps on 1 processor divided by the time to run 10^4 steps on P processors

$$ParEff = \frac{CPU(1)}{CPU(P)}. \tag{2.1}$$

This definition assumes that for the same number of timesteps, the effective time simulated using P replicas is P times the time simulated using straight MD. In situations where the time spent dephasing and checking for events is negligible compared to the time spent trapped in a single basin, this is a good assumption. Using the same assumption, the parallel speed-up was defined as

$$SpeedUp = P * \frac{CPU(1)}{CPU(P)}. \tag{2.2}$$

Figure 2.3 shows parallel speed-up and parallel efficiency (inset) versus processor count. The speed-up obtained on 1024 processors was 873.5. The inset shows that the parallel efficiency drops slowly with increasing processor count, giving an efficiency of about 85% on 1024 processors. Most of this loss is due to time spent communicating the results of the event detection between the processors.

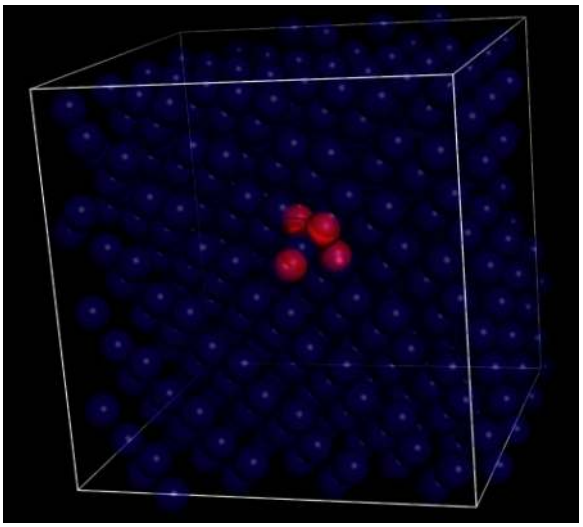


Figure 2.1. Visualization of a vacancy in a 4x4x4 supercell of a silicon crystal. The four atoms adjacent to the vacancy are colored red, while all other atoms are colored transparent blue

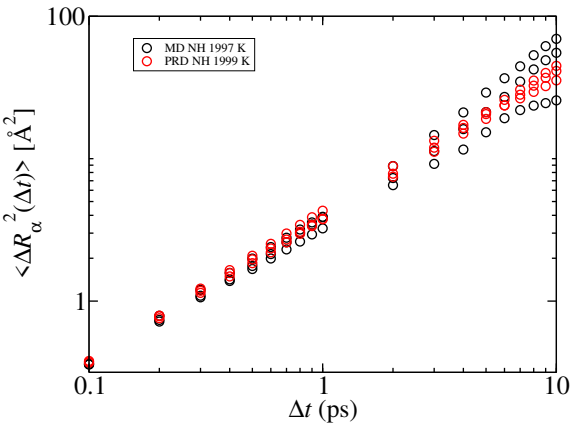


Figure 2.2. Comparison of MD (black) and PRD (red) results for mean squared displacement of a silicon vacancy versus time at 2000 K. In both cases, the three statistically independent and equivalent x, y, and z displacement components are plotted separately, indicating the level of statistical variation for each sample point.

2.3 Nudged elastic band (NEB)

2.3.1 Original Implementation of NEB

The Nudged Elastic Band (NEB) method for finding minimum energy paths was originally developed by Mills, Jónsson and Schenter [30, 31]. However, the method is detailed in a straight-forward manner in the later work by Henkelman and Jónsson [14]. The method is as follows:

1. Starting with two known configurations to act as the endpoints of a minimum energy path (MEP), $\mathbf{R}_0$ and $\mathbf{R}_N$, create $N-1$ images between these configurations. The endpoints $\mathbf{R}_0$ and $\mathbf{R}_N$ should be states corresponding to different energy minima of the system, else the minimization procedure described below will collapse the MEP to a single point. The endpoints are often chosen to be actual energy minima, though this is not required. The intermediate states can be generated in different ways; the most common is simply to linearly interpolate atom positions between configurations 0 and N.

2. Once the images are created calculate the unit tangent vector $\hat{\boldsymbol{\tau}}_i = \boldsymbol{\tau}_i / |\boldsymbol{\tau}_i|$ for each configuration $i = 1, 2, \ldots, N-1$, where

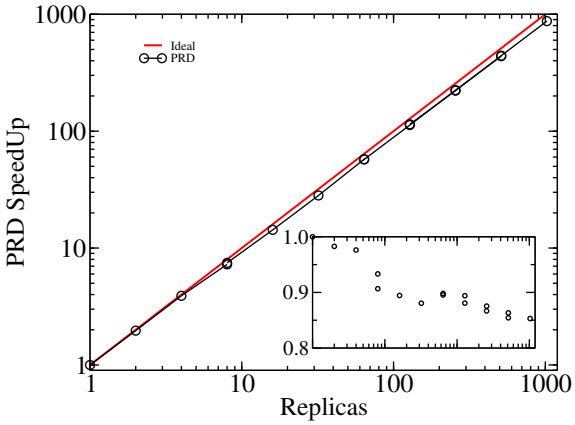


Figure 2.3. Parallel speed-up versus processor count (replicas) for PRD simulations (circles) of vacancy diffusion in silicon at 300 K. Ideal scaling is indicated by the solid red line. The inset figure shows corresponding values for parallel efficiency. See text for definitions of speed-up and efficiency.

$$\boldsymbol{\tau}_i = \frac{\mathbf{R}_i - \mathbf{R}_{i-1}}{|\mathbf{R}_i - \mathbf{R}_{i-1}|} + \frac{\mathbf{R}_{i+1} - \mathbf{R}_i}{|\mathbf{R}_{i+1} - \mathbf{R}_i|}.$$

3. Calculate the spring force along the unit tangent direction,

$$\mathbf{F}_i^s = k\left(\mathbf{R}_i^s \cdot \hat{\boldsymbol{\tau}}_i\right)\hat{\boldsymbol{\tau}}_i$$

where $\mathbf{R}_i^s \equiv (\mathbf{R}_{i+1} - \mathbf{R}_i) - (\mathbf{R}_i - \mathbf{R}_{i-1})$ and k is a spring constant. k does not have to be the same constant for all springs, but if it is then the images will be equally spaced. Although no guidance is provided in [14] regarding what to use for k, according to the theory for NEB, the magnitude of k should not affect the motion of the elastic band away from or to the minimum energy path. k merely affects the motion along the band.

4. Calculate the true force from the governing inter-atomic potential, $\mathbf{F}_i^t = -\boldsymbol{\nabla} V(\mathbf{R}_i)$, and take its component perpendicular to the tangent direction,

$$\mathbf{F}_i^\perp = \mathbf{F}_i^t - \left(\mathbf{F}_i^t \cdot \hat{\boldsymbol{\tau}}_i\right)\hat{\boldsymbol{\tau}}_i.$$

5. Combine these two forces to get the total force acting on each image,

$$\mathbf{F}_i = \mathbf{F}_i^s + \mathbf{F}_i^\perp.$$

6. A minimization algorithm (*e.g.* conjugate gradient) is applied to minimize the total force on each image, repeating steps 2 - 5 as often as needed.

7. Once the forces on all images have been minimized, the minimum energy path (MEP) is now defined and can be scanned and interpolated to determine the energy of the saddle point, *i.e.* the maximum point on the MEP. In the Appendix of [14], interpolation is done using a cubic polynomial between adjacent images i and $i+1$:

$$\begin{aligned} V(x) &= a_i x^3 + b_i x^2 + c_i x + d_i \\ a_i &= -\frac{2\left(V_{i+1} - V_i\right)}{R^3} - \frac{F_i + F_{i+1}}{R^2} \\ b_i &= \frac{3\left(V_{i+1} - V_i\right)}{R^2} + \frac{2F_i + F_{i+1}}{R} \\ c_i &= -F_i \\ d_i &= V_i \end{aligned}$$

NOTE: The relations for a_i and b_i are incorrect as they appear in the Appendix of [14]. The relations given above are correct. In these expressions, R is described as the "length of the ith segment". However, it is probably given by the relation $R = |\mathbf{R}_{i+1} - \mathbf{R}_i|$. Similarly, x is the distance from $\mathbf{R}_i$ for an arbitrary configuration $\mathbf{X}$ where $x = |\mathbf{X} - \mathbf{R}_i|$ and $0 < x < R$. Alternatively, we can express this arbitrary configuration as $\mathbf{X} = \mathbf{R}_i + \frac{x}{R}\left(\mathbf{R}_{i+1} - \mathbf{R}_i\right)$. From this, we notice that

$$\frac{\partial V}{\partial x} = \frac{\partial V}{\partial \mathbf{X}} \cdot \frac{\partial \mathbf{X}}{\partial x} = \boldsymbol{\nabla} V(\mathbf{X}) \cdot \frac{1}{R}\left(\mathbf{R}_{i+1} - \mathbf{R}_i\right) = -\mathbf{F}^t(\mathbf{X}) \cdot \hat{\mathbf{R}}$$

where $\hat{\mathbf{R}} \equiv \left(\mathbf{R}_{i+1} - \mathbf{R}_i\right)/R$. Hence, we can deduce that $F_i = \mathbf{F}_i^t \cdot \hat{\mathbf{R}}$ and $F_{i+1} = \mathbf{F}_{i+1}^t \cdot \hat{\mathbf{R}}$. In [14], F_i and F_{i+1} are referred to as "parallel forces". Clearly, the term "parallel" means with respect to the direction $\hat{\mathbf{R}}$.

The above expressions produce the results that when $x = 0$, $V = V_i$ and when $x = R$, $V = -2V_{i+1} + 2V_i - \left(F_i + F_{i+1}\right)R + 3V_{i+1} - 3V_i + \left(2F_i + F_{i+1}\right)R - F_i R + V_i = V_{i+1}$. The maximum and minimum of $V(x)$ are located at:

$$x_{\text{extrema}} = \frac{-b_i \pm \sqrt{b_i^2 - 3a_i c_i}}{3a_i}$$

By evaluating the second derivative of $V(x)$, $V''(x) = 6a_i x + 2b$, we can determine that the maximum within this segment occurs at $x_{\max} = \left(-b_i - \sqrt{b_i^2 - 3a_i c_i}\right)/\left(3a_i\right)$ and has the value

$$V(x_{\max}) = \frac{b_i}{27a_i^2}\left(2b_i^2 - 9a_i c_i\right) + \frac{\left(2b_i^2 - 6a_i c_i\right)}{27a_i^2}\sqrt{b_i^2 - 3a_i c_i} + d_i.$$

The maximum of $V(x_{\max})$ among all segments along the elastic band is the saddle point.

2.3.2 Improved tangent estimate for NEB

In [14], Henkelman and Jónsson show that the original NEB method is prone to the occurrence of kinks and an instability in the elastic band if the number of images is sufficiently large. The then propose the following revision to the method (a different definition of the local tangent) in order to eliminate kinks:

1. Repeat step 1 of the original implementation.
2. The tangent vector $\boldsymbol{\tau}_i$ is now defined as follows:

$$\boldsymbol{\tau}_i = \begin{cases} \boldsymbol{\tau}_i^+ & \text{if } V_{i+1} > V_i > V_{i-1}, \\ \boldsymbol{\tau}_i^- & \text{if } V_{i+1} < V_i < V_{i-1}, \\ \boldsymbol{\tau}_i^+ \Delta V_i^{\max} + \boldsymbol{\tau}_i^- \Delta V_i^{\min} & \text{if } V_{i+1} < V_i > V_{i-1} \text{ or } V_{i+1} > V_i < V_{i-1} \text{ and } V_{i+1} > V_{i-1}, \\ \boldsymbol{\tau}_i^+ \Delta V_i^{\min} + \boldsymbol{\tau}_i^- \Delta V_i^{\max} & \text{if } V_{i+1} < V_i > V_{i-1} \text{ or } V_{i+1} > V_i < V_{i-1} \text{ and } V_{i+1} < V_{i-1}. \end{cases}$$

   where $\boldsymbol{\tau}_i^+ = \mathbf{R}_{i+1} - \mathbf{R}_i$, $\boldsymbol{\tau}_i^- = \mathbf{R}_i - \mathbf{R}_{i-1}$, $V_i = V(\mathbf{R}_i)$, $\Delta V_i^{\max} = \max(|V_{i+1} - V_i|, |V_{i-1} - V_i|)$ and $\Delta V_i^{\min} = \min(|V_{i+1} - V_i|, |V_{i-1} - V_i|)$. The unit tangent vector is determined by $\hat{\boldsymbol{\tau}}_i = \boldsymbol{\tau}_i / |\boldsymbol{\tau}_i|$.
3. The spring force is also altered in its definition:

$$\mathbf{F}_i^s = k R_i^s \hat{\boldsymbol{\tau}}_i$$

   where $R_i^s \equiv |\mathbf{R}_{i+1} - \mathbf{R}_i| - |\mathbf{R}_i - \mathbf{R}_{i-1}|$ and k is the spring constant.
4. Steps 4-7 are the same as in the original implementation.

2.3.3 Climbing image NEB

The theory above provides a good estimate for the saddle point while avoiding kinks in the elastic band. However, since the configuration images are equal spaced (or arbitrarily spaced if each image has its own spring constant k), there is no guarantee that the interpolation estimate for the saddle point has converged on its true value. Henkelman *et al.* [16] point this out by stating that "When the energy barrier is narrow compared with the length of the MEP, few images land in the neighborhood of the saddle point and the interpolation can be inaccurate". To rectify this, they suggest the following modification to the NEB algorithm:

1. Perform a few iterations of the normal NEB method, steps 1-7 of the previous section that uses the improved tangent estimate.

2. Identify the image with the highest energy, $i_{\max}$. Modify the force on this image to the following:

$$\mathbf{F}_{i_{\max}} = -\boldsymbol{\nabla} V(\mathbf{R}_{i_{\max}}) + 2\left(\boldsymbol{\nabla} V(\mathbf{R}_{i_{\max}}) \cdot \hat{\boldsymbol{\tau}}_{i_{\max}}\right) \hat{\boldsymbol{\tau}}_{i_{\max}}$$

Henkelman *et al.* note that this expression is the full force due to the potential with the component along the elastic band inverted. Obviously, this expression is not affected by spring forces.

3. Henkelman *et al.* also suggest adjusting the spring constant k for images near the saddle point. The use of strong springs results in images moving closer to the saddle point, thereby improving the approximation made with interpolation. This adjustment is expressed as

$$k_i \rightarrow k_i' = \begin{cases} k_{\max} - \Delta k \left(\frac{V_{\max} - V_{i,\max}}{V_{\max} - V_{\text{ref}}} \right) & \text{if } V_{i,\max} > V_{\text{ref}} \\ k_{\max} - \Delta k & \text{if } V_{i,\max} < V_{\text{ref}} \end{cases}$$

where $V_{i,\max} = \max\{V_i, V_{i-1}\}$ is the higher energy of the two images connected by spring i (Presumably, segment i that contains spring k_i' connects images i and $i-1$. Hence, there are N images along the elastic band that are numbered $i = 1, 2, \ldots, N$.), $V_{\max}$ is the maximum value of $V_{i,\max}$ along the elastic band, and V_{ref} is a reference value (in [16], V_{ref} is the energy of the higher endpoint of the MEP). No guidance is given in [16] regarding what values to specify for either $k_{\max}$ or Δk. Since the springs on either side of image i no longer have the same spring constant, a modification of the spring force is warranted. While Henkelman *et al.* do not provide an expression for this modification, a reasonable form is

$$\mathbf{F}_i^s = \left(k_{i+1}' \left|\mathbf{R}_{i+1} - \mathbf{R}_i\right| - k_i' \left|\mathbf{R}_i - \mathbf{R}_{i-1}\right|\right) \hat{\boldsymbol{\tau}}_i.$$

2.3.4 Details of LAMMPS implementation

The NEB algorithm was implemented in LAMMPS as a new "neb" input script command which follows the details of Section 2.3.3 and its preceding sections, except for step (3) of Section 2.3.3. The NEB calculation within LAMMPS runs as a multi-replica model. Each of the M "configurations" of the system, denoted as $\mathbf{R}_0, \mathbf{R}_1, \ldots \mathbf{R}_{M-1}$, for the two end-point configurations and $M-2$ intermediate images are treated as a replica.

See the discussion at the beginning of Section 2.2.2 for how such a simulation is invoked in parallel with LAMMPS. Note that a NEB calculation is often performed on a small number of atoms which are assumed to reconfigure as the system transitions over the energy barrier. These mobile atoms may be a geometrically local subset of a larger, essentially static, background of atoms which can be held motionless during the minimization procedure. Thus it is often not necessary to use more than one processor per replica, since the system size is

small, though LAMMPS allows a NEB calculation to be configured with multiple processors per replica.

Also note that as with PRD, more NEB replicas can be used than there are physical processors, since MPI enables a simulation to run on virtual processors. An entire NEB calculation with multiple replicas can even be run on a single processor.

The syntax for the "neb" command added to LAMMPS is as follows:

```
neb etol ftol N1 N2 Nevery filename
```

where

- etol = stopping tolerance for energy
- ftol = stopping tolerance for force
- N1 = max # of timesteps to run initial NEB
- N2 = max # of timesteps to run barrier-climbing NEB
- Nevery = print NEB status every this many timesteps
- filename = file specifying configuration on other side of barrier

The user must also precede the "neb" command with a "min_style" command and a "fix_neb" command in the input script. The "min_style" command must select either the QuickMIN or FIRE minimizer:

```
min_style quickmin
min_style fire
```

The syntax for the "fix_neb" command is as follows:

```
fix ID group-ID neb Kspring
```

where

- ID = ID of fix command
- group-ID = ID of group of atoms to include in NEB calculation
- neb = style name of this fix command

- Kspring = inter-replica spring constant

The meaning of all these parameters is described below. Note that the group of atoms specified in the "fix neb" command can be a subset of the atoms in the system. Inter-replica spring forces are only applied to these atoms, which we term the NEB atoms. Non-NEB atoms can move (slightly) during the NEB procedure, or can be held fixed by setting forces on them to zero via the "fix setforce" command in LAMMPS.

The current configuration of atoms (when the "neb" command is issued) becomes the initial endpoint configuration for the NEB calculation. It should represent a state with the NEB atoms (and all other atoms) having coordinates on one side of the energy barrier. These coordinates become the first replica. A perfect energy minimum is not required, since the NEB calculation will relax the configuration of the first replica to a true energy minimum, but it will converge faster if the initial state is already at a minimum. For example, for a system with a free surface, the surface should be fully relaxed before attempting a NEB calculation.

The final endpoint configuration is specified in the file *filename*, which is simply a list of atom IDs and coordinates. Only NEB atoms or a subset of them should be included in the file. As in the initial case, this configuration should be at or near an energy minimum. The final coordinates of atoms not listed in the file are set equal to their initial coordinates. This set of coordinates becomes the last replica. Note that a final coordinate for a NEB atom need not be specified if it is only expected to displace a small amount during the NEB procedure. For example, only the final coordinate of the single atom diffusing into a vacancy need be listed in the file if the surrounding atoms will only relax slightly in the final configuration.

The intermediate configurations for the other replicas are set to values linearly interpolated between the corresponding atoms in the initial and final configurations.

The first stage of NEB runs by relaxing the collective coordinates of the set of replicas. As described above, this involves two kinds of forces: inter-atomic forces within a replica, and inter-replica forces between an atom and its image in neighboring replicas. The strength of the spring is specified by $Kspring$ in the "fix neb" command.

The energy minimization of the overall multi-replica system is not performed using a conjugate-gradient (CG) style minimizer for two reasons. First, it would require calculation of global information across all replicas, e.g. the norms of vectors of length $3NM$ where N is the number of atoms (per replica) and M is the number of replicas.

Second, the global energy and force functions are actually inconsistent in a mathematical sense, which means that a gradient-based minimizer (e.g. CG) will not work effectively. This is because the NEB forces applied to the atoms contain several modifications to the global energy (negative) derivatives. Spring forces are projected on to the tangent direction, while potential energy forces have the tangent component removed. Moreover, we do not require the two endpoint configurations to initially be in minimum energy states, but allow them to relax independently to a nearby minimum energy state. This is done by turning off

inter-replica forces on atoms in the endpoint states. Thus a NEB atom in the second replica feels a force from its image in the first (endpoint) replica, but not vice versa.

Because of these issues, we implemented two damped-dynamics minimizers in LAMMPS, which are commonly used by NEB practitioners in the literature. These are the QuickMin [51, 22] and FIRE [3] minimizers. Prior to the "neb" command, one of these must be explicitly selected using the "min_style" command described above. They work by timestepping positions and velocities in a simple Euler integration scheme while damping the velocity of each atom in a prescribed manner as a function of the magnitude and direction of the force on it. This has the added benefit of allowing the atoms some freedom to escape local minima, which can be beneficial in stand-alone energy minimization of a (single replica) system; the two minimizers added to LAMMPS can also be used in this mode.

The first NEB stage runs until the specified *etol* or *ftol* criteria for energy or force tolerance is met for every replica. If one of the tolerances is set to 0.0, the other tolerance effectively sets the precision of the calculation. If both tolerances are 0.0, then no more than *N1* timesteps are run.

At the end of the first stage, each replica's state should be equally spaced (in a reaction coordinate sense) over the energy barrier. The replica with the highest energy (near the top of the barrier) is flagged, and the second stage begins. As discussed in step (2) of Section 2.3.3, the forces on atoms in this replica are altered, so that it will climb to the top of the energy barrier as the NEB calculation proceeds. The second stage continues using the same *etol* or *ftol* tolerances or for at most *N2* steps.

Four kinds of output can be generated during a NEB calculation: (1) energy barrier statistics, (2) thermodynamic output by each replica, (3) dump files, and (4) restart files.

The energy barrier data is printed to the screen and main log file, once every *Nevery* timesteps. It includes the timestep, the maximum force per replica, the maximum force per atom (in any replica), potential gradients in the initial, final, and climbing replicas, the forward and backward energy barriers, the total reaction coordinate (RDT), and the normalized reaction coordinate and potential energy of each replica.

The "maximum force per replica" is the two-norm of the $3N$-length force vector for the atoms in each replica, maximized across replicas, which is what the *ftol* setting is checking against. In this case, N is all the atoms in each replica. The "maximum force per atom" is the maximum force component of any atom in any replica. The potential gradients are the two-norm of the $3N$-length force vector solely due to the interaction potential i.e. without adding in inter-replica forces. Note that inter-replica forces are zero in the initial and final replicas, and only affect the direction in the climbing replica. For this reason, the "maximum force per replica" is often equal to the potential gradient in the climbing replica. In the first stage of NEB, there is no climbing replica, and so the potential gradient in the highest energy replica is reported, since this replica will become the climbing replica in the second stage of NEB.

The "reaction coordinate" (RD) for each replica is the two-norm of the 3N-length vector

of distances between its atoms and the preceding replica's atoms, added to the RD of the preceding replica. The RD of the first replica RD1 = 0.0; the RD of the final replica RDN = RDT, the total reaction coordinate. The normalized RDs are divided by RDT, so that they form a monotonically increasing sequence from zero to one. When computing the RD, N only includes NEB atoms.

The forward (reverse) energy barrier is the potential energy of the highest replica minus the energy of the first (last) replica.

When running on multiple partitions, LAMMPS produces additional log files for each partition, e.g. "log.lammps.0", "log.lammps.1", etc. For a NEB calculation, these contain the thermodynamic output for each replica.

If dump commands in the input script define a filename that includes a universe or uloop style variable, then one dump file (per dump command) will be created for each replica. At the end of the NEB calculation, the final snapshot in each file will contain the sequence of snapshots that transition the system over the energy barrier. Earlier snapshots will show the convergence of the replicas to the minimum-energy path (MEP).

Likewise, restart filenames can be specified with a universe or uloop style variable, to generate restart files for each replica. These may be useful if the NEB calculation fails to converge properly to the MEP, and you wish to restart the calculation from an intermediate point with altered parameters.

We also wrote two Python scripts, `neb_combine.py` and `neb_final.py`, included in the `tools/python` directory of the LAMMPS distribution, which are useful in analyzing output from a NEB calculation. Assume a NEB simulation with M replicas, and the NEB atoms labeled with a specific atom type. Script `neb_combine.py` extracts atom coordinates for the NEB atoms from all M dump files and creates a single dump file where each snapshot contains the NEB atoms from all the replicas and one copy of non-NEB atoms from the first replica (presumed to be identical in other replicas). This can be visualized/animated to see how the NEB atoms relax as the NEB calculation proceeds. Script `neb_final.py` extracts the final snapshot from each of the M dump files to create a single dump file with M snapshots. This can be visualized to watch the system make its transition over the energy barrier.

2.4 Temperature-accelerated dynamics (TAD)

2.4.1 Standard TAD Algorithm

The steps for performing a TAD simulation, as described in the paper by Sørensen & Voter [55] are as follows:

1. Preselect the desired low temperature, T_{low}, a lower bound for prefactors used to estimate transition rate constants, ν_{min}, and a probability value, δ, for the TAD simulation.

2. From the initial configuration of the system, determine its nearest energy minimized state A using a conjugate gradient (CG) or steepest descent (SD) algorithm.

3. Start an MD simulation at a high temperature, T_{high}, using a Langevin thermostat. At specified time intervals, use CG/SD to check whether A is still the minimized state for the current configuration.

4. At a certain point, the system will converge to a different minimized state B. At that point,

   (a) If the time interval used is large ($\sim$2 ps), then go back to configuration of last check and perform CG/SD for a series of states separated by smaller intervals ($\sim$0.1 ps) to determine when the $A - B$ transition occurs. S&V suggest storing these intermediate states for easy calculation, but this amounts to storing about 20 states of the system between large interval checks, which can be memory intensive.

   (b) If the time interval is small, then presumably one already has the two configurations of interest. While use of a small time interval will be efficient with regard to memory storage, it would be inefficient as many more CG/SD evaluations are done for a given amount of simulated time at T_{high}.

   The two configurations of interest are referred to as X_1 and X_2. These are not equivalent to A and B, respectively, as the former are configurations at T_{high} while the latter are energy minimized configurations.

5. Using configurations A and B, compute the energy barrier of the transition, E_a by determining the saddle point of the dividing energy ridge between the two configurations. S&V suggest using a linearized trajectory as input to an nudged elastic band (NEB) calculation to determine the minimum energy path between A and B. The saddle point is the highest energy point on the minimum energy path.

6. Once this is done, store configuration X_2, its energy barrier E_a, and its low temperature time extrapolation,

$$t_{\text{low}} = t_{\text{high}} \exp\left[E_a \left(\beta_{\text{low}} - \beta_{\text{high}}\right)\right],$$

   where t_{high} is the amount of MD simulation time elapsed between the start of the simulation and when the transition to configuration X_2 occurs, $\beta_{\text{low}} = (k_B T_{\text{low}})^{-1}$, $\beta_{\text{high}} = (k_B T_{\text{high}})^{-1}$, and k_B is Boltzmann's constant. Also, compute and store an estimate of the high temperature stop time,

$$t_{\text{high,stop}} = \frac{\ln(1/\delta)}{\nu_{\text{min}}} \left(\frac{\nu_{\text{min}} t_{\text{low}}}{\ln(1/\delta)}\right)^{\frac{\beta_{\text{high}}}{\beta_{\text{low}}}}$$

   Presumably, the atomic velocities for configuration X_2 are also stored.

7. Return the system to configuration X_1 and reverse the velocities for all atoms. As a Langevin thermostat is used (which injects noise into the atomic trajectories), this will not result in time reversed trajectory of the system.

8. Proceed from step 3, continuing to check and discover new minimized states different from either A or B. Repeat steps 4 - 6, with one exception: instead of using a new configuration's t_{low} to estimate $t_{\text{high,stop}}$, use the shortest among all "new" configurations, $t_{\text{low,short}}$. With $t_{\text{low,short}}$ and $t_{\text{high,stop}}$ defined, δ represents the probability that further simulation at T_{high} for times $t > t_{\text{high,stop}}$ will result in finding a state with a shorter event time $t < t_{\text{low,short}}$. S&V try to keep this value at 0.001 to 0.05 for most problems. Since $t_{\text{high,stop}} \propto \ln(1/\delta)^{1-\frac{T_{\text{low}}}{T_{\text{high}}}}$, the lower δ is set, the higher $t_{\text{high,stop}}$ will be.

9. Repeat steps 3 - 8 until $t_{\text{high}} \geq t_{\text{high,stop}}$. Then, select the configuration that corresponds to $t_{\text{low,short}}$ and repeat steps 2 - 8. It probably makes sense to reset all temporary storage containers at this point, but keep track of total elapsed low temperature time, $\sum_{\text{(all transitions)}} t_{\text{low,short}}$, and low temperature configurations.

2.4.2 Details of LAMMPS implementation

The standard TAD algorithm of Section 2.4.1 was implemented in LAMMPS as a new "tad" input script command. Although it differs from PRD in many respects, the code implementation reuses many of the internal features that were developed for PRD. It also makes direct use of LAMMPS' NEB functionality described in the previous section. Similar to NEB and PRD calculations, TAD within LAMMPS runs as a multi-partition simulation. In the current LAMMPS implementation of TAD, the partitions are only used to run TAD's NEB calculations. All the non-NEB TAD operations are performed on the first partition, while the other partitions remain idle. Multiple partitions are invoked in a LAMMPS simulation, by using the "-partition" command-line switch to allocate M partitions of processors for the simulation. See the discussion at the beginning of Section 2.2.2 for how such a simulation is invoked in parallel with LAMMPS.

As described in Section 2.3.4, the first TAD implementation in LAMMPS was restricted to exactly one processor per replica. This restriction has now been lifted, so that it is possible to have more than one processor per partition.

The syntax for the "tad' command added to LAMMPS is as follows:

```
tad Nstep t_event T_lo T_hi delta tmax compute-ID keyword value ...
```

where

- Nstep = # of timesteps to run (not including dephasing/quenching)

- t_event = timestep interval between event checks
- T_lo = temperature at which event times are desired
- T_hi = temperature at which MD simulation is performed
- delta = desired confidence level for stopping criterion
- tmax = reciprocal of lowest expected preexponential factor (time units)
- compute-ID = ID of the compute used for event detection

As described in Section 2.4.1, a TAD run has several stages, which are repeated each time an event is performed. These stages are summarized below and each stage is cross-referenced to the corresponding step of the standard TAD algorithm given in Section 2.4.1.

```
while (time remains):
  while (time < tstop):
    until (event occurs):
      run dynamics for t_event steps (step 3)
      quench (step 3)
    run neb calculation using all replicas (step 5)
    compute tlo from energy barrier (step 6)
    update earliest event (step 6)
    update tstop (step 6)
    reflect back into current basin (step 7)
  execute earliest event (step 9)
```

Before the outer loop begins, the initial potential energy basin is identified by quenching (an energy minimization, see below) the initial state and storing the resulting coordinates for reference. Inside the inner loop, dynamics is run continuously according to whatever integrator has been specified by the user, stopping every *t_event* steps to check if a transition event has occurred. This check is performed by quenching the system and comparing the resulting atom coordinates to the coordinates from the previous basin. A quench is an energy minimization and is performed by whichever algorithm has been defined by the *min* and *min_style* keywords or their default values. Note that typically, you do not need to perform a highly-converged minimization to detect a transition event.

The event check is performed by a compute with the specified compute-ID. Currently there is only one compute that works with the TAD command, which is the "compute event/displace" command. Other event-checking computes may be added. Compute event/displace checks whether any atom in the compute group has moved further than a specified threshold distance. If so, an "event" has occurred. The NEB calculation is similar to that invoked by the *neb* command, except that the final state is generated internally, instead of being read

in from a file. The TAD implementation provides default values for the NEB settings, which can be overridden using the *neb* and *neb_style* keywords.

A key aspect of the TAD method is setting the stopping criterion appropriately. If this criterion is too conservative, then many events must be generated before one is finally executed. Conversely, if this criterion is too aggressive, then high-entropy, high-barrier events will be over-sampled, while low-entropy low-barrier events will be under-sampled. If the lowest pre-exponential factor is known fairly accurately, then it can be used to estimate *tmax*, and the value of *delta* can be set to the desired confidence level e.g. $delta = 0.05$ corresponds to 95% confidence. However, for systems where the dynamics are not well characterized (the most common case), it will be necessary to experiment with the values of *delta* and *tmax* to get a good trade-off between accuracy and performance.

A second key aspect is the choice of the high temperature *t_hi*. A larger value greatly increases the rate at which new events are generated. However, too large a value introduces errors due to anharmonicity (not accounted for within hTST). Once again, for any given system, experimentation is necessary to determine the best value of *t_hi*.

Five kinds of output can be generated during a TAD run: (1) event statistics, (2) NEB statistics, (3) thermodynamic output by each replica, (4) dump files, and (5) restart files.

Event statistics are printed to the screen and master "log.lammps" file each time an event is executed. The quantities are the timestep, CPU time, global event number N, local event number M, event status, energy barrier, time margin, *t_lo* and *delt_lo*. The timestep is the usual LAMMPS timestep, which corresponds to the high-temperature time at which the event was detected, in units of timestep. The CPU time is the total processor time since the start of the TAD run. The global event number N is a counter that increments with each executed event. The local event number M is a counter that resets to zero upon entering each new basin. The event status is E when an event is executed, and is D for an event that is detected, while DF is for a detected event that is also the earliest (first) event at the low temperature. The time margin is the ratio of the high temperature time in the current basin to the stopping time. This last number can be used to judge whether the stopping time is too short or too long (see above). *t_lo* is the low-temperature event time when the current basin was entered, in units of timestep. *delt_lo* is the time at which each is detected event, measured since entering the current basin. *delt_lo* is calculated by rescaling the high-temperature time since entering the current basin, as explained in step 6 of the standard TAD algorithm given in Section 2.4.1. On lines for executed events, with status E, the global event number is incremented by one, the local event number and time margin are reset to zero, while the global event number, energy barrier, and *delt_lo* match the last event with status DF in the immediately preceding block of detected events. The low-temperature event time *t_lo* is incremented by *delt_lo*.

The NEB statistics are written to the file specified by the *neb_log* keyword. If the keyword value is *none*, then no NEB statistics are printed out. The statistics are written every $Nevery$ timesteps. See the description of the NEB implementation in Section 2.3.4 for a full description of the NEB output statistics. When invoked from TAD, NEB statistics are never

printed to the screen.

Because the NEB calculation must run on multiple partitions, LAMMPS produces additional screen and log files for each partition, e.g. "log.lammps.0", "log.lammps.1", etc. For the TAD command, these contain the thermodynamic output of each NEB replica. In addition, the log file for the first partition, "log.lammps.0", will contain thermodynamic output from short runs and minimizations corresponding to the dynamics and quench operations, as well as a line for each new detected event, as described above. After the TAD command completes, timing statistics for the TAD run are printed in each replica's log file, giving a breakdown of how much CPU time was spent in each stage (NEB, dynamics, quenching, etc).

Any dump files defined in the input script will be written to during a TAD run at timesteps when an event is executed. This means the requested dump frequency in the dump command is ignored. There will be one dump file (per dump command) created for all partitions. The atom coordinates of the dump snapshot are those of the minimum energy configuration resulting from quenching following the executed event. The timesteps written into the dump files correspond to the timestep at which the event occurred and NOT the clock. A dump snapshot corresponding to the initial minimum state used for event detection is written to the dump file at the beginning of each TAD run.

If the restart command is used, a single restart file for all the partitions is generated, which allows a TAD run to be continued by a new input script in the usual manner. The restart file is generated after an event is executed. The restart file contains a snapshot of the system in the new quenched state, including the event number and the low-temperature time. The restart frequency specified in the restart command is interpreted differently when performing a TAD run. It does not mean the timestep interval between restart files. Instead it means an event interval for executed events. Thus a frequency of 1 means write a restart file every time an event is executed. A frequency of 10 means write a restart file every 10th executed event. When an input script reads a restart file from a previous TAD run, the new script can be run on a different number of replicas or processors.

Note that within a single state, the dynamics will typically temporarily continue beyond the event that is ultimately chosen, until the stopping criterion is satisfied. When the event is eventually executed, the timestep counter is reset to the value when the event was detected. Similarly, after each quench and NEB minimization, the timestep counter is reset to the value at the start of the minimization. This means that the timesteps listed in the replica log files do not always increase monotonically. However, the timestep values printed to the master log file, dump files, and restart files are always monotonically increasing.

2.4.3 Example TAD calculation

We have validated the LAMMPS implementation of TAD by direct comparison with trajectories generated using standard MD at a high temperature. We chose the same vacancy

diffusion example used to validate the PRD implementation, described in Section 2.2.3. The TAD low temperature was set to $T\_lo = 2000$ K, the same as that used for the NVT MD simulations. The high temperature was chosen to be $T\_hi = 2300$ K; higher values caused melting of the crystal. The total duration of the high temperature TAD simulation was 1000 ps. Only 3 processors were used, so that the NEB calculation consisted of two end-point replicas and one intermediate replica. Because of the high symmetry of the vacancy hopping pathway, a single intermediate replica was sufficient to accurately locate the transition state and calculate its energy. The compute event/displace threshold displacement was 0.5 Å. The following *tad* command was used:

```
tad 1000000 50 2000 2300 0.01 0.01 event     &
min 1e-05 1e-05 100 100      &
neb 0.0 0.01 200 200 20      &
min_style cg     &
neb_style fire     &
neb_log log.neb
```

Figure 2.4 shows a comparison of the mean square displacement of the vacancy versus time for TAD and NVT MD. The x, y, and z components are reported separately, to indicate the statistical uncertainties in both data sets. The TAD results lie within the statistical range of the NVT MD results, but appear to be slightly lower, on average.

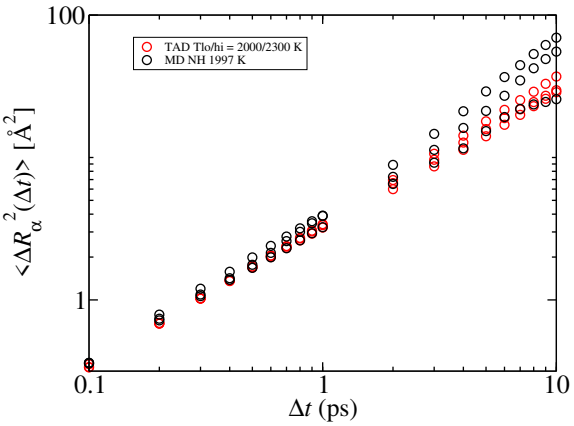


Figure 2.4. Comparison of MD (black) and TAD (red) results for mean squared displacement of a silicon vacancy versus time at 2000K.

In order to demonstrate the large speed-ups that are achievable using temperature acceleration, we repeated the 1000 ps TAD simulation using values of T_lo equal to 1000 K,

500 K, and 300 K, while keeping $T\_hi = 2300$ K. Whereas each of these simulations required roughly the same amount of CPU time, the low temperature time simulated increase exponentially with decreasing temperature. We define the effective speed-up to be:

$$SpeedUp = \frac{[CPU_{MD}(t)/t]_{2300K}}{[CPU_{TAD}(t_{lo})/t_{lo}]}. \tag{2.3}$$

where $CPU_{MD}(t)$ is the CPU time for NVT MD at 2300 K as a function of simulation time t, and $CPU_{TAD}(t_{lo})$ is the CPU time for TAD as a function of the low temperature simulation time t_{lo}. Figure 2.5 shows speed-up as a function of t during the course of the simulations. For the case of $T\_lo = 2000$ K, the speed-up is less than one, due to the overhead of performing NEB calculations, etc. However, the speed-up increases greatly with decreasing temperature. At 300 K, the speed-up is about 10^7.

2.5 Enhancements to SPPARKS

Another tool used in this project to model dynamics at the "fine" scale was the SPPARKS package for performing parallel kinetic Monte Carlo (KMC) simulations [40]. KMC models of materials can be parameterized at various length and time scales. For example, the atomic scale can be modeled as atoms on an fcc lattice, or the mesoscale can be modeled as chunks of aggregated material on a simpler cubic lattice. "Events" are defined which represent, for example in the atomic case, atoms diffusing to neighboring lattice sites with associated rate constants. These could depend on the height of energy barriers which in turn depend on the energetics of different defect configurations, which can be calculated from more fundamental methods such as molecular dynamics or *ab initio* quantum formulations.

The KMC method generates a sequence of events with correct relative probabilities so that the system evolves over time with accurate dynamics. The method is efficient, since the computational effort is focused on actual events. The SPPARKS package enables user definition of new KMC models and their events, either at the atomic or mesoscale, and can evolve systems in parallel, by breaking up the physical domain into sub-sections, one per processor.

Aside from developing new models specific to this project, several enhancements were made to SPPARKS to enable larger-scale simulations:

- The internal "group" solver was made more robust and easily usable for materials style models. This solver is unique in that it can select an event from a list of N possible events (e.g. N lattice sites) in constant $O(1)$ time, *independent* of the size of N.
- The internal labeling of lattice sites and other associated logic was extended to be 64-bit, which now allows models with up to $\sim 10^{19}$ sites to be simulated. The previous limit was around 2 billion sites (due to 32-bit integers).

- An on-the-fly parallel visualization capability was added to make it easier to view and analyze the results from very large simulations. On a requested timestep, each processor renders (draws) the lattice sites it owns into an JPEG image buffer. The set of buffers are merged (one for each of P processors) into a single image which is written to disk. While this approach is not interactive (the user must specify view parameters in advance, such as resolution and viewing angle), it dramatically reduces the volume of output for large models and gives the user an animation of the model dynamics as soon as the simulation has finished, without the need for further post-processing.

An example of these enhancements in action, presented at a recent conference [42], is shown in Figure 2.6, for a grain growth (microstructural evolution) simulation using a Potts model, done in collaboration with materials scientists from 1800. These are snapshots at the end of two simulations of a billion-site model (1000 by 1000 by 1000 periodic lattice), each run for about 2 hours on 1728 cores of a Cray XT5 machine.

The colors represent regions of the model where all the material has the same underlying lattice orientation, i.e. a grain in the polycrystalline material. The model was initialized with each of the billion sites having a different randomized orientation (a billion grains). Over time some grains grow and others vanish, leaving a final state with a few large grains. The left panel shows "normal" grain growth driven by curvature at the boundaries between grains, whereby the boundary energy can be reduced if large grains grow at the expense of small grains. The model in the right panel contains a small 0.1% fraction of pinning sites which represent impurities in the material. Grain growth is inhibited as boundaries migrate though these pinning sites, and growth eventually stalls when all boundaries become "pinned". This leads to a different distribution of grain sizes and shapes which can be compared to electron microscope (TEM) images to validate and parameterize the computational models.

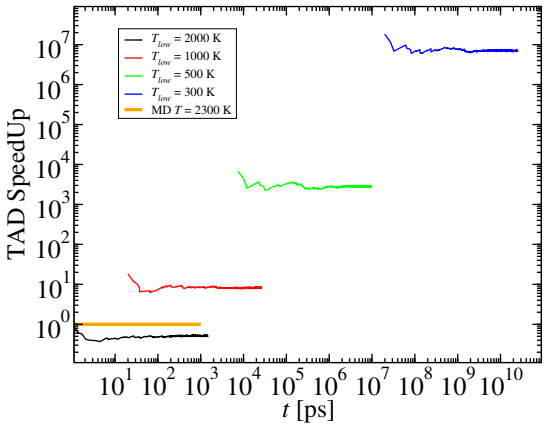


Figure 2.5. TAD speed-up versus simulation time, t or t_{lo}, for simulations of vacancy diffusion in silicon at different temperatures. The orange line represents the conventional MD simulation at 2300 K, whose speed-up is unity, by definition.

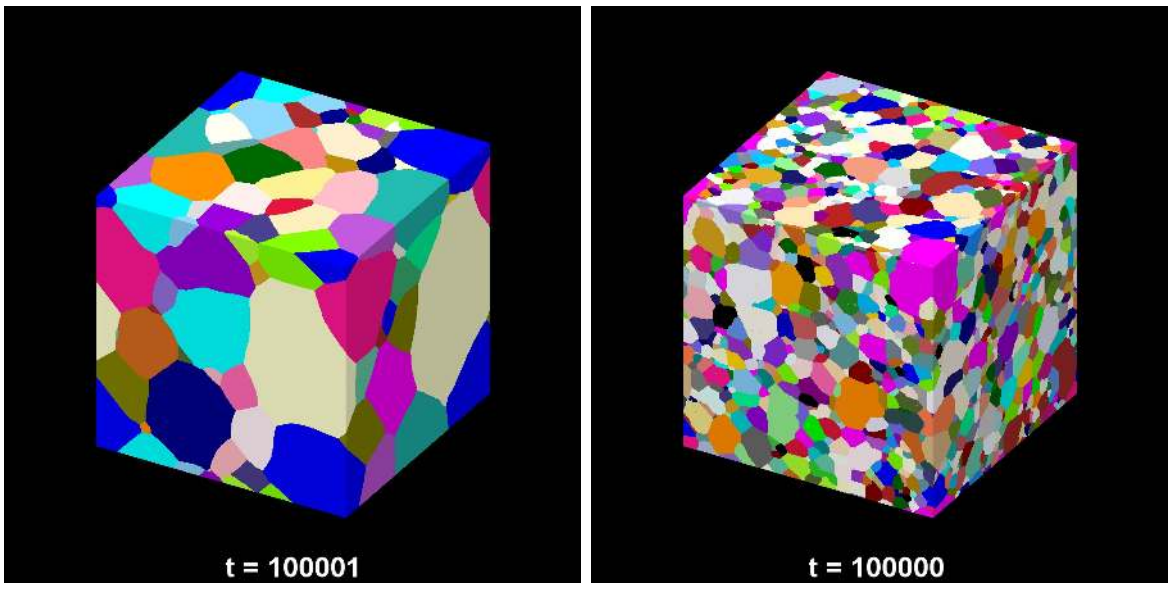


Figure 2.6. *Normal (left) and pinned (right) grain growth.*

Chapter 3

The Equation-Free Method for Surface Evolution in 2D

This section is in large part a reproduction of the paper "Equation-free accelerated simulations of the morphological relaxation of crystal surface." by Wagner, Zhou and Plimpton [64].

3.1 Introduction

The evolving morphology of solid surfaces due to diffusion is an important phenomenon in the synthesis, performance and aging of micro- and nano-structured materials and devices, and the ability to model these processes depends on the ability to quantitatively predict this diffusion. Continuum models for surface diffusion exist [32, 35], but for crystals these models are in general only valid above the roughening temperature of the material, below which singularities in the surface energy as a function of surface orientation lead to the formation of facets and other structures that cannot be described by existing coarse scale models.

An alternative to solving continuum equations is to simulate the individual atomic motions that drive surface diffusion. Since accurate interatomic potentials exist to describe many materials of interest at the atomic scale, these problems would seem to be candidates for modeling via molecular dynamics (MD) simulation. However, MD requires integration of timescales on the order of atomic vibrations (picoseconds), while many processes of interest in solids, including surface diffusion, occur on much larger timescales. Diffusion in solid crystals is a process driven by so-called rare events – motion of individual atoms or groups of atoms between lattice sites – that occur at a frequency much slower (by many orders of magnitude) than the atomic vibrational timescale. Methods designed to circumvent this timescale restriction eliminate or at least reduce the time spent simulating atom vibrations, focusing only on the rare events themselves. In some cases it is feasible to tabulate the possible events and their rates for any configuration, leading to an atomistic kinetic Monte Carlo (KMC) method [6, 17]. In other situations it is difficult or impossible to enumerate the possible events *a priori*, and a number of approaches have been developed to compute the most relevant events on-the-fly as the simulation proceeds, either through dynamic simulation

[63, 62, 55, 52, 61] or direct exploration of the energy surface [15].

Unfortunately, for large systems of atoms (such as might be observable in an experiment), these accelerated methods themselves have timescale limitations because of the very large number of events taking place. Simulation of morphological changes due to surface evolution, for example, becomes unfeasible for feature sizes more than a few nanometers in size, especially at low or moderate temperatures. To make further progress, we can note that the quantities most of interest to a modeler, such as the mean surface profile shape, can be described at a much coarser level than the atomic scale; the dilemma is that while we are interested in coarse scales over long times, we can only simulate fine scales over short times. The "equation-free" method was introduced by Kevrekidis et al. [23] to address problems of this type. In this approach, microscale computations (such as MD or KMC) are used as a set of numerical experiments from which can be distilled macroscale information, including coarse scale variables and their time derivatives. In this way the coarse scales of the system can be evolved in time even though a closed-form kinetic equation for these scales is unknown. The equation-free method has been applied to a number of multi-scale problems, including molecular dynamics of fluid diffusion [7], flow in a random medium [66], acoustic waves in a plasma [49], and flow in carbon nanotubes [56]; see Kevrekidis *et al.* [24] for a review and other applications.

In this paper we apply the equation-free projective integration (EFPI) method to a simple KMC model of surface diffusion in an attempt to accelerate the simulation of mean surface profile evolution. An important step in the application of EFPI to a given problem is the identification of an appropriate set of coarse variables with which to describe the problem; proper selection of these variables often requires physical intuition and insight. As will be shown, the most obvious choice of variables for the surface diffusion problem does not adequately parameterize the dynamics, and one of the main contributions of our work is the identification of a sufficient set of coarse variables. Furthermore, we show that some care must be taken to integrate these coarse variables in time in a stable manner, and propose a simple approximation that allows stable integration and large time acceleration of the simulation. Since our goal in this initial work is to study the feasibility of EFPI for this class of problems, we focus on the two-dimensional problem (a one-dimensional surface); this small problem size allows us to accurately simulate the full problem using the unaccelerated KMC model to provide a baseline for comparison. A successful method will allow the acceleration of problems of larger size, dimension and complexity that are too expensive to simulate using KMC alone. However, as will be discussed, extension to large 3D problems requires further development of some of our techniques, another motivation to limit the scope of this work to the 2D problem. Extension to 3D will be explored in future papers.

Finally, it should be noted that in this work we are concerned with the evolution of surface features that may be large, in the normal direction, compared with the atomic layer thickness. Features of this size are important, for example, when considering the formation or aging of nanopatterned surfaces or nanostructured materials. This problem is fundamentally different in its coarse scale description from surface phenomena that take place within a single monolayer, such as island growth (see, e.g., Bartelt *et al.* [2]). The large feature size in our

problem of interest allows us to consider the dynamics of the average surface profile, obtained through an ensemble average over many realizations of the same stochastic simulation. This form of profile averaging is not appropriate for island growth, for which a different set of statistics, such as average island size and density, must be considered.

3.2 Solid-on-Solid Model

The model for diffusion used in out test studies is the one-dimensional solid-on-solid (SOS) model, often referred to in the literature as "1+1 dimensional" since in effect it models the 1D surface of a 2D solid. The equilibrium statistical mechanics of this system have been elucidated by Leamy *et al.* [27]; an important characteristic of this model is that the roughening temperature is 0 K, so that any finite temperature is above the roughening transition (this is not the case for the equivalent model of the 2D surface of a 3D solid, the "2+1 dimensional" model). Both the 1+1 and 2+1 dimensional models have been used by several authors to study the evolution of crystal surface profiles [46, 48, 47, 45, 19, 34, 33, 57, 25].

The SOS model in 1D is parameterized by a set of positions i and the integer surface heights h_i. The energy of each site is proportional to the height difference between that site and its neighbors:

$$\begin{aligned} E &= J \sum_{i=1}^{L} \left[\frac{1}{2}|h_i - h_{i-1}| + \frac{1}{2}|h_{i+1} - h_i| \right] \\ &= J \sum_{i=1}^{L} |h_i - h_{i-1}| \end{aligned} \tag{3.1}$$

where L is the number of sites along the length of the system. The system is periodic, so that sites $i = 0$ and $i = L$ are equivalent. The bond energy J is taken to be 1.0 throughout this work.

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. For consistency with the 1+1 dimensional Monte Carlo simulations of Selke and Duxbury [47], we assume that the expected rate for any event can be computed as a function of the energy difference between the initial state (1) and final state (2):

$$P_{1\to 2} = \begin{cases} \frac{1}{2} & \text{if } \Delta E_{12} \leq 0, \\ \frac{1}{2} e^{-\Delta E_{12}/T} & \text{if } \Delta E_{12} > 0 \end{cases} \tag{3.2}$$

where ΔE_{12} is the change in total energy of the system going from state 1 to state 2, and T is the temperature of the system. The prefactor of 1/2 in Eqn. (3.2) is chosen so that the

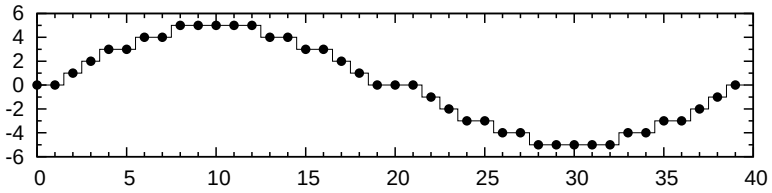


Figure 3.1. Initial condition from equation (3.3) with $L = 40$ and $b_0 = 5.5$.

timescale matches that of Selke and Duxbury [47], who report time in units of Monte Carlo steps per site (MCS/S).

To understand the dynamics of surface evolution described by this model, the decay of a sinusoidal surface is studied using KMC. In anticipation of eventual extension of the method to larger and more complex systems, we have used the SPPARKS KMC code for all simulations [41, 40]; this is a parallel Monte Carlo code for on-lattice and off-lattice models that includes algorithms for KMC as well as Metropolis Monte Carlo (MMC).

The initial condition for all simulations is:

$$h_i(t = 0) = \text{trunc}(b_0 \sin(2\pi i/L)) \tag{3.3}$$

where b_0 is the initial amplitude and trunc() is a function that truncates its argument to the next lower integer in magnitude. Figure 3.1 shows the initial condition corresponding to $L = 40$ and $b_0 = 5.5$.

Surface evolution data is computed by averaging over a large number of realizations, where a different random number seed is used to initialize the Monte Carlo simulation for each realization. The averaging operation will be denoted by an overline, so that $\bar{h}_i$ represents the height at position i averaged over all realizations:

$$\bar{h}_i = \frac{1}{N_R} \sum_{n=1}^{N_R} h_i^n \tag{3.4}$$

where h_i^n is the height at position i in realization n and N_R is the total number of realizations. Because this 1D problem runs very quickly, a large number of realizations can be run to obtain smooth data; 10^4 realizations are used to generate data in this paper.

The evolution of the surface can characterized by the Fourier coefficients of the average profiles:

$$A_k = \frac{2}{L} \sum_{i=0}^{L-1} \bar{h}_i \cos(2\pi ki/L) \tag{3.5a}$$

$$B_k = \frac{2}{L} \sum_{i=0}^{L-1} \bar{h}_i \sin(2\pi ki/L). \tag{3.5b}$$

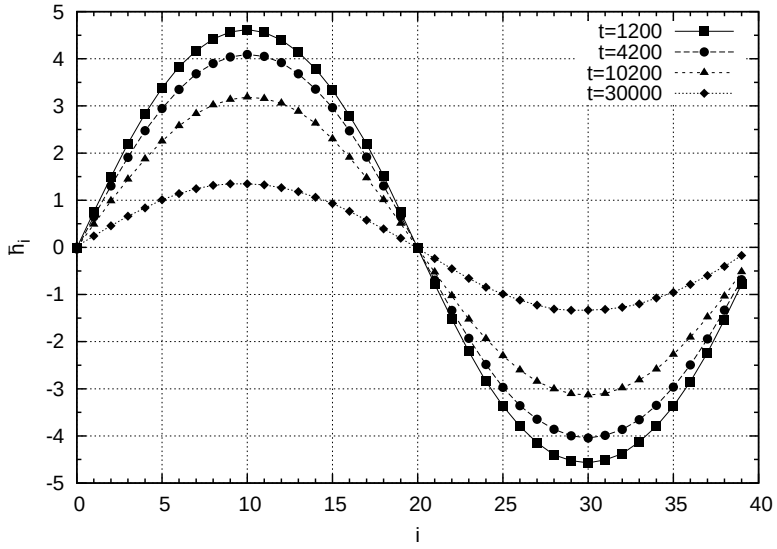


Figure 3.2. Average profile $\bar{h}(t)$ over time for $L = 40$, $b_0 = 5.5$, $T = 0.8$.

where A_k and B_k are respectively the sine and cosine Fourier coefficients. Also useful is the magnitude of the components, C_k:

$$C_k = (A_k^2 + B_k^2)^{1/2} \tag{3.6}$$

Figure 3.2 shows the average profile height $\bar{h}(t)$ for several different times for the case $L = 40$, $b_0 = 5.5$ at a temperature $T = 0.8$. The profile retains its sinusoidal shape. In Figure 3.3 the leading Fourier coefficient C_1 is plotted in time for various system sizes, along with a best fit for each curve to an exponential function; straight lines plotted on the semi-log axes imply that C_1 decays exponentially in time. The parameter τ given in the plot is the characteristic decay time for each curve, i.e. $C_1 \sim \exp(-t/\tau)$. Figure 3.4 shows the variation of τ with L on a log-log plot; the best-fit slope indicates that $\tau \sim L^{4.17}$.

Since the SOS model is intended to model the evolution of a surface profile under diffusion, these results can be compared with a continuum model of diffusion. The theoretical continuum equation for surface diffusion, assuming a small-amplitude surface profile, is a fourth-order PDE [32]:

$$\frac{\partial \bar{h}}{\partial t} = -B \frac{\partial^4 \bar{h}}{\partial x^4} \tag{3.7}$$

where B is a temperature-dependent surface mobility. This equation can be derived by assuming the flux of surface atoms is driven by gradients in the chemical potential, which is proportional to the surface curvature; for small amplitudes, the curvature is given by the second derivative of the profile height, leading to a fourth-order equation for the rate of change of the height. Self-similar solutions to Eqn. (3.7) on a periodic domain of length L

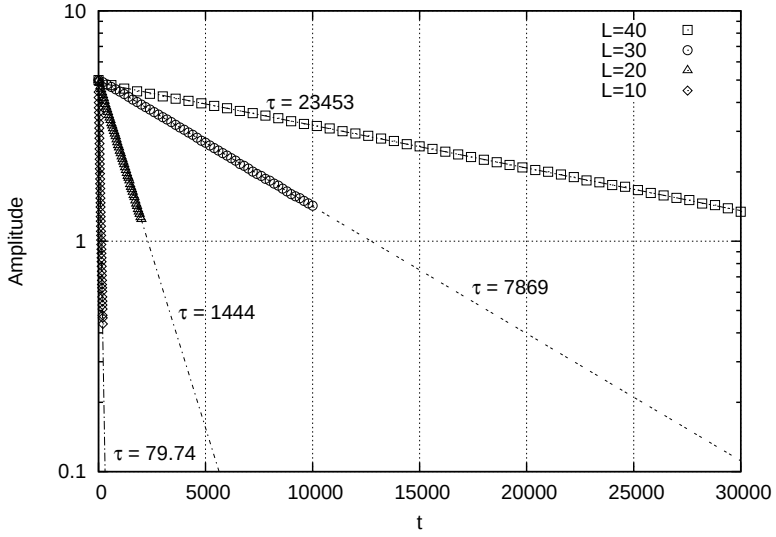


Figure 3.3. Semi-log plot and exponential fit of the decay of leading Fourier coefficient C_1 for various system sizes, with $b_0 = 5.5$, $T = 0.8$. Relaxation time τ for each value of L is the negative reciprocal of the slope of the line on the semi-log plot.

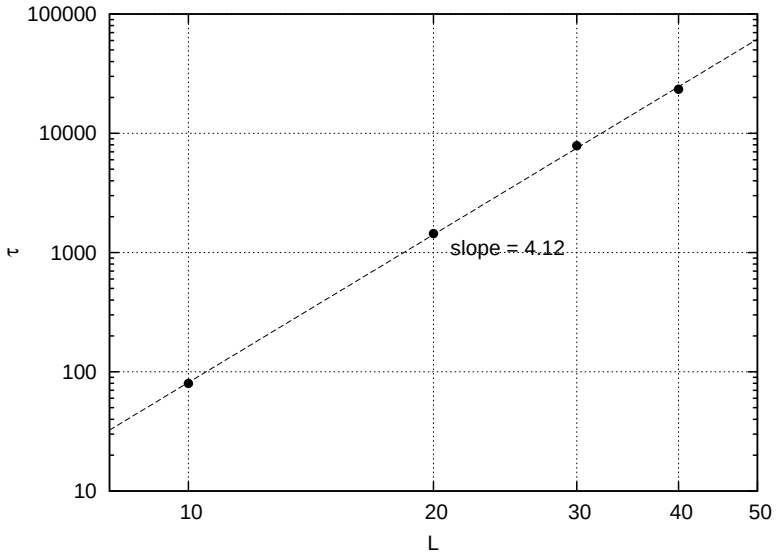


Figure 3.4. Variation of τ with L for $b_0 = 5.5$, $T = 0.8$. The slope of 4.17 is very close to the continuum theory prediction of 4.0.

have the form

$$\bar{h}(x,t) = b_0 \sin\left(\frac{2\pi n x}{L}\right) \exp\left(-\frac{B\,(2\pi n)^4\,t}{L^4}\right) \tag{3.8}$$

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

$$\tau = \frac{L^4}{B(2\pi n)^4} \tag{3.9}$$

This dependence of τ on L ($\tau \sim L^4$) is in good agreement with that computed using the SOS model for $n = 1$ ($\tau \sim L^{4.17}$), indicating that, at least for long wavelengths modes, surface evolution under the SOS model is well-described by the continuum model of surface diffusion.

However, as discussed in the Introduction, the continuum equation in (3.7) has known deficiencies in modeling real material surfaces. On the other hand, the SOS model (and all diffusion models that compute motion of individual atoms) are limited by computational expense in the domain size and time scale over which they can be used. The variables of interest, such as surface profile height as characterized by the leading Fourier coefficient, vary on timescales much slower than the individual atom hop rate; the difficulty is that Eqn. (3.7) may be inaccurate for the general case (such as 3D systems below the roughening temperature) and an accurate evolution equation for these slow variables is unknown.

3.3 Equation-Free Projective Integration

In order to extend simulations using the SOS model to longer timescales and larger systems, we will explore the use of equation-free projective integration (EFPI) [23]. EFPI is an approach to evolving the coarse scales of a system over a long time even though we are only able to explicitly simulate the fine scales of the system over short times. EFPI is related to, and in some cases a generalization of, temporal multiscale methods that have been developed for specific applications, such as the method of homogenization. For example, Oskay and Fish [36] have developed a time integration scheme for structural fatigue simulations using multiple time scales, in which the slow ("macro-chronological") solution is updated over long times based on the dynamics of the fast ("micro-chronological") solution over short times. One difference between EFPI and other techniques is the lack of any assumptions about the macroscale governing equation in EFPI; coarse scale evolution is driven solely by the fine scale dynamics. Techniques that require manipulation of a governing equation, such as homogenization, are not applicable in these cases.

In this report we will largely follow the notation of Kevrekidis et al. [23]. Consider a system described by a (possibly large) number of fine scale variables, represented by $\mathbf{u}$, and suppose that we can write a time evolution equation for these variables:

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}), \quad \mathbf{u} \in \mathbb{R}^m, \quad m \gg 1 \tag{3.10}$$

Equation (3.10) may represent deterministic or stochastic dynamics, and $\mathbf{u}$ may even represent an ensemble of systems (as it will in our application). The important point is that we are able to solve numerically for the time evolution of $\mathbf{u}$, although possibly over only very short times. We represent the solution of this equation symbolically through the *fine time stepper* $\mathcal{T}_f^t$, an operator defined such that:

$$\mathbf{u}(s+t) = \mathcal{T}_f^t \mathbf{u}(s) \tag{3.11}$$

if $\mathbf{u}$ is a solution to (3.10).

For many problems, we are not interested in all details of the solution, but in a reduced set of coarse scale variables that can be denoted $\mathbf{U} \in \mathbb{R}^M$, with $M \ll m$. These coarse scale variables are assumed to be a function of the fine scales through a *restriction* operator $\mathcal{M}$:

$$\mathbf{U} = \mathcal{M}\mathbf{u}. \tag{3.12}$$

The restriction operator may involve averaging over space, time, an ensemble of realizations, or some combination of all of these. A successful coarse scale model will require that all components of $\mathbf{U}$ are slowly varying compared with the fine scales $\mathbf{u}$. Ideally, the statistics of the variables that are not directly captured by $\mathbf{U}$ should be well-approximated as functionals of the selected coarse scale variables, even if the forms of these functionals are not known. If these conditions are true, the variables $\mathbf{U}$ parameterize a fast-attracting slow manifold for the dynamics.

We are interested in solving for the evolution of $\mathbf{U}$, which is assumed to behave according to its own evolution equation:

$$\frac{d\mathbf{U}}{dt} = \mathbf{F}(\mathbf{U}). \tag{3.13}$$

where $\mathbf{F}(\mathbf{U})$ may be unavailable. To summarize the problem, we want to simulate Eqn. (3.13) for large or infinite times, but are only able to simulate Eqn. (3.10) over short times.

As a first step toward solving this problem, we will approximate the solution to (3.13) over some length of time[1] τ, where τ is short enough so that the solution of (3.10) over this time, i.e. the evaluation of $\mathcal{T}_f^\tau \mathbf{u}$, is tractable, but long enough to allow a measurable change in the coarse scales $\mathbf{U}$. We introduce a *lifting* operator μ that maps the macroscopic description $\mathbf{U}$ to a consistent microscopic description $\mathbf{u}$. "Consistent" here means that $\mathcal{M}\mu = I$, i.e. lifting macroscopic to microscopic followed by restricting back to macroscopic has no net effect. The lifting operator is non-unique and involves the creation of information. With this operator in hand, we can define a *coarse time stepper* $\mathcal{T}_c^\tau$ over time period τ through a series of three steps. Beginning with $\mathbf{U}(t)$ known at some time t: 1) Lift to a a microscale description $\mathbf{u}(t) = \mu\mathbf{U}(t)$; 2) Evolve over time τ using the fine time stepper, $\mathbf{u}(t+\tau) = \mathcal{T}_f^\tau \mathbf{u}(t)$; and 3) restrict back to a macroscale description and define the coarse time stepper as $\mathcal{T}_c^\tau \mathbf{U}(t) \equiv \mathcal{M}\mathbf{u}(t+\tau)$. So the coarse time stepper is equivalent to

$$\mathcal{T}_c^\tau = \mathcal{M}\mathcal{T}_f^\tau \mu \tag{3.14}$$

[1]Note that τ in this section is unrelated to the characteristic decay time denoted with the same symbol in Section 3.2.

One detail that is important in designing the lifting operator and choosing the time τ is how quickly the fine scale variables, once initialized, approach the slow manifold parameterized by $\mathbf{U}$. That is, if the statistics of the fine scales are functions of the variables in $\mathbf{U}$, then regardless of the way in which these fine scales are initialized they should approach the correct values within a short amount of time. This "healing" time should ideally be much less than the coarse integration time τ.

With these concepts, we can describe projective integration over time scales that are long compared with τ. Let $\Delta t_c > \tau$ be a large time step (on the order of the slow coarse scale dynamics), and Δt_f be a small time step (on the order of τ). Introduce numerical approximations of the coarse solution $\mathbf{U}(t)$ as $\mathbf{U}^N \approx \mathbf{U}(N\Delta t_c)$, and let $\mathbf{u}^{N,0} = \mu\mathbf{U}^N$ be the lifted fine scale approximation at time $N\Delta t_c$. Let $t_j = j\Delta t_f$, where j is an integer and $t_j < \Delta t_c$. Using the coarse time stepper, we can compute

$$\mathbf{U}^{N,j} = \mathcal{T}_c^{t_j}\mathbf{U}^{N,0} \tag{3.15}$$

for a series of values of j in the range $[0, j_{max}]$; this is affordable as long as $t_{j_{max}}$ remains of the order of τ. Using these values of $\mathbf{U}^{N,j}$, we can design a time integration scheme by approximating the time derivative $\mathbf{F}(\mathbf{U})$ from equation (3.13) at these time points. For example, a version of the forward Euler integrator is obtained by extrapolating linearly to time $t = (N+1)\Delta t_c$:

$$U^{N+1} = U^{N,j_{max}} + (\Delta t_c - j_{max}\Delta t_f)\tilde{\mathbf{F}}(\mathbf{U}^{N,j_{max}}), \tag{3.16}$$

where $\tilde{\mathbf{F}}(\mathbf{U}^{N,j_{max}})$ is an approximation of $\mathbf{F}(\mathbf{U})$ at time $t = N\Delta t_c + j_{max}\Delta t_f$. A general form for this approximation can be written:

$$\tilde{\mathbf{F}}(\mathbf{U}^{N,j_{max}}) = \sum_{j=0}^{j_F} \alpha_j \mathbf{U}^{N,j_{max}-j} \tag{3.17}$$

where j_F controls the number of time points used in the approximation ($0 < j_F \leq j_{max}$) and α_j are a set of coefficients selected to best approximate the first order derivative of $\mathbf{U}^{N,j_{max}}$. For example, $j_F = 1, \alpha_0 = 1/\Delta t_f, \alpha_1 = -1/\Delta t_f$ gives a simple two-point approximation of the derivative. By using projective integration in this way, we can simulate the evolution of $\mathbf{U}(t)$ using time steps of size Δt_c while only needing to evolve the fine-scale system over times of size $j_{max}\Delta t_f$.

3.4 EFPI for the SOS Model

3.4.1 Coarse Time Stepper Tests

The first step in the application of EFPI to the SOS diffusion model is the choice of appropriate lift and restrict operators. As a first test of our operators, we will require that

they should allow the coarse scales of the system to be well approximated over time through repeated application of the coarse time stepper with no projective integration used at all. That is, by repeating the lift-integrate-restrict procedure multiple times, we should observe the same time evolution of the coarse scales as would be obtained by simply integrating the fine scales over the entire time using Equation (3.10). In the notation of Section 3.3, this is

$$\mathcal{T}_c^{n\tau}\mathbf{U}(0) \approx (\mathcal{T}_c^{\tau})^n\mathbf{U}(0), \tag{3.18}$$

or equivalently,

$$\mathcal{M}\mathcal{T}_f^{n\tau}\mu\mathbf{U}(0) \approx (\mathcal{M}\mathcal{T}_f^{\tau}\mu)^n\mathbf{U}(0). \tag{3.19}$$

Obviously, testing this requires that we solve on a system of only moderate size, so that computing the time evolution of the fine scales over a long time is affordable. As will be seen, satisfaction of (3.19) is not trivial for some obvious choices of lift/restrict operators, and attempts to do so elucidate some key points.

A natural choice for the set of coarse variables $\mathbf{U}$ is the set of ensemble averaged heights $\bar{h}_i$ defined in (3.4); the restriction operator $\mathcal{M}$ represents the averaging operation in that equation. Our first choice of lifting operator will be very simple: initialize h_i by setting each profile height for all realizations to the integer value closest to $\bar{h}_i$:

$$h_i^n = \text{round}(\bar{h}_i) \quad \forall n. \tag{3.20}$$

Figure 3.5 shows the evolution of the Fourier coefficient C_1 over time for one of the same cases examined in Section 3.2: $L = 40$, $T = 0.8$, and $b_0 = 5.5$. The solid line shows the curve obtained by using the fine scale time stepper for the entire time of 20,000 time units, i.e. $\tau = 20{,}000$ and $n = 1$ in equation (3.19). The other two lines show the results obtained if the lift and restrict operators are applied every 1000 or 2000 time units; for these cases $n = 20$ and $n = 10$, respectively, so that the total time is the same for all three runs. Clearly, the overall dynamics is disturbed by applying the restrict and lift operators, so much that in the case of $\tau = 1000$ the expected decay of the profile stagnates completely. In this case, every time the restrict and lift operators are applied, the coefficient C_1, although initially unchanged, begins to increase rapidly over a short period of time before slowly decaying again. The full solution itself (i.e. the solid curve) undergoes this brief increase at time $t = 0$, indicating that this behavior may be related to an initial equilibration of the solution after initialization to a smooth solution. For $\tau = 2000$, the dynamics follow the solid curve more closely, but still incur deviations every time the restrict and lift operators are applied.

More insight can be gained by examining a similar plot of the energy over time (Figure 3.6). The continuously integrated solution (solid line), after an initial transient, fluctuates around a slowly decreasing mean energy as expected. But the other two curves in the figure show that every time the restrict-lift operation is performed, the average energy decreases by a large amount because of the smoothness of the profile that results from the lift operation. For the $\tau = 2000$ case the solution approaches the fully integrated solution before the next perturbation, but for $\tau = 1000$ there is not enough time to re-establish the correct energy, and the profile statistics begin to deviate from the dynamics of the fully integrated solution.

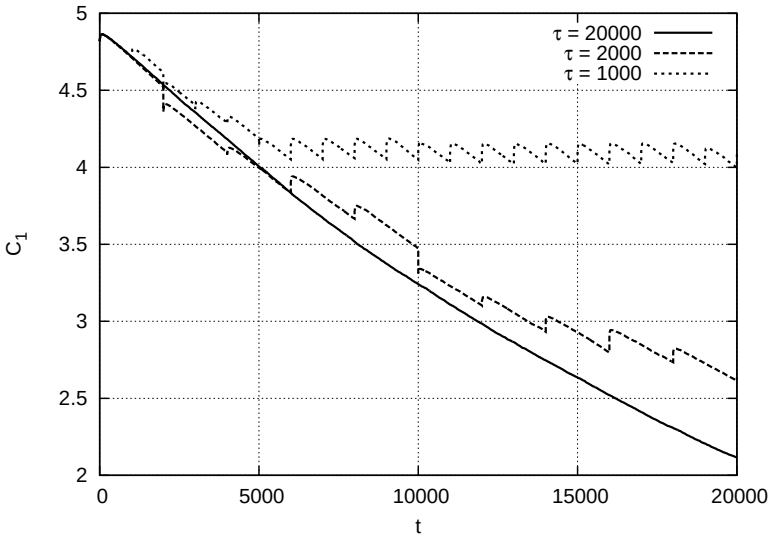


Figure 3.5. Magnitude of leading Fourier coefficient C_1 for various choices of coarse time step τ in equation (3.19), using lift and restrict operators based on the ensemble-averaged height array. Results are shown for $L = 40$, $T = 0.8$, and $b_0 = 5.5$.

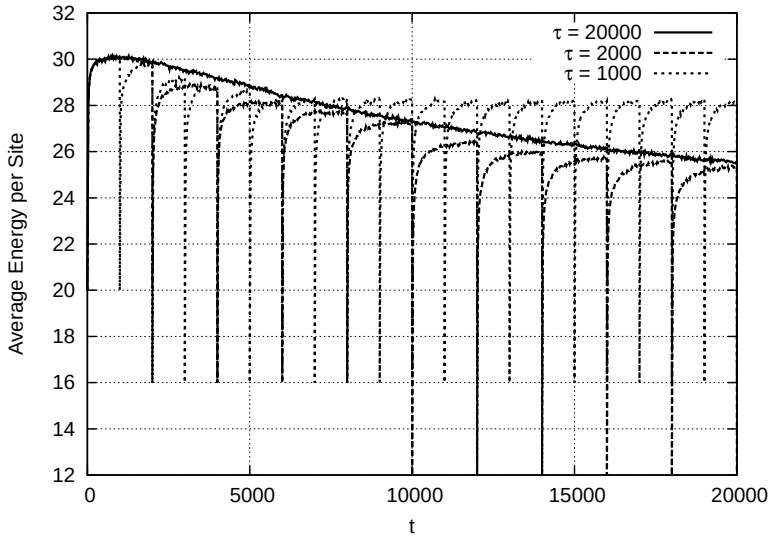


Figure 3.6. Average energy per site for various choices of coarse time step τ in equation (3.19), using lift and restrict operators based on the ensemble-averaged height array. Results are shown for $L = 40$, $T = 0.8$, and $b_0 = 5.5$.

The long healing time of the system does not allow a fast enough return to the slowly-varying manifold indicated by the solid line.

It is not hard to develop a hypothesis for the cause of this behavior. The surface profile can be thought of as a series of steps and ledges. Whenever our simple lift operator is applied, the resulting profile is smooth, with a minimum number of steps needed to give the correct average amplitude. As the fine-scale simulation of each individual realization proceeds, atoms begin to break away from ledges and diffuse across the steps, and adatom-vacancy pairs may spontaneously form on the steps. The motion of individual atoms and vacancies is fast compared with the decay of the profile shape, and within a moderate amount of time the atoms and vacancies on each step reach some equilibrium distribution. The motion of atoms and vacancies from step to step within this equilibrium drives the slower dynamics of the average profile decay. However, when we intermittently apply the restrict-lift operation, adatoms and vacancies are destroyed. If the time period τ is too short, there is not enough time to re-establish equilibrium on the steps, and an incorrect profile decay is predicted.

Therefore, to improve the design of our lift-restrict operators, we must incorporate statistical information about the fluctuations in the height profiles. Instead of computing the mean surface height everywhere, we can simplify the system by using only the magnitude of the first Fourier coefficient C_1 to parameterize the mean surface profile. We define $\tilde{h}_i$ according to the leading Fourier mode of our decaying sine wave, and $\hat{h}_i$ as the fluctuation around this value:

$$\tilde{h}_i = C_1 \sin\left(\frac{2\pi i}{L}\right) \tag{3.21a}$$

$$\hat{h}_i = h_i - \tilde{h}_i \tag{3.21b}$$

The average value $\bar{h}_i$ used in the simplest operator represents the first order moment of the profile. We can capture more information by using the second-order moment, represented here by the two-point correlation function $G(d)$:

$$G(d) \equiv \frac{1}{L}\sum_{i=0}^{L-1} \overline{\hat{h}_i \hat{h}_{i+d}} \tag{3.22}$$

where it is understood that the periodic boundary conditions are observed if $i + d > L$. As before, the overbar denotes ensemble averaging. Note that by its definition $G(d) = G(-d)$, and because of the periodic boundary condition $G(d) = G(L - d)$.

Also useful is the Fourier transform of $G(d)$, denoted by $G_K(k)$:

$$G_K(k) = \frac{1}{L}\sum_{d=0}^{L-1} e^{i2\pi kd/L} G(d) \tag{3.23}$$

The corresponding inverse transform is

$$G(d) = \sum_{k=0}^{L-1} e^{-i2\pi kd/L} G_K(k) \tag{3.24}$$

An important property of $G_K(k)$ is that it is related to the complex Fourier transform of the height fluctuations, $\hat{h}_K$:

$$\begin{aligned} G_K(k) &= \overline{\hat{h}_K^*(k)\hat{h}_K(k)} \\ &= \overline{|\hat{h}_K(k)|^2} \end{aligned} \tag{3.25}$$

where an asterisk denotes the complex conjugate.

A few more remarks on these quantities:

- It is a property of discrete Fourier transforms that if $f(d) = f^*(L-d)$ for all d, then the transform $f_K(k)$ is real for all k. Because $G(d)$ is real and $G(d) = G(L-d)$, then $G_K(k)$ is real for all k. Of course, this is also demonstrated by (3.25).
- Also by equation (3.25), $G_K(k) \geq 0$ for all k.
- Since $\hat{h}_i$ is real for all i, $h_K(k) = h_K^*(L-k)$ for all k.
- By periodicity, $h_K(0) = h_K(L)$; but by the previous statement, $h_K(0) = h_K^*(L)$. So $h_K(0) = h_K^*(0)$, implying that $h_K(0)$ is real. Likewise, if L is even, $h_K(L/2) = h_K^*(L/2)$ so that $h_K(L/2)$ is also real.

The single scalar value C_1 together with the values of $G(d)$ for $d = 0$ to $L/2$ make up a set of coarse scale variables for which we can write corresponding restrict and lift operators. To perform a restriction given a set of realizations for the height array, we first use Eqns. (3.4) - (3.6) to compute C_1, the magnitude of the leading order Fourier coefficient, then compute $G(d)$ from Eqns. (3.21) and (3.22). To lift from these coarse variables to a set of realizations, we take the Fourier transform of G and make use of Eqn. (3.25). At each realization n, for each value of k between 0 and $L/2$ we select a random phase ϕ with uniform probability on $[0, 2\pi]$, and assign $\hat{h}_K^n(k) = \hat{h}_K^{n*}(L-k) = (G_K(k))^{1/2}\, e^{i\phi}$ (while enforcing $\phi = 0$ for $k = 0$ and $k = L/2$). We then compute the inverse Fourier transform of $\hat{h}_K^n$ and add this fluctuation to the leading Fourier mode:

$$h_i^n = C_1 \sin\left(\frac{2\pi i}{L}\right) + \hat{h}_i^n \tag{3.26}$$

We repeated our test of the coarse scale time stepper using this new lift/restrict operator pair. Results are shown in Figures 3.7 and 3.8. The evolution of C_1 (Figure 3.7) shows much better agreement between the solutions, with only a small amount of deviation for the $\tau = 1000$ case; the $\tau = 2000$ curve follows the full solution almost exactly. The small increases in C_1 seen previously after every restrict/lift step are nearly completely removed. Even more interesting is the behavior of the system energy (Figure 3.8). Contrary to the previous choice of operators, in this case the energy actually *increases* significantly after each re-application. However, while the operators based on ensemble-averaging showed a long healing time back to the slowly varying dynamics, the operators using the 2-point correlation show a very fast healing time back to the continuously integrated solution. An explanation for this may be

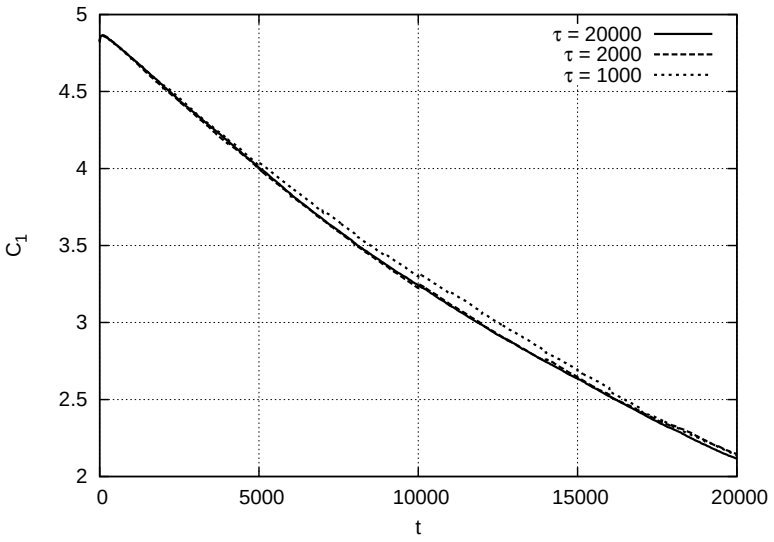


Figure 3.7. Magnitude of leading Fourier coefficient C_1 for various choices of coarse time step τ in equation (3.19), using lift and restrict operators based on C_1 and the 2-point correlation function $G(d)$. Results are shown for $L = 40$, $T = 0.8$, and $b_0 = 5.5$.

that, while the full statistics of the system are not perfectly reconstructed using the 2-point correlation function operator (as indicated by the increases in energy), fine scales in the system may be reproduced more accurately than for the ensemble-average operator, and healing can occur through processes taking place over much shorter length scales, allowing fast recovery. Again, we stress that in the equation-free method, we are less concerned with temporary deviations from the slow manifold, even if they are large, than with how quickly the solution heals and returns to the slow manifold.

3.4.2 Projective Integration Tests

Using the restrict/lift operators based on the 2-point correlation function described in the previous section, we can use EFPI to accelerate the time evolution of our simulations. The algorithm used follows equations (3.15)-(3.17), using a linear regression over $j_F + 1$ points to approximate the time derivative of the coarse scale variables. This is shown schematically in Figure 3.9. Given a set of coarse scale variables at time $t = N\Delta t_c$, represented by $\mathbf{U}^N$, we apply the lift operator to obtain a set of realizations for the height array, represented by $\mathbf{u}^{N,0}$. We then perform KMC simulations to advance the system by a total time of $j_{max}\Delta t_f$; we compute $\mathbf{U}^{N,j}$ for $j = 0$ to j_{max} by applying the restrict operator at intervals of Δt_f. We then fit a line to the final $j_F + 1$ points. By fitting through more than two points ($j_F \geq 2$) we can mitigate the effects of noise in the system; by choosing $j_F < j_{max}$ we can ignore the initial equilibration phase of the detailed simulation, during which the system may not

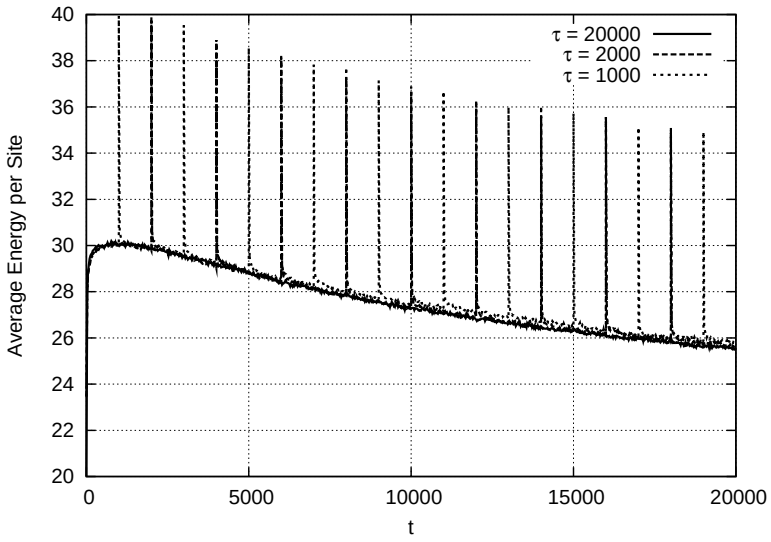


Figure 3.8. Average energy per site for various choices of coarse time step τ in equation (3.19), using lift and restrict operators based on C_1 and the 2-point correlation function $G(d)$. Results are shown for $L = 40$, $T = 0.8$, and $b_0 = 5.5$.

have returned to the correct slow manifold. Finally, we use this line to extrapolate the solution at time $t + \Delta t_c$. Note that this linear regression and extrapolation is computed for each individual element of the vector $\mathbf{U}$—in our case, the Fourier coefficient C_1 and all independent values of the 2-point correlation $G(d)$.

Application of EFPI to systems with lengths $L = 20$ and $L = 40$ are shown in Figure 3.10. Both cases use $\Delta t_f = 25$, $j_F = 6$, and $j_{max} = 12$, so that over each coarse time step a total time of $j_{max}\Delta t_f = 300$ is simulated using the full KMC simulation. The coarse time step is $\Delta t_c = 1000$, so that the effective speed-up of the method in these case is 10/3. In Figure 3.10 and subsequent similar plots, filled circles give the values of the fine scale system after restriction, sampled at intervals of Δt_f; dotted lines represent the best-fit line used to extrapolate over each coarse step. The two cases plotted highlight different sources of error in the method. For $L = 20$, the error is clearly dominated by that caused by the first-order time integration method used for projection (see Eqn. 3.16); at each coarse step the linear approximation to the curve causes an under-prediction of the correct value. This error can be alleviated by using a higher-order integration method, or by shrinking the coarse time step size.

For the $L = 40$ case, on the other hand, the first-order integration error does not appear to be the main source of error, as evidenced by the fact that the EFPI curve actually over-predicts the fully integrated value at times. Instead, the solution error apparently comes at least in part from errors in the derivative itself, i.e. those incurred in the approximation $\tilde{\mathbf{F}}(\mathbf{U})$ to the real derivative (Eqn. 3.17). Some of these errors are inherent to the approximation

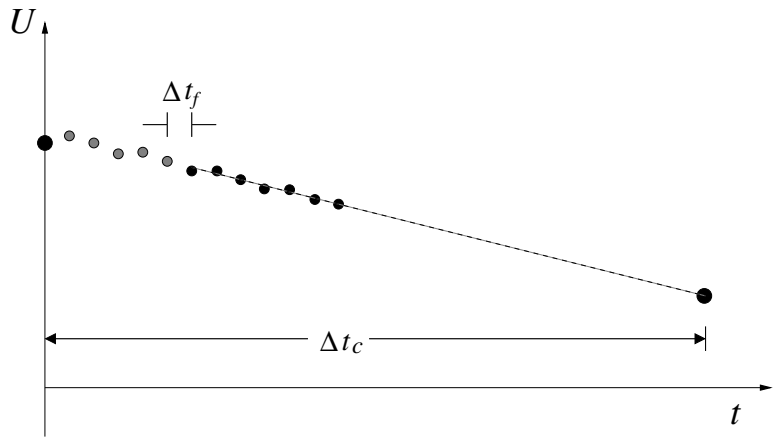


Figure 3.9. Schematic of a single coarse step in the projective integration scheme. The initial set of coarse variables **U** is lifted to a corresponding set of fine scale variables. The fine scale solution is updated for a time of $j_{max}\Delta t_f$ using KMC, and restrict operators are performed at intervals of Δt_f. For each coarse scale variable U, a linear regression is fit to the last $j_F + 1$ points (the black circles in the diagram), and this line is used to predict U at time interval of Δt_c. Using multiple points for the regression reduces the effects of noise on the solution. In the figure, $j_{max} = 12$ and $j_F = 6$.

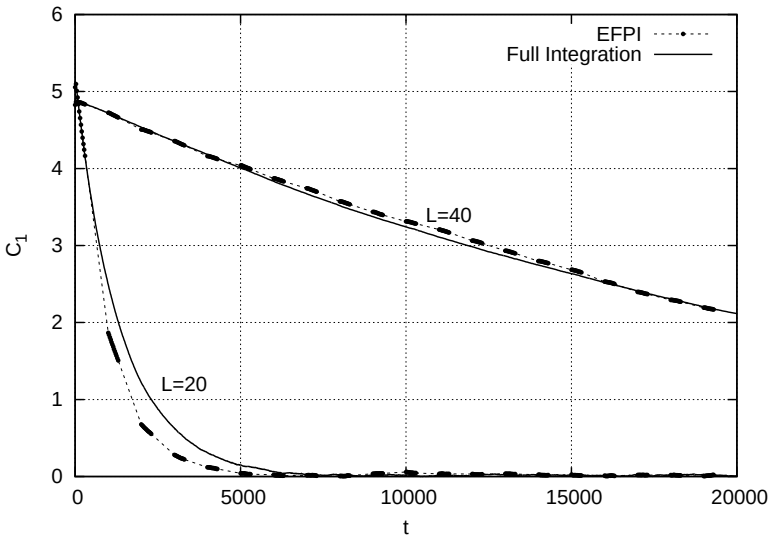


Figure 3.10. Application of EFPI to the SOS model with lengths of 20 and 40, using lift and restrict operators based on C_1 and the 2-point correlation function $G(d)$. Solid lines show the fully integrated solution (no projection). For the EFPI solutions, circles show the values of the fine scale system after restriction, sampled at intervals of $\Delta t_f = 25$; dotted lines represent the best-fit line used to extrapolate over each coarse step. For both cases, $j_{max} = 12$, $j_F = 6$, $\Delta t_c = 1000$, $T = 0.8$, and $b_0 = 5.5$.

in Eqn. (3.17) itself, but some may also be caused by fluctuations in the fine scale solution due to noise or to the application of the restrict/lift operators.

For given values of j_{max} and Δt_f, the effective speed-up of projective integration is proportional to the choice of the coarse time step size Δt_c. To improve speed-up, we tested coarse time steps of 1000, 2000, 4000, and 6000. Results are shown in Figure 3.11. It is clear that increasing Δt_c much above the original value of 1000 leads to poor results, limiting the speed-up attainable with this method.

To remedy this, we begin by noting that it was shown by Gear and Kevrekidis [11] that projective integration schemes similar to ours are stable for problems with both small and large eigenvalues, but are unstable for moderate eigenvalues if the ratio of Δt_c to $j_{max}\Delta t_f$ is too large. Furthermore, the region of instability in the eigenvalue spectrum grows as this ratio grows larger. Although we cannot directly compute the eigenvalues of our discrete problem, we can do so for the related continuum diffusion equation (3.7). For that problem, the eigenmodes are simply the Fourier modes, and the corresponding eigenvalue for each mode is related to the inverse of its characteristic relaxation time (Equation 3.9). Small eigenvalues correspond to slowly varying Fourier modes, large eigenvalues to fast modes. The Fourier modes of the height profile are related in turn to the two-point correlation function

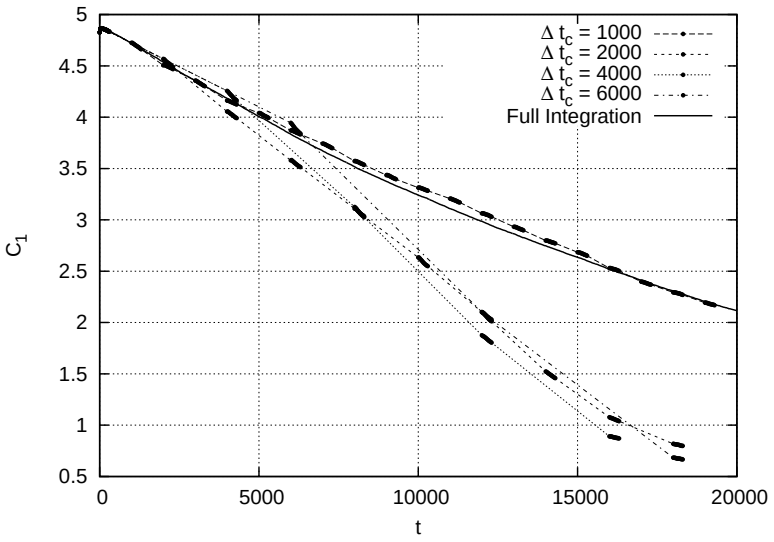


Figure 3.11. Application of EFPI to the SOS model with lengths of $L = 40$ for various choices of Δt_c. For all cases, $j_{max} = 12$, $j_F = 6$, $\Delta t_f = 25$, $T = 0.8$, and $b_0 = 5.5$.

through 3.25. Therefore, assuming that our discrete SOS model mimics the dynamics of the continuum diffusion equation, time integration of the model is equivalent to integrating a stiff set of equations, with a large range in eigenvalues. The values of $G_K(k)$ for various k provide a set of variables that can be conveniently separated into slow, moderate, and fast modes.

To demonstrate this, Figure 3.12(a) shows the evolution of $G_K(k,t)$ for the first few modes of the system, where we have started with a flat surface profile; in this case, the dynamics is driven by thermal fluctuations and each mode evolves toward an equilibrium amplitude that is determined by the finite system temperature. Clearly, $k = 1$ corresponds to a slow variable that is still evolving at the end of the simulation ($t = 2 \times 10^4$), while $k = 4$ is a fast variable that quickly attains its equilibrium value; here, "fast" is defined as having a rise time that is small compared to our fine scale time step Δt_f, and "slow" implies a time large relative to the size range of our coarse time step Δt_c. Modes associated with $k = 2$ and $k = 3$ have moderate timescales compared with our integration time step sizes. Figure 3.12(b) shows the effects that these time scales have on the projective integration of the variables. In this run, we have used the values of $G_K(k)$ for different values of k as our coarse scale variables for projective integration; since we require $G_K \geq 0$, we clip any negative values to zero after the projection step. A coarse time step of $\Delta t_c = 2000$ was used for the simulation shown.

The projective integration of $G_K(k,t)$ for $k = 1$ gives a solution that is similar to fully integrated curve in 3.12(a). However, the solutions for the $k = 2$ and $k = 3$ variables show large oscillations as the inadequately-sized timestep causes large overshoots. These

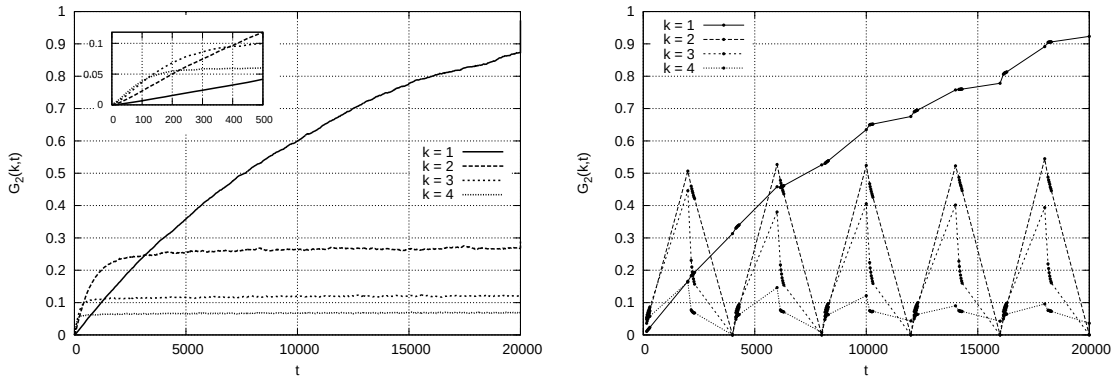


Figure 3.12. (a) Time evolution of the first four components of $G_K(k,t)$, computed using full KMC integration beginning with an initially flat profile. Inset shows early time. Simulation parameters were $L = 40$, $T = 0.8$. (b) Equation-free projective integration of the same problem, using the components of $G_K(k,t)$ as the projected coarse scale variables. $j_{max} = 12$, $j_F = 6$, $\Delta t_c = 2000$, $\Delta t_f = 25$.

oscillations prevent us from ever approaching the correct long-time dynamics. Interestingly, while the $k = 4$ undergoes oscillations, after a time it does remain close to the correct steady-state value; the healing time for this faster variable is nearly short enough to be captured within the fine scale integration time of $j_{max}\Delta t_f$.

To stabilize the integration, we must replace the projection of the fast and moderate variables, which leads to instability, with a different scheme. This new scheme is based on the recognition that the slow manifold we are trying to simulate should be parameterized only by the slow variables in our system; the fast variables should be "slaved" to the slow ones, as long as the system lies on the slow manifold. The problem then becomes how to specify the fast variables as a function of the slow ones. One simple approach is to set these fast values to their steady-state values, which can be computed in a single fully integrated simulation as in figure 3.12(a). However, anticipating applications where such steady states are unknown *a priori*, we will avoid this technique. Another approach to initializing the fast variables has been developed by Gear and coworkers [10, 12], in which an iterative scheme is used to project the fast variables toward the slow manifold; however, it is not clear that this method will be successful for the moderate-speed variables that lead to instability in our problem.

An approach that we have found to yield stable results with acceptable accuracy is to simply "freeze" the fast and moderate variables over the projection step. That is, rather than using linear extrapolation to compute the updated fast variables at time $t + \Delta t_c$, as we

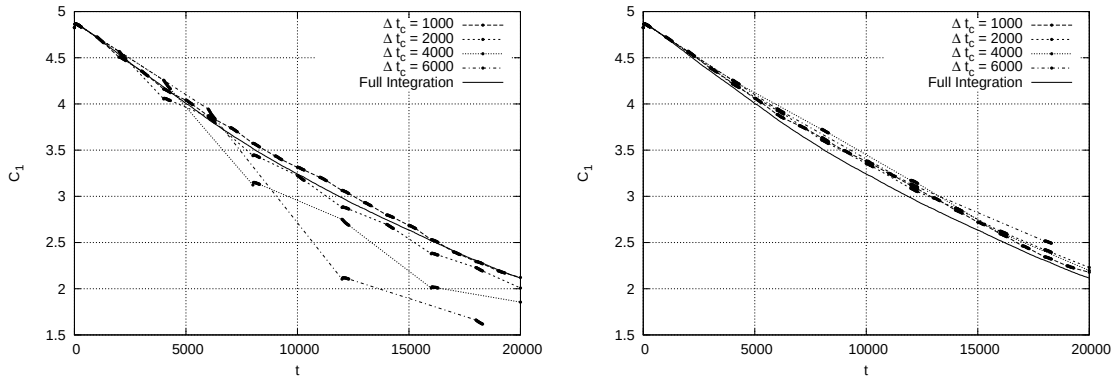


Figure 3.13. (a) EFPI of the SOS model with $L = 40$. Parameters are identical to those used in Figure 3.11, except that the elements of $G_K(k)$ (rather than $G(d)$) are used as the coarse variables. (b) EFPI of the same problem, but using Eqn. (3.27) to set the fast variables ($k \geq 2$) at each projection step.

do for the slow variables, we set:

$$G_K(k, t + \Delta t_c) = G_K(k, t + j_{max}\Delta t_f) \tag{3.27}$$

for $k \geq 2$. Application of this scheme is demonstrated in figure 3.13. Figure 3.13(a) shows a simulation identical to that in 3.11, except that the elements of $G_K(k)$ (rather than $G(d)$) are used as the coarse variables, with projective integration used for all values of k. Oscillations and loss of accuracy are seen for large values of Δt_c. In figure 3.13(b), the same cases are plotted using Eqn. (3.27) to update the fast variables at each projection step. Solutions remain stable and give good overall prediction of profile height decay, although some loss of accuracy is seen for large values of Δt_c.

3.5 Discussion and Conclusions

Our work demonstrates that equation-free projective integration can successfully be used to accelerate simulations of surface diffusion using the solid-on-solid model. However, at least two points seem to be crucial in obtaining accurate, stable results. The first is the recognition that the most obvious choice of coarse variables for the problem, describing the average surface profile, do not adequately describe the dynamics. This was demonstrated in our inability to recover the correct profile evolution after applying a lift operator based only on these variables. Instead, the coarse variables must include statistics that capture information about the fine scales in the system, since it is the fine scales themselves that drive the motion. The 2-point correlation function or its Fourier transform have been shown

here to adequately capture this fine scale information in a statistical sense and reconstruct the correct dynamics after lifting.

One limitation of the technique we have used to reconstruct the random system given the 2-point correlation function is the requirement that the correlation function $G(d)$ is known for all values of d so that the Fourier transform can be computed; this limitation is shared by the Karhunen-Loève expansion technique [29], another common method for reconstructing a Gaussian random field from its correlation function. The computation and storage of $G(d)$ may be unfeasible, especially for large problems in three dimensions. However, other techniques for reconstructing random fields have been introduced (e.g. [43, 67]), and their application to larger problems will be studied in future work.

A more fundamental question that is raised by our search for the correct set of coarse scale variables is whether there is a more systematic way of determining the appropriate parameterization than the trial-and-error approach used here. One possible approach is the "Baby-Bathwater Scheme" of Li *et al.* [28], in which the sensitivities of the macroscopic behavior to spatial derivatives of the coarse scale concentration profile are systematically explored. While this scheme can shed light on the nature of the coarse scale equation (assuming it exists), it is unclear whether it can be used to explore the possibly very large space of possible coarse scale variables in a complex problem. Kevrekidis and Samaey [24] point out the link between the identification of coarse variables in equation-free computations with dimensionality reduction techniques used in data mining (e.g. [59]); this analogy is a key area for future research. It should also be recognized that selection of the ideal set of coarse variables may not lead to accurate lifting if secondary ("fast") variables cannot be properly initialized. For example, in our surface diffusion problem the mean profile shape might be sufficient for lifting, *if* we were able to prescribe the 2-point correlation function as a function of the mean profile; because this function is unknown, we must track the correlation function with additional coarse variables. Gear *et al.* [10] have introduced a method for initializing fast variables by iteratively enforcing that time derivatives of these variables are equal to zero after a lift operation. Extension and testing of this idea on more complex problems is also an area of research.

The second important point in applying EFPI that our results demonstrate is that even when a set of coarse variables that correctly parameterize the system has been found, care must be taken to identify fast variables that lead to instabilities when projective integration is applied. This can be difficult, as when the fast variables are related to modes that may be "hidden" depending on the choice of parameterization. For example, when the values of the 2-point correlation function $G(d)$ for various d are used as a set of coarse variables, a separation into fast and slow variables is not obvious; only by transforming to $G_K(k)$ can we identify fast and slow modes. In the case of the SOS model for surface diffusion, the timescales of the various modes are most easily understood by referring to the continuum equation for surface diffusion, but in general it is precisely the lack of such a known continuum description that motivates the use of the equation-free method in the first place. An interesting direction for future research is to develop techniques for identifying the correct separation of fast and slow variables using only the fine scale description, without appealing to a continuum model.

Once the fast and moderate speed variables have been identified, the problem remains of how to initialize them at each projection step (assuming they cannot be stably integrated through projective integration). The method employed in our work, of keeping these values constant over the projection step, seems valid for the very fast variables (which presumably have reached their slow-manifold values during the fine scale simulation), but not necessarily for the more moderate-speed variables; in this case, we are simply constraining the variables to avoid instability rather than accurately predicting their new values. It is likely that this gives acceptable results only because it is the fastest variables (the smallest length scales) that drive the dynamics we are interested in. For problems where this is not the case, better strategies for initializing the unprojected variables must be developed.

Chapter 4

Improved Lift Operators for Surface Evolution

4.1 Introduction

In Section 3, we showed that the lift operator for 2D surface evolution simulations had to include the spatial correlation function as a coarse scale variable in order to successfully reproduce the dynamics of the system after lifting. This approach has several shortcomings, especially as we attempt to apply it to more complex systems in 3D:

- The 2-point correlation function may itself vary spatially in real problems, whereas we assumed in Section 3 that it was a function only of the distance d between two points (see for example equation 3.22); this extra dependence makes it impossible to reconstruct the height field using a convolution in Fourier space.
- The correlation between two sites in 3D has a directional dependence on the vector between the two sites that must be stored; this makes it much more expensive both to store the correlation function and reconstruct the height field.
- The height field for a given realization is a set of integers, while the values reconstructed using the correlation function are real values that must be rounded to the nearest integer; even in 2D, this rounding operation leads to modified statistics (which helps explain why we still see a jump in the energy function after lifting in Section 3).
- More generally, the importance of the 2-point correlation function was identified only after trial and error. We would like to develop an algorithm for lifting that is less problem-specific and does not require such *a priori* knowledge of which higher-order statistics are important.

The last point above is an important one. We expect that for these multi-scale systems, higher-order statistics are in some way slaved to the coarse mean field values; i.e. assuming the system is already on some slow manifold, there is a one-to-one mapping between the coarse slow variables (the average height profile, in this case) and the higher order statistics (the correlation function). Unfortunately, we do not know what this mapping is, and if the

higher order statistics are incorrectly initialized in our problem, we are no longer on the slow manifold.

Since we know only the coarse scale description of the system, one way to proceed in the lift operation might be to run the microscale system using the fine scale time integrator (e.g. KMC in our surface diffusion examples) while holding the coarse scale – and only the coarse scale – fixed. Over time, we may hope that the higher order statistics come into equilibrium with the constrained coarse scales, effectively giving the correct mapping and ensuring that the system is on the desired slow manifold. This procedure can be thought of as an artificial "healing" step, where the constrained variables do not change during the healing process (recall that this was not the case with the healing time in our examples in Section 3, when coarse variables tended to drift away from the desired values). Unfortunately, it is not always obvious how to constrain a general set of variables during a KMC simulation. One approach to doing this that holds promise in our early tests is the maximum entropy method.

4.2 Maximum Entropy Method

The maximum entropy method has its roots in the work of E. T. Jaynes [18]; a summary of the basic approach is given here.

For simplicity we will assume that we have a system that can be described in terms of discrete states, although extension to continuous systems is straightforward. These discrete states might be finite in number (e.g. the roll of a die has 6 possible outcomes) or infinite (e.g. our solid-on-solid surface model in Section 3 allows any combination of integer heights that preserve the total mass of the system). Let P_i be the probability of finding the system in state i. Our goal is to find the most likely values for the individual P_i's, given the known constraints of the system.

According to the maximum entropy principle, the appropriate values of P_i are those that maximize the entropy S, defined as:

$$S(P_1, P_2, ...) = -\sum_i P_i \ln P_i, \tag{4.1}$$

subject to the known system constraints. One constraint that is always present is that the sum of the probabilities must equal unity:

$$\sum_i P_i = 1 \tag{4.2}$$

For the case where no other information is known about the system, there are no other constraints, and the probabilities can be solved by maximizing the augmented entropy function $\bar{S}$:

$$\bar{S} = -\sum_i P_i \ln P_i - \nu(\sum_i P_i - 1) \tag{4.3}$$

where ν is a Lagrange multiplier. This function is maximized by taking the partial derivative with respect to P_i and setting equal to zero:

$$\frac{\partial \bar{S}}{\partial P_i} = 0 = -\ln P_i - 1 - \nu \tag{4.4}$$

giving

$$P_i = e^{-\nu-1}. \tag{4.5}$$

Setting the partial derivative of $\bar{S}$ with respect to ν to zero results in constraint (4.2). To solve for the Lagrange multiplier, we can substitute this result into this constraint; the final result for the probability is

$$P_i = \frac{1}{N}. \tag{4.6}$$

where N is the total number of possible states. Thus, in the absence of any other knowledge of the system, we conclude that all states are equally likely.

4.2.1 Average Energy Constraint

A simple non-trivial example of the use of the maximum energy principle is the case where the average system energy $\langle E \rangle$ is known; this is expressed through an additional system constraint:

$$\sum_i P_i E_i = \langle E \rangle \tag{4.7}$$

where E_i is the energy of the system in state i. We use an additional Lagrange multiplier β in the augmented entropy function to enforce this constraint:

$$\bar{S} = -\sum_i P_i \ln P_i - \nu\big(\sum_i P_i - 1\big) - \beta\big(\sum_i P_i E_i - \langle E \rangle\big) \tag{4.8}$$

Taking the partial derivative with respect to P_i and setting equal to zero yields:

$$P_i = e^{-1-\nu-\beta E_i} \tag{4.9}$$

To eliminate ν, we can again substitute into (4.2), giving:

$$e^{-1-\nu} = Z^{-1}, \tag{4.10a}$$

$$Z \equiv \sum_i e^{-\beta E_i}. \tag{4.10b}$$

The resulting expression for P_i is then:

$$P_i = Z^{-1} e^{-\beta E_i}. \tag{4.11}$$

We recognize this as the probability distribution function for the canonical ensemble, where Z is the partition function. The solution for the Lagrange multiplier β is more difficult; it must be specified so as to satisfy the average energy constraint (4.7):

$$\sum_i Z^{-1} e^{-\beta E_i} = \langle E \rangle \tag{4.12}$$

which can be rewritten:

$$\langle E \rangle = -\frac{\partial}{\partial \beta} \ln Z. \tag{4.13}$$

This is in general a nonlinear equation that must be solved by incorporating information about the form of the energy E_i; this is difficult to solve in closed form without a concise expression for Z as a function of β, which is only available for very simple systems. Note that the maximum entropy principle alone does not give us the relationship between β and temperature ($\beta = 1/k_B T$) that we know from classical statistical mechanics. To derive this expression requires more insight into the physics of the problem.

4.2.2 General Constraints

We can use the maximum energy principle to derive the form of the probability distribution function for the more general case of m system constraints, in addition to the constraint on the sum of the probabilities (4.2). Suppose we have m different functions, $f_k(\mathbf{x})$ for $k = 1$ to m, where $\mathbf{x}$ is a vector of all of the system variables; the ensemble average of each function f_k (designated $\langle f_k \rangle$) is to be constrained to a known value, F_k. These constraints can be written:

$$\sum_i P_i f_k(\mathbf{x}_i) = F_k. \tag{4.14}$$

Following the same procedure as above, we can write an augmented entropy function using Lagrange multipliers λ_k:

$$\bar{S} = -\sum_i P_i \ln P_i - \nu(\sum_i P_i - 1) - \sum_k \lambda_k (\sum_i P_i f_k(\mathbf{x}_i) - F_k) \tag{4.15}$$

The resulting form of the probability distribution is:

$$P_i = Z^{-1} \exp\big[-\sum_k \lambda_k f_k(\mathbf{x}_i)\big] \tag{4.16a}$$

$$Z = \sum_i \exp\big[-\sum_k \lambda_k f_k(\mathbf{x}_i)\big] \tag{4.16b}$$

We recognize that the microcanonical ensemble distribution derived in the previous section is just a special case of this more general expression. The Lagrange multipliers must be solved through the set of m nonlinear equations obtained by substituting into the constraints:

$$F_k = \langle f_k \rangle = -\frac{\partial}{\partial \lambda_k} \ln Z(\lambda_1, \lambda_2, ...), k = 1...m \tag{4.17}$$

A few additional properties of this general maximum entropy expression can be explored. Taking the partial derivative of $\langle f_k \rangle$ with respect to a given Lagrange multiplier λ_j gives:

$$\frac{\partial \langle f_k \rangle}{\partial \lambda_j} = -\frac{\partial^2}{\partial \lambda_j \partial \lambda_k} \ln Z \tag{4.18}$$

Assuming $\ln Z$ is a smooth function of the Lagrange multipliers, the right hand side of the above equation is insensitive to the order of j and k, implying the symmetric result:

$$\frac{\partial \langle f_k \rangle}{\partial \lambda_j} = \frac{\partial \langle f_j \rangle}{\partial \lambda_k} \tag{4.19}$$

These derivatives can be related to the statistical fluctuations of the f functions. First, note that the ensemble average of the product of two functions, $\langle f_j f_k \rangle$, is

$$\begin{aligned}
\langle f_j f_k \rangle &= \sum_i P_i f_j(\mathbf{x}_i) f_k(\mathbf{x}_i) \\
&= \sum_i Z^{-1} \exp\big[-\sum_l \lambda_l f_l(\mathbf{x}_i)\big] f_j(\mathbf{x}_i) f_k(\mathbf{x}_i) \\
&= Z^{-1} \sum_i \frac{\partial^2}{\partial \lambda_j \partial \lambda_k} \exp\big[-\sum_l \lambda_l f_l(\mathbf{x}_i)\big] \\
&= Z^{-1} \frac{\partial^2 Z}{\partial \lambda_j \partial \lambda_k}
\end{aligned} \tag{4.20}$$

This can be used to write the fluctuations of the f functions in terms of derivatives of the partition function:

$$\begin{aligned}
\langle (f_j - \langle f_j \rangle)(f_k - \langle f_k \rangle) \rangle &= \langle f_j f_k \rangle - \langle f_j \rangle \langle f_k \rangle \\
&= Z^{-1} \frac{\partial^2 Z}{\partial \lambda_j \partial \lambda_k} - Z^{-2} \frac{\partial Z}{\partial \lambda_j} \frac{\partial Z}{\partial \lambda_k} \\
&= \frac{\partial^2}{\partial \lambda_j \partial \lambda_k} \ln Z \\
&= -\frac{\partial \langle f_k \rangle}{\partial \lambda_j} = -\frac{\partial \langle f_j \rangle}{\partial \lambda_k}
\end{aligned} \tag{4.21}$$

4.3 A Solution Algorithm for the Maximum Entropy Lagrange Multipliers

The maximum entropy principle gives a probability distribution function (PDF) that, for the correct choice of Lagrange multipliers, obeys the desired constraints. This PDF can be used, for example, in a Metropolis Monte Carlo algorithm to sample from the desired ensemble and obtain realizations that obey the constraints; other variables, such as the higher order statistics of the system, can be expected to be at equilibrium in the sense that they maximize the entropy.

However, determination of the correct values of the Lagrange multipliers λ_k is not trivial. Several authors have used a method based on the maximum entropy principle, the so-called MaxEnt method, to generate random microstructures [26, 68]. In these works, iterative

methods are used to solve the nonlinear set of equations defined by equation (4.17) for the set of Lagrange multipliers. At the core of these methods is the use of a stochastic simulation method, such as the Metropolis Monte Carlo (MMC) method [9], to generate realizations with a given PDF, as well as corresponding ensemble averages. This algorithm will be given in the next section, while a general approach for iterative solution of the Lagrange multipliers will be outlined in Section 4.3.2.

4.3.1 The Metropolis Monte Carlo Algorithm using the MaxEnt PDF

The PDF in equation (4.16) has the form:

$$P_i = Z^{-1} \exp\big[-H(\mathbf{x}_i, \boldsymbol{\lambda})\big] \tag{4.22}$$

where the generalized Hamiltonian function is:

$$H(\mathbf{x}_i, \boldsymbol{\lambda}) = \sum_k \lambda_k f_k(\mathbf{x}_i). \tag{4.23}$$

The function H plays the same role as the term $E(\mathbf{x}_i)/k_BT$ in the classical MMC method [9]: at each step, a trial move is generated, and the change in value of $H(\mathbf{x}_i, \boldsymbol{\lambda})$ is computed. If $\Delta H < 0$, the trial move is accepted automatically; otherwise, the move is accepted with probability $\exp\big[-\Delta H\big]$. The algorithm is given in Algorithm 1.

A few remarks on this algorithm:

- The form of the trial move depends on the system being simulated, as well as any constraints on the system that are not explicitly captured in the functions f_k. For example, in our simulations of surface diffusion using the SOS model (Section 3.2), the total amount of material is assumed to be conserved. A reasonable form of a trial move in this case is to move individual "atoms" from one site to another, i.e. increment a column at random by 1 while decrementing a different column by 1.
- Note that in order to compute the change in the functions f_k, it is usually not necessary to actually compute the function at the new state; rather, the change in the function can be computed directly, and more efficiently, especially if effects of an event are local. For example, if f_k is the total energy of the system, and can be written as the sum of the energies of individual sites, Δf_k can be calculated as the change in energy at only the sites whose values are affected by the trial move (i.e. the sites involved in the move, and their nearest neighbors).
- It is important to compute updated statistics even when the trial move has been rejected. In this way, configurations that are energetically favorable, which are more likely to persist for multiple steps, are weighted more heavily in the computation of ensemble averages.

Algorithm 1 Metropolis Monte Carlo Method using a generalized maximum entropy PDF

Given a set of m Lagrange multipliers λ_k and functions of interest $f_k(\mathbf{x})$, $k = 1$ to m, and initial state $\mathbf{x}_0$
Compute each function of interest $f_{k,0} = f_k(\mathbf{x}_0)$ at the initial state.
for $i = 1$ to N_{steps} **do**
  Generate a trial new state, $\mathbf{x}_{trial}$
  Compute the change in each function of interest caused by the trial move: $\Delta f_k = f_k(\mathbf{x}_{trial}) - f_k(\mathbf{x}_{i-1})$
  Compute the change in H caused by the trial move: $\Delta H = \sum_k \lambda_k \Delta f_k$
  if $\Delta H < 0$ **then**
    accept = true
  else
    Generate random real number r with uniform distribution on $[0, 1]$
    if $r < \exp\big(-\Delta H\big)$ **then**
      accept = true
    else
      accept = false
    end if
  end if
  if *accept* = true **then**
    $\mathbf{x}_i = \mathbf{x}_{trial}$
    $f_{k,i} = f_{k,i-1} + \Delta f_k$ for $k = 1$ to m
  else
    $\mathbf{x}_i = \mathbf{x}_{i-1}$
    $f_{k,i} = f_{k,i-1}$ for $k = 1$ to m
  end if
  Update statistics (e.g. ensemble averages of each f_k, pair correlations, etc.)
end for

4.3.2 Iterative Methods for Solving for the Lagrange Multipliers

As discussed above, the set of Lagrange multipliers needed for the construction of the maximum entropy PDF, i.e. the solution to the nonlinear set of equations given in (4.17), can be solved for iteratively. A convenient way to understand this approach is as the minimization of the scalar quantity G:

$$G(\boldsymbol{\lambda}) = \ln Z(\boldsymbol{\lambda}) + \sum_k \lambda_k F_k \tag{4.24}$$

where as before F_k are the desired values of the constrained ensemble averages of function f_k. To see that this minimization is equivalent to solving (4.17), note that setting the derivatives of G with respect to the individual Lagrange multipliers to zero simply gives:

$$F_k = -\frac{\partial}{\partial \lambda_k} \ln Z(\boldsymbol{\lambda}) = \langle f_k \rangle. \tag{4.25}$$

Furthermore, the components of the Hessian matrix (the second derivatives with respect to $\boldsymbol{\lambda}$) are:

$$\frac{\partial^2 G}{\partial \lambda_j \partial \lambda_k} = \frac{\partial^2}{\partial \lambda_j \partial \lambda_k} \ln Z = \langle f_j f_k \rangle - \langle f_j \rangle \langle f_k \rangle \tag{4.26}$$

This is recognized as the covariance matrix for the random vector $\mathbf{f}$. Since covariance matrices are positive definite, the surface described by $G(\boldsymbol{\lambda})$ is concave upward everywhere, and the solution to (4.25) therefore gives a unique global minimum.

This minimization problem can now be solved using a gradient-based iterative method, such as nonlinear conjugate gradient. Details of such methods depend on the specific choice of method and are given elsewhere ([26, 68]), but the basic approach is as follows: Given an initial guess of $\boldsymbol{\lambda}$, the residual of equation (4.25) is computed. This is done by using the given values of λ_k to form the probability distribution function given by (4.16), and then using the resulting PDF in a stochastic simulation, such as the Metropolis Monte Carlo algorithm described in Section (4.3.1), to compute the ensemble averages of the functions f_k.

Typically, iterative methods also require the Hessian matrix to be computed. Again, this can be done using a Monte Carlo simulation to compute the variances of the functions f_k; by equation (4.26), these fluctuations give the components of the Hessian. A single Monte Carlo simulation at each iteration can give the ensemble averages and covariances of all of the f_k functions for a given set of values for λ_k. Using these values, the $\boldsymbol{\lambda}$ vector is updated according to the chosen iterative method. This process is repeated until the residuals of equation (4.25) are driven below a given tolerance.

With values for the λ_k Lagrange multipliers in hand, yet another MMC simulation can be used to generate realizations with the probability distribution function given by (4.16). In this case, configurations obtained at the end of each step of the MMC simulation can be taken as the set of realizations that form the result of the lift operation. Alternatively, a subset of realizations can be used (e.g. the result of every n^{th} step of the algorithm where n is an integer); these realizations will still be distributed according to the desired PDF.

In practice, the MaxEnt method described in this section has a number of shortcomings, all of which are mentioned in the literature (e.g. [26, 37]) and verified by our own numerical experiments:

1. Because there is an inherent statistical error in the MMC method that only goes away as the number of steps becomes very large, there is an uncertainty in the ensemble averages and fluctuations (i.e. the first and second derivatives of the function being minimized) at each iteration of the algorithm. This uncertainty can slow down or destroy the convergence of the method. This problem can be somewhat alleviated by modifying the method to allow for this uncertainty (see for example the stochastic conjugate gradient methods of Schraudolph and Graepel [44]), but in our own test problems we have found this issue to be difficult to surmount for large systems.

2. The problem of error in the MMC method is compounded by what Koutsourelakis [26] calls the *critical slowing down* phenomenon, in which the MMC method becomes

"stuck" in a low energy region of phase space and cannot efficiently sample the space. This lowers the accuracy of the MMC method even further. One possible remedy for this that we have explored is to use a method similar to the Kinetic Monte Carlo method, rather than MMC, which can improve efficiency for simulations close to equilibrium (see e.g. Bortz *et al.* [4]). This seems to improve results for some cases, but not all.

3. Because of the above issues, the computational cost of the method is very high. For example Patelli and Schueller [37] compared the MaxEnt method to two other methods for generating random microstructures, simulated annealing and the genetic algorithm, to reconstruct microstructures based on given sets of statistical correlation functions. They concluded that the MaxEnt method is too expensive for all but very small systems.

The high computational cost of the MaxEnt method makes it unattractive for use in conjunction with the Equation-Free Projective Integration (EFPI) method, since the purpose of EFPI is to allow acceleration of time integration; for this to be useful, the computation saved by projective integration must outweigh the cost of the lift operation. However, there are at least two advantages to the MaxEnt method that make it attractive, especially compared with other methods for reconstructing microstructures such as simulated annealing [67]:

1. The MaxEnt method can reconstruct a microstructure based on only a small number of coarse variables, while higher-order statistics come into equilibrium in a natural way.

2. Unlike simulated annealing, which provides a single realization at a time, the MaxEnt method allows the generation of a large number of realizations very quickly once the correct values of the Lagrange multipliers have been obtained. If our EFPI system requires an ensemble average over many realizations (as in our surface diffusion simulations), this is advantageous.

Because of these properties, we seek a more efficient implementation of the maximum entropy principle for generation of realizations in our lift operator.

4.4 Efficient Generation of Realizations using the Maximum Entropy Principle

4.4.1 The SOS Model in 3D

It is useful to make our ideas in this section concrete by considering our particular system of interest: the solid-on-solid (SOS) model of surface evolution described in Chapter 3. Recall

that in that model, the energy of each site is proportional to the height difference between that site and its neighbors:

$$E = \sum_{i=1}^{N} E_i \tag{4.27a}$$

$$E_i = \frac{J}{2} \sum_{j \in neigh_i} |h_i - h_j| \tag{4.27b}$$

where N is the total number of sites, J is the the bond energy (taken to be 1.0 throughout this chapter), and $neigh_i$ is the set of neighbors of site i. In our 3D simulations, where h_i is the height defined on a 2D square lattice, we count interactions with the 4 nearest neighbors, i.e. the neighbors that share an edge in the square lattice (and not the corner neighbors).

As in Chapter 3, we are interested in developing a lift operation to be used in an Equation-Free Projective Integration simulation. As before, the fine scale dynamics over short times will be simulated using a KMC method. The system evolves by executing a series of events with rates that can be computed based on the energy change. In the 3D case, an event consists of motion of a single atom from its original site to any one of that site's 8 nearest neighboring sites (including the corner neighbors). The rate for each event transforming the system from state 1 to state 2 is given by:

$$P_{1\to 2} = \begin{cases} \frac{1}{8} & \text{if } \Delta E_{12} \leq 0, \\ \frac{1}{8} e^{-\Delta E_{12}/k_B T} & \text{if } \Delta E_{12} > 0 \end{cases} \tag{4.28}$$

where ΔE_{12} is the change in total energy of the system going from state 1 to state 2, k_B is Boltzmann's constant, and T is the system temperature. Note that the only difference with the corresponding rate expression for the 2D system (Equation 3.2) is the prefactor, 1/8 instead of 1/2; in both cases this prefactor is the reciprocal of the number of possible events originating at each site (the number of neighboring sites to which an atom can move), and is chosen to make our effective timescale consistent with other authors [48].

As in the 2D case, the initial condition for our simulations will be a sine wave in the x direction, with wavelength equal to the system size and amplitude b_0. Let L_x and L_y be the number of sites in the x and y direction on our rectangular lattice of sites. The initial condition is simply:

$$h_i(t=0) = \text{trunc}(b_0 \sin(2\pi x_i / L_x)) \tag{4.29}$$

where again trunc() is a function that truncates its argument to the next lower integer in magnitude. We assume a grid spacing of 1, so that x_i (the x location of site i) is an integer in the range $[0, L_x - 1]$.

Surface evolution data is again computed by ensemble averaging over many realizations of the system:

$$\bar{h}_i = \frac{1}{N_R} \sum_{n=1}^{N_R} h_i^n \tag{4.30}$$

where h_i^n is the height at position i in realization n and N_R is the total number of realizations. The evolution can be characterized in terms of the Fourier coefficients of the height profiles. We will be especially interested in the pure x-directional modes, so that the coefficients for mode k for a given realization are:

$$A_k^n = \frac{2}{L_x L_y} \sum_{i=1}^{N} h_i^n \cos(2\pi k x_i / L_x) \tag{4.31a}$$

$$B_k^n = \frac{2}{L_x L_y} \sum_{i=1}^{N} h_i^n \sin(2\pi k x_i / L_x) \tag{4.31b}$$

The main quantities tracked in the evolution of a given set of realizations will be the ensemble averages and standard deviations of the Fourier coefficients:

$$A_k = \frac{1}{N_R} \sum_{n=1}^{N_R} A_k^n \tag{4.32a}$$

$$B_k = \frac{1}{N_R} \sum_{n=1}^{N_R} B_k^n \tag{4.32b}$$

$$\sigma_{A_k} = \left(\sum_{n=1}^{N_R} (A_k^n - A_k)^2 \right)^{1/2} \tag{4.32c}$$

$$\sigma_{B_k} = \left(\sum_{n=1}^{N_R} (B_k^n - B_k)^2 \right)^{1/2} \tag{4.32d}$$

Note that the ensemble averages of the Fourier coefficients, A_k and B_k, are equivalent to the Fourier coefficients of the ensemble averaged heights $\bar{h}_i$. The coefficients of the individual realizations are required, however, to compute the standard deviations of A_k and B_k.

Because the evolving surface, in the ensemble average, is fairly smooth, the average profile can be described well in terms of a small number of Fourier coefficients. Because the initial condition is a sine wave, the cosine modes (the values of A_k) are on average very close to zero; additionally, because of symmetry, the even-numbered sine modes are also negligible. Thus, only the odd-numbered sine waves (B_1, B_3, etc.) are typically needed to describe the ensemble-averaged surface.

Finally, as in the 2D case and by analogy with continuum surface diffusion, the short-wavelength modes are expected to evolve quickly, so that the slow dynamics of the system can be parameterized by just a few of the longest modes, i.e. the smallest values of k. Our goal, then, will be to trace the evolution of the first few odd-numbered sine modes (typically just B_1 and B_3). Because we have an ensemble of realizations, each of which will display slightly different evolution because of the stochastic nature of the dynamics, we expect the standard deviation of these variables to grow with time, and a complete description of the ensemble must also include these standard deviations. So the set of coarse variables that will be used for the EFPI method will include 4 variables: B_1, B_3, σ_{B_1}, and σ_{B_3}. The lift operator we design must generate a set of realizations consistent with these coarse variables. This is the operator we will develop in the next section.

4.4.2 Lift Operator Development

The simplest approach to generating a set of realizations consistent with our set of coarse scale parameters is to run a KMC simulation, just as is done for the evolution of the fine scale dynamics during the EFPI process, but with the coarse variables constrained in some way to equal the intended values. However, applying general constraints within a Monte Carlo simulation (either MMC or KMC) is not trivial, and requires that the PDF of the constrained system be known. The form of this PDF is assumed to be that derived using the maximum entropy principle, Equation (4.16).

Before forming our PDF, we note that in addition to the ensemble means and standard deviations of the leading Fourier sine modes, we expect our PDF to be dependent on the total energy of the system. To see that this must be true, consider what results from an unconstrained energy: higher modes of the system, which are independent of the constrained modes, can grow without bound, which is clearly not physical. To avoid this, we will assume that our PDF depends on the energy in the same way as the unconstrained PDF used to evolve the fine scale dynamics: through a term proportional to $\exp(-\beta_0 E_I)$.

Combining this with the form of the maximum entropy PDF given in Equation (4.16), and including constrained functions B_1, B_3, σ_{B_1}, and σ_{B_3}, gives:

$$P_I = Z^{-1} \exp\left[-\beta_0 E_I - \beta_1 B_{1I} - \beta_3 B_{3I} - \alpha_1 \left(B_{1I} - \bar{B_1}\right)^2 - \alpha_3 \left(B_{3I} - \bar{B_3}\right)^2\right] \tag{4.33a}$$

$$Z = \sum_I \exp\left[-\beta_0 E_I - \beta_1 B_{1I} - \beta_3 B_{3I} - \alpha_1 \left(B_{1I} - \bar{B_1}\right)^2 - \alpha_3 \left(B_{3I} - \bar{B_3}\right)^2\right] \tag{4.33b}$$

A few remarks on this PDF:

- The notation B_{1I} and B_{3I} is used to denote the Fourier sine coefficients computed at state I, and E_I is the energy of state I, i.e. $E_I = E(\mathbf{h}_I)$. Note that we have switched in this section to using uppercase I to denote the state, rather than lowercase i which may refer to the site; i.e. h_i and E_i are the height and energy of site i, while $\mathbf{h}_I$ and E_I are the vector of all heights and the total energy for a given state I.
- Rather than constraining the standard deviations directly, we constrain the square of each standard deviation, i.e. the variance. This is done because the variance can be written as an ensemble average (it is the ensemble average of squared fluctuations), and the maximum entropy principle is used to constrain ensemble averages.
- The Lagrange multipliers enforcing the constraints (the values of λ_k in Equation 4.16) are now β_1, β_3, α_1, and α_3, enforcing respectively the goal values B_1, B_3, σ_{B_1}, and σ_{B_3}.
- Comparing with Equation (4.16), we recognize the energy term as a constraint on the ensemble average of the total system energy, with Lagrange multiplier β_0. However, we do not prescribe a given value for the system energy, just as we do not prescribe an average energy for the KMC simulations used to simulate the fine scale dynamics. Instead, we prescribe the Lagrange multiplier to be $\beta_0 = 1/k_B T$, just as for the fine scale evolution.

- The terms weighted by α_1 and α_3 are the squared differences with respect to $\bar{B}_1$ and $\bar{B}_3$, which denote the given, "goal" values of the ensemble averages of B_1 and B_3. Only for the correct choice of Lagrange multipliers are these goal values equal to the actual averages $\langle B_1 \rangle$ and $\langle B_3 \rangle$ that would be computed directly from the PDF. For that choice of Lagrange multipliers, the quantities constrained by α_1 and α_3 are equal to the variances of B_1 and B_3 according to the standard definition.

4.4.3 Solution for the Lagrange Multipliers

As in Section 4.3, once the form of the PDF has been defined as in Equation (4.33), the problem remains to solve for the Lagrange multipliers. It would be straightforward to compute or approximate these multipliers analytically, but for one complication: the energy of the system, E_I, is not independent of the Fourier coefficients.

To demonstrate the ramifications of this, and to make some analytical headway, we start by making the approximation that the site heights can take any real value, rather than just integers. In this case the heights can be written as a sum of modes whose coefficients can also take any real value:

$$h_i = \sum_{k=1}^{n_\phi} C_k \phi_k(x_i, y_i) \tag{4.34}$$

where $\phi_k(x, y)$ are the set of modes and C_k are the coefficients. For the case of our 3D solid-on-solid model, these modes can be taken as the set of Fourier modes, including modes that vary in the y direction. We assume that the total mass of the system is conserved, so that the sum of all site heights is constant. Because of this constraint, the number of degrees of freedom in the system, and therefore the number of modes, is one less than the number of sites in the array: $n_\phi = L_x L_y - 1$. We will take the first two modes to be the two odd-numbered Fourier sine modes, so that $C_1 = B_1$ and $C_2 = B_3$.

Since the coefficients can take any value, the partition function Z given in (4.33b) is no longer a sum over discrete states, but an integral over all possible values of all of the modal coefficients, including B_1 and B_3:

$$\begin{aligned} Z = \int_{-\infty}^{+\infty} \dots \int_{-\infty}^{+\infty} \exp\big[& -\beta_0 E(B_1, B_3, C_3, \dots, C_{n_\phi}) - \beta_1 B_1 - \beta_3 B_3 \\ & - \alpha_1 \left(B_1 - \bar{B}_1\right)^2 - \alpha_3 \left(B_3 - \bar{B}_3\right)^2 \big] dB_1 dB_3 dC_3 dC_4 \dots dC_{n_\phi} \end{aligned} \tag{4.35}$$

In this expression the dependence of the energy on the modal coefficients has been made explicit. This dependency makes it impossible to factor this integral any further without making some approximations.

As a first step in approximating the partition function, we will decompose the energy into two components, the first capturing the dependence on B_1 and B_3, the second the dependence on the other C_k coefficients:

$$E(B_1, B_3, C_3, \dots, C_{n_\phi}) = E_0(B_1, B_3) + E'(B_1, B_3, C_3, \dots, C_{n_\phi}) \tag{4.36}$$

where $E_0(B_1, B_3)$ is defined as the minimum energy over all possible values of the C_k coefficients. That is,

$$E_0(B_1, B_3) \equiv E(B_1, B_3, C_3^{min}, \ldots, C_{n_\phi}^{min}) \tag{4.37}$$

where the coefficients C_k^{min} are the values that give the minimum energy. The second part of the energy decomposition, E', is then by definition positive and equal to zero when $C_k = C_k^{min}$ for all $k \geq 3$.

We can further approximate $E_0(B_1, B_3)$ by noting that if α_1 and α_3 are positive and of large enough value in Equation (4.35), the corresponding terms in the exponential will drive the integrand to zero except where $(B_1 - \bar{B_1})^2$ and $(B_3 - \bar{B_3})^2$ are small; in other words, states have low probabilities except in the vicinity of $B_1 = \bar{B_1}$ and $B_3 = \bar{B_3}$. This justifies a Taylor expansion of E_0 about these values, in which we keep only the first order terms:

$$E_0(B_1, B_3) \approx E_0(\bar{B_1}, \bar{B_3}) + \frac{\partial E_0}{\partial B_1}(B_1 - \bar{B_1}) + \frac{\partial E_0}{\partial B_3}(B_3 - \bar{B_3}) \tag{4.38}$$

The second part of the energy, E', can be expanded about the minimum coefficient values, C_k^{min}. In a typical Taylor expansion about a minimum function value of zero, the leading order terms would be of second order. However, a simple Taylor expansion is not appropriate, because the energy may in fact be a discontinuous function of some of the C_k values at the minimum, particularly if the energy depends on absolute values, as does the SOS model energy. To account for this, we will separate the coefficients into two sets: $k \in K_1$, for which the dependence of E' on C_k is continuous, and $k \in K_2$, for which the dependence of E' on C_k is discontinuous at C_k^{min}. We can write an expansion of E' as:

$$E'(B_1, B_3, C_3, \cdots, C_{n_\phi}) = \sum_{k \in K_1} \frac{1}{2} \frac{\partial^2 E}{\partial C_k^2}(B_1, B_3)(C_k - C_k^{min})^2 + \sum_{k \in K_2} \left| \frac{\partial E}{\partial C_k}(B_1, B_3) \right| \left| C_k - C_k^{min} \right| \tag{4.39}$$

It has been assumed that the dependence on C_k for $k \in K_2$ is symmetric, so that to leading order it can be written in terms of the absolute value, and the absolute value of the partial derivative is equal on both sides of the discontinuity. However, we will see in the ensuing analysis that the exact form of this expansion is not crucial, since this term will not contribute to the approximation for the Lagrange multipliers. It is important to note, however, that the partial derivatives in (4.39) may themselves depend on B_1 and B_3.

This decomposition and expansion for the energy can be substituted into the expression

for the partition function, Equation (4.35). After some rearrangement, we obtain:

$$Z \approx \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \exp\Big[-\left(\beta_0\frac{\partial E_0}{\partial B_1}+\beta_1\right)\left(B_1-\bar{B}_1\right)-\alpha_1\left(B_1-\bar{B}_1\right)^2\Big] \exp\Big[-\left(\beta_0\frac{\partial E_0}{\partial B_3}+\beta_3\right)\left(B_3-\bar{B}_3\right)-\alpha_3\left(B_3-\bar{B}_3\right)^2\Big] \left[\prod_{k\in K_1}\left(\int_{-\infty}^{+\infty}\exp\big[-\frac{1}{2}\beta_0\frac{\partial^2 E}{\partial C_k^2}(C_k-C_k^{min})^2\big]dC_k\right)\right] \left[\prod_{k\in K_2}\left(\int_{-\infty}^{+\infty}\exp\big[-\beta_0\big|\frac{\partial E}{\partial C_k}(B_1,B_3)\big|\big|C_k-C_k^{min}\big|\big]dC_k\right)\right] dB_3 dB_1 \quad (4.40)$$

Here we have taken advantage of the fact that multiplying the partition function by a constant does not change the statistics of the system, as long as the same factor is applied to the exponential term in the PDF. To see that this is true, note that ensemble averages are related to derivatives of $\ln Z$ (Equation 4.17); multiplying Z by a constant increases $\ln Z$ by an additive constant, which does not contribute to the derivative. This has allowed us to remove a factor of $\exp\left[-\beta_0 E_0(\bar{B}_1,\bar{B}_3)\right]$ in (4.40), and also to write everything in terms of $(B_1-\bar{B}_1)$ and $(B_3-\bar{B}_3)$. Strictly speaking we should use a different symbol for the partition function in (4.40) to reflect this multiplicative constant, but throughout this section we will assume that such constants can be absorbed into Z an avoid introducing more notation.

The integrals in C_k can be evaluated analytically, giving:

$$Z \approx \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \exp\Big[-\left(\beta_0\frac{\partial E_0}{\partial B_1}+\beta_1\right)\left(B_1-\bar{B}_1\right)-\alpha_1\left(B_1-\bar{B}_1\right)^2\Big] \exp\Big[-\left(\beta_0\frac{\partial E_0}{\partial B_3}+\beta_3\right)\left(B_3-\bar{B}_3\right)-\alpha_3\left(B_3-\bar{B}_3\right)^2\Big] \left(\left(\frac{2\pi}{\beta_0}\right)^{M_1/2}\prod_{k\in K_1}\left(\frac{\partial^2 E}{\partial C_k^2}(B_1,B_3)\right)^{-1/2}\right) \left(\left(\frac{2}{\beta_0}\right)^{M_2}\prod_{k\in K_2}\Big|\frac{\partial E}{\partial C_k}\Big|^{-1}\right) dB_3 dB_1 \quad (4.41)$$

where M_1 and M_2 are the number of modes k in the sets K_1 and K_2, respectively.

To complete the approximation, we must estimate the dependence of the partial derivatives of E on C_k on the variables B_1 and B_3. Although it is almost certainly not universally true, we have found both through numerical experiments and analysis of simple cases that typically, the second derivative with respect to C_k behaves as $1/B_1$ for $k \in K_1$, and that the first derivative does not depend strongly on either B_1 or B_3 for $k \in K_2$. Using these estimates, and again noting that multiplying Z by a constant factor does not affect statistics,

allows us to write:

$$Z \approx \int_{-\infty}^{+\infty} B_1^{M_1/2} \exp\Big[-\left(\beta_0 \frac{\partial E_0}{\partial B_1} + \beta_1\right)\left(B_1 - \bar{B}_1\right) - \alpha_1 \left(B_1 - \bar{B}_1\right)^2\Big] dB_1 \\ \times \int_{-\infty}^{+\infty} \exp\Big[-\left(\beta_0 \frac{\partial E_0}{\partial B_3} + \beta_3\right)\left(B_3 - \bar{B}_3\right) - \alpha_3 \left(B_3 - \bar{B}_3\right)^2\Big] dB_3 \quad (4.42)$$

Finally, by expanding the term $B_1^{M_1/2}$ about $\bar{B}_1$, we can incorporate this term into the exponential:

$$\begin{aligned} B_1^{M_1/2} &= \exp\Big[\frac{M_1}{2} \ln B_1\Big] \\ &\approx \exp\Big[\frac{M_1}{2}\left(\ln \bar{B}_1 + \frac{1}{\bar{B}_1}(B_1 - \bar{B}_1) - \frac{1}{2\bar{B}_1{}^2}(B_1 - \bar{B}_1)^2\right)\Big] \\ &\approx \bar{B}_1{}^{M_1/2} \exp\big[\frac{M_1}{2\bar{B}_1}(B_1 - \bar{B}_1) - \frac{M_1}{4\bar{B}_1{}^2}(B_1 - \bar{B}_1)^2\big] \end{aligned} \quad (4.43)$$

Substituting into the expression for Z and again removing multiplicative constants leaves:

$$Z \approx \int_{-\infty}^{+\infty} \exp\Big[-\left(\beta_0 \frac{\partial E_0}{\partial B_1} - \frac{M_1}{2\bar{B}_1} + \beta_1\right)\left(B_1 - \bar{B}_1\right) - \left(\alpha_1 + \frac{M_1}{4\bar{B}_1{}^2}\right)\left(B_1 - \bar{B}_1\right)^2\Big] dB_1 \\ \times \int_{-\infty}^{+\infty} \exp\Big[-\left(\beta_0 \frac{\partial E_0}{\partial B_3} + \beta_3\right)\left(B_3 - \bar{B}_3\right) - \alpha_3 \left(B_3 - \bar{B}_3\right)^2\Big] dB_3 \quad (4.44)$$

This partition function is finally in a form that can be used to compute analytically the mean and standard deviation of the coarse scale variables B_1 and B_3, and thus solve for the Lagrange multipliers. In fact, it is now easy to see by inspection that the following choice of variables leads to the desired statistics:

$$\beta_1 = -\beta_0 \frac{\partial E_0}{\partial B_1} + \frac{M_1}{2\bar{B}_1} \quad (4.45a)$$

$$\beta_3 = -\beta_0 \frac{\partial E_0}{\partial B_3} \quad (4.45b)$$

$$\alpha_1 = \frac{1}{2\sigma_{B_1}^2} - \frac{M_1}{4\bar{B}_1{}^2} \quad (4.45c)$$

$$\alpha_3 = \frac{1}{2\sigma_{B_3}^2} \quad (4.45d)$$

With these values for the Lagrange multipliers, the partition function takes the simple form:

$$Z = \int_{-\infty}^{+\infty} \exp\Big[-\frac{1}{2\sigma_{B_1}^2}\left(B_1 - \bar{B}_1\right)^2\Big] dB_1 \times \int_{-\infty}^{+\infty} \exp\Big[-\frac{1}{2\sigma_{B_3}^2}\left(B_3 - \bar{B}_3\right)^2\Big] dB_3 \quad (4.46)$$

This partition function has the standard Gaussian form in each variable and gives means of $\bar{B}_1$ and $\bar{B}_3$, and standard deviations of σ_{A_1} and σ_{A_3}.

The derivation of these values required many approximations, and in general the results given in (4.45) are only meant as an estimate, a starting point for an iterative solution for a more precise set of multipliers. This iterative approach will be outlined in Section 4.4.5. In particular, the value of M_1, the number of modes for which the dependence of energy on the modal coefficient C_k is continuous at the minimum value, is difficult to compute for real systems and must itself be estimated. However, for some simple systems, this approximate partition function can be computed in more detail and shown to give good results. An example of such a simple system is shown in the next section.

For clarity, it should be pointed out that the approximate partition function derived here is *only* meant to help derive approximate values for the Lagrange multipliers, and to understand how the interdependency of the modal coefficients and the system energy affects the form of the PDF. Once initial estimates have been made for the Lagrange multipliers, all stochastic simulations, whether to solve iteratively for more precise multiplier values or to generate realizations in the lift operator, use the full form of the PDF given by the maximum entropy principle in Equation (4.33).

4.4.4 A Simple Example of the Maximum Entropy Partition Function

Consider a periodic 2D system with length L_x (equivalent to a 1D system with thickness $L_y = 1$). To simplify analysis, we will assume that although the system is described by L_x discrete sites, these sites can have any height (i.e. h_i is not limited to integers, as in the real SOS model). Furthermore, shape changes are such that only 3 modes of the system are nonzero: the first and third Fourier sine modes parameterized by B_1 and B_3, along with the first cosine mode, which is parameterized by A_1 (see Equation 4.31). By the notation introduced in the previous section,, we have $n_\phi = 3$, and $C_3 = A_1$. The height profile is given by:

$$h(x_i) = B_1 \sin\left(\frac{2\pi x_i}{L_x}\right) + B_3 \sin\left(\frac{6\pi x_i}{L_x}\right) + C_3 \cos\left(\frac{2\pi x_i}{L_x}\right). \tag{4.47}$$

The system energy, normally written as a sum of site energies (Equation 4.27), can be approximated as a continuous integral (we choose bond energy $J = 1$):

$$E = \int_0^{L_x} \left|\frac{\partial h}{\partial x}\right| dx \tag{4.48}$$

Alternatively, the system energy can be thought of as summing the absolute values of each peak-to-valley distance in the function; thus the energy of a pure sine wave is equal to 4 times the amplitude.

We will choose goal values of $\bar{B_1} = 4.0$, $\bar{B_3} = 0.0$, $\sigma_{B_1} = 0.5$, and $\sigma_{B_3} = 0.1$. Now we need to compute the dependence of the energy on the coefficients. Even for this simple system, the absolute value in the energy function makes it difficult to write this dependency

for arbitrary values of the coefficients. However, for small values of B_3, we have:

$$E_0(B_1, B_3) = 4B_1 - 4B_3 \tag{4.49}$$

To see that this is true, consider that when $B_3 = 0$, E_0 is 4 times the amplitude of the sine wave ($E_0 = 4B_1$). Perturbing this profile by a sine wave with wavelength $L_x/3$ decreases the amplitude by an amount B_3 at both extremes, giving a new energy of $E_0 = 4(B_1 - B_3)$.

The perturbation to the energy due to the C_3 coefficient can be approximated by again considering only the vicinity of $B_3 = 0$. In that case, the height profile is:

$$h(x_i)\Big|_{B_3=0} = B_1 \sin\left(\frac{2\pi x_i}{L_x}\right) + C_3 \cos\left(\frac{2\pi x_i}{L_x}\right). \tag{4.50}$$

which we recognize as a phase-shifted sine wave with amplitude $(B_1^2 + C_3^2)^{1/2}$. Thus the energy is $4(B_1^2 + C_3^2)^{1/2}$, which is continuous in C_3, the minimum value is $C_3^{min} = 0$, and the second partial derivative of energy with respect to C_3 at the minimum is:

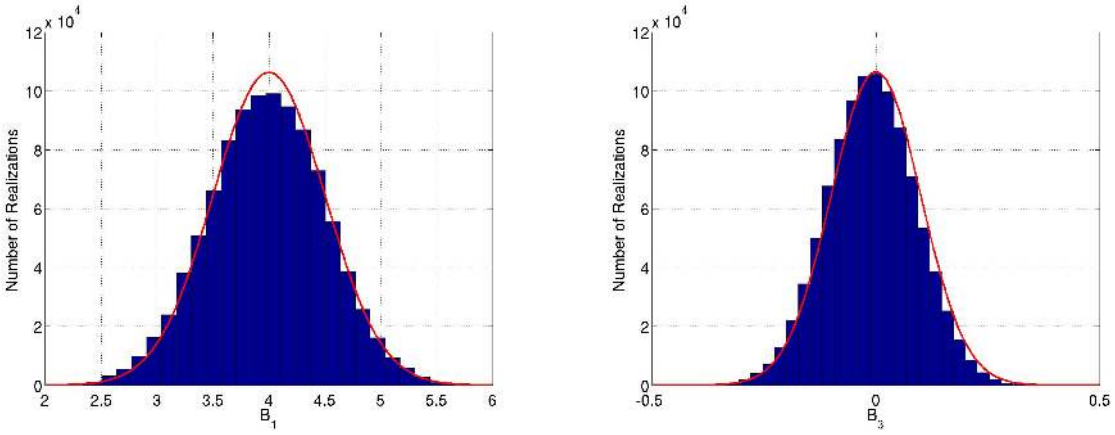
$$\frac{\partial^2 E}{\partial C_3^2} = \frac{4}{B_1} \tag{4.51}$$

Note that this has the $1/B_1$ dependency noted in the previous section for typical modes with $k \in K_1$.

Summarizing, we have $\frac{\partial E_0}{\partial B_1} = 4$, $\frac{\partial E_0}{\partial B_3} = -4$, and $M_1 = 1$. Using Equation (4.45) gives $\beta_1 = 0.125 - 4\beta_0$, $\beta_2 = 4\beta_0$, $\alpha_1 \approx 1.984$, and $\alpha_3 = 50$ (recall that $\beta_0 = 1/k_BT$ is a given parameter of the simulation). We have written a small Metropolis Monte Carlo simulation code in Matlab to use the PDF given in (4.33) to generate a number of realizations by choosing trial moves that perturb one of the 3 active modes of the system, at random, by a small increment of random size. The resulting set of realizations is consistent with the PDF in (4.33), and from these we can compute the mean and standard deviation of each variable of interest.

Figure 4.1 shows the results of this simulation, using 10^6 MMC steps, system size $L_x = 20$ and a temperature of $k_BT = 1.0$. These two plots compare histograms of the values of B_1 and B_3 with the Gaussian distribution generated using the desired mean and standard deviation for each variable, as in Equation (4.46); the agreement is very good. The mean values computed in the simulation are $\langle B_1 \rangle = 3.979$, $\langle B_3 \rangle = -0.010$, $\sigma_{B_1} = 0.518$, and $\sigma_{B_3} = 0.099$. These are all very close to the specified goal values, demonstrating that we have achieved our goal of finding values of the Lagrange multipliers that give the desired statistics, at least for this simple system.

It is interesting to explore what happens when the Lagrange multipliers are chosen more arbitrarily. Suppose that we naïvely choose values of the Lagrange multiplier that, when substituted into (4.33), more closely mimic a Gaussian form of the PDF. That is, choose $\beta_1 = \beta_3 = 0$, $\alpha_1 = 1/(2\sigma_{B_1}^2)$, $\alpha_3 = 1/(2\sigma_{B_3}^2)$; when substituted into (4.33), these values give a partition function is similar to that in (4.46), but include a dependence on energy (we emphasize again that it is the full PDF in (4.33) that is used in the MMC simulation,



(a) (b)

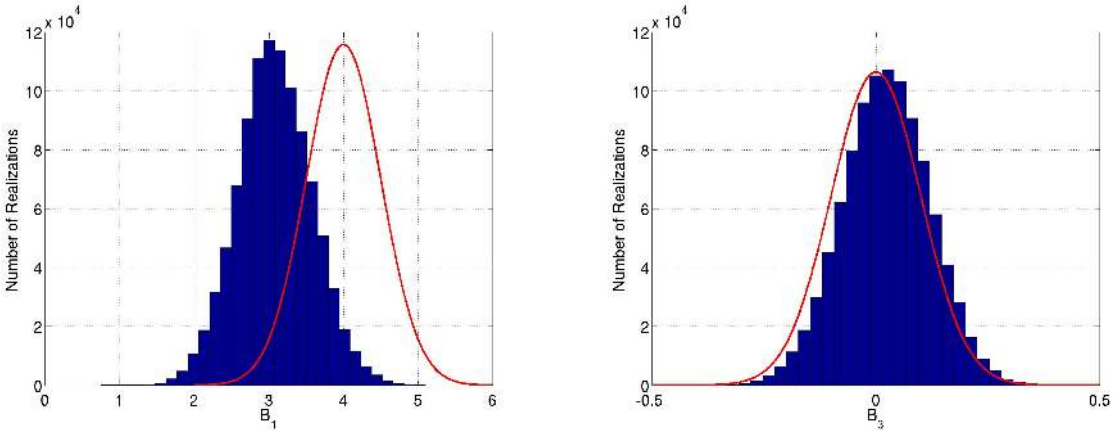
Figure 4.1. MMC results for a simplified 2D system, using Lagrange multipliers computed from Equation (4.45). Blue bars are histograms of states generated in the simulation, showing distribution of variables (a) B_1 and (b) B_3. Red lines show Gaussian distribution computed using goal values of mean and standard deviation for each variable, $P(B) = \exp\left[(B - \bar{B})^2/(2\sigma^2)\right]$.

including the energy term). Results of the simulation using these values are shown in Figure (4.2). Now the comparison with the desired PDF is poor, especially for B_1. Computed statistics are $\langle B_1 \rangle = 3.036$, $\langle B_3 \rangle = 0.026$, $\sigma_{B_1} = 0.498$, and $\sigma_{B_3} = 0.098$. Although standard deviations are well-constrained, the mean values are far from the goals. We can conclude from this that the modifications given in (4.45) to the "naïve" values of the Lagrange multipliers are necessary to achieve the desired statistics; these modifications come from the dependence of the PDF on the system energy.

4.4.5 Iterative Solution for the Lagrange Multipliers

With an initial guess in hand for the Lagrange multipliers, based on an analytical approximation, we now seek to solve more precisely for the values of the multipliers that give the desired statistics in a real simulation of a complex system. As in the general approach outlined in Section 4.3, we will use an iterative method to achieve this goal. However, there are several differences between the approach we will take here and the methods that have been applied previously in the literature, as described in Section 4.3:

- We will use a KMC algorithm, rather than the MMC algorithm applied in the literature (as in Algorithm 1), to compute realizations; in this way we hope to alleviate the "critical slowing down problem" noted elsewhere [26].



(a) (b)

Figure 4.2. MMC results for a simplified 2D system, using Lagrange multipliers $\beta_1 = \beta_3 = 0$, $\alpha_1 = 1/(2\sigma^2_{B_1})$, $\alpha_3 = 1/(2\sigma^2_{B_3})$. Blue bars are histograms of states generated in the simulation, showing distribution of variables (a) B_1 and (b) B_3. Red lines show Gaussian distribution computed using goal values of mean and standard deviation for each variable, $P(B) = \exp\left[(B - \bar{B})^2/(2\sigma^2)\right]$.

- We are including a constraint on the standard deviation of each of the coarse scale variables of interest, which drives the probability distribution function to zero if the coarse variables at a given state are too far away from the desired mean values. This leads to tighter control of the coarse variables and better convergence behavior, especially when the desired standard deviation is small (i.e. the Lagrange multipliers α are large).

- We have a reasonable initial guess for the multipliers (Equation 4.45) from which to start our iterative algorithm.

- Instead of a nonlinear conjugate-gradient method, we use a very simple quasi-Newton method in which the Hessian matrix of the system is assumed to be diagonal, i.e. each statistical quantity (mean or standard deviation) that we are trying to constrain is assumed to be a function of a single Lagrange multiplier.

We will make this last point clearer by presenting our iterative algorithm in more detail. Returning to the notation of Section 4.3.2, we want to solve for the vector of Lagrange multipliers $\boldsymbol{\lambda}$ that minimizes the scalar quantity $G(\boldsymbol{\lambda})$ defined in Equation (4.24). Denote the derivatives of G with respect to $\boldsymbol{\lambda}$ as $\mathbf{g}$; the minimization problem is equivalent to setting $\mathbf{g} = \mathbf{0}$:

$$g_k = \frac{\partial G}{\partial \lambda_k} = F_k - \langle f_k \rangle = 0 \tag{4.52}$$

where, as previously, F_k are the desired values of the constrained ensemble averages of function f_k. To make the connection with the case in which we constrain the means and variances of the Fourier coefficients B_1 and B_3, we have:

$$\boldsymbol{\lambda} = \{\beta_1, \beta_3, \alpha_1, \alpha_3\} \tag{4.53a}$$
$$\mathbf{F} = \{\bar{B_1}, \bar{B_3}, \sigma^2_{B_1}, \sigma^2_{B_3}\} \tag{4.53b}$$
$$\mathbf{f} = \{B_1, B_3, (B_1 - \langle B_1 \rangle)^2, (B_3 - \langle B_3 \rangle)^2\} \tag{4.53c}$$

Note that although our PDF (Equation 4.33) includes a Lagrange multiplier β_0 that multiplies the system energy, we do not include either β_0 or the energy in our vectors $\boldsymbol{\lambda}$, $\mathbf{F}$ or $\mathbf{f}$, because we are not constraining the energy. In all cases we use the same value of β_0 used for the fine scale simulations, i.e. $\beta_0 = 1/k_B T$.

The algorithm we use is based on Newton's method, which we write for a given iteration as:

$$\mathbf{g}(\boldsymbol{\lambda}^{n+1}) = \mathbf{0} \approx \mathbf{g}(\boldsymbol{\lambda}^n) + \mathbf{B}\Delta\boldsymbol{\lambda}^n \tag{4.54}$$

where superscripts denote an iteration number, and $\mathbf{B}$ is an approximation to the Hessian matrix:

$$B_{jk} \approx \frac{\partial^2 G}{\partial\lambda_j \partial\lambda_k} = \frac{\partial g_k}{\partial\lambda_j} \tag{4.55}$$

Note that in a true Newton method $\mathbf{B}$ would be exactly equal to the Hessian, while for a quasi-Newton method it is only an approximation. Solving for $\Delta\boldsymbol{\lambda}^n$ and using it to update $\boldsymbol{\lambda}$ gives:

$$\Delta\boldsymbol{\lambda}^n = -\mathbf{B}^{-1}\mathbf{g}(\boldsymbol{\lambda}^n) \tag{4.56a}$$
$$\boldsymbol{\lambda}^{n+1} = \boldsymbol{\lambda}^n + \Delta\boldsymbol{\lambda}^n = \boldsymbol{\lambda}^n - \mathbf{B}^{-1}\mathbf{g}(\boldsymbol{\lambda}^n) \tag{4.56b}$$

This completes an iteration, but it remains to specify the approximation used for $\mathbf{B}$. A typical strategy in a quasi-Newton method is to use a secant method, in which approximate Hessian is computed from the n and $n-1$ states as:

$$\mathbf{g}(\boldsymbol{\lambda}^n) - \mathbf{g}(\boldsymbol{\lambda}^{n-1}) = \mathbf{B}(\boldsymbol{\lambda}^n - \boldsymbol{\lambda}^{n-1}) = \mathbf{B}\Delta\boldsymbol{\lambda}^{n-1} \tag{4.57}$$

If $\boldsymbol{\lambda}$ is a vector of dimension greater than one, $\mathbf{B}$ is underdetermined, and various quasi-Newton methods take different strategies to solve for $\mathbf{B}$. In our case, we will make the simple approximation that $\mathbf{B}$ is diagonal, $B_{jk} = B_{\overline{jj}}\delta_{jk}$, leading to:

$$B_{\overline{jj}} = \frac{g_j(\boldsymbol{\lambda}^n) - g_j(\boldsymbol{\lambda}^{n-1})}{\lambda_j^n - \lambda_j^{n-1}} \tag{4.58}$$

and

$$\lambda_j^{n+1} = \lambda_j^n - \Big(\frac{\lambda_j^n - \lambda_j^{n-1}}{g_j^n - g_j^{n-1}}\Big) g_j^n \tag{4.59}$$

This expression uses the shorthand $g_j^n \equiv g_j(\boldsymbol{\lambda}^n)$. This defines a single iteration of the algorithm.

At each iteration, the function $\mathbf{g}$ must be computed by calculating the ensemble averages of the functions f_k for a given set of Lagrange multipliers. As mentioned above, this is done using a KMC simulation that uses the PDF given in Equation(4.33) in place of the usual canonical ensemble PDF. That is, we write the PDF as:

$$P_I = Z^{-1} \exp\left[-H(\mathbf{h}_I, \boldsymbol{\lambda})\right] \tag{4.60}$$

where now

$$H(\mathbf{h}_I, \boldsymbol{\lambda}) \equiv \beta_0 E(\mathbf{h}_I) + \sum_k \lambda_k f_k(\mathbf{h}_I) \tag{4.61}$$

Note that the PDF we use for these KMC simulations is similar to that used in the MMC simulations of Section 4.3.1, but with a more explicit dependence on system energy and the multiplier β_0. The expected rate for an event transforming the system from state 1 to state 2 is now

$$P_{1\to 2} = \begin{cases} \frac{1}{8} & \text{if } \Delta H_{12} \leq 0, \\ \frac{1}{8} e^{-\Delta H_{12}} & \text{if } \Delta H_{12} > 0 \end{cases} \tag{4.62}$$

Possible system events are again chosen to be atom hops between neighboring sites. Ensemble averages of the functions of interest are computed by collecting data points at regular intervals in the computed time (and *not* at regular intervals in the number of steps, which for KMC would give incorrect statistics). The mean of these data points gives the ensemble average over the ensemble defined by the given PDF. These ensemble averages are the values $\langle f_k \rangle$,

Iterations can be continued either until the norm of the vector $\mathbf{g}$ is less than some tolerance, or a maximum number of iterations is reached. The full algorithm is given in Algorithm 2.

4.5 Results

In this section we present some preliminary results of the use of our newly developed lift operator on SOS model systems. All of our tests are similar to the lift/restrict operator tests described in Section 3.4.1, in which the operators are applied repeatedly as the system evolves in time, but without a projective integration step. Ideally, restricting to a set of coarse variables followed by lifting back to a set of re-initialized fine scale realizations should have minimal effect on either the values or the time evolution of the coarse variables.

In these tests, we are interested in several things:

- The effect of system size on our ability to effectively apply lifting operators
- The effect of leaving Fourier coefficient B_3 unconstrained
- The effect of choosing not to update β_3, the Lagrange multiplier associated with B_3, during the iterative solution

Algorithm 2 Iterative quasi-Newton algorithm to solve for the Lagrange multipliers

Given a set of functions of interest $\mathbf{f}$ (e.g. the coarse-scale Fourier coefficients of a surface profile) and goal values $\mathbf{F}$ for the ensemble averages of these functions (the means and variances).
Choose a maximum number of iterations n_{max}, and a residual tolerance ϵ
Set the initial value of the set of Lagrange multipliers $\boldsymbol{\lambda}^0$ to be used in the maximum entropy PDF using Equation 4.45
Call a KMC simulation to compute $\langle \mathbf{f} \rangle(\boldsymbol{\lambda}^0)$, and set $\mathbf{g}^0 = \mathbf{F} - \langle \mathbf{f} \rangle(\boldsymbol{\lambda}^0)$.
Compute the norm of the residuals, $r = \|\mathbf{g}^0\|$
if $r \leq \epsilon$ **then**
    Stop.
end if
Choose a small initial value $\Delta\boldsymbol{\lambda}^0$, and set $\boldsymbol{\lambda}^1 = \boldsymbol{\lambda}^0 + \Delta\boldsymbol{\lambda}^0$.
Call a KMC simulation to compute $\langle \mathbf{f} \rangle(\boldsymbol{\lambda}^1)$, and set $\mathbf{g}^1 = \mathbf{F} - \langle \mathbf{f} \rangle(\boldsymbol{\lambda}^1)$.
for $n = 1$ to n_{max} **do**
    Compute the norm of the residuals, $r = \|\mathbf{g}^n\|$
    if $r \leq \epsilon$ **then**
        Stop.
    end if
    Update λ via $\lambda_j^{n+1} = \lambda_j^n - g_j^n \left(\lambda_j^n - \lambda_j^{n-1}\right) / \left(g_j^n - g_j^{n-1}\right)$.
    Call a KMC simulation to compute $\langle \mathbf{f} \rangle(\boldsymbol{\lambda}^{n+1})$, and set $\mathbf{g}^{n+1} = \mathbf{F} - \langle \mathbf{f} \rangle(\boldsymbol{\lambda}^0)$.
end for

4.5.1 Test 1: $L_x = 40$, $L_y = 1$

We have first applied our results to a 2D system of length $L_x = 40$, similar to the systems explored in Chapter 3. For these 2D simulations, our initial condition is a sine wave with wavelength L_x and an initial amplitude of $b_0 = 5.5$ (see Equation 4.29). The system is run for a total time of $t_{tot} = 2000$, and the restrict and lift operator is applied at intervals of $t_{R/L} = 400$. The system temperature is $k_B T = 0.8$, and number of realizations in the ensemble N_R is $10,000$. These parameters, along with those used for all other example simulations in this chapter, are given in Table 4.1.

In previous 2D simulations shown in Chapter 3, the evolving surface was well-described by the first Fourier mode, parameterized by B_1. We are interested in exploring how important the variable B_3 is for reproduction of the coarse dynamics. We therefore test three cases: 1) Set $\beta_3 = \alpha_3 = 0$ to leave B_3 completely unconstrained, 2) Set β_3 and α_3 according to the approximate initial conditions computed from (4.45), but do not update them in the iterative scheme to improve efficiency, and 3) Update β_3 and α_3 at each iteration. In this last case, we have chosen to constrain the standard deviation of B_3 to a small value of 0.05 at each lift step. We have found that a small value of σ_{B_3} improves the robustness of the iterative method because B_3 is constrained to a smaller neighborhood around the goal mean value, $\bar{B_3}$. This adjustment of σ_{B_3} has not been found to affect the dynamics of the system.

Table 4.1. Parameters used in lift operator example simulations

	Section	L_x	L_y	b_0	t_{tot}	$t_{L/R}$	k_BT	N_R
Test 1	4.5.1	40	1	5.5	2×10^3	4×10^2	0.8	1×10^4
Test 2	4.5.2	40	4	5.5	1.2×10^4	3×10^3	0.8	2×10^3
Test 3	4.5.3	80	32	5.5	6×10^4	3×10^4	0.8	96
Test 4	4.5.4	120	32	15.5	2×10^5	1×10^5	0.8	96

Results are plotted in Figures 4.3 and 4.4. All cases are plotted together against a continuous simulation over the total time, with no restrict or lift operators applied; this continuous simulation can be treated as the "correct" solution. Figures 4.3(a) and 4.3(b) show average coefficients $\langle B_1 \rangle$ and $\langle B_3 \rangle$, respectively, and Figure 4.3(c) shows the standard deviation in B_1. All three cases show good agreement with the continuous curve, with the exception of the B_3 for the case with $\beta_3 = \alpha_3 = 0$, for which B_3 is unconstrained. Interestingly, for this unconstrained case B_3 jumps to a value close to the long-time value eventually reached by the other cases. The conclusion is that this is a long-time equilibrium value that has not yet been reached by the continuous solution for B_3 by time $t = 400$. This is a validation of the maximum entropy method; when Lagrange multipliers are used properly the value of B_3 can be constrained to a non-equilibrium value, but otherwise it finds a value that is at equilibrium with respect to the rest of the system. Also interesting is that even in case 2 (fixed values of β_3 and α_3), B_3 is constrained closely to the goal value. This indicates that, at least for this simple system, our approximated values in Equation (4.45) are good estimates for the ideal values of those multipliers. The average height profiles just after application of the first lift operation at time $t = 400$ are plotted in Figure 4.3(d), and compared with the profile just before the application of the restrict/lift. Again, agreement is good.

However, a plot of the evolution of the system energy (Figure 4.4(a)) shows some disagreement with the continuous solution for all cases, beginning at the first lift operation. A possible reason for this discrepancy is shown in Figure 4.4(b), which compares the evolution of the average total amplitude of the leading order Fourier mode, $\langle (A_1^2 + B_1^2)^{1/2} \rangle$, where A_1 is the leading Fourier cosine mode. Comparing with Figure 4.3(a), this jump is clearly coming from a non-zero value of A_1 in the lifted cases. We conclude that this leading cosine mode is finding a non-zero equilibrium value in the maximum entropy lifting operation, but that because this is a long-wavelength, slow mode, it has not yet had time to reach this equilibrium value in the continuous solution. Interestingly, this difference is not manifest in the average height profiles (Figure 4.3(d)). One way to think about this is that each realization generated by the lift operation has a random phase shift that averages to zero, giving the correct average height profile but an incorrect average amplitude. To correctly deal with this, we could either include A_1 in the list of coarse variables to be constrained (since it is in fact a slow variable that parameterizes the system), or to remove this random phase shift from each state that get generated during our lift operation. We have not yet tested either of these strategies.

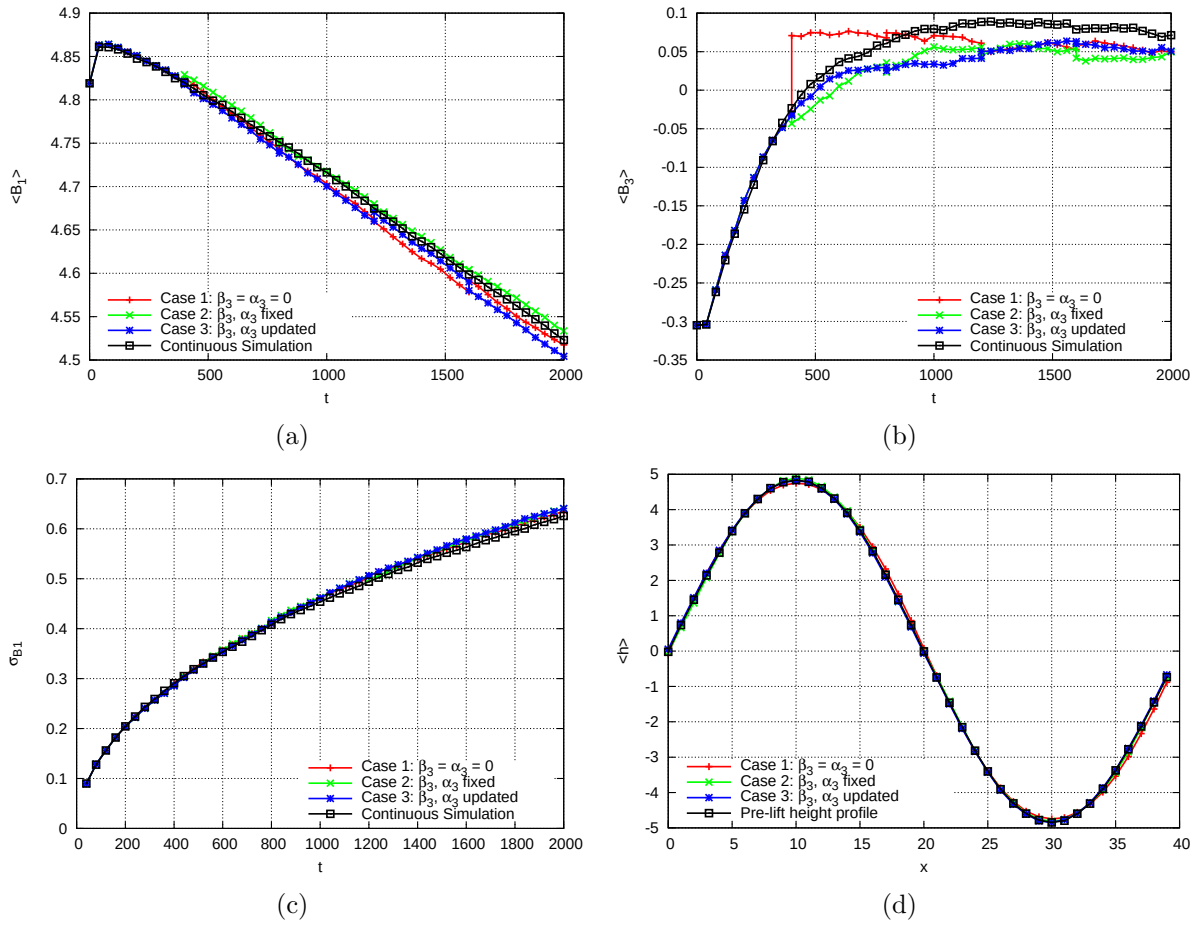
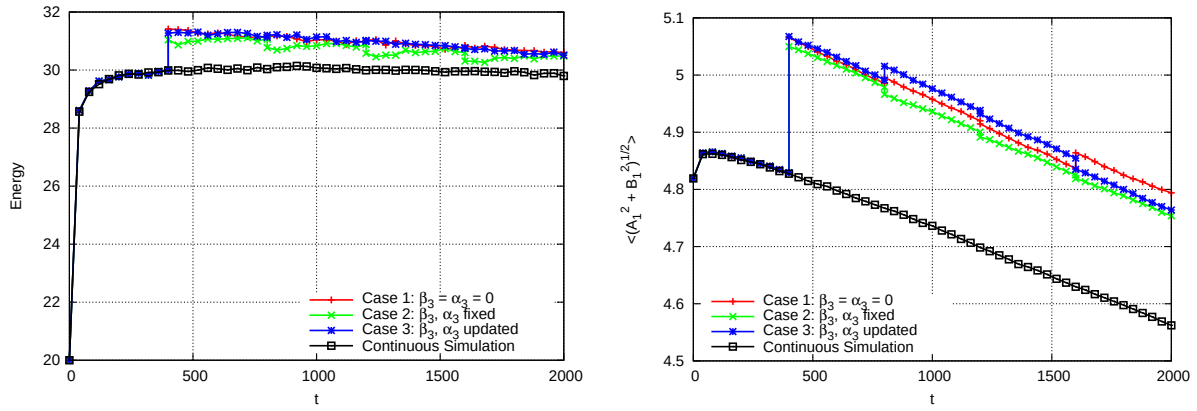


Figure 4.3. Comparison of effects on system evolution of 1D, $L_x = 40$ system with lift operator applied at fixed intervals. (a) $\langle B_1 \rangle$ vs. time, (b) $\langle B_3 \rangle$ vs. time, (c) σ_{B_1} vs. time, (d) Average height profile after the first lift operation at $t = 400$, compared with profile before lifting.



(a) (b)

Figure 4.4. Comparison of effects on system evolution of 2D, $L_x = 40$ system (Test 1) with lift operator applied at fixed intervals. (a) Ensemble-averaged energy vs. time, (b) $\langle (A_1^2 + B_1^2)^{1/2} \rangle$ vs. time. See text for discussion.

4.5.2 Test 2: $L_x = 40$, $L_y = 4$

We now demonstrate that, unlike in the 2D case, the inclusion of the coarse parameter B_3 is important in capturing the dynamics correctly in 3D. We test on a fairly small system of size 40×4; other parameters are given in Table 4.1. Two cases are run for this system: 1) Set $\beta_3 = \alpha_3 = 0$, and 2) Set β_3 and α_3 according to the approximate initial conditions, but do not update during the iterative solve.

The time histories of $\langle B_1 \rangle$, $\langle B_3 \rangle$, and σ_{B_1}, as well as the average height profiles after the first lift operation, are shown in Figure 4.5. In this case, the dynamics is not well preserved unless B_3 is constrained through the use of the Lagrange multipliers. The reason is clear from looking at the height profiles (Figure 4.5(d)); when B_3 is unconstrained, that mode takes on a value that leads to a "flattening" of the profile peaks. It appears that this flattening leads to a lower total energy of the system. The continuous system has not yet reached this lower-energy profile, and so the real system dynamics can only be reproduced by constraining B_3 to evolve as it does in the continuous system. Note that the profiles in Figure 4.5(d) are averaged both over the ensemble and over the y direction, to give profiles that depends on x.

4.5.3 Test 3: $L_x = 80$, $L_y = 32$

As the system size increases, we find that the initial estimate for the Lagrange multipliers is less accurate, and it is necessary to update β_3 during the iterative solution. This is demonstrated on a system of size 80×32; other parameters are given in Table 4.1. Note that

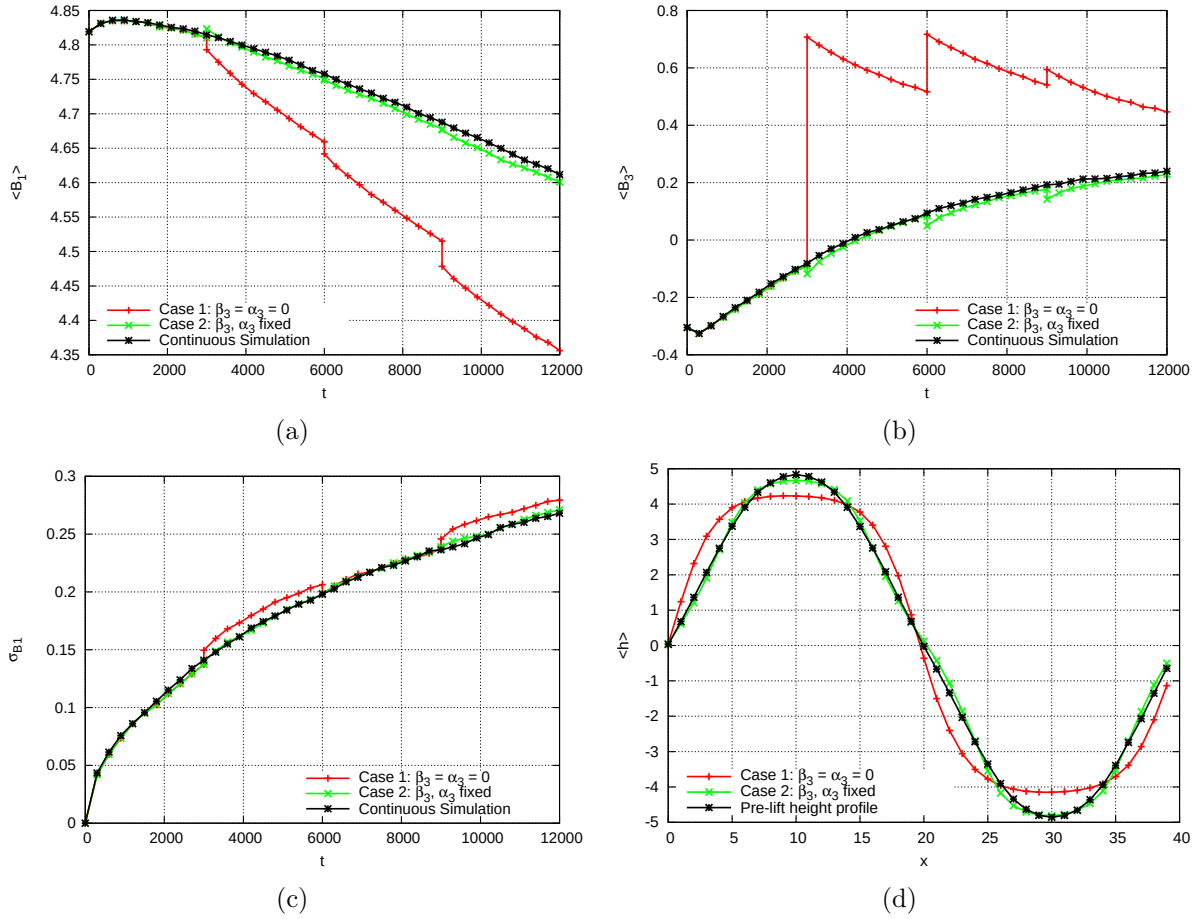


Figure 4.5. Comparison of effects on system evolution of 3D, $L_x = 40$, $L_y = 4$ system (Test 2) with lift operator applied at fixed intervals. (a) $\langle B_1 \rangle$ vs. time, (b) $\langle B_3 \rangle$ vs. time, (c) σ_{B_1} vs. time, (d) Height profile, averaged along the y direction and over the ensemble, after the first lift operation at $t = 3000$, compared with profile before lifting.

as the system size increases, longer simulation times are necessary to capture evolution of the system. Two cases are compared: 1) β_3 fixed at its initial condition, and 2) β_3 updated during the iterative solution.

The time histories of $\langle B_1 \rangle$, $\langle B_3 \rangle$, and σ_{B_1}, as well as the average height profiles after the first lift operation, are shown in Figures 4.6. The value of B_3 is not well-constrained unless β_3 is included in the iterative update. For this system, the error in the reproduction of B_1 is actually larger when β_3 is update than when it is fixed; however, we attribute this to fluctuations in the system that lead to an imperfect solution in our iterative algorithm; the numerical difference between the goal and realized average values of B_1 is only around 0.02. We may be able to reduce this error by using a tighter tolerance on the iterative solve or averaging over more realizations, both of which will increase computational cost.

The effects of the incorrect Fourier coefficient B_3 are also seen in the profile shape (Figure 4.6(d)); the updated-β_3 more closely matches the profile before lifting.

4.5.4 Test 4: $L_x = 120$, $L_y = 32$

The largest system we have tested is size 120×32, with other parameters given in Table 4.1. For this system we ran with a larger initial amplitude of $b_0 = 15.5$. As in Test 3, two cases are run: 1) β_3 fixed at its initial condition, and 2) β_3 updated during the iterative solution. Results are shown in Figure 4.7; as in Test 3, we conclude that the value of β_3 must be updated to properly constrain the B_3 Fourier coefficient.

Figure 4.8 shows the full ensemble-averaged surface before and after lifting for Case 2, with updated β_3. There is a slight flattening of the peaks of the profile visible after lifting, but otherwise the profile is well-predicted (in agreement with Figure 4.7(d), which shows the y-averaged profiles).

4.6 Maximum Entropy Method: Conclusions

We have developed and implemented an algorithm based on the maximum entropy principle that initializes a set of realizations based on the known means and standard deviations of a small number of coarse scale variables. We have also demonstrated that by using this lift operator, we are able to reproduce the time rate of change of the coarse variables in our surface diffusion model, as long as care is taken to solve accurately for the Lagrange multipliers that constrain our system.

However, there are some disadvantages to this method that make it difficult to apply to real systems in its current form. The first, and most important, is computational expense. Each iteration of the algorithm to solve for Lagrange multipliers requires a KMC simulation that that must be run for long enough to obtain good statistics. Since our goal is to use this

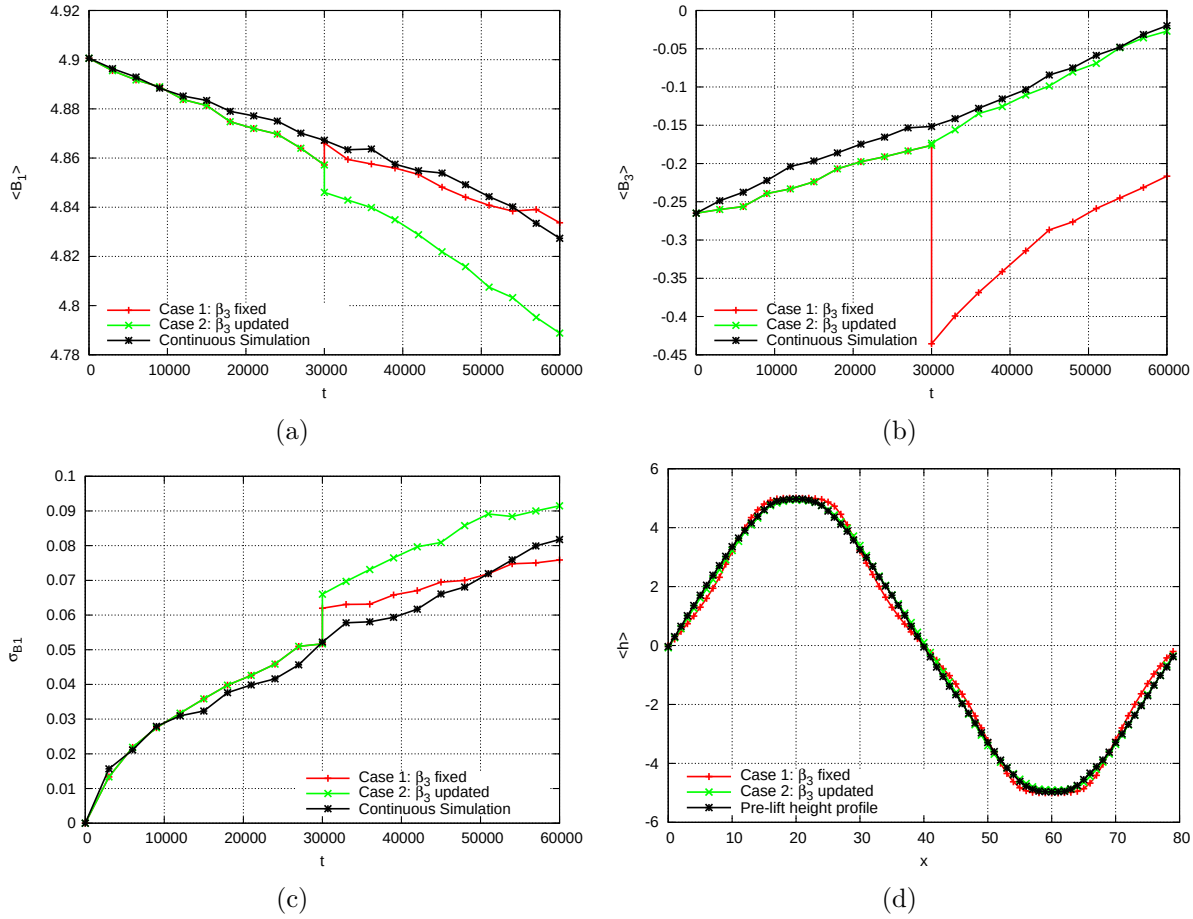


Figure 4.6. Comparison of effects on system evolution of 3D, $L_x = 80$, $L_y = 32$ system (Test 3) with lift operator applied at fixed intervals. (a) $\langle B_1 \rangle$ vs. time, (b) $\langle B_3 \rangle$ vs. time, (c) σ_{B_1} vs. time, (d) Height profile, averaged along the y direction and over the ensemble, after the lift operation at $t = 30,000$, compared with profile before lifting.

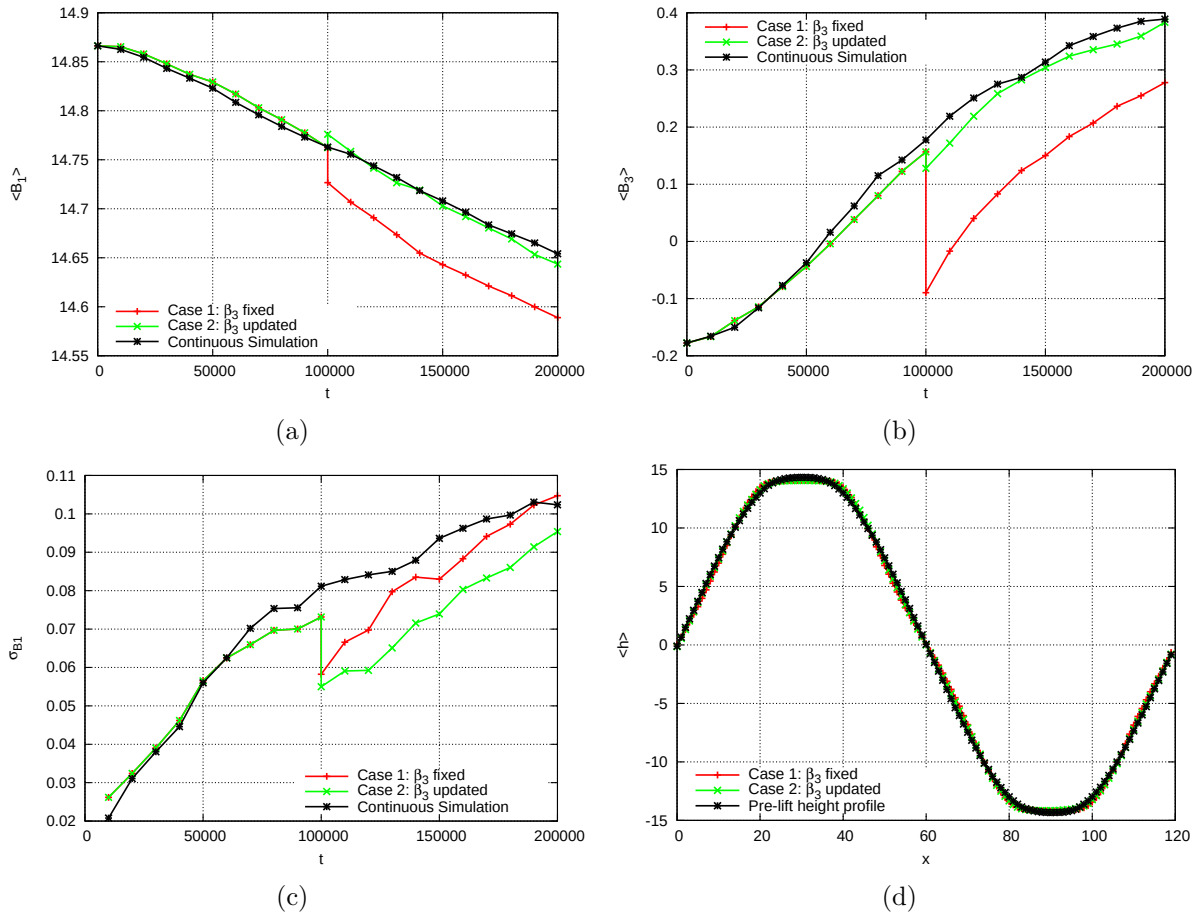
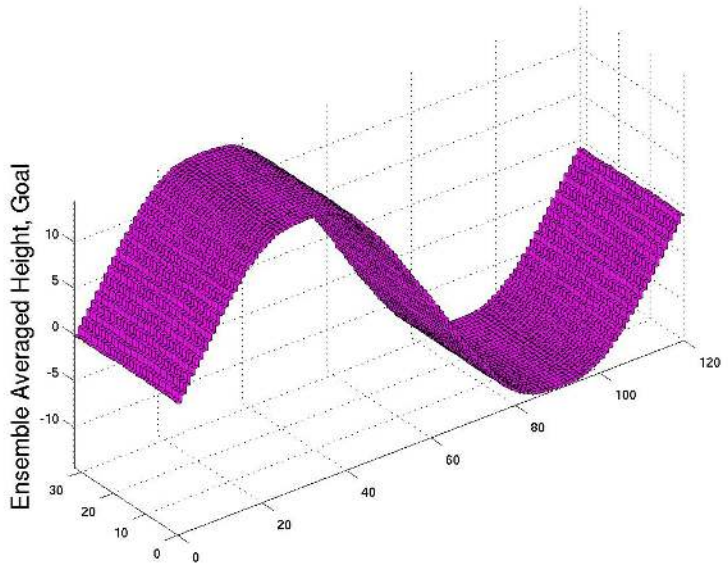
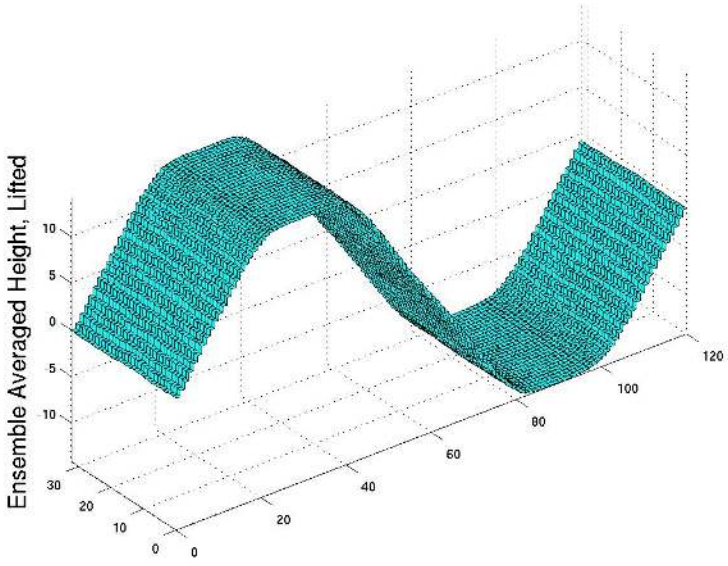


Figure 4.7. Comparison of effects on system evolution of 3D, $L_x = 120$, $L_y = 32$ system (Test 4)with lift operator applied at fixed intervals. (a) $\langle B_1 \rangle$ vs. time, (b) $\langle B_3 \rangle$ vs. time, (c) σ_{B_1} vs. time, (d) Height profile, averaged along the y direction and over the ensemble, after the lift operation at $t = 100,000$, compared with profile before lifting.



(a)



(b)

Figure 4.8. Comparison of ensemble-averaged heights for the $L_x = 120$, $L_y = 32$ system (Test 4). (a) Before lift operation. (b) After lift operation, using updated β_3 (Case 2).

method together with a projective integration scheme in time to achieve speed-up, it may still be possible to obtain a substantial gain with this method, as long as a large projection step is possible. However, any gain in efficiency of our lift operation will lead to improvements in the efficiency of the entire EFPI scheme.

One route to achieving this may be to derive better approximations for the initial values of the Lagrange multipliers. We have shown in our calculations that for simple systems, the approximate values given in Equation (4.45) give well-constrained values of the multipliers, but that these estimates are less effective for larger systems in 3D. With more analysis, it may be possible to achieve better estimates even in 3D. In particular, the correct value of the parameter M_1 in these estimates is unclear for 3D systems; we have typically set it equal to the number of pure x-direction modes that are unconstrained, but more in-depth analysis may lead to better results.

Another way to improve the method is to implement better quasi-Newton algorithms that will speed up convergence compared with the simple approach presented in Section 4.4.5. For example, Abramov [1] has shown improvements in efficiency for a similar maximum entropy algorithm using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) technique. Such a technique may also help reduce the effects of iteration-to-iteration fluctuations inherent in our method, which otherwise can only be reduced by increasing the number of realizations in the ensemble, which adds to computational expense. This approach should be studied more thoroughly for our system.

4.7 Choice of Coarse Scale Variables

One topic that is important for EFPI, but that has not been explored in depth in our project, is the choice of coarse scale variables. In our surface diffusion systems we have assumed that the long-wavelength Fourier modes give a good parameterization of the slow manifold in the system, but there is no reason to think *a priori* that this choice is optimal. We have made progress through our use of the maximum entropy method lift operator, removing the need for non-intuitive, higher-order statistics among our coarse variables. But the question remains: For a general system, how can we find the smallest set of variables that fully parameterizes the slow dynamics of the system?

This question is closely related to the very active research topics of dimensionality reduction and reduced order modeling. In this last section of the chapter, we summarize two systematic approach for finding low-dimensional projections of high-dimensional spaces that have been developed in recent years: diffusion maps, and proper orthogonal decomposition.

4.7.1 Diffusion maps

Tenebaum *et al.* [59] developed Isomap, which is a diffusion mapping technique capable of finding a non-linear, low-dimensional embedding of a high-dimensional data set. Isomap uses time snapshots of the full system and maps them to points in hyperspace. The algorithm then finds an optimal lower-dimensional space using a distance measure between these points in hyperspace. The cut-off for the size of the dimension is up to the user, but this method does order the dimensions by the magnitudes of their eigenvalues. There is also Matlab code available online for this procedure. [1]

A data mining approach can be used to identify the optimal set of coarse variables for a problem of our type. Such a method has been used by Sonday *et al.* for the Ising model problem of a moving surface within a solid material, where the dimensionality of the problem is large and there are no intrinsic macroscopic variables [54]. These authors use a kinetic Monte Carlo simulator for the fine scale dynamics and a Fokker-Planck equation governing the position of the surface at the coarse scale. Their lift operator is a combination of simulated annealing alternating with KMC healing steps to get to the target fine scale following time projection. Diffusion maps are a dimensionality reduction tool that can help find the best slow manifold for a high-dimensional problem without relying on intuition. The method helps determine which features are important to the overall system dynamics by finding hidden variables that may not be slaved to macroscopic observables during a simulation. It does this by looking at distances of snapshots of the simulation. The crux of the method is providing the diffusion map with the best distance measure. This part is not at all obvious or systematic for a general class of problems, but finding the dimension and best coarse variables once a distance measure is defined is straightforward. Sonday *et al.* [54] also discuss how to compare the correlation between the coarse variables found by diffusion maps to physical observables that may be a more attractive option for the choice of coarse scale variables.

4.7.2 Model reduction techniques: POD

Proper Orthogonal Decomposition (POD), also known as Principle Component Analysis (PCA), is a way of rewriting a space-time solution by separating and ordering the time dependent modes of that solution [5],

$$u(\mathbf{x}, t) = \sum_k \phi_k(\mathbf{x}) a_k(t), \tag{4.63}$$

where $\phi_k(\mathbf{x})$ are orthonormal functions (or vectors in discretized space) and $a_k(t)$ are the time dependent modes with corresponding magnitudes λ_k. In order to solve for the optimal set of ϕ_k's, we can define a matrix over a given time interval T

$$\mathbf{B}_{ij} = \frac{1}{T} \int_{t_o}^{t_o+T} u(\mathbf{x}_i, t) u(\mathbf{x}_j, t) dt$$

[1] http://isomap.stanford.edu/

and determine the basis functions via the eigenvalue problem,

$$\mathbf{B}_{ij}\phi_k = \lambda_k \phi_k.$$

However, it is clear from this setup that the basis functions are not unique, non-constant in time, and perhaps also not the optimal basis functions for times outside of the interval $[t_o, t_o + T]$. One needs to choose this time interval wisely in order for the basis to be optimal. The time dependent modes can then be extracted using the face that the ϕ_k's are orthogonal,

$$a_k(t) = \int u(\mathbf{x}, t)\phi_k(\mathbf{x})d\mathbf{x}.$$

The basic idea behind POD is that one can approximate a full solution by truncating the infinite series in Equation 4.63 to obtain the best lower-dimensional description.

The Equation-free method has been used in practice with POD for incompressible Navier Stokes equations by Sirisup et al [53]. The authors report good speed-up of their method on a test problem of periodic flow around a cylinder.

Chapter 5

Design of Lift and Restrict Operators for Equation-Free Projective Integration of Vacancy Diffusion in Solid Materials

5.1 Motivation

Our objective is to formulate a coarse scale representation of vacancy concentration in a solid material to be used in the framework of equation-free projective integration (EFPI) as developed by Kevrekidis and colleagues. Use of EFPI requires the design and implementation of consistent lift (transfer information from the coarse scale to the fine scale) and restrict (transfer information from the fine scale to the coarse scale) operators. The definition of consistency is such that, starting from a coarse scale description of vacancy concentration, successive application of the lift and restrict operators does not change the coarse scale description. While not new, this objective of defining specialized operators that exchange fine and coarse representations of material systems is complex. For example, work has been done by Torquato and colleagues [67, 50, 60, 20, 21] to mathematically represent complex microstructures with a limited number of correlation and other statistical functions. This body of research shows accuracy of fine scale reconstructions requires multiple coarse scale metrics in order to establish even a minimal amount of consistency between the two representations.

In mathematical terms, c_v represents the coarse scale variable of vacancy concentration or, more accurately, a vacancy fraction equal to the number of vacancies (N_v) divided by the number of atoms that would be present for a bulk lattice (*i.e.* the number of sites, N_s). This is equivalent to mole fraction for an atomic system. At the atomic scale (our fine scale), c_v is determined through simple counting of unoccupied atomic lattice sites and is defined at a system level.

At the coarse scale, c_v is a localized quantity and is considered to be a function of spatial position, $c_\mathrm{v}(\mathbf{x}, t)$. Practically, c_v is evaluated at the nodal positions on a finite element (FE) mesh that overlays the system of interest, *i.e.* $c_I = c_\mathrm{v}(\mathbf{x}_I, t)$. An example of such an atomic system with an overlaying FE mesh is shown in Figure 5.1. Values of c_v at an arbitrary

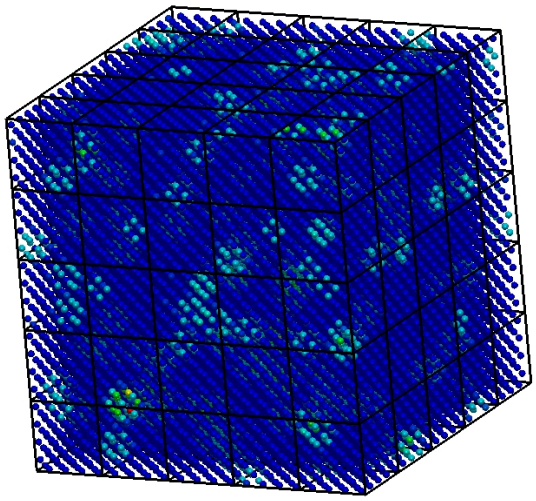


Figure 5.1. Atomic system of Cu atoms with vacancies and an overlaying FE mesh of 125 nodes. Atoms are colored by potential energy, with vacancies corresponding with clusters of atoms at higher-than-bulk potential energy values.

spatial location between nodes is determined through the use of interpolation functions, as will be defined shortly.

The set of nodal values $\mathbf{C} = \{c_I\}$ is defined over the coarse scale domain, $\mathbf{C} \in \mathbb{R}^{N_n}$. In general, the size of $\mathbf{C}$ is far less than the size of a set of variables defined at the fine scale, in our case the atomic scale, *i.e.* $\mathbf{c} \in \mathbb{R}^{N_a}$ where N_a is the number of atoms and $N_n \ll N_a$. The coarse scale variable $\mathbf{C}$ is assumed to be a function of the fine scale variable $\mathbf{c}$ through a restriction operator $\mathcal{M}$:

$$\mathbf{C} = \mathcal{M}\mathbf{c} \tag{5.1}$$

Similarly, a lifting operator μ can be introduced that maps the coarse scale description $\mathbf{C}$ to a consistent fine scale description $\mathbf{c}$, *i.e.*

$$\mathbf{c} = \mu\mathbf{C}. \tag{5.2}$$

Mathematically, consistency equates to the condition $\mathcal{M}\mu = I$, *i.e.* lifting followed by restricting has no net effect on the coarse scale description. The lifting operator is non-unique and involves the creation of information.

5.2 Restrict Operator

5.2.1 Estimation using Hardy's approach

We seek to define a mathematical operator that quantifies the vacancy fraction c_{v} localized to the spatial position $\mathbf{x}$. One obvious route is to define a localized number-density of atoms, normalize it relative to a number-density expected for a bulk lattice and then subtract this fraction from unity,

$$c_{\mathrm{v}}(\mathbf{x},t) = 1 - \frac{\rho_n(\mathbf{x},t)}{\rho_{n0}}. \tag{5.3}$$

Defining $\rho_n(\mathbf{x},t)$ can be accomplished using the method by Hardy [13, 65], who defined a localization function ψ that spatially averages atomic properties to evaluate the equivalent continuum property at a given position and time. Hence,

$$\rho_n(\mathbf{x},t) \equiv \sum_{\alpha=1}^{N_a} \psi(\mathbf{x}^\alpha - \mathbf{x}) \tag{5.4}$$

and

$$c_{\mathrm{v}}(\mathbf{x},t) = 1 - V_a \sum_{\alpha=1}^{N_a} \psi(\mathbf{x}^\alpha - \mathbf{x}), \tag{5.5}$$

where the quantity $V_a = \rho_{n0}^{-1}$ is the volume per atom attributed to a bulk lattice. In general, the localization function $\psi(\mathbf{r})$ is non-negative, is of compact-support, has the dimensions of inverse volume, and is normalized such that $\int_\Omega \psi(\mathbf{r}) d^3r = 1$, where Ω is the domain of interest containing the collection of atoms.

Equation (5.5) is not general in the sense that deformation of the material would have the effect of changing the number-density, thereby leading to erroneous estimates of vacancy fraction even in the case of a lattice with zero porosity. This equation can be modified to account for deformation as such:

$$c_{\mathrm{v}}(\mathbf{x},t) = 1 - V_a J_{\mathbf{F}} \sum_{\alpha=1}^{N_a} \psi(\mathbf{x}^\alpha - \mathbf{x}), \tag{5.6}$$

where $J_{\mathbf{F}} = \det(\mathbf{F})$ and $\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}}$ is the deformation gradient evaluated at position $\mathbf{x}$. $\mathbf{F}$ itself can be estimated using Hardy's approach to calculate the displacement field:

$$\mathbf{u}(\mathbf{x},t) = \frac{\sum_{\alpha=1}^{N_a} m^\alpha \mathbf{u}^\alpha(t) \psi(\mathbf{x}^\alpha - \mathbf{x})}{\sum_{\alpha=1}^{N_a} m^\alpha \psi(\mathbf{x}^\alpha - \mathbf{x})}, \tag{5.7}$$

where m^α is the mass of atom α, and $\mathbf{u}^\alpha$ is the displacement of the atom from its reference (initial) position. This definition for displacement field was proposed by Zimmerman *et al.* [69]. As discussed by these authors in [69], Equation (5.7) can be differentiated with respect to $\mathbf{x}$ to define a displacement gradient,

$$\nabla_{\mathbf{x}} \mathbf{u}(\mathbf{x},t) = \frac{\sum_{\alpha=1}^{N_a} m^\alpha \left(\mathbf{u}^\alpha(t) - \mathbf{u}(\mathbf{x},t)\right) \otimes \nabla_{\mathbf{x}} \psi(\mathbf{x}^\alpha - \mathbf{x})}{\sum_{\alpha=1}^{N_a} m^\alpha \psi(\mathbf{x}^\alpha - \mathbf{x})}, \tag{5.8}$$

where $\nabla_{\mathbf{x}} \equiv \frac{\partial}{\partial \mathbf{x}}$, and $\mathbf{F}$ is found from the usual relation $\mathbf{F} = (\mathbf{1} - \nabla_{\mathbf{x}}\mathbf{u})^{-1}$. Alternatively, the displacement gradient can be defined using nodal values of displacement and the gradients of the (dimensionless) interpolation functions associated with each node,

$$\nabla_{\mathbf{x}}\mathbf{u}(\mathbf{x}, t) = \sum_{I=1}^{N_n} \varphi_I(\mathbf{x})\mathbf{u}_I, \tag{5.9}$$

where $\mathbf{u}_I = \mathbf{u}(\mathbf{x}_I, t)$ is the displacement of node I, φ_I is a dimensionless interpolation (shape) function associated with node I and $I \in N_n$. Both equations (5.8) and (5.9) use Hardy's definition of velocity ($\mathbf{v}(\mathbf{x}, t)$) with the approximation of ignoring time dependency of the localization function itself to connect $\mathbf{v}$ with $\mathbf{u}$. This approximation is exact if reference positions ($\mathbf{X}$) instead of current positions ($\mathbf{x}$) are used in the formulation. In such a case, $\mathbf{F} = \mathbf{1} + \nabla_{\mathbf{x}}\mathbf{u}$.

5.2.2 Estimation using atomic quadrature

In the previous section, we used Hardy's approach to define a restriction operator for estimating vacancy concentration at the coarse scale. Restricting ourselves to cases in which only vacancy diffusion occurs and the material is otherwise undeformed, we choose to evaluate $c_{\mathrm{v}}(\mathbf{x})$ at the coordinates of the nodal positions for an overlaying FE mesh. Using the shorthand expression $\psi_I(\mathbf{x}^\alpha) = \psi(\mathbf{x}^\alpha - \mathbf{x}_I)$:

$$c_I(t) = 1 - V_a \sum_{\alpha=1}^{N_a} \psi_I(\mathbf{x}^\alpha), \tag{5.10}$$

It is important to note that localization functions ψ_I are distinct from conventional nodal interpolation functions φ_I in the following ways:

1. They are different in terms of their units: ψ_I has units of inverse volume whereas φ_I is dimensionless.

2. In general, they do not possess the same functional form nor have the same region of influence.

3. FE mesh interpolation functions typically obey the partition of unity rule, $\sum_{I=1}^{N_n} \varphi_I(\mathbf{x}) = 1$, while an equivalent relation for ψ_I does not necessarily exist.

These differences introduce a bit of inconsistency with regard to the use of an FE mesh and nodal values of the field $c_{\mathrm{v}}(\mathbf{x})$. Typically, spatially varying fields are expressed as summations of products of nodal values and their interpolation functions, *i.e.* $c_{\mathrm{v}}(x) = \sum_{I=1}^{N_n} c_I\varphi_I(\mathbf{x})$. However, substitution of equation (5.10) into this expression does not result in the originating expression given in (5.5).

In addition to these issues, we also note that in equation (5.10) we have used an assumption by defining V_a, the volume per atom attributed to a bulk lattice. Certainly, V_a has a well-understood meaning for a physical crystal. However, use of a FE mesh to spatially partition a body presents an interesting problem in how to relate the subdivisions of elemental or nodal volume to subdivisions of atomic volume. It must be the case that for a bulk crystal where all atoms are occupying lattice sites, both the sum of nodal volumes and the sum of atomic volumes must add up to the same value of the total volume of the crystal. However, it is not clear that subsets of atomic volumes for all sites located within a single element will sum to that element's volume. When the discreteness of the mesh is uneven with respect to the discreteness of the underlying atomic lattice in one or more spatial dimensions, it is conceivable that different elements of the same size will contain different numbers of atomic sites. In such a scenario, a uniform value of volume per atomic site will not be sufficient to satisfy any site-element volume correspondence.

An alternative approach that can resolve these issues is to replace our localization functions in equation (5.10) with the specific form $\psi_I = V_I^{-1}\varphi_I$, where V_I is the volume associated with node I, and replace V_a with a weight specific to the atomic site occupied (w_α). Hence our new expression for vacancy concentration at a given node is:

$$c_I(t) = 1 - \frac{1}{V_I}\sum_{\alpha=1}^{N_a} \varphi_I(\mathbf{x}^\alpha) w_\alpha. \tag{5.11}$$

V_I will be a constant for a uniform mesh that overlays an atomic system with periodic boundary conditions. For a non-uniform mesh, the value of V_I will vary for each node. For a system where periodic boundary conditions were not used or not applied to the mesh, V_I would differ between internal and boundary nodes. Equation (5.11) essentially defines our restrict operation, $\mathcal{M}$, which maps the set of atomic positions $\{\mathbf{x}^\alpha\}$ to the set of nodal vacancy concentration $\{c_I\}$ at any given instant of time. We emphasize the important distinction that although we use the notation w_α, the weight and the volume it represents is associated with the site occupied by atom α, and not the atom itself.

5.2.3 Defining weights for atomic quadrature

Given a set of nodes N_n and elements N_e that overlap a region of atom sites N_s, we would like to compute the best quadrature weight for each atomic site, $\mathbf{w} = \{w_s\}$, where $s \in N_\mathrm{s}$. By 'best', we mean that we seek to satisfy the criteria:

1. The sum of the weights should equal the known volume of the domain, Ω.

2. Integration of the shape functions associated with each node $I \in N_n$ should yield the known volume associated with the node, V_I.

3. Variability of the weights should be as small as possible.

Here, we again make the distinction between atoms ($\alpha \in N_a$) and atomic sites ($s \in N_s$). In equation (5.11) and its predecessors, we are referring to atoms present within our material system, *i.e.* atomic lattice sites that are occupied. For the case where all sites are occupied ($N_a = N_s$), we expect that $c_I = 0$. In this section we are defining the individual weights ($\mathbf{w}$) that produce this equality exactly. Hence, the set of weights $\mathbf{w}$ is associated with the atomic sites, whether or not those sites are occupied. It is only for occupied sites that these weights are used within the expression given in (5.11).

We begin by noting that if criterion 2 is satisfied, criterion 1 follows by partition of unity. So, we essentially have 2 criteria:

$$V_I \equiv \int_\Omega \varphi_I(\mathbf{x}) d\mathbf{x} = \sum_{s=1}^{N_s} \varphi_I(\mathbf{x}_s) w_s, \tag{5.12}$$

and the error function

$$E = \frac{1}{2} \sum_{s=1}^{N_a} (w_s - \bar{w})^2 \tag{5.13}$$

is minimized with respect to the set of weights $\mathbf{w}$. Here, $\bar{w} = \Omega / N_s$. The relations can be combined through the use of Lagrange multipliers to enforce constraints:

$$E \rightarrow E_{mod} = \frac{1}{2} \sum_{s=1}^{N_s} (w_s - \bar{w})^2 + \sum_{I=1}^{N_n} \lambda_I \left(\sum_{s=1}^{N_s} \varphi_I(\mathbf{x}_s) w_s - V_I \right) \tag{5.14}$$

The combined function E_{mod} is minimized with respect to both the atomic weights and Lagrange multipliers:

$$\frac{\partial E_{mod}}{\partial w_s} = (w_s - \bar{w}) + \sum_{I=1}^{N_n} \lambda_I \varphi_I(\mathbf{x}_s) = 0 \tag{5.15}$$

$$\frac{\partial E_{mod}}{\partial \lambda_I} = \sum_{s=1}^{N_s} \varphi_I(\mathbf{x}_s) w_s - V_I = 0 \tag{5.16}$$

These equations can be re-cast as:

$$w_s + \sum_{I=1}^{N_n} \lambda_I \varphi_I(\mathbf{x}_s) = \bar{w} \tag{5.17}$$

$$\sum_{s=1}^{N_s} \varphi_I(\mathbf{x}_s) w_s = V_I \tag{5.18}$$

These equations can be put into vector/matrix form:

$$\mathbf{w} + \boldsymbol{\varphi} \boldsymbol{\lambda} = \bar{\mathbf{w}} \tag{5.19}$$

$$\boldsymbol{\varphi}^T \mathbf{w} = \mathbf{V} \tag{5.20}$$

where $\mathbf{w} = \{w_s\}$, $\bar{\mathbf{w}} = \bar{w}\mathbf{1}_\mathrm{s}$ ($\mathbf{1}_\mathrm{s}$ is a vector of length N_s where each component equals 1), $\boldsymbol{\varphi} = [\varphi_{sI}]$, $\varphi_{sI} = \varphi_I(\mathbf{x}_s)$, and $\mathbf{V} = \{V_I\}$. Substitution of (5.19) in (5.20) yields:

$$\begin{aligned}\boldsymbol{\varphi}^T\left(\bar{\mathbf{w}} - \boldsymbol{\varphi}\boldsymbol{\lambda}\right) &= \mathbf{V}\\ \boldsymbol{\varphi}^T\bar{\mathbf{w}} - \boldsymbol{\varphi}^T\boldsymbol{\varphi}\boldsymbol{\lambda} &= \mathbf{V}\\ \boldsymbol{\varphi}^T\boldsymbol{\varphi}\boldsymbol{\lambda} &= \boldsymbol{\varphi}^T\bar{\mathbf{w}} - \mathbf{V}\end{aligned}$$

Thus,

$$\boldsymbol{\lambda} = \mathbf{M}^{-1}\left(\boldsymbol{\varphi}^T\bar{\mathbf{w}} - \mathbf{V}\right), \tag{5.21}$$

where $\mathbf{M} \equiv \boldsymbol{\varphi}^T\boldsymbol{\varphi}$, and

$$\mathbf{w} = \bar{\mathbf{w}} - \boldsymbol{\varphi}\mathbf{M}^{-1}\left(\boldsymbol{\varphi}^T\bar{\mathbf{w}} - \mathbf{V}\right). \tag{5.22}$$

This relation can be simplified further provided that $\bar{\mathbf{w}}$ can be expressed with regards to the interpolation functions $\boldsymbol{\varphi}$, *i.e.* $\bar{\mathbf{w}} = \boldsymbol{\varphi}\bar{\mathbf{W}}$ or $\bar{w}_s = \sum_{I=1}^{N_n}\varphi_I(\mathbf{x}_s)\bar{W}_I\ \forall s \in N_\mathrm{s}$. Then,

$$\begin{aligned}\mathbf{w} &= \boldsymbol{\varphi}\bar{\mathbf{W}} - \boldsymbol{\varphi}\mathbf{M}^{-1}\left(\boldsymbol{\varphi}^T\boldsymbol{\varphi}\bar{\mathbf{W}} - \mathbf{V}\right)\\ &= \boldsymbol{\varphi}\bar{\mathbf{W}} - \boldsymbol{\varphi}\mathbf{M}^{-1}\mathbf{M}\bar{\mathbf{W}} + \boldsymbol{\varphi}\mathbf{M}^{-1}\mathbf{V}\\ &= \boldsymbol{\varphi}\bar{\mathbf{W}} - \boldsymbol{\varphi}\bar{\mathbf{W}} + \boldsymbol{\varphi}\mathbf{M}^{-1}\mathbf{V}\end{aligned}$$

Thus,

$$\mathbf{w} = \boldsymbol{\varphi}\mathbf{M}^{-1}\mathbf{V}. \tag{5.23}$$

We note that in the final expression given in (5.23), neither $\bar{w}$ nor $\bar{W}$ appear.

An additional constraint to consider is that the sum of weights for atoms in a specific element should equal the volume of that element, *i.e.*

$$\sum_{s=1,s\in e}^{N_\mathrm{s}} w_s = V_e, \tag{5.24}$$

where $e \in N_e$. This can alternatively be written as

$$\sum_{s=1}^{N_\mathrm{s}} \hat{\varphi}_e(\mathbf{x}_s)w_s = V_e, \tag{5.25}$$

where

$$\hat{\varphi}_e(\mathbf{x}_s) \equiv \begin{cases}1, & \text{if } s \in e\\ 0, & \text{if } s \notin e\end{cases} \tag{5.26}$$

As before, we define a modified error function that incorporates the set of nodal Lagrange multipliers $\boldsymbol{\lambda} = \{\lambda_I\}$ and a set of elemental Lagrange multipliers $\boldsymbol{\theta} = \{\theta_e\}$:

$$E_{mod} = \frac{1}{2}\sum_{s=1}^{N_\mathrm{s}}(w_s - \bar{w})^2 + \sum_{I=1}^{N_n}\lambda_I\left(\sum_{s=1}^{N_\mathrm{s}}\varphi_I(\mathbf{x}_s)w_s - V_I\right) + \sum_{e=1}^{N_e}\theta_e\left(\sum_{s=1}^{N_\mathrm{s}}\hat{\varphi}_e(\mathbf{x}_s)w_s - V_e\right) \tag{5.27}$$

Minimizing E_{mod} with respect to the weights and all multipliers leads to the equations:

$$w_s + \sum_{I=1}^{N_n} \lambda_I \varphi_I(\mathbf{x}_s) + \sum_{e=1}^{N_e} \theta_e \hat{\varphi}_e(\mathbf{x}_s) = \bar{w} \tag{5.28}$$

$$\sum_{s=1}^{N_s} \varphi_I(\mathbf{x}_s) w_s = V_I \tag{5.29}$$

$$\sum_{s=1}^{N_s} \hat{\varphi}_e(\mathbf{x}_s) w_s = V_e \tag{5.30}$$

or, in vector/matrix form:

$$\mathbf{w} + \boldsymbol{\varphi}\boldsymbol{\lambda} + \hat{\boldsymbol{\varphi}}\boldsymbol{\theta} = \bar{\mathbf{w}} \tag{5.31}$$

$$\boldsymbol{\varphi}^T \mathbf{w} = \mathbf{V} \tag{5.32}$$

$$\hat{\boldsymbol{\varphi}}^T \mathbf{w} = \mathbf{V}_e \tag{5.33}$$

The set of matrix equations is most easily solved by defining the quantities $\tilde{\boldsymbol{\lambda}} = \begin{Bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\theta} \end{Bmatrix}$, $\tilde{\boldsymbol{\varphi}} = [\boldsymbol{\varphi}\ \hat{\boldsymbol{\varphi}}]$, and $\tilde{\mathbf{V}} = \begin{Bmatrix} \mathbf{V} \\ \mathbf{V}_e \end{Bmatrix}$. This allows us to put equations (5.31)-(5.33) in the form:

$$\mathbf{w} + \tilde{\boldsymbol{\varphi}}\tilde{\boldsymbol{\lambda}} = \bar{\mathbf{w}} \tag{5.34}$$

$$\tilde{\boldsymbol{\varphi}}^T \mathbf{w} = \tilde{\mathbf{V}} \tag{5.35}$$

Comparing these equations with (5.19) and (5.20), we realize that the solution is:

$$\mathbf{w} = \bar{\mathbf{w}} - \tilde{\boldsymbol{\varphi}} \tilde{\mathbf{M}}^{-1} \left(\tilde{\boldsymbol{\varphi}}^T \bar{\mathbf{w}} - \tilde{\mathbf{V}} \right), \tag{5.36}$$

where $\tilde{\mathbf{M}} \equiv \tilde{\boldsymbol{\varphi}}^T \tilde{\boldsymbol{\varphi}}$. We note that our original matrix $\mathbf{M}$ was of dimensions $N_n \times N_n$, while $\tilde{\mathbf{M}}$ has dimensions $(N_n + N_e) \times (N_n + N_e)$.

5.2.4 Properties of restrict operator

Using the restrict operator defined by equation (5.11) where the atomic site weights are defined by (5.22), (5.23) or (5.36), we consider some of its properties. We first evaluate the

mean value of the vacancy concentration field over the system volume, Ω:

$$\begin{aligned}
\bar{c}(t) &= \frac{1}{\Omega}\int_{\Omega} c(\mathbf{x},t)d\Omega \\
&= \frac{1}{\Omega}\int_{\Omega} \sum_{I=1}^{N_n} c_I\varphi_I(\mathbf{x})d\Omega \\
&= \frac{1}{\Omega}\sum_{I=1}^{N_n} c_I \int_{\Omega} \varphi_I(\mathbf{x})d\Omega \\
&= \frac{1}{\Omega}\sum_{I=1}^{N_n} c_I V_I \\
&= \frac{1}{\Omega}\sum_{I=1}^{N_n} \left(1 - \frac{1}{V_I}\sum_{\alpha=1}^{N_a} \varphi_I(\mathbf{x}^\alpha)w_\alpha\right) V_I \\
&= \frac{1}{\Omega}\sum_{I=1}^{N_n} \left(V_I - \sum_{\alpha=1}^{N_a} \varphi_I(\mathbf{x}^\alpha)w_\alpha\right) \\
&= \frac{1}{\Omega}\left(\Omega - \sum_{I=1}^{N_n}\sum_{\alpha=1}^{N_a} \varphi_I(\mathbf{x}^\alpha)w_\alpha\right) \\
&= 1 - \frac{1}{\Omega}\sum_{I=1}^{N_n}\sum_{\alpha=1}^{N_a} \varphi_I(\mathbf{x}^\alpha)w_\alpha
\end{aligned}$$

We then use the partition of unity rule discussed above ($\sum_{I=1}^{N_n}\varphi_I = 1$) to further simplify the above expression to:

$$\bar{c}(t) = 1 - \frac{1}{\Omega}\sum_{\alpha=1}^{N_a} w_\alpha \tag{5.37}$$

The expression above has a clear meaning. The term $\sum_{\alpha=1}^{N_a} w_\alpha$ is the total volume associated with sites occupied by atoms in the system. The division of this term by the total system volume Ω is the fraction of the system occupied by matter. Vacancy concentration is, of course, the difference of this term from unity. $\sum_{\alpha=1}^{N_a} w_\alpha$ can also be equated to an average volume for occupied sites times the number of occupied sites, *i.e.* $\sum_{\alpha=1}^{N_a} w_\alpha = \bar{w}_a N_a$. Earlier, we expressed the system volume as $\Omega = \bar{w}N_\mathrm{s}$, where $\bar{w}$ is the average volume for all sites. Hence,

$$\bar{c}(t) = 1 - \frac{\bar{w}_a N_a}{\bar{w}N_\mathrm{s}}. \tag{5.38}$$

We recall that in Section 5.2.3 we defined our site weights to possess as little variability as possible. Thus, $\bar{w}_a \approx \bar{w}$, and

$$\bar{c}(t) \approx 1 - \frac{N_a}{N_\mathrm{s}} = \frac{N_\mathrm{v}}{N_\mathrm{s}}, \tag{5.39}$$

where $N_\mathrm{v} = N_\mathrm{s} - N_a$ is the total number of vacancies in the system. This equation confirms that the system average of our nodal measure of vacancy concentration is consistent with the physical notion of vacancy concentration as applied to the entire atomic system.

$\bar{c}$ is the mean value of the coarse-grained, spatially-varying vacancy concentration field. Alternatively, we can calculate an average value of vacancy concentration for all the nodes in the system, $\bar{c}_n$:

$$\begin{aligned}
\bar{c}_n(t) &= \frac{1}{N_n}\sum_{I=1}^{N_n} c_I(t) \\
&= \frac{1}{N_n}\sum_{I=1}^{N_n}\left\{1 - \frac{1}{V_I}\sum_{\alpha=1}^{N_a}\varphi_I(\mathbf{x}^\alpha)w_\alpha\right\} \\
&= \frac{1}{N_n}\left\{N_n - \sum_{I=1}^{N_n}\frac{1}{V_I}\sum_{\alpha=1}^{N_a}\varphi_I(\mathbf{x}^\alpha)w_\alpha\right\} \\
&= \frac{1}{N_n}\left\{N_n - \sum_{\alpha=1}^{N_a} w_\alpha \sum_{I=1}^{N_n}\frac{1}{V_I}\varphi_I(\mathbf{x}^\alpha)\right\} \\
&= 1 - \sum_{\alpha=1}^{N_a} w_\alpha \frac{1}{N_n}\sum_{I=1}^{N_n}\frac{1}{V_I}\varphi_I(\mathbf{x}^\alpha)
\end{aligned}$$

This expression can be simplified further for the case that the amount of volume associated with any given node, V_I, is the same for all nodes, *i.e.* $V_I = V_n = \Omega/N_n$. Given this assumption,

$$\begin{aligned}
\bar{c}_n(t) &= 1 - \sum_{\alpha=1}^{N_a} w_\alpha \frac{1}{N_n V_n}\sum_{I=1}^{N_n}\varphi_I(\mathbf{x}^\alpha) \\
&= 1 - \sum_{\alpha=1}^{N_a} w_\alpha \frac{1}{\Omega}\sum_{I=1}^{N_n}\varphi_I(\mathbf{x}^\alpha) \\
&= 1 - \frac{1}{\Omega}\sum_{\alpha=1}^{N_a} w_\alpha = \bar{c}(t)
\end{aligned}$$

Thus, we see that for an equipartition of volume among nodes, the nodal and spatial means of vacancy concentration are equivalent. In general, the two values would be different from each other.

We can also evaluate the standard deviation of the vacancy concentration field, σ_c:

$$\begin{aligned}
\sigma_c^2 &= \frac{1}{\Omega}\int_\Omega \left(c(\mathbf{x},t)-\bar{c}(t)\right)^2 d\Omega \\
&= \frac{1}{\Omega}\int_\Omega \left(c^2 - 2\bar{c}c + \bar{c}^2\right) d\Omega \\
&= \frac{1}{\Omega}\left(\int_\Omega c^2 d\Omega - 2\bar{c}\int_\Omega c d\Omega + \bar{c}^2\int_\Omega d\Omega\right) \\
&= \frac{1}{\Omega}\left(\int_\Omega c^2 d\Omega - 2\Omega\bar{c}^2 + \Omega\bar{c}^2\right) \\
&= \left(\frac{1}{\Omega}\int_\Omega c^2 d\Omega\right) - \bar{c}^2
\end{aligned}$$

This expression can be somewhat simplified as follows:

$$\begin{aligned}
\sigma_c^2 &= \left(\frac{1}{\Omega}\int_\Omega c^2 d\Omega\right) - \bar{c}^2 \\
&= \left(\frac{1}{\Omega}\int_\Omega \left(\sum_{I=1}^{N_n} c_I\varphi_I(\mathbf{x})\right)^2 d\Omega\right) - \bar{c}^2 \\
&= \left(\frac{1}{\Omega}\int_\Omega \left(\sum_{I=1}^{N_n} c_I\varphi_I(\mathbf{x})\right)\left(\sum_{J=1}^{N_n} c_J\varphi_J(\mathbf{x})\right) d\Omega\right) - \bar{c}^2 \\
&= \left(\sum_{I=1}^{N_n}\sum_{J=1}^{N_n} c_I c_J \left(\frac{1}{\Omega}\int_\Omega \varphi_I(\mathbf{x})\varphi_J(\mathbf{x}) d\Omega\right)\right) - \bar{c}^2 \\
&= \left(\sum_{I=1}^{N_n}\sum_{J=1}^{N_n} c_I c_J \Lambda_{IJ}\right) - \bar{c}^2,
\end{aligned}$$

where $\Lambda_{IJ} \equiv \frac{1}{\Omega}\int_\Omega \varphi_I(\mathbf{x})\varphi_J(\mathbf{x}) d\Omega$. $\mathbf{\Lambda}$ is a dimensionless version of the commonly used "mass matrix" in FE analysis (c.f. [58]). Taking the definition of c_I and c_J from (5.11) and using the shorthand notation of $W_I = \sum_{\alpha=1}^{N_a} \varphi_I(\mathbf{x}^\alpha) w_\alpha$, we obtain

$$\sigma_c = \left\{\left(\sum_{I=1}^{N_n}\sum_{J=1}^{N_n}\left(1-\frac{W_I}{V_I}\right)\left(1-\frac{W_J}{V_J}\right)\Lambda_{IJ}\right) - \bar{c}^2\right\}^{1/2} \tag{5.40}$$

Equation (5.40) is the simplest form we can obtain without further information about how atoms are placed with respect to the mesh, which is necessary to evaluate terms W_I and W_J, or specification of the interpolation functions, needed to evaluate W_I, W_J and Λ_{IJ}. In the trivial case of 1 element (node) for a periodic system, $\varphi_1(\mathbf{x}) = \varphi_1(\mathbf{x}^\alpha) = 1$ and $V_1 = \Omega$. Thus, $W_1 = \sum_{\alpha=1}^{N_a} w_\alpha$, $\Lambda_{11} = 1$ and $\sigma_c = \sqrt{\bar{c}^2 - \bar{c}^2} = 0$. This makes sense since there is only a single value of c for the system.

As before, we can also calculate the standard deviation of the distribution of nodal values

of vacancy concentration$\{c_I\}$, σ_{c_n}:

$$\begin{aligned}
\sigma_{c_n}^2 &= \frac{1}{N_n}\sum_{I=1}^{N_n}(c_I - \bar{c}_n)^2 \\
&= \frac{1}{N_n}\sum_{I=1}^{N_n}\left(c_I^2 - 2\bar{c}_n\, c_I + \bar{c}_n^2\right) \\
&= \frac{1}{N_n}\left\{\sum_{I=1}^{N_n} c_I^2 - 2\bar{c}_n\sum_{I=1}^{N_n} c_I + \bar{c}_n^2\sum_{I=1}^{N_n} 1\right\} \\
&= \frac{1}{N_n}\left\{\sum_{I=1}^{N_n} c_I^2 - 2N_n\bar{c}_n^2 + N_n\bar{c}_n^2\right\} \\
&= \left(\frac{1}{N_n}\sum_{I=1}^{N_n} c_I^2\right) - \bar{c}_n^2
\end{aligned}$$

Substituting equation (5.11) into this relation, one obtains:

$$\begin{aligned}
\sigma_{c_n}^2 &= \left(\frac{1}{N_n}\sum_{I=1}^{N_n}\left(1 - \frac{1}{V_I}\sum_{\alpha=1}^{N_a}\varphi_I(\mathbf{x}^\alpha)w_\alpha\right)^2\right) - \bar{c}_n^2 \\
&= \left(\frac{1}{N_n}\sum_{I=1}^{N_n}\left(1 - \frac{W_I}{V_I}\right)^2\right) - \bar{c}_n^2,
\end{aligned}$$

and thus,

$$\sigma_{c_n} = \left\{\left(\frac{1}{N_n}\sum_{I=1}^{N_n}\left(1 - \frac{W_I}{V_I}\right)^2\right) - \bar{c}_n^2\right\}^{1/2}. \tag{5.41}$$

This expression is similar in form to the one given in (5.40). Again, we note that more information regarding the functional form of $\varphi_I(\mathbf{x})$ and how atoms are placed with respect to the mesh is needed for further simplification of this expression.

5.3 Lift Operator

We now seek to define the lift operator, μ, that creates an atomistic or fine scale ensemble that is reflective of the desired coarse scale vacancy concentration profile $c_{\mathrm{v}}(\mathbf{x})$ as specified through the nodal values of vacancy concentration $\mathbf{C} = \{c_I\}$ and the interpolation functions $\{\varphi_I\}$. It is important to remember that whatever the form μ takes, it must be the case that $\mathcal{M}\mu = I$.

A reasonable starting point for a lift operation is to apply the coarse scale vacancy concentration on a full atomic lattice with zero porosity, using it to guide the deletion of

specific atoms. We first evaluate the coarse scale field $c_{\mathrm{v}}(\mathbf{x})$ at the positions of every atomic site (*i.e.* $s = 1, 2, \ldots, N_{\mathrm{s}}$),

$$c^s = c_{\mathrm{v}}(\mathbf{x}^s) = \sum_{I=1}^{N_n} \varphi_I(\mathbf{x}^s) c_I. \tag{5.42}$$

Using this "atomic" value of vacancy concentration, we then use a procedure that randomly considers specific sites from the set of all sites and then uses a random number together with the value of c^s for that site to determine if the atom at that site should be deleted. A few important considerations should be noted before detailing our deletion algorithm:

- A random integer in the range of [1, N_{s}] should be used to select each site for consideration/evaluation. This makes the probability for considering each site uniform, as opposed to a orderly progression through this range, which would favor consideration and possible deletion of lower-indexed sites.
- As the value of $c^s \in [0, 1]$ for each site, the random number chosen, r^s, should also be from this range. The evaluation criteria for deletion amounts to deleting the atom from site s if $r^s \leq c^s$.
- It may be insufficient to consider each atomic site only once when forming a deletion list. Ideally, upon completion our deletion list would contain $N_{\mathrm{v}} = \bar{c} N_{\mathrm{s}}$ atoms. However, the introduction of randomness into the site selection and evaluation processes will generally result in the number of atoms to be deleted, N_{del}, not equal to N_{v}. As such, we choose to keep a tally of the number of deleted atoms and continue the selection and evaluation process until the criterion of $N_{\mathrm{del}} = \bar{c} N_{\mathrm{s}}$ is satisfied.

Given these considerations, we now relate our initial design of a deletion algorithm, shown in Algorithm 3. Once the appropriate atoms have been deleted, the process is complete as an atomistic system has been created/initialized that should be consistent with the coarse scale field $\mathbf{C}$.

In practice, we have found that Algorithm 3 yields (upon successive application of the lift and restrict operators) the same mean nodal concentration $\bar{c}$, but produces variations in the individual nodal concentration values, c_I. We can perform a simple analysis to predict the amount of variation to be expected. We start with defining the square of this variation as $\sigma^2_{c_I} = \left\langle (c_I - \langle c_I \rangle)^2 \right\rangle$ where the notation $\langle\rangle$ is used to denote expectation value of the enclosed function or property. This expression can be simplified to $\sigma^2_{c_I} = \langle c_I^2 \rangle - \langle c_I \rangle^2$. Taking each term separately, we first note that an alternative to equation (5.11) is:

$$c_I(t) = 1 - \frac{1}{V_I} \sum_{s=1}^{N_{\mathrm{s}}} \varphi_I(\mathbf{x}^s) w_s H(r^s - \langle c^s \rangle), \tag{5.43}$$

where $H(r)$ is the Heaviside function equal to 0 for $r < 0$ and 1 for $r > 0$, and $\langle c^s \rangle$ is the expectation value of c^s. For this exercise, we assume that $\langle c^s \rangle = \bar{c} \ \ \forall s = 1, 2, \ldots, N_{\mathrm{s}}$. Hence,

$$c_I(t) = 1 - \frac{1}{V_I} \sum_{s=1}^{N_{\mathrm{s}}} \varphi_I(\mathbf{x}^s) w_s H(r^s - \bar{c}), \tag{5.44}$$

Algorithm 3 Atom deletion algorithm for N_s atomic sites

start with the coarse scale description of vacancy concentration, $\{c_I\}$
calculate the mean value of the coarse scale field, $\bar{c}$
for $s = 1$ to N_s **do**
  compute c^s using equation (5.42)
end for
initialize N_del and integer array (of size N_s) for flagging of atoms to be deleted (del_flag)
while $N_\mathrm{del} < \bar{c} N_\mathrm{s}$ **do**
  pick a random site index s from the range [1,N_s]
  if del_flag[s] = 0 **then**
    choose a random number from the range [0,1] (r^s)
    if $r^s \leq c^s$ **then**
      add the atom at site s to the deletion list and set del_flag[s] = 1
      increase N_del by 1
    end if
  end if
end while
delete atoms from the sites specified in the deletion list

We now calculate $\langle c_I \rangle$:

$$\begin{aligned}\langle c_I \rangle &= \left\langle 1 - \frac{1}{V_I} \sum_{s=1}^{N_\mathrm{s}} \varphi_I(\mathbf{x}^s) w_s H(r^s - \bar{c}) \right\rangle \\ &= 1 - \frac{1}{V_I} \sum_{s=1}^{N_\mathrm{s}} \varphi_I(\mathbf{x}^s) w_s \left\langle H(r^s - \bar{c}) \right\rangle\end{aligned}$$

Realizing that $\langle H(r^s - \bar{c}) \rangle = 1 - \bar{c}$, and $\sum_{s=1}^{N_\mathrm{s}} \varphi_I(\mathbf{x}^s) w_s = V_I$, this expression simplifies to

$$\langle c_I \rangle = 1 - \frac{1}{V_I} V_I \left(1 - \bar{c}\right) = 1 - (1 - \bar{c}) = \bar{c} \tag{5.45}$$

We also calculate $\langle c_I^2 \rangle$:

$$\begin{aligned}
\langle c_I^2 \rangle &= \left\langle \left(1 - \frac{1}{V_I} \sum_{s=1}^{N_s} \varphi_I(\mathbf{x}^s) w_s H(r^s - \bar{c}) \right)^2 \right\rangle \\
&= \left\langle 1 - 2\frac{1}{V_I} \sum_{s=1}^{N_s} \varphi_I(\mathbf{x}^s) w_s H(r^s - \bar{c}) + \left(\frac{1}{V_I} \right)^2 \left(\sum_{s=1}^{N_s} \varphi_I(\mathbf{x}^s) w_s H(r^s - \bar{c}) \right)^2 \right\rangle \\
&= 1 - 2\frac{1}{V_I} \sum_{s=1}^{N_s} \varphi_I(\mathbf{x}^s) w_s \langle H(r^s - \bar{c}) \rangle + \left(\frac{1}{V_I} \right)^2 \sum_{s=1}^{N_s} \sum_{t=1}^{N_s} \varphi_I(\mathbf{x}^s) w_s \varphi_I(\mathbf{x}^t) w_t \left\langle H(r^s - \bar{c}) H(r^t - \bar{c}) \right\rangle \\
&= 1 - 2(1 - \bar{c}) \frac{1}{V_I} \sum_{s=1}^{N_s} \varphi_I(\mathbf{x}^s) w_s + \left(\frac{1}{V_I} \right)^2 \sum_{s=1}^{N_s} \sum_{t=1}^{N_s} \varphi_I(\mathbf{x}^s) w_s \varphi_I(\mathbf{x}^t) w_t \left\langle H(r^s - \bar{c}) H(r^t - \bar{c}) \right\rangle \\
&= 1 - 2(1 - \bar{c}) + \left(\frac{1}{V_I} \right)^2 \sum_{s=1}^{N_s} \sum_{t=1}^{N_s} \varphi_I(\mathbf{x}^s) w_s \varphi_I(\mathbf{x}^t) w_t \left\langle H(r^\alpha - \bar{c}) H(r^\beta - \bar{c}) \right\rangle
\end{aligned}$$

$$\langle c_I^2 \rangle = 2\bar{c} - 1 + \left(\frac{1}{V_I} \right)^2 \sum_{s=1}^{N_s} \sum_{t=1}^{N_s} \varphi_I(\mathbf{x}^s) w_s \varphi_I(\mathbf{x}^t) w_t \left\langle H(r^s - \bar{c}) H(r^t - \bar{c}) \right\rangle \tag{5.46}$$

In order to further manipulate the right hand side of the above equation, we need to examine the double sum separately for the cases where $t = s$ and where $t \neq s$. For the first case,

$$\begin{aligned}
&\sum_{\substack{s=1,\\ t=s}}^{N_s} \varphi_I(\mathbf{x}^s) w_s \varphi_I(\mathbf{x}^t) w_t \left\langle H(r^s - \bar{c}) H(r^t - \bar{c}) \right\rangle \\
&\qquad = \sum_{s=1}^{N_s} \left(\varphi_I(\mathbf{x}^s) w_s \right)^2 \left\langle \left(H(r^s - \bar{c}) \right)^2 \right\rangle \\
&\qquad = (1 - \bar{c}) \sum_{s=1}^{N_s} \left(\varphi_I(\mathbf{x}^s) w_s \right)^2.
\end{aligned} \tag{5.47}$$

For the case of $t \neq s$, the expectation values can be evaluated separately:

$$\begin{aligned}
&\sum_{s=1}^{N_s} \sum_{\substack{t=1,\\ t \neq s}}^{N_s} \varphi_I(\mathbf{x}^s) w_s \varphi_I(\mathbf{x}^t) w_t \left\langle H(r^s - \bar{c}) H(r^t - \bar{c}) \right\rangle \\
&\qquad = \sum_{s=1}^{N_s} \sum_{\substack{t=1,\\ t \neq s}}^{N_s} \varphi_I(\mathbf{x}^s) w_s \varphi_I(\mathbf{x}^t) w_t \left\langle H(r^s - \bar{c}) \right\rangle \left\langle H(r^t - \bar{c}) \right\rangle \\
&\qquad = (1 - \bar{c})^2 \sum_{s=1}^{N_s} \sum_{\substack{t=1,\\ t \neq s}}^{N_s} \varphi_I(\mathbf{x}^s) w_s \varphi_I(\mathbf{x}^t) w_t
\end{aligned} \tag{5.48}$$

Equation (5.48) can be further modified by adding and subtracting the $t = s$ term:

$$\begin{aligned}
&\sum_{s=1}^{N_\mathrm{s}} \sum_{\substack{t=1,\\ t\neq s}}^{N_\mathrm{s}} \varphi_I(\mathbf{x}^s) w_s \varphi_I(\mathbf{x}^t) w_t \left\langle H(r^s - \bar{c}) H(r^t - \bar{c}) \right\rangle \\
&\qquad = (1-\bar{c})^2 \sum_{s=1}^{N_\mathrm{s}} \sum_{t=1}^{N_\mathrm{s}} \varphi_I(\mathbf{x}^s) w_s \varphi_I(\mathbf{x}^t) w_t - (1-\bar{c})^2 \sum_{s=1}^{N_\mathrm{s}} \left(\varphi_I(\mathbf{x}^s) w_s\right)^2 \\
&\qquad = (1-\bar{c})^2 \left(\sum_{s=1}^{N_\mathrm{s}} \varphi_I(\mathbf{x}^s) w_s \right)^2 - (1-\bar{c})^2 \sum_{s=1}^{N_\mathrm{s}} \left(\varphi_I(\mathbf{x}^s) w_s\right)^2 \\
&\qquad = (1-\bar{c})^2 V_I^2 - (1-\bar{c})^2 \sum_{s=1}^{N_\mathrm{s}} \left(\varphi_I(\mathbf{x}^s) w_s\right)^2
\end{aligned}$$

Thus,

$$\sum_{s=1}^{N_\mathrm{s}} \sum_{\substack{t=1,\\ t\neq s}}^{N_\mathrm{s}} \varphi_I(\mathbf{x}^s) w_s \varphi_I(\mathbf{x}^t) w_t \left\langle H(r^s - \bar{c}) H(r^t - \bar{c}) \right\rangle = (1-\bar{c})^2 \left\{ V_I^2 - \sum_{s=1}^{N_\mathrm{s}} \left(\varphi_I(\mathbf{x}^s) w_s\right)^2 \right\} \quad (5.49)$$

Combining equations (5.46), (5.47), and (5.49), we obtain:

$$\begin{aligned}
\left\langle c_I^2 \right\rangle &= 2\bar{c} - 1 + \left(\frac{1}{V_I}\right)^2 \left\{ (1-\bar{c}) \sum_{s=1}^{N_\mathrm{s}} \left(\varphi_I(\mathbf{x}^s) w_s\right)^2 + (1-\bar{c})^2 \left(V_I^2 - \sum_{s=1}^{N_\mathrm{s}} \left(\varphi_I(\mathbf{x}^s) w_s\right)^2 \right) \right\} \\
&= 2\bar{c} - 1 + (1-\bar{c})^2 + \left(1 - \bar{c} - (1-\bar{c})^2\right) \left(\frac{1}{V_I}\right)^2 \sum_{s=1}^{N_\mathrm{s}} \left(\varphi_I(\mathbf{x}^s) w_s\right)^2 \\
&= 2\bar{c} - 1 + 1 - 2\bar{c} + \bar{c}^2 + \left(1 - \bar{c} - 1 + 2\bar{c} - \bar{c}^2\right) \left(\frac{1}{V_I}\right)^2 \sum_{s=1}^{N_\mathrm{s}} \left(\varphi_I(\mathbf{x}^s) w_s\right)^2 \\
&= \bar{c}^2 + \left(\bar{c} - \bar{c}^2\right) \left(\frac{1}{V_I}\right)^2 \sum_{s=1}^{N_\mathrm{s}} \left(\varphi_I(\mathbf{x}^s) w_s\right)^2 \\
&= \bar{c}^2 + \bar{c}\,(1-\bar{c}) \left(\frac{1}{V_I}\right)^2 \sum_{s=1}^{N_\mathrm{s}} \left(\varphi_I(\mathbf{x}^s) w_s\right)^2
\end{aligned}$$

Hence,

$$\sigma_{c_I}^2 = \bar{c}\,(1-\bar{c})\, \frac{1}{V_I^2} \sum_{s=1}^{N_\mathrm{s}} \left(\varphi_I(\mathbf{x}^s) w_s\right)^2 \quad (5.50)$$

While this expression cannot be simplified further, we do notice that if we approximate w_s by a constant value ($\bar{w}$) for all sites, then

$$\sigma_{c_I}^2 = \gamma \left(\frac{\bar{w}}{V_I}\right)^2 \bar{c}\,(1-\bar{c})\,, \quad (5.51)$$

where $\gamma = \sum_{s=1}^{N_\mathrm{s}} \left(\varphi_I(\mathbf{x}^s)\right)^2$. Thus, as larger element volumes are used with a fixed atomic density (*i.e.* $\bar{w}/V_I$ gets smaller), the accuracy of our lift operator improves.

5.4 Improvements to the Lift Operator

As the amount of variance inherent to the lift operator can be quite large, we identify several possible ways to reduce it:

1. Use of larger elements (more atoms per element) to decrease the variance.

2. Instead of a single realization at the fine scale, the lift operator would create multiple realizations. The restrict operator would then construct the coarse scale field $\{c_I\}$ based on information from all realizations. This would improve the consistency between μ and $\mathcal{M}$.

3. Additional metrics could be used to iteratively adjust the fine scale system to make it more consistent with starting values of the coarse scale description.

An example of the third method would be as follows: First, we define a nodal error ε_I as

$$\varepsilon_I = c_I^{\text{post}} - c_I^{\text{pre}}, \tag{5.52}$$

where c_I^{pre} is the starting value of c_I before the lift operation begins and c_I^{post} is a value of c_I calculated by performing the restrict operation at the end of an iteration of the lift operation (prior to any dynamics that alters the newly created fine scale configuration). From this nodal error, we also define a global error E as

$$E = \frac{1}{\Omega} \sum_{I}^{N_n} V_I \varepsilon_I^2, \tag{5.53}$$

where $\Omega \equiv \sum_I^{N_n} V_I$. We then perform some type of optimization that minimizes the global error E.

One such optimization procedure is the simulated annealing method where modifications to the underlying atomic system are made that change the values of the nodal concentrations, Δc_I and resulting nodal and global errors, $\Delta\varepsilon_I$ and ΔE. Here, we choose to modify the atomic system by deleting an atom occupying a given random site (η), while simultaneously inserting an atom into a random vacant site (ν). This type of modification preserves the the mean vacancy concentration of the entire system, $\bar{c}$. In accordance with the sign convention used in equation (5.11), the resulting change in nodal vacancy concentration is

$$\Delta c_I = \frac{1}{V_I} \left(\varphi_I(\mathbf{x}^\eta) w_\eta - \varphi_I(\mathbf{x}^\nu) w_\nu \right). \tag{5.54}$$

This change in nodal vacancy concentration modifies the nodal error,

$$\varepsilon_I \to \varepsilon_I + \Delta c_I, \tag{5.55}$$

and therefore also the global error,

$$E \to \frac{1}{\Omega} \sum_{I}^{N_n} V_I \left(\varepsilon_I + \Delta c_I\right)^2 . \tag{5.56}$$

Thus, the change in global error is

$$\Delta E = E - \frac{1}{\Omega} \sum_{I}^{N_n} V_I \varepsilon_I^2 = \frac{1}{\Omega} \sum_{I}^{N_n} V_I \Delta c_I \left(\Delta c_I + 2\varepsilon_I\right) . \tag{5.57}$$

In the simulated annealing algorithm, the acceptance of this atom-vacancy swap occurs if $\Delta E < 0$, or it occurs with some finite probability if $\Delta E > 0$. The probability of accepting a modification that increases global error is estimated using the Boltzmann relation $p = \exp(-\Delta E/T)$, where T is the 'temperature' of the system and represents a measure of ambient inertia available to increase the global error. For a finite, non-zero value of T, p is between 0 and 1. The modification is accepted if a random number drawn from this range (r) is less than this probability, *i.e.* $r < p$. T is systematically reduced over the course of attempted modifications in order to drive the global error to a minimum. Algorithm 4 shows the lift operation that combines this error metric and simulated annealing method with the original deletion process.

The performance of algorithm 4 depends on several user-specified pieces of information, including the initial value of temperature T_0, the cooling schedule, and the maximum number of iterations. Literature suggests that T_0 should be set such that for a large number ($maximum\_number\_iterations$) of atom-vacancy swaps considered, some fixed percentage (*e.g.* 50%) of them would be selected, *i.e.* median$(p) = 0.5$. However, we have found that this produces a much higher value of T_0 than needed and instead opt to set $T_0 = 0.01$.

There has been much research done to design an optimum cooling schedule, *i.e.* a prescription for $T(k)$ that drives down the global error to the smallest value possible over the minimum number of iterations. This prescription is difficult to optimize; ideally it should force the global error function to its minimum as quickly as possible and a fast rate of decay of T accomplishes this. However, if T decays too quickly the system will be frozen in a local minimum error state such that increasing the number of iterations performed does little to decrease the final global error. Difficult also is striking a balance between practicality and robustness when prescribing the maximum number of iterations (atom-vacancy swaps) done. Too few results in too high a global error while too many increases the computational cost beyond reasonableness. One example cooling schedule (shown in Algorithm 4) is $T = T_0 a^k$ where $0 < a < 1$. In our simulations, a is chosen such that temperature decreases by six orders of magnitude by the end of the maximum number of iterations, *i.e.* $\ln(a) = \ln(10^{-6})/(maximum\_number\_iterations)$.

Algorithm 4 Lift operator algorithm with reduced error

start with the coarse scale description of vacancy concentration, $\{c_I^{\text{pre}}\}$
for $s = 1$ to N_{s} **do**
  compute c^s using equation (5.42)
end for
initialize N_{del} and integer array (of size N_{s}) for flagging of atoms to be deleted (del_flag)
while $N_{\text{del}} < \bar{c} N_{\text{s}}$ **do**
  pick a random site index s from the range [1,N_{s}]
  if del_flag[s] = 0 **then**
    choose a random number from the range [0,1] (r^s)
    if $r^s < c^s$ **then**
      add atomic site s to the deletion list and set del_flag[s] = 1
      increase N_{del} by 1
    end if
  end if
end while
using the deletion list, construct lists (deques) of atoms (sites occupied by an atom) and vacancies (sites occupied by a vacancy)
execute the restrict operator, equation (5.11), to compute $\left\{c_I^{\text{post}}\right\}$
calculate the nodal errors $\{\varepsilon_I\}$ using equation (5.52) and the global error E using equation (5.53)
if $E > E_{\text{tol}}$ **then**
  set initial value of $T = T_0$ and iteration index $k = 1$; store initial value of E as E_0
  while $\frac{E}{E_0} >$ tol **and** $k \leq max\_number\_iterations$ **do**
    pick a random site from the atom list and a random site from the vacancy list
    calculate Δc_I using equation (5.54) and accumulate global error change ΔE using equation (5.57)
    if $\Delta E \leq 0$ **then**
      modify deletion list value for the two sites
      remove sites from their respective lists and add them to the opposite list
      modify c_I, ε_I and E using the calculated changes
    else
      calculate the acceptance probability $p = \exp(-\Delta E/T)$
      choose a random number from the range [0,1] (r)
      if $r \leq p$ **then**
        modify deletion list value for the two sites
        remove sites from their respective lists and add them to the opposite list
        modify c_I, ε_I and E using the calculated changes
      end if
    end if
    recalculate the sizes of the atom and vacancy lists to verify no change in $\bar{c}$
    increment k by 1
    modify T according to some cooling schedule, *e.g.* $T = T_0 a^k$ where $0 < a < 1$
  end while
end if

5.5 Example Simulations

In this section, we examine the behavior of our restrict (equation (5.11)) and lift (algorithm 4) operators. We use atomistic systems of copper ($a_0 = 3.615$Å) as modeled using the embedded atom method by Foiles, Baskes and Daw [8]. These systems contain significant amounts of vacancies, on the order of 1 to 3%, in both uniform and non-uniform arrangements (as will be described). Simulations are performed using the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) code [38] developed by Sandia National Laboratories in conjunction with a specially written code to implement the lift operator.

5.5.1 Uniformly random vacancy concentration

We first examine a system of 32,000 atomic sites (20 x 20 x 20 unit cells) for which 1% of the sites are randomly chosen to be unoccupied, *i.e.* $\bar{c} = 0.01$. From this initial distribution of vacancies, we use our restrict operator to establish the initial coarse scale distribution of vacancy concentration $c_v(\mathbf{x})$. We then perform 20 consecutive lift/restrict operations. Figure 5.2 shows the initial coarse-scale vacancy concentration field (the 'zeroth' iteration), after the 1st lift/restrict iteration, and after the 20th iteration. We notice that our lift

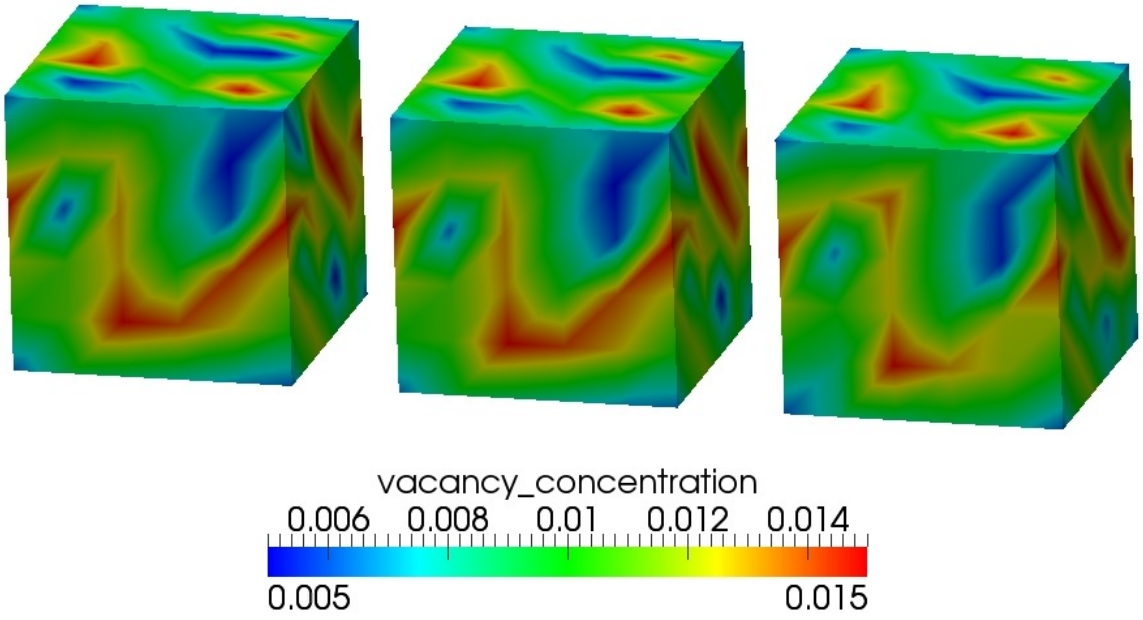


Figure 5.2. $c_v(\mathbf{x})$ for 3 different lift / restrict iterations (left to right: 0, 1, 20) for a system with $\bar{c} = 0.01$.

algorithm does a good job at creating new fine-scale realizations that produce very similar

coarse-scale distributions of the vacancy concentration field. In particular, the distribution of $c_{\mathrm{v}}(\mathbf{x})$ for the zeroth and first iterations appear nearly identical. The 20^{th} iteration does show some divergence from the initial distribution. However, within an EFPI simulation our lift/restrict approach would only be used as a single iteration, thereby minimizing this divergence.

Figure 5.3 shows the atomic configuration initially used (iteration 0) along with the fine-scale configurations produced by the lift operator at iterations 1 and 20. In this figure

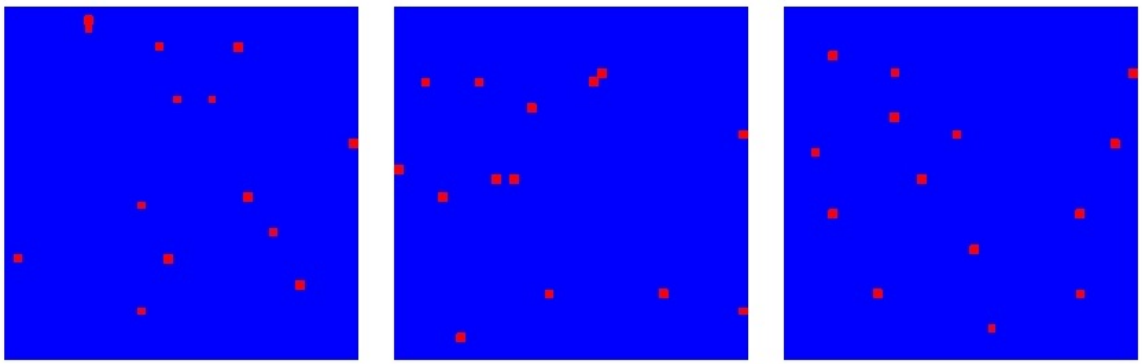
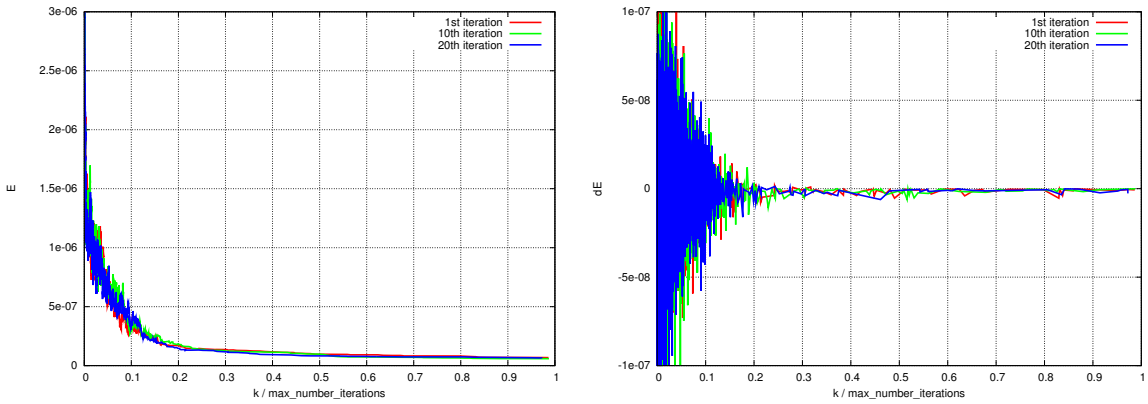


Figure 5.3. Atomic sites colored by occupancy (blue - atom, red - vacancy) for 3 different lift/restrict iterations (left to right: 0, 1, 20) for a system with $\bar{c} = 0.01$.

atomic sites are colored according to their occupancy where blue sites are atoms and red sites are vacancies. It is clear that while the lift operator creates a fine-scale configuration that corresponds to the same coarse-scale distribution of vacancy concentration upon successive application of the restrict operator (as shown in Figure 5.2), it does not produce the same fine-scale configuration each time. Not only is this behavior anticipated in the use of an equation-free simulation approach, but conceivably our lift operator could be used to create multiple fine-scale realizations of the same coarse-scale distribution. These realizations would then be run in parallel to more effectively form system averages of time-integrated behavior, as recently shown by Wagner *et al.* [64].

Figure 5.4 shows how the total and incremental global error (as defined in equations (5.56) and (5.57), respectively) evolve over the course of the simulated annealing algorithm for the 1^{st} , 10^{th} and 20^{th} application of the lift operator. We observe that the error converges very quickly initially, and then slowly later, reaching the approximate final value within 60% of the maximum number of iterations (equal to 10 x N_a = 320,000 for this simulation).

Figure 5.5 shows how the standard deviation in the distribution of nodal vacancy concentration values evolves with successive iterations of the lift/restrict operations. We note that the values observed lie between 0.00294 and 0.00315. We also note that σ_c appears to be decreasing with successive applications of the lift/restrict operations. The cause of this



(a) (b)

Figure 5.4. (a) E and (b) ΔE as a function of simulated annealing iteration number for a 32,000 atom system with initial mean porosity of 1%.

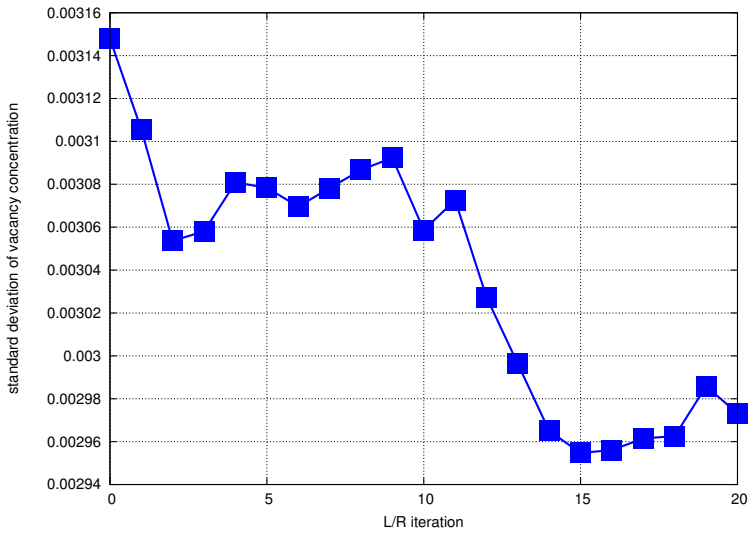


Figure 5.5. σ_c as a function of lift/restrict iteration number for repeated calls of μ and $\mathcal{M}$ for a 32,000 atom system with initial mean porosity of 1%.

behavior is not apparent.

Before leaving this example, we examine how application of the restrict and lift operator affects the dynamics of the atomistic ensemble used for fine scale simulation. In such a study, we are not yet performing projective integration but merely using our restrict operator to convert the fine scale system to a coarse scale representation, and then using the lift operator to create a new fine scale system with the same coarse scale characteristics before continuing dynamics at the fine scale. Nevertheless, performing this exercise gives us knowledge about the minimum disruption expected by application of our operators to an actual EFPI simulation.

Figure 5.6 shows the evolution of system (a) potential energy, (b) kinetic energy, (c) total energy, and (d) pressure for a molecular dynamics simulation of our 1% vacancy system initially at 0 K. We perform 10,000 timesteps of 0.001 ps (a total of 10 ps) between restrict / lift iterations. During this time, a time integration method consistent with NVE ensemble dynamics is used. From this figure, we observe that the system's excess potential energy (originating from the presence of the vacancies) is quickly equipartitioned into excess potential and kinetic energy. However, once that process occurs the system maintains a constant total energy over the course of the 10,000 step fine scale simulation. As application of the restrict and lift operators causes a slightly different vacancy distribution, the amount of this excess energy differs slightly. In Figure 5.6(c), we observe a variation of total energy of about 2 eV, or 0.0018%. Figures 5.6(b) and (d) show that after each restrict / lift interrupt (which causes a spike in values due to reinitialization of the system) the system returns to approximately the same values of kinetic energy and pressure. Overall, the figure shows that the restrict and lift operators have minimal effect on system dynamics. It does suggest, however, that the lift process should include a reinitialization of atomic velocities to be consistent with the amount of kinetic (or perhaps total) energy present prior to the restrict operation, rather than a resetting to the initial values used (zero in this case).

Figure 5.7 shows the results for a similar simulation run where the atomic system is initially set with a velocity distribution consistent with a temperature of 500 K. As for the 0 K case, this system experiences small variations of total energy ($\sim$1 eV, 0.0009%) and nearly constant levels of potential energy, kinetic energy and pressure with step-to-step variations consistent with MD simulation. Large spikes in these quantities still appear due to the reinitialization process.

Figure 5.8 shows the results for a similar simulation where the atomic system is again initially set with a velocity distribution consistent with a temperature of 500 K, and the system is treated as an NVT (isochoric-isothermal) ensemble. With regards to total energy, this system does display a larger variation during the reinitialization process. However, close examination of Figure 5.8(c) shows that the total energy stays essentially at the same value of approximately -107600 eV for all fine scale simulation periods. This value has a somewhat large variance initially ($\sim$50 eV, 0.047%) that decreases significantly over the 10,000 timesteps of simulation. Similar trends are noticed in the other graphed quantities.

From these studies, we conclude that our restrict and lift operators can maintain system

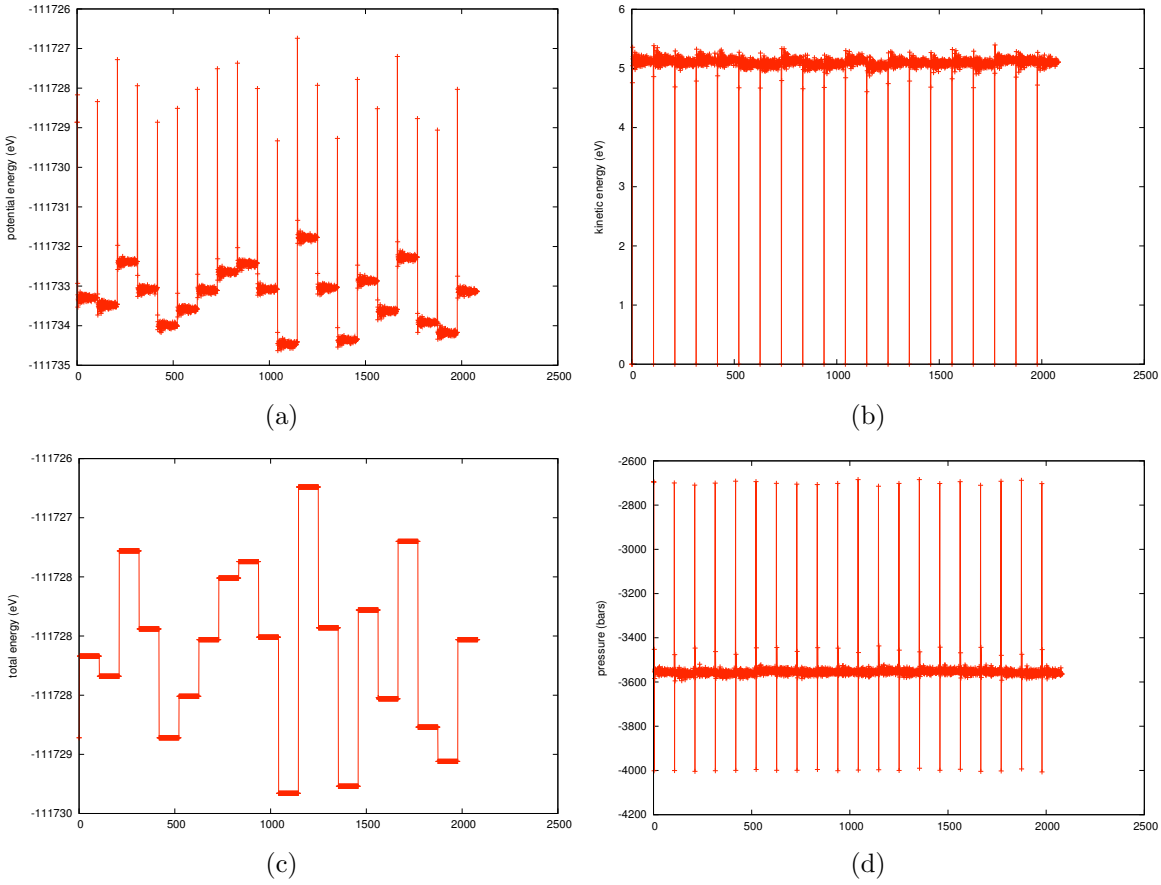


Figure 5.6. MD simulation of 1% vacancy system with 10,000 time-steps between restrict / lift interrupts, NVE at initial temperature of 0 K. Horizontal axis is in units of 100 timesteps.

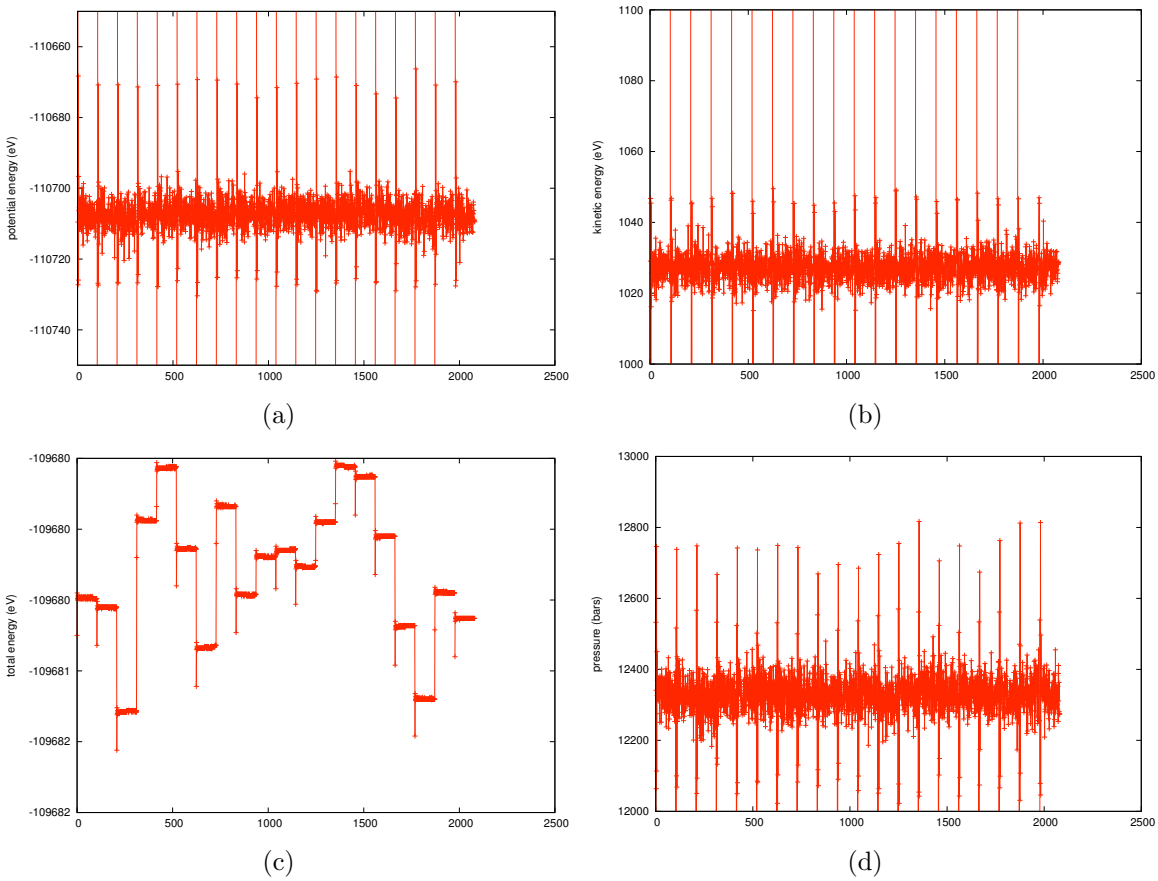


Figure 5.7. MD simulation of 1% vacancy system with 10,000 time-steps between restrict / lift interrupts, NVE at initial temperature of 500 K. Horizontal axis is in units of 100 timesteps.

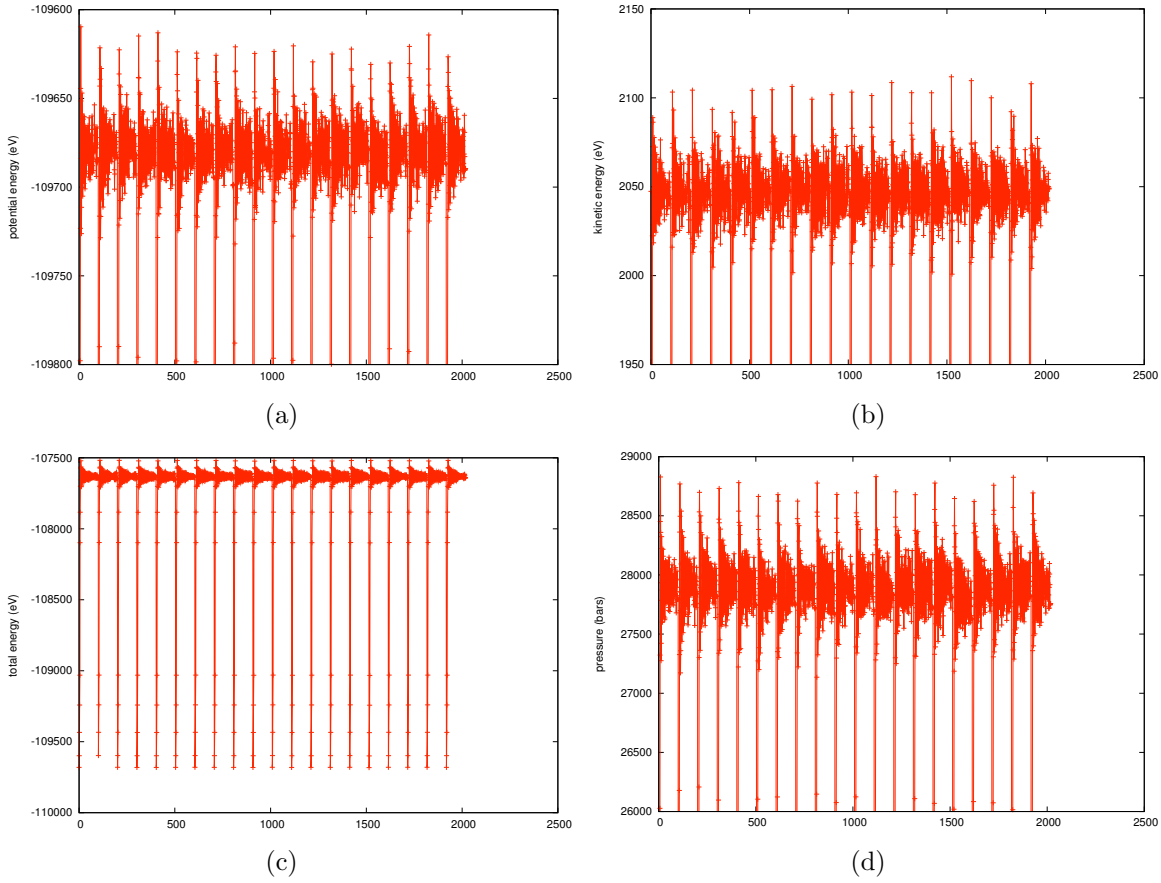


Figure 5.8. MD simulation of 1% vacancy system with 10,000 time-steps between restrict / lift interrupts, NVT at temperature of 500 K. Horizontal axis is in units of 100 timesteps.

statistics for interrupted dynamics run. Clearly, some room for improvement exists with regard to reinitialization of atomic velocities at the end of the lift process. However, even without this improvement restrict / lift interrupts appear to have minimal impact on fine scale system dynamics.

5.5.2 Bilinear vacancy concentration profile

In the previous example, we used our restrict and lift operators on a system with a uniform vacancy concentration, with no pre-defined spatial distribution of the coarse scale concentration field. In this section, we examine a somewhat longer system (96,000 sites, 20 x 20 x 60 unit cells) in which a bilinear distribution is initially set for the coarse scale concentration field. In this system, the system is divided into a 5 x 5 x 14 element grid with $c_{\mathrm{v}}(\mathbf{x})$ set to values varying from 3% for the middle plane of nodes along the long (z) direction and a value of 0% at the periodic ends. Analytic functions are used to prescribe nodal values to vary linearly between these values of 0 and 3%. As in the previous case, 20 successive restrict / lift operations are performed.

Figure 5.9 shows the coarse scale vacancy concentration distribution after the 1^{st}, 2^{nd}, 10^{th} and 20^{th} lift / restrict iterations. We observe that our lift / restrict process does a good job at maintaining the coarse scale distribution between two consecutive iterations. As in the case of the uniform distribution, we notice a variation of this coarse scale field that evolves over the span of 20 iterations. As before, we point out that within an EFPI simulation our lift / restrict approach would only be used as a single iteration, thereby minimizing this variation.

Figure 5.10 shows the atomic configurations produced by the lift operator at iterations 1, 2, 10, and 20. In this figure atomic sites are colored according to their occupancy where blue sites are atoms and red sites are vacancies. As before, we note that fine scale configurations are such that they produce similar coarse scale distributions of vacancy concentration, even though they themselves are distinctly different from one another.

Figure 5.11 shows the evolution of global error E and change in E between iterations for the simulated annealing algorithm for the 1^{st}, 10^{th} and 20^{th} lift / restrict iterations. Again, as with the uniform concentration example we observe quick convergence of this error, reaching its final value within 50 to 60% of the total number of simulated annealing steps used.

One feature different for this bilinear case than the previous example is that while the total number of vacancies in our system remains constant (1,440), our lift and restrict algorithms no longer maintain the same value of $\bar{c}$. Figure 5.12(a) shows a small decrease of $\bar{c}$ over the 20 lift / restrict iterations, from an initial value of 0.015 to a final value of 0.0149989. While this decrease does not impact the number of vacancies created in the fine scale configurations, it certainly will at some point. Based on the rate of decrease observed in Figure 5.12(a), we can estimate that at about 109 iterations the coarse scale field will be such that application of the lift process will result in one less vacancy. As we noted above, our lift and restrict

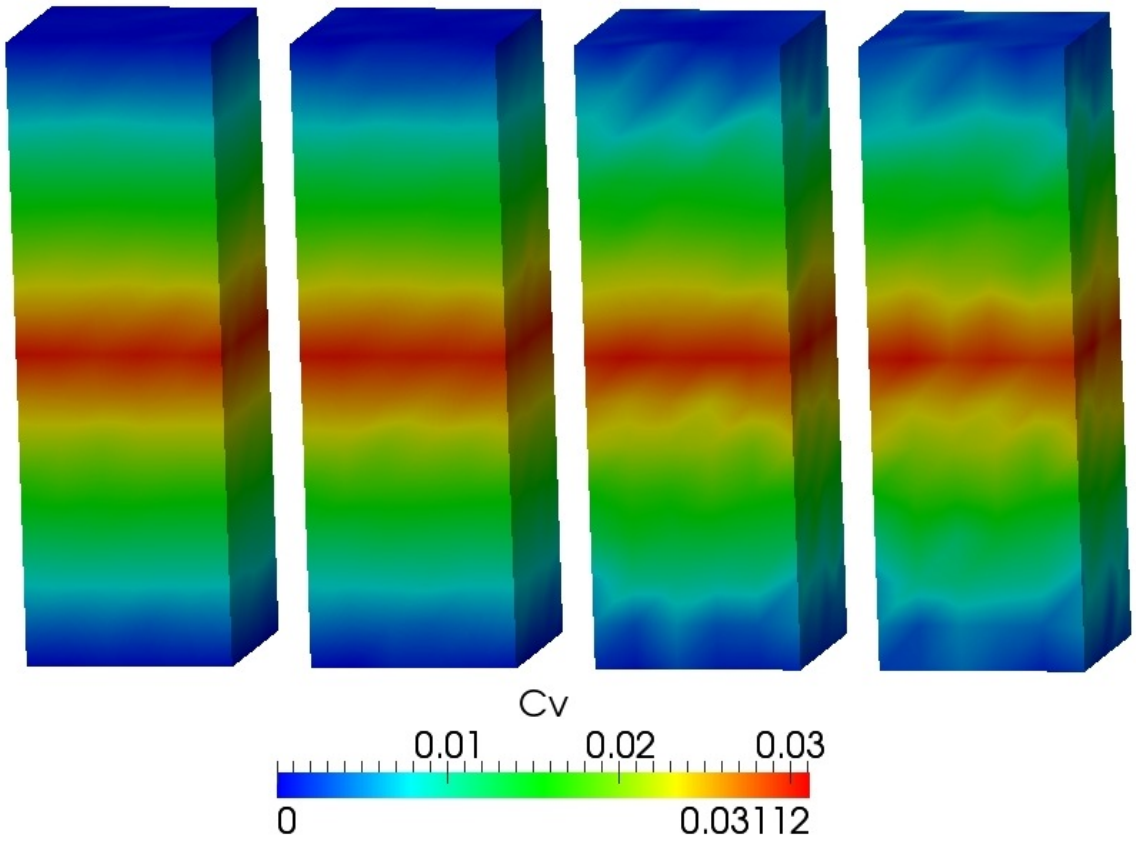


Figure 5.9. $c_v(\mathbf{x})$ for 4 different lift / restrict iterations (left to right: 1, 2, 10, 20) for a system with a bilinear distribution varying from 0 at the ends to 0.03 at the middle plane of nodes on a grid of 14 elements in the z-direction.

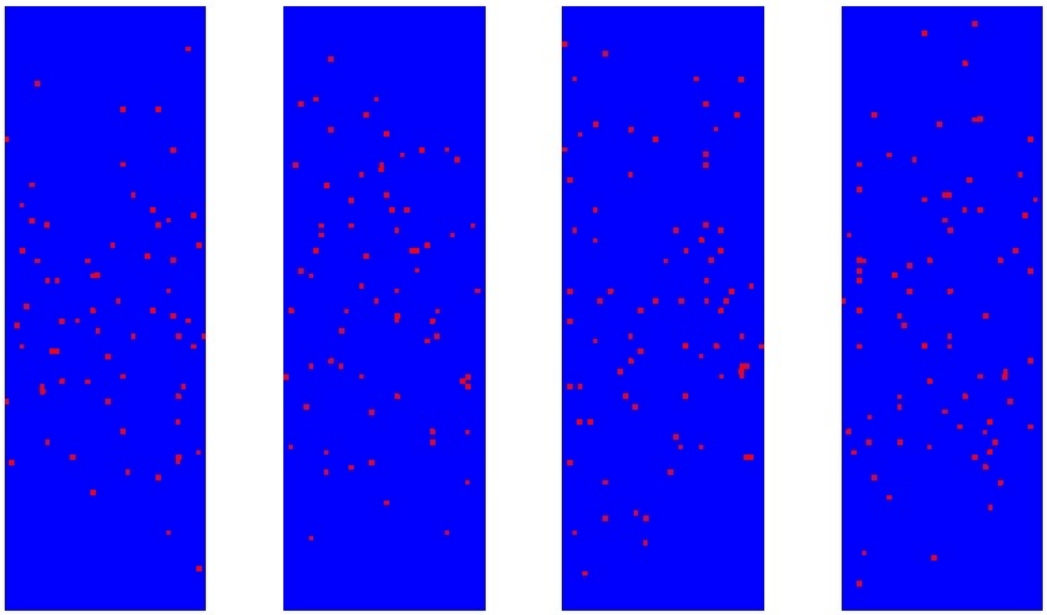
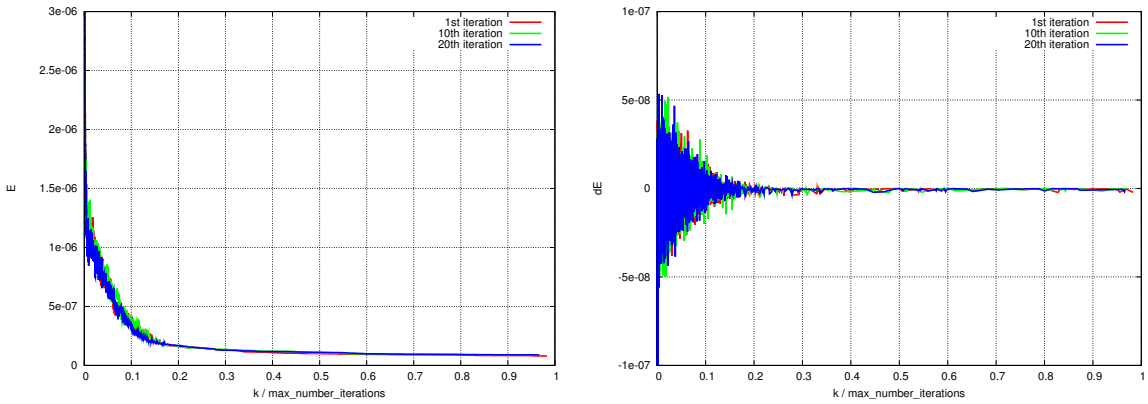
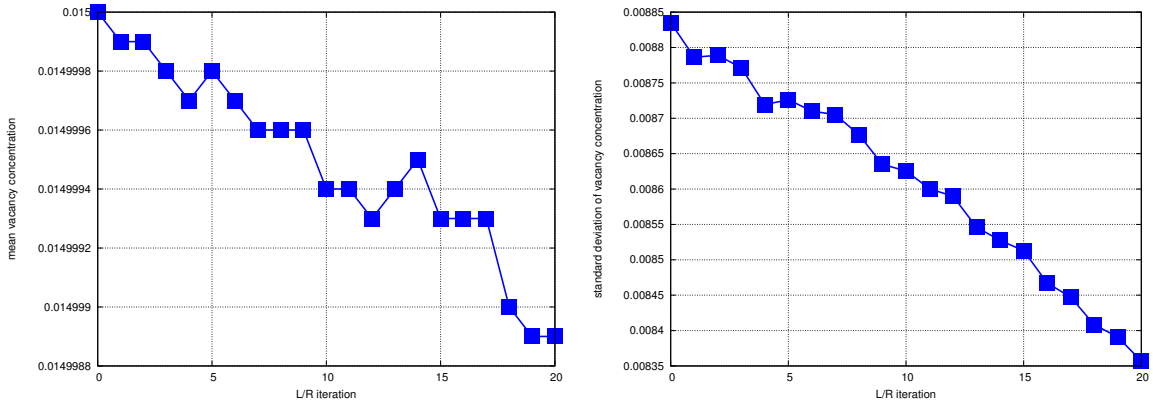


Figure 5.10. Atomic sites colored by occupancy (blue - atom, red - vacancy) for 4 different lift / restrict iterations (left to right: 1, 2, 10, 20) for a system with a bilinear distribution varying from 0 at the ends to 0.03 at the middle plane of nodes on a grid of 14 elements in the z-direction.



(a) (b)

Figure 5.11. (a) E and (b) ΔE as a function of simulated annealing iteration number for a 96,000 atom system with a bilinear distribution of c_v that varies between 0 and 0.03 on a grid of 14 elements in the z-direction.



(a) (b)

Figure 5.12. (a) $\bar{c}$ and (b) σ_c as a function of lift / restrict iteration number for repeated calls of μ and $\mathcal{M}$ for a 96,000 atom system with a bilinear distribution of c_v that varies between 0 and 0.03 on a grid of 14 elements in the z-direction.

operators will not ordinarily be used in a repetitive fashion as was done here. However, it is disconcerting that some error is introduced such that $\mu\mathcal{M}$ is not exactly equal to unity.

We can investigate this behavior further by examining how the variation of site weights and the average weight of occupied sites interacts with the gradient imposed by c_v. To start, we recall the necessity of solving for the site weights for use in atomic quadrature due to the fact that the discreteness of the mesh reflected by element size can be uneven with respect to the discreteness of the underlying atomic lattice. Thus, different elements of the same size can contain different numbers of atomic sites. For this example, we used 60 unit cells of sites (120 atomic planes) in the z-direction broken into 14 elements. As $60/14 = 4.2857...$, we do not have the same number of sites in each element and expect a distribution of site weights. This distribution is shown in histogram form in Figure 5.13. This figure shows a non-normal distribution of site weights for the system. As compared with the mean site weight value $\bar{w}$ (shown as a pink bar in the figure), we observe non-uniformly spaced peaks in the distribution at both higher and lower values.

In equation (5.38), we expressed the system-averaged vacancy concentration as $\bar{c} = 1 - (\bar{w}_a N_a)/(\bar{w} N_\mathrm{s})$, where $\bar{w}_a$ is the mean weight for occupied sites and $\bar{w}$ is the mean weight for all sites. If we express $\bar{w}_a$ as $\bar{w} + \delta\bar{w}$, we can recast this as

$$\bar{c} = \bar{c}_\mathrm{ideal} - \frac{\delta\bar{w} N_a}{\bar{w} N_\mathrm{s}}, \tag{5.58}$$

where $\bar{c}_\mathrm{ideal} = 1 - N_a/N_\mathrm{s} = N_\mathrm{v}/N_\mathrm{s}$ is the "ideal" mean vacancy concentration based only on this numbers of occupied and unoccupied sites. Equation (5.58) shows that depending on the sign of the variance $\delta\bar{w}$, $\bar{c}$ can be either higher or lower than its ideal value. Figure 5.14

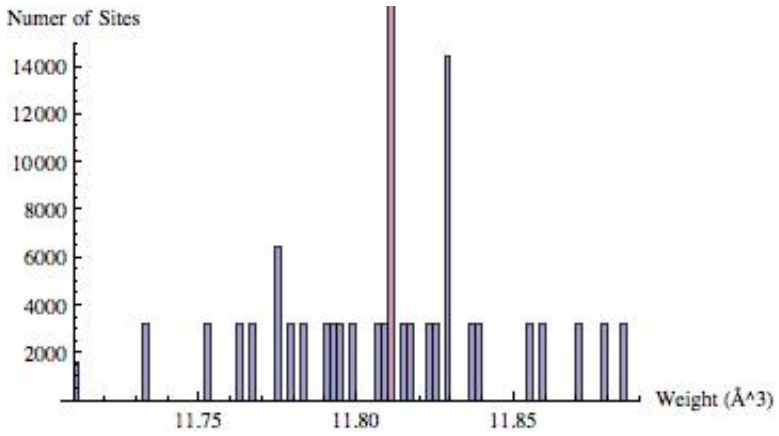


Figure 5.13. Histogram of site weights for the 96,000 atomic site system where 60 unit cells and 14 elements are used in the z-direction. The pink bar shows the mean site weight, $\bar{w}$

shows this variance as a function of number of lift / restrict iterations for the 14 element case examined here. We observe that for this case, $\delta\bar{w}$ is positive at all iterations. This corresponds well with $\bar{c} < \bar{c}_{\text{ideal}}$ as observed in Figure 5.12(a). We also examine how the site weight distribution affects the deviation of our initial bilinear distribution of c_{v}. Figure 5.15 shows this variation as a function of nodal position in the z-direction. For this figure, values have been averaged among all nodes lying in the x-y plane at the same value of z. This figure shows a few illuminating features. First, as previously noted the distribution produced after the 2^{nd} lift / restrict iteration is virtually identical to the one before it. Second, by the 20^{th} iteration there has been a slight reduction in c_{v} at nodes in the middle of the system (at the higher values of c_{v}) and a noticeable increase in c_{v} at nodes towards the ends of the system (at the lower values of c_{v}). Figure 5.15(b) shows a definitive pattern in the spatial distribution of site weights, but little correspondence between value of w_α and value of c_{v}. There may be some correspondence between the slope of how w_α varies in the z-direction and the deviation of c_{v} from its original bilinear form. We observe that positions at which the slope has a high value coincide coincide with positions of the lowest and highest values of c_{v}. This slope is lower around 55 and 160 Å, where there appears to be closer agreement with the bilinear form.

We also investigate how the choice of number of elements impacts these features. Figures 5.16 and 5.17 shows the variation of $\bar{c}$ and σ_c with lift / restrict iteration number for grids composed of 10 and 16 elements in the z-direction, respectively. As the use of 10 elements produces a uniform number of atoms in each element, we observe that $\bar{c}$ remains at its initial value of 0.015. By comparison, the use of 16 elements (as with 14) gives an uneven number of atomic planes per element in the z-direction, which again produces a small variation of $\bar{c}$ such that $\bar{c} < 0.015$ for all lift / restrict iterations. It is interesting that this

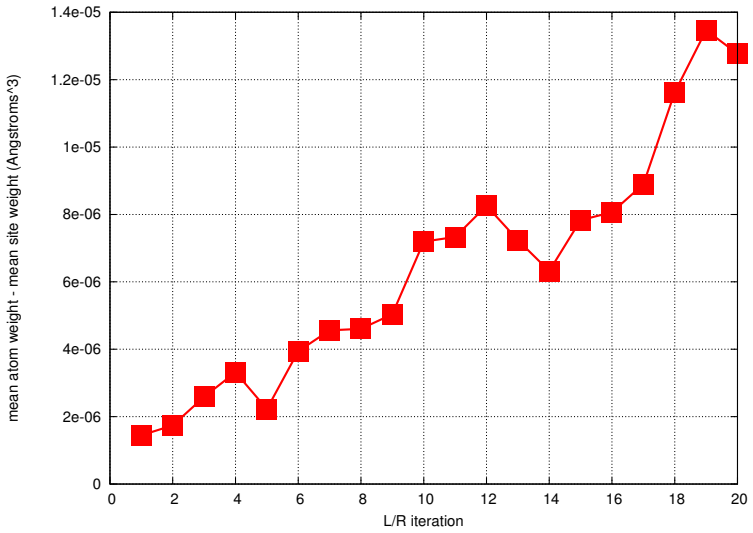
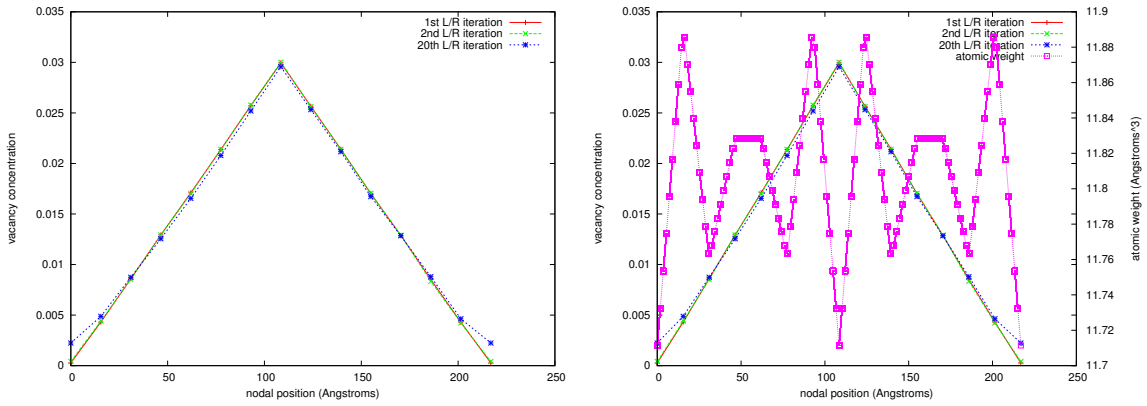
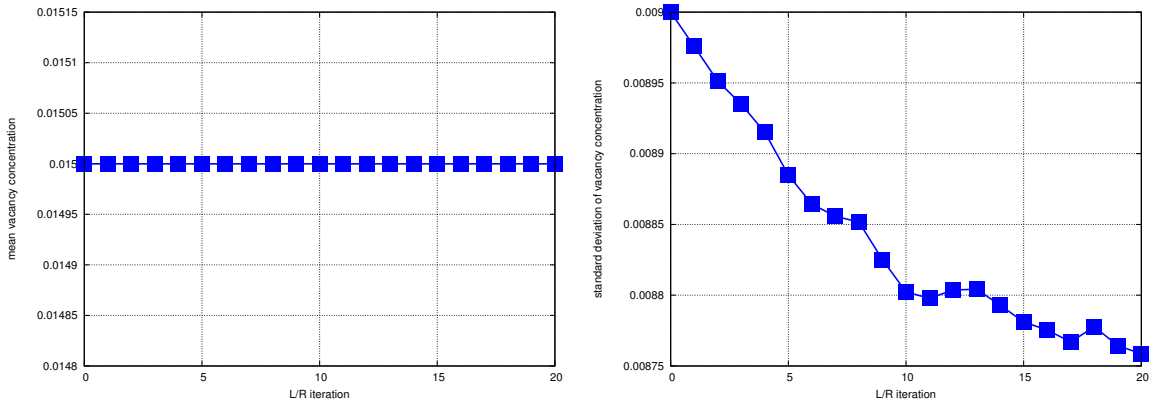


Figure 5.14. Difference between mean atom (occupied site) weight and mean site weight ($\delta\bar{w}$) as a function of lift / restrict iteration number for the bilinear example.



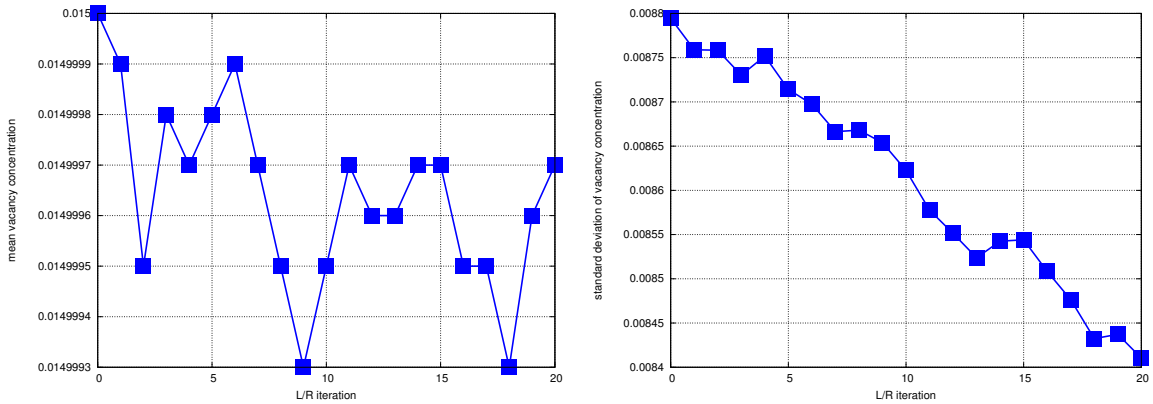
(a) (b)

Figure 5.15. (a) Variation of vacancy concentration as a function of nodal position in the z-direction for the 1^{st}, 2^{nd} and 20^{th} lift / restrict iterations. (b) Same graph also showing z-direction variation of site weights.



(a) (b)

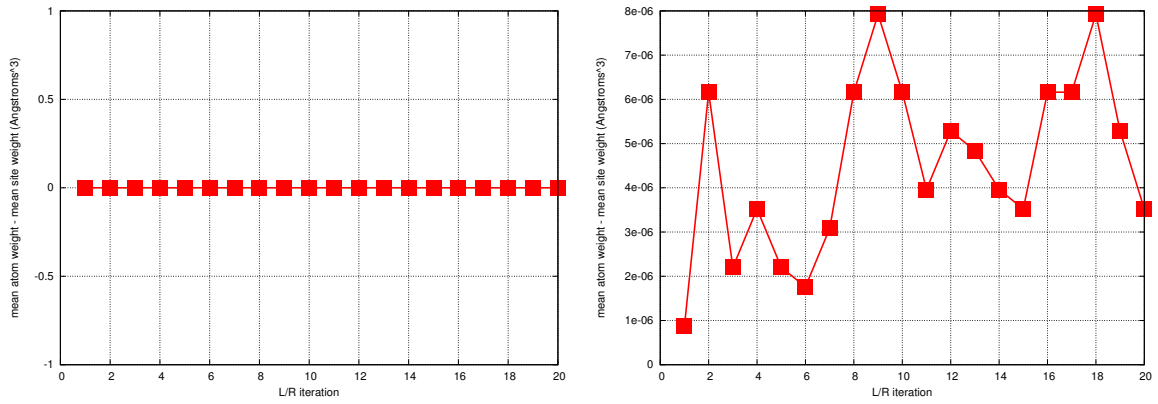
Figure 5.16. (a) $\bar{c}$ and (b) σ_c as a function of lift / restrict iteration number for repeated calls of μ and $\mathcal{M}$ for a 96,000 atom system with a bilinear distribution of c_{v} that varies between 0 and 0.03 on a grid of 10 elements in the z-direction.



(a) (b)

Figure 5.17. (a) $\bar{c}$ and (b) σ_c as a function of lift / restrict iteration number for repeated calls of μ and $\mathcal{M}$ for a 96,000 atom system with a bilinear distribution of c_{v} that varies between 0 and 0.03 on a grid of 16 elements in the z-direction.

variation, however, is less monotonic in its form than for the 14 element case. The behavior of $\bar{c}$ for these systems again correlates well with the variance $\delta\bar{w}$, as shown in Figure 5.18. As



(a) (b)

Figure 5.18. Difference between mean atom (occupied site) weight and mean site weight ($\delta\bar{w}$) as a function of lift / restrict iteration number for the bilinear example: (a) 10 elements in the z-direction, (b) 16 elements in the z-direction.

seen in Figure 5.12(b), Figures 5.16(b) and 5.17(b) also show a decrease of σ_c with increasing lift / restrict iteration, the rate of which is somewhat less for the 10 element system than for the 14 or 16 element ones.

Figure 5.19 shows the variation of c_{v} as a function of nodal position in the z-direction for the 10 and 16 element cases. As we would expect, we see very small variations of the exact bilinear form for the 10 element case (the largest observable variation being at the ends of the system where $c_{\mathrm{v}} \to 0$), and variations for the 16 element case are comparable to those for our original 14 element system. For clarity, Figure 5.20 compares the distribution of c_{v} after the 20^{th} lift / restrict iteration for all three systems.

5.6 Concluding Remarks

We have formulated a coarse scale representation of vacancy concentration in a solid material to be used in the framework of equation-free projective integration. This restrict operation, given in equation (5.11), used site weights determined from atomic quadrature with linear interpolation functions commonly used in finite element analysis. We have also developed a corresponding lift operation that takes a given coarse scale distribution of vacancy concentration and creates a consistent fine scale configuration. This operation is shown as Algorithm 4 and uses a simulated annealing process to minimize the error between a given coarse scale

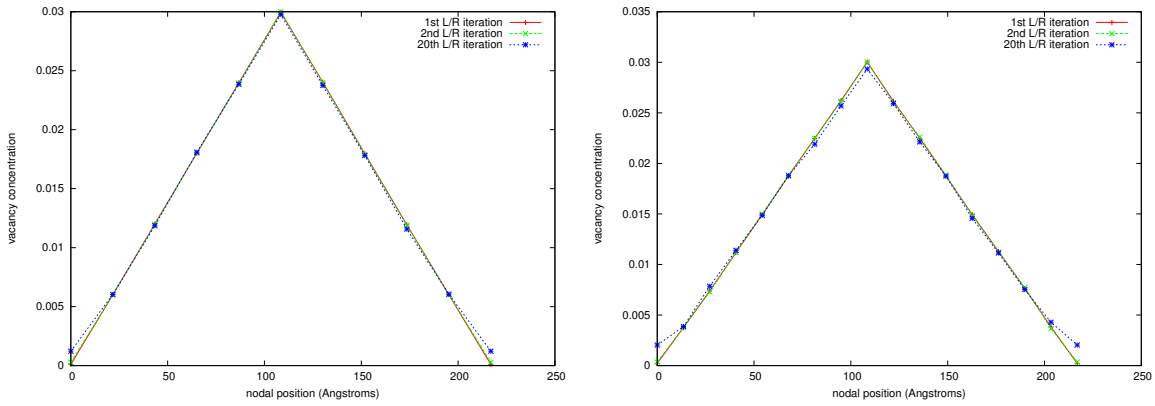


Figure 5.19. Variation of vacancy concentration as a function of nodal position in the z-direction for the 1^{st}, 2^{nd} and 20^{th} lift / restrict iterations: (a) 10 elements in the z-direction, (b) 16 elements in the z-direction.

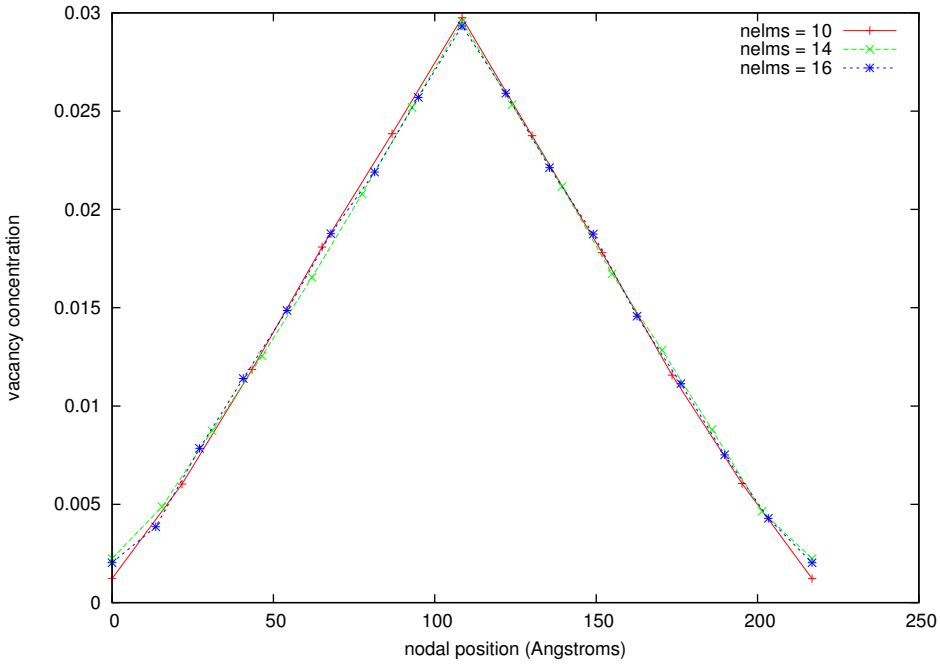


Figure 5.20. Variation of vacancy concentration as a function of nodal position in the z-direction for the 20^{th} lift / restrict iteration for the 10, 14 and 16 element systems.

field, and the resultant of the restrict operator applied to a given fine scale configuration. We have applied these operators to simple examples to establish their properties and the fidelity with which they exhibit consistency. Potential methods for future improvement of our operators include the creation and use of multiple fine scale configurations within the lift process, and the need for attention to be paid when spatial gradients of the coarse scale field are large relative to the scale over which site weights vary.

References

[1] R. V. Abramov. The multidimensional moment-constrained maximum entropy problem: A BFGS algorithm with constraint scaling. *J. Comput. Phys.*, 228:96–108, 2009.

[2] N. C. Bartelt, W. Theis, and R. M. Tromp. Ostwald ripening of two-dimensional islands on Si(001). *Phys. Rev. B*, 54(16):11741–11751, 1996.

[3] E. Bitzek, P. Koskinen, F. Gahler, M. Moseler, and P. Gumbsch. Structural relaxation made simple. *Phys. Rev. Lett.*, 97:170201, 2006.

[4] 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.

[5] A. Chatterjee. An introduction to the proper orthogonal decomposition. *Current Science*, 78(7):808—817, 2000.

[6] A. Chatterjee and D. G. Vlachos. An overview of spatial microscopic and accelerated kinetic Monte Carlo methods. *J. Comput.-Aided Mater.*, 14(2):253–308, 2007.

[7] L. Chen, P. G. Debenedetti, C. W. Gear, and I. G. Kevrekidis. From molecular dynamics to coarse self-similar solutions: a simple example using equation-free computation. *J. Non-Newton. Fluid*, 120(1-3):215–223, 2004.

[8] S. M. Foiles, M. I. Baskes, and M. S. Daw. Embedded-atom-method functions for the fcc metals Cu, Ag, Au, Ni, Pd, Pt, and their alloys. *Phys. Rev. B*, 33:7983–7991, 1986.

[9] D. Frenkel and B. Smit. *Understanding Molecular Simulation: From Algorithms to Applications*. Academic Press, San Diego, 1996.

[10] C. W. Gear, T. J. Kaper, I. G. Kevrekidis, and A. Zagaris. Projecting to a slow manifold: Singularly perturbed systems and legacy codes. *SIAM J. Appl. Dyn. Syst.*, 4(3):711–732, 2005.

[11] C. W. Gear and I. G. Kevrekidis. Projective methods for stiff differential equations: Problems with gaps in their eigenvalue spectrum. *SIAM J. Sci. Comput.*, 24(4):1091–1106, 2003.

[12] C. W. Gear and I. G. Kevrekidis. Constraint-defined manifolds: A legacy code approach to low-dimensional computation. *J. Sci. Comput.*, 25(1):17–28, 2005.

[13] R. J. Hardy. Formulas for determining local properties in molecular-dynamics simulations: Shock waves. *J. Chem. Phys.*, 76(1):622–628, 1982.

[14] G. Henkelman and H. Jónsson. Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points. *J. Chem. Phys.*, 113:9978–9985, 2000.

[15] G. Henkelman and H. Jonsson. Long time scale kinetic monte carlo simulations without lattice approximation and predefined event table. *J. Chem. Phys.*, 115(21):9657–9666, 2001.

[16] G. Henkelman, B. P. Uberuaga, and H. Jónsson. A climbing image nudged elastic band method for finding saddle points and minimum energy paths. *J. Chem. Phys.*, 113:9901–9904, 2000.

[17] H. C. Huang, G. H. Gilmer, and T. D. de la Rubia. An atomistic simulator for thin film deposition in three dimensions. *J. Appl. Phys.*, 84(7):3636–3649, 1998.

[18] E. T. Jaynes. Information theory and statistical mechanics. *Phys. Rev.*, 106(4):620–630, 1957.

[19] Z. Jiang and C. Ebner. Simulations of low-temperature annealing of crystal surfaces. *Phys. Rev. B*, 53(16):11146–11151, 1996.

[20] Y. Jiao, F. H. Stillinger, and S. Torquato. Modeling hetergeneous materials via two-point correlation functions: Basic principles. *Phys. Rev. E.*, 76:031110, 2007.

[21] Y. Jiao, F. H. Stillinger, and S. Torquato. Modeling hetergeneous materials via two-point correlation functions: II. Algorithmic details and applications. *Phys. Rev. E.*, 77:031135, 2008.

[22] H. Jonsson, G. Mills, and K. W. Jacobsen. Nudged elastic band method for finding minimum energy paths of transitions. In B. J. Berne, G Ciccotti, and D. F Coker, editors, *Classical and Quantum Dynamics in Condensed Phase Simulations*, pages 385–404. World Scientific, Singapore, 1998.

[23] I. G. Kevrekidis, C. W. Gear, J. M. Hyman, P. G. Kevrekidis, O. Runborg, and C. Theodoropoulos. Equation-free, coarse-grained multiscale computation: enabling microscopic simulators to perform system-level analysis. *Comm. Math. Sci.*, 1(4):715–762, 2003.

[24] I. G. Kevrekidis and G. Samaey. Equation-free multiscale computation: Algorithms and applications. *Annu. Rev. Phys. Chem.*, 60:321–344, 2009.

[25] J. Klars and W. Selke. Disordered flat phase of a crystal surface: Critical and dynamic properties. *Phys. Rev. B*, 74(7):073405, 2006.

[26] P. S. Koutsourelakis. Probabilistic characterization and simulation of multi-phase random media. *Probabilist Eng. Mech.*, 21(3):227–234, 2006.

[27] 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*. Academic Press, New York, 1975.

[28] J. Li, P. G. Kevrekidis, C. W. Gear, and I. G. Kevrekidis. Deciding the nature of the coarse equation through microscopic simulations: The baby-bathwater scheme. *Multiscale Model. Sim.*, 1(3):391–407, 2003.

[29] M. Love. *Probability Theory*. Springer-Verlag, 4th edition, 1977.

[30] G. Mills and H. Jónsson. Quantum and thermal effects in h_2 dissociative adsorption: Evaluation of free energy barriers in multidimensional quantum systems. *Phys. Rev. Lett.*, 72:1124–1128, 1994.

[31] G. Mills, H. Jónsson, and G. K. Schenter. Reversible work transition state theory: application to dissociative adsorption of hydrogen. *Surf. Sci.*, 324:305–337, 1995.

[32] W. W. Mullins. Theory of thermal grooving. *J. Appl. Phys.*, 28(3):333–339, 1957.

[33] M. V. R. Murty. Morphological stability of nanostructures. *Phys. Rev. B*, 62(24):17004–17011, 2000.

[34] M. V. R. Murty and B. H. Cooper. Dynamics of surface profile evolution through surface diffusion. *Phys. Rev. B*, 54(15):10377–10380, 1996.

[35] F. A. Nichols and W. W. Mullins. Morphological changes of a surface of revolution due to capillarity-induced surface diffusion. *J. Appl. Phys.*, 36(6):1826–1835, 1965.

[36] C. Oskay and J. Fish. Fatigue life prediction using 2-scale temporal asymptotic homogenization. *Int. J. Numer. Meth. Eng.*, 61(3):329–359, 2004.

[37] E. Patelli and G. Schueller. On optimization techniques to reconstruct microstructures of random heterogeneous media. *Comp. Mater. Sci.*, 45(2):536–549, 2009.

[38] S. J. Plimpton. LAMMPS Molecular Dynamics package. `http://lammps.sandia.gov`, 2011.

[39] S. J. Plimpton. LAMMPS Molecular Dynamics package, interatomic potential comparisons. `http://lammps.sandia.gov/bench.html/#potentials`, 2011.

[40] S. J. Plimpton. SPPARKS kinetic Monte Carlo package. `http://www.sandia.gov/~sjplimp/spparks.html`, 2011.

[41] S. J. Plimpton, C. C. Battalie, M. E. Chandross, E. A. Holm, A. P. Thompson, V. Tikare, G. J. Wagner, E. B. Webb, X. Zhou, C. Garcia Cardona, and A. Slepoy. Crossing the mesoscale no-man's land via parallel kinetic monte carlo. Technical Report SAND2009-6226, Sandia National Laboratories, October 2009.

[42] S. J. Plimpton and A. P. Thompson. Efficient exact and approximate kinetic Monte Carlo algorithms. *ICIAM 2011 conference, Symposium on Multiscale Approximations of Kinetic Monte Carlo Simulations*, Vancouver, Canada, 2011.

[43] J. A. Quiblier. A new 3-dimensional modeling technique for studying porous-media. *J. Colloid Interf. Sci.*, 98(1):84–102, 1984.

[44] N. N. Schraudolph and T. Graepel. Towards stochastic conjugate gradient methods. *ICONIP'02: Proceedings of the 9th International Conference on Neural Information Processing*, pages 853–856, 2002.

[45] P. C. Searson, R. Li, and K. Sieradzki. Surface-diffusion in the solid-on-solid model. *Phys. Rev. Lett.*, 74(8):1395–1398, 1995.

[46] W. Selke and T. Bieker. Morphological-changes of periodic surface profiles. *Surf. Sci.*, 281(1-2):163–177, 1993.

[47] W. Selke and P. M. Duxbury. Surface profile evolution above roughening. *Z. Phys. B Con. Mat.*, 94(3):311–318, 1994.

[48] W. Selke and P. M. Duxbury. Equilibration of crystal surfaces. *Phys. Rev. B*, 52(24):17468–17479, 1995.

[49] M. A. Shay, J. F. Drake, and B. Dorland. Equation free projective integration: A multiscale method applied to a plasma ion acoustic wave. *J. Comput. Phys.*, 226(1):571–585, 2007.

[50] N. Sheehan and S. Torquato. Generating microstructures with specified correlation functions. *J. Appl. Phys.*, 89(1):53–60, 2001.

[51] D. Sheppard, R. Terrell, and G. Henkelman. Optimization methods for finding minimum energy paths. *J. Chem. Phys.*, 128:134106, 2008.

[52] Y. Shim, J. G. Amar, B. P. Uberuaga, and A. F. Voter. Reaching extended length scales and time scales in atomistic simulations via spatially parallel temperature-accelerated dynamics. *Phys. Rev. B*, 76(20):205439, 2007.

[53] S. Sirisup, G. E. Karniadakis, D. Xiu, and I.G. Kevrekidis. Equation-free/Galerkin-free POD-assisted comptuation of incompressible flows. *J. Comput. Phys.*, 207:568–587, 2005.

[54] B. E. Sonday, M. Haataja, and I. G. Kevrekidis. Coarse-graining the dynamics of a driven interface in the presence of mobile impurities: Effective description via diffusion maps. *Phys. Rev. E*, 80:031102–1–031102–11, 2009.

[55] M. R. Sørensen and A. F. Voter. Temperature-accelerated dynamics for simulation of infrequent events. *J. Chem. Phys.*, 112:9599–9606, 2000.

[56] S. Sriraman, I. G. Kevrekidis, and G. Hummer. Coarse nonlinear dynamics and metastability of filling-emptying transitions: Water in carbon nanotubes. *Phys. Rev. Lett.*, 95(13):130603, 2005.

[57] F. Szalma, W. Selke, and S. Fischer. Dynamics of surface steps. *Physica A*, 294(3-4):313–322, 2001.

[58] J. A. Templeton, R. E. Jones, J. W. Lee, J. A. Zimmerman, and B. M. Wong. A long-range electric field solver for molecular dynamics based on atomistic-to-continuum modeling. *J. Chem. Theory Comput.*, 7:1736–1749, 2011.

[59] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[60] S. Torquato. Statistical Description of Microstructures. *Annu. Rev. Mater. Res.*, 32:77–111, 2002.

[61] B. P. Uberuaga, R. G. Hoagland, A. F. Voter, and S. M. Valone. Direct transformation of vacancy voids to stacking fault tetrahedra. *Phys. Rev. Lett.*, 99(13):135501, 2007.

[62] A. F. Voter. Parallel replica method for dynamics of infrequent events. *Phys. Rev. B*, 57(22):R13985–R13988, 1998.

[63] A. F. Voter, F. Montalenti, and T. C. Germann. Extending the time scale in atomistic simulation of materials. *Annu. Rev. Mater. Res.*, 32:321–346, 2002.

[64] G. J. Wagner, X. W. Zhou, and S. J. Plimpton. Equation-free accelerated simulations of the morphological relaxation of crystal surfaces. *Int. J. Multiscale Comp. Eng.*, 8(4):423–439, 2010.

[65] E. B. Webb III, J. A. Zimmerman, and S. C. Seel. Reconsideration of continuum thermomechanical quantities in atomic scale simulations. *Math. Mech. Solids*, 13:221–266, 2008.

[66] D. B. Xiu and I. G. Kevrekidis. Equation-free, multiscale computation for unsteady random diffusion. *Multiscale Model. Sim.*, 4(3):915–935, 2005.

[67] C. L. Y. Yeong and S. Torquato. Reconstructing random media. *Phys. Rev. E*, 57(1):495–506, 1998.

[68] N. Zabaras and S. Sankaran. A maximum entropy approach for property prediction of random microstructures. *Acta Mater.*, 54(8):2265–2276, 2006.

[69] J. A. Zimmerman, R. E. Jones, and J. A. Templeton. A material frame approach for evaluating continuum variables in atomistic simulations. *J. Comput. Phys.*, 229(6):2364–2389, 2010.

DISTRIBUTION:

1	MS 0316	R.C. Schmidt, 1444
1	MS 0372	H.E. Fang, 1524
1	MS 0747	V. Tikare, 6223
1	MS 0825	J.S. Lash, 1510
1	MS 0889	C.C. Battaile, 1814
1	MS 0889	M.E. Chandross, 1814
1	MS 0889	J.M. Lane, 1814
1	MS 1315	M.J. Stevens, 1814
1	MS 1316	S.J. Plimpton, 1426
1	MS 1318	R.J. Hoekstra, 1426
1	MS 1318	R. Hooper, 1445
1	MS 1318	R.P. Pawlowski, 1444
1	MS 1320	R.B. Lehoucq, 1444
1	MS 1320	M.L. Parks, 1444
1	MS 1322	J.B. Aidun, 1425
1	MS 1322	P.S. Crozier, 1426
1	MS 1322	S. Jayaraman, 1425
1	MS 1322	R.P. Muller, 1425
1	MS 1322	P.A. Schultz, 1425
1	MS 1322	T.-R. Shan, 1425
1	MS 1322	A.P. Thompson, 1425
1	MS 1411	S.F. Foiles, 1814
1	MS 1411	A.L. Frischknecht, 1814
1	MS 1411	L.M. Hall, 1814
1	MS 1411	E.A. Holm, 1814
1	MS 1411	R.A. Roach, 1823
1	MS 1411	G. Tucker, 1814
1	MS 1411	C.R. Weinberger, 1814
1	MS 9042	J.W. Foulk III, 8246
1	MS 9042	C.D. Moen, 8246
1	MS 9042	J.T. Ostien, 8246
1	MS 9051	B.J. Debusschere, 8351
1	MS 9054	R.W. Carling, 8300
1	MS 9153	R.G. Miller, 8200
1	MS 9154	M.E. Gonzales, 8240
1	MS 9161	R.H. Nilson, 8365
1	MS 9402	D. Ward, 8131
1	MS 9403	L.M. Hale, 8246

1	MS 9403	B.M. Wong, 8223
1	MS 9404	R.E. Jones, 8246
1	MS 9404	N.R. Moody, 8222
1	MS 9404	A. Mota, 8246
1	MS 9404	X. Zhou, 8246
1	MS 9404	J.A. Zimmerman, 8246
1	MS 9409	J. Deng, 8365
1	MS 9409	L.C. Erickson, 8365
1	MS 9409	N.R. Fornaciari, 8365
1	MS 9409	J. Lee, 8365
1	MS 9409	K.K. Mandadapu, 8365
1	MS 9409	J.A. Templeton, 8365
1	MS 9409	G.J. Wagner, 8365
1	MS 0899	Technical Library, 9536 (electronic copy)
1	MS 0359	D. Chavez, LDRD Office, 1911

Sandia National Laboratories