A Tutorial on Advanced Dynamic Monte Carlo Methods for Systems with Discrete State Spaces

M. A. Novotny

School of Computational Science and Information Technology,
Florida State University, Tallahassee, FL 32306-4120, U.S.A.

Permanent Address: Dept. of Physics and Astronomy
Mississippi State University
Mississippi State, MS 39762, USA

September 11, 2001

Abstract

Advanced algorithms are necessary to obtain faster-than-real-time dynamic simulations in a number of different physical problems that are characterized by widely disparate time scales. Recent advanced dynamic Monte Carlo algorithms that preserve the dynamics of the model are described. These include the n-fold way algorithm, the Monte Carlo with Absorbing Markov Chains (MCAMC) algorithm, and the Projective Dynamics (PD) algorithm. To demonstrate the use of these algorithms, they are applied to some simplified models of dynamic physical systems. The models studied include a model for ion motion through a pore such as a biological ion channel and the metastable decay of the ferromagnetic Ising model. Non-trivial parallelization issues for these dynamic algorithms, which are in the class of parallel discrete event simulations, are discussed. Efforts are made to keep the article at an elementary level by concentrating on a simple model in each case that illustrates the use of the advanced dynamic Monte Carlo algorithm.

1.0 Introduction

Classical Monte Carlo methods have been used to study a wide variety of different systems since they were first proposed by Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller [1]. The Monte Carlo method can be viewed as a method to perform multi-dimensional integrals [2]. For example, to obtain static critical exponents [3] for model systems such as Ising, Potts, and classical Heisenberg models, Monte Carlo simulations are used to calculate multidimensional sums or integrals [4, 5]. A significant number of advanced Monte Carlo algorithms have been developed for static critical phenomena. These include cluster algorithms [6–8], multicanonical algorithms [9–15], and histogram reweighting methods [16–18]

In its simplest form, the Monte Carlo algorithm is like a movie with each frame a three-step process. First, generate a (pseudo-)random number. In this work we will only use random numbers uniformly distributed between 0 and 1. Second, choose a trial move from the current state to a new state. Third, accept or reject this trial move depending on the current random number and an acceptance probability for the trial move. The sums or integrals to be calculated are then updated using the new frame, and the procedure is iterated repeatedly.

Calculations of static behaviors, such as multi-dimensional integrals or static critical exponents, allow a wide variety of trial moves. This includes, for example, cluster and multicanonical algorithms. This is so because in static calculations only the final sum (or integral) is of interest. The particular trial moves have no physical meaning. Such freedom of trial moves no longer exists if the Monte Carlo algorithm is applied to a dynamic problem where the physical meaning of the trial moves is an essential part of the model. Fortunately, some advanced Monte Carlo algorithms exist to make many of these calculations efficient without changing the meaning of the trial moves. It is these advanced algorithms that will be described and applied in this article.

2.0 Standard Dynamic Monte Carlo Methods

It is also possible to ascribe physical meaning to certain trial moves. Consider a one-dimensional random walker on a lattice. At each tick of a clock the walker has a certain probability to move to the left, the right, or to stay where it is. In this case, the actual Monte Carlo trial (move left, move right, or stay) has a physical meaning. Now imagine that the probabilities of moving to the left or to the right are small. Then, most of the time the walker stays where it is, and it will require a large amount of computer time before the walker moves a substantial distance. The advanced dynamic algorithms for this problem, described in Sec. 4, keep this physical dynamic intact, but decrease the amount of computer time required for the walker to reach a specified location.

Consider the ferromagnetic Ising model on a regular lattice with periodic boundary conditions. Each site of the lattice has a spin σ_i=$\pm$1. From the Hamiltonian, the energy of a particular spin configuration is given by

$$\mathcal{E} = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j - H \sum_i \sigma_i, \tag{1}$$

where the coupling constant J>0 and the external magnetic field is H. The first sum of

Eq. (1) is over all nearest-neighbor spin pairs and the second sum is over all N Ising spins. One can use the simple Monte Carlo procedure described in this section or advanced methods such as a cluster method, to obtain the static critical behavior, such as the internal energy, specific heat, susceptibility, or order parameter at a particular temperature T and field H. A particularly simple trial move involves a single spin flip, i.e. letting $\sigma_i \rightarrow -\sigma_i$. However, one can also consider these Monte Carlo trial moves to be due to the interaction of the Ising spins with a heat bath. Then the dynamics of the simulation is of physical relevance, and methods such as the cluster method *cannot* be used since they change the dynamics.

The dynamics for the kinetic Ising model can be derived using a Master equation approach [19, 20, 21]. In 1977 Martin [22] considered a lattice of quantum spin $\frac{1}{2}$ particles, each coupled to their own heat bath, and showed that under certain assumptions the same dynamic as for the kinetic Ising model was obtained. This dynamic is:

- Randomly choose an Ising spin. If there are N Ising spins, the probability of choosing each spin is $1/N$.
- Calculate a random number $\bar{r}$.
- Calculate (or look up in a stored table) the energy $E_{\rm old}$ of the current Ising configuration and $E_{\rm new}$ of the configuration with the chosen Ising spin flipped.
- Change to the new configuration, i.e. flip the chosen Ising spin, if

$$\bar{r} \leq \exp(-E_{\rm new}/k_{\rm B}T)/\left[\exp(-E_{\rm old}/k_{\rm B}T) + \exp(-E_{\rm new}/k_{\rm B}T)\right] \ . \qquad (2)$$

- Increment the time by one Monte Carlo step (mcs), i.e. by one spin-flip attempt.

This Monte Carlo dynamic has physical meaning. Here T is the temperature and $k_{\rm B}$ is Boltzmann's constant. Furthermore, the time scale for a single spin-flip trial is set by the heat bath. This means that for Ising spins interacting with phonons, one spin-flip attempt per spin (one Monte Carlo step per spin, defined to be one MCSS) should correspond to a time on the order of an inverse phonon frequency, roughly 10^{-13} seconds. Thus, to simulate this simple model for a time of one second would take 10^{13} Monte Carlo steps per spin. If you could get your computer to perform one spin-flip attempt for each CPU clock tick (roughly 10^{-9} sec), then a one-second simulation would take about three hours of CPU time per spin! Your simulation would take $10^4 N$ times longer than physical time. To simulate metastable decay for the Ising model, as discussed below in Sec. 4.1, one is interested in time scales of years to decades. To simulate your Ising model for a N=100 spin system would take about 12 days of CPU time to simulate for one second, about 2 years of CPU time to simulate for one minute, and about one century of CPU time to simulate for one hour. These CPU times are much longer than I can wait for computer results, and we are interested in simulating much larger lattices and much longer times. The sections below describe advanced dynamic algorithms that allow these simulations to be performed in a reasonable amount of CPU time *without* changing the underlying dynamic. The goal is to obtain faster-than-real-time dynamic simulations.

The dynamic algorithm described above, using Eq. (2) for the Ising model, is called the Glauber dynamic. It was introduced by Glauber [19] to study the dynamics of a one-dimensional Ising chain. For higher-spin models, and sometimes also for the Ising model,

this type of dynamic is called a heat-bath dynamic. Another popular choice [4, 5] for the flip probability for Ising spins is the Metropolis dynamic [1], which always flips the spin if the energy of the new configuration is lower than that of the old. Thus the spin flips if

$$\bar{r} \leq \min\left\{1, \exp\left[(E_{\rm old} - E_{\rm new})/k_{\rm B}T\right]\right\} \ . \tag{3}$$

The dynamic that should be used in a particular dynamic simulation is the one that has been derived from the underlying physical system. Both of these dynamics satisfy detailed balance, and hence will obtain the correct statics for the Ising model.

Note that the dynamic involves a *random* choice of the spin on which a spin flip will be attempted. Performing spin-flip attempts sequentially on the lattice will lead to a different dynamic. It has been shown in nucleation-and-growth studies of the lifetime of the metastable state in an Ising model that the use of sequential instead of random updates can lead to different physical results. In particular, the pre-factor for the nucleation rate is different [23, 24].

3.0 A Brief Review of Absorbing Markov Chains

Many of the advanced algorithms described in the following sections will use absorbing Markov chains. A brief introduction to some of the important features of absorbing Markov chains is presented here.

Consider an absorbing Markov chain with s transient states and r absorbing states. The system starts in one of the s transient states, and remains in the transient subspace of the s transient states until it is absorbed into one of the r absorbing states. We will use the mathematician's notation of the current state vector being operated on by a matrix from the right. (As opposed to the normal physicist's convention.) This will allow the reader to make easy contact with the literature on absorbing Markov chains, for example [25, 26].

The transition matrix defining an absorbing Markov chain has the form

$$\mathbf{M}_{(r+s)\times(r+s)} = \begin{pmatrix} \mathbf{I}_{r\times r} & \mathbf{0}_{r\times s} \\ \mathbf{R}_{s\times r} & \mathbf{T}_{s\times s} \end{pmatrix} \tag{4}$$

where the sizes of the matrices have been explicitly shown. The matrix $\mathbf{I}$ is an identity matrix, the matrix $\mathbf{0}$ is a matrix with all zero elements, the matrix $\mathbf{T}$ is called the transient matrix, and the matrix $\mathbf{R}$ is called the recurrent matrix. $\mathbf{M}$ is a Markov matrix, i.e., each of its row sums equal one, since at each clock tick the system must jump to a new state or remain in the current state.

The matrix that governs the motion of the system after m time steps (or m clock ticks) is given by matrix multiplication to be

$$\mathbf{M}^m_{(r+s)\times(r+s)} = \begin{pmatrix} \mathbf{I}_{r\times r} & \mathbf{0}_{r\times s} \\ (\mathbf{I} + \mathbf{T} + \cdots + \mathbf{T}^{m-1})_{s\times s}\mathbf{R}_{s\times r} & \mathbf{T}^m_{s\times s} \end{pmatrix} \ . \tag{5}$$

The system must initially be in the transient subspace, so the s-state initial vector is $\vec{v}_{\rm I}^{\rm T}$ and the total initial vector for the matrix $\mathbf{M}$ is given by $\begin{pmatrix} \vec{0}^{\rm T} & \vec{v}_{\rm I}^{\rm T} \end{pmatrix}$ where the vector of

zeros has r elements. (The superscript T denotes the transponse.) Applying $\mathbf{M}^m$ to this initial vector gives the $(r+s)$-dimensional vector

$$(\,\vec{0}^{\rm T}\quad \vec{v}_{\rm I}^{\rm T}\,)\,\mathbf{M}^m = (\,\vec{v}_{\rm I}^{\rm T}\,(\mathbf{I}+\mathbf{T}+\cdots+\mathbf{T}^{m-1})\,\mathbf{R}\quad \vec{v}_{\rm I}^{\rm T}\mathbf{T}^m\,)\ . \tag{6}$$

Introduce the vector $\vec{e}$ with all elements equal to unity. Then the probability of still being in the space of the s transient states after m time steps is given by

$$p_{\text{still in transient subspace}} = \vec{v}_{\rm I}^{\rm T}\mathbf{T}^m\vec{e}\,. \tag{7}$$

The probability of the state having exited to each of the r absorbing states after m time steps is given by the elements of the vector

$$\vec{p}^{\rm T}_{\text{absorption after } m \text{ time steps}} = \vec{v}_{\rm I}^{\rm T}\left(\mathbf{I}+\mathbf{T}+\cdots+\mathbf{T}^{m-1}\right)\mathbf{R}\,. \tag{8}$$

The probability of having exited to each of the r absorbing states after m time steps is the sum of the probabilities of having exited in the first time step (which is the term $\mathbf{IR}$) plus the probability of having exited after the second time step (the term $\mathbf{TR}$) plus the probability of having exited in time steps 3, 4, $\cdots$, m. Thus the probability that the system exits to each of the r absorbing states, *given* that it exits at time step m, is given by the elements of the vector

$$\vec{p}^{\rm T}_{\text{absorption given exit at time } m} = \vec{q} = \frac{\vec{v}_{\rm I}^{\rm T}\mathbf{T}^{m-1}\mathbf{R}}{\vec{v}_{\rm I}^{\rm T}\mathbf{T}^{m-1}\mathbf{R}\vec{e}}\,. \tag{9}$$

Equations (7) and (9) are the equations needed to perform Monte Carlo with Absorbing Markov Chain calculations.

In the absorbing Markov chain literature the $s\times s$ matrix

$$\mathbf{N} = (\mathbf{I}-\mathbf{T})^{-1} = \mathbf{I}+\mathbf{T}+\mathbf{T}^2+\cdots\ , \tag{10}$$

called the fundamental matrix, is introduced. Then the probability that the system ends up in a particular absorbing state is given by the elements of the vector

$$\vec{p}^{\rm T}_{\text{absorbed into particular state}} = \vec{v}_{\rm I}^{\rm T}\mathbf{N}\mathbf{R}\,. \tag{11}$$

The average lifetime to exit from the transient subspace is given by

$$\langle\tau\rangle = \vec{v}_{\rm I}^{\rm T}\mathbf{N}\vec{e}\,. \tag{12}$$

The units for $\langle\tau\rangle$ are the time units of the clock, with one attempted move every tick of the clock. This can be derived using the fact that the probability to be absorbed at time step m is

$$p_{\text{absorbed at time } m} = \vec{v}_{\rm I}^{\rm T}\left(\mathbf{T}^{m-1}-\mathbf{T}^m\right)\vec{e}\,, \tag{13}$$

so that

$$\begin{aligned}\langle\tau\rangle &= \sum_{m=1}^{\infty} m\vec{v}_{\rm I}^{\rm T}\left(\mathbf{T}^{m-1}-\mathbf{T}^m\right)\vec{e}\\ &= \vec{v}_{\rm I}^{\rm T}\left(\mathbf{I}+\mathbf{T}^1+\mathbf{T}^2+\cdots\right)\vec{e}\\ &= \vec{v}_{\rm I}^{\rm T}\left(\mathbf{I}-\mathbf{T}\right)^{-1}\vec{e}\ =\ \vec{v}_{\rm I}^{\rm T}\mathbf{N}\vec{e}\,.\end{aligned} \tag{14}$$

The fundamental matrix can also be used to calculate the moments of the lifetime for the system, that is the moments of the exit time probabilities. These are given by [25]

$$\left\langle \tau^2 \right\rangle = \vec{v}_{\rm I}^{\rm T} \left(2\mathbf{N}^2 - \mathbf{N} \right) \vec{e} \tag{15}$$

and

$$\left\langle \tau^3 \right\rangle = \vec{v}_{\rm I}^{\rm T} \left(6\mathbf{N}^3 - 6\mathbf{N}^2 + \mathbf{N} \right) \vec{e} \tag{16}$$

and

$$\left\langle \tau^4 \right\rangle = \vec{v}_{\rm I}^{\rm T} \left(24\mathbf{N}^4 - 36\mathbf{N}^3 + 14\mathbf{N}^2 - \mathbf{N} \right) \vec{e}\,. \tag{17}$$

4.0 Particle Motion Through a Pore

In order to illustrate the advanced dynamic Monte Carlo algorithms, we introduce an extremely simple one-dimensional walk. This one-dimensional walk can be viewed as a first step in simulating motion of ions through biological ionic channels [27–32]. These ion channels control the motion of ions through cell membranes, and they are consequently of importance in understanding cell function and drug delivery to cells. The model is kept simple to allow us to introduce the dynamic Monte Carlo algorithms in a transparent fashion.

The model is a one-dimensional walk on a lattice. Our particle (the ion) starts at lattice site zero (on the left) and performs a random walk until it is absorbed at the right-most lattice site. Each lattice site i has an energy E_i. We will study a 16-site lattice (16 transient states and one absorbing state) with energies shown in Fig. 1. The energies for lattice sites 0 through 16 are (in dimensionless units) $-\frac{1}{2}$, 1, 1, 0, $\frac{1}{2}$, 0, 0, $\frac{1}{2}$, 0, 0, $\frac{1}{2}$, $\frac{1}{2}$, $\frac{3}{4}$, 0, $\frac{1}{2}$, 0, -1. The dynamic we introduce is that at each time step the walker randomly picks whether it will try to perform a move to the left or to the right. The probability of actually performing this move is given by

$$p_{\rm move} = \frac{\exp(-E_{i\pm1}/k_{\rm B}T)}{\exp(-E_i/k_{\rm B}T) + \exp(-E_{i\pm1}/k_{\rm B}T)}\,, \tag{18}$$

where the + (−) sign is for the attempted move to the right (left). When the walker is in lattice site 0, the move to the right is always attempted. We are interested in the average lifetime $\langle \tau \rangle$ for a walker starting at lattice site 0 before it is absorbed at lattice site 16.

4.1 Ion Channel Standard Monte Carlo

The standard Monte Carlo dynamic for this problem is extremely simple to program. Each algorithmic step uses two uniformly distributed random numbers, $\tilde{r}$ and $\bar{r}$. The algorithm is:

- Place the walker at lattice site 0.
- Set the time t to zero.
- Perform the following steps until the walker is absorbed when it reaches lattice site 16.

- Set the time t to t=t+1.
- Calculate two random numbers, $\tilde{r}$ and $\bar{r}$.
- If the walker is not at lattice site 0 and $\tilde{r}$ is less than $\frac{1}{2}$, then attempt a move to the left. Otherwise attempt a move to the right.
- Calculate (or look up in a stored table) the probability of moving, $p_{\rm move}$ from Eq. (18).
- Move the walker if $\bar{r} \leq p_{\rm move}$.

This algorithm is executed K times, and the statistics for the mean lifetime are accumulated. The average lifetime, calculated for K=1000 escapes, is shown in Fig. 2. The time unit is Monte Carlo steps (mcs), so the lifetime corresponds to the average number of clock ticks before the walker reaches lattice site 16.

4.2 Ion Channel n-fold Way Monte Carlo (Event Driven Simulation)

As can be seen from Fig. 2, the average lifetime at low temperatures can be very long. The time unit is Monte Carlo steps, mcs, for our single walker. At T=0.05 (in dimensionless units with $k_{\rm B}$=1) the value for τ is about 3×10^{13} mcs. If each mcs took a nanosecond of computer time (a very fast implementation on an extremely fast computer), the computational time required for 1000 escapes would require 3×10^{7} seconds of CPU time. This corresponds to about one year of computer time! (For my implementation of the standard algorithm on a Cray SV1 supercomputer this calculation would have taken about 32 years!) I did not spend this amount of computer time on the calculation. Rather, I utilized an advanced algorithm to perform my dynamic simulation at this temperature in 35 minutes on a Cray SV1 computer.

The first advanced algorithm I used corresponds to the n-fold way algorithm introduced by Bortz, Kalos, and Lebowitz in 1975 [33], as implemented in discrete time [34]. This and related similar schemes have been put forward and utilized in the simulations of many problems. Just a few of these include crystal growth [35, 36], renormalization group methods for obtaining critical exponents [37], growth of nanostructures [38, 39], dynamics of simple models of glasses [40, 41], lattice models for protein folding [42], thermal decay of recorded information [43], Potts models for domain growth [44], catalytic surface reactions [45, 46], aging on long time scales [47], ground state calculations of spin glass models [48], crossover phenomena in the anisotropic Ising model [49], domain boundaries in anisotropic Ising models [50], multidimensional minimization of functions via simulated annealing [51], dynamic recrystallization [52], and collective surface diffusion [53].

These algorithms are also called event-driven algorithms in the discrete-event simulation community [54]. A discrete-event simulation is one in which the state vector of the system changes discontinuously at particular (possibly random) time intervals. In this case, the time intervals correspond to the times at which the walker jumps from one state to another (*not* the times at which a step is attempted but then rejected). The idea behind event-driven simulations is to change the state of the system in one algorithmic step, and to increment the time by the time required to make that transition. This is where the absorbing Markov chain methodology is useful.

Let the transient state for an absorbing Markov chain be the current state, and the two absorbing states be the walk to the left or to the right. Then the Markov matrix is

$$\mathbf{M}_{\text{ion }1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ p_{\text{left}} & p_{\text{right}} & 1 - p_{\text{left}} - p_{\text{right}} \end{pmatrix} \tag{19}$$

where p_{right} is the probability of jumping to the right and p_{left} is the probability of jumping to the left. These probabilities are given by $p_{\text{move}}/2$ [from Eq. (18)] where the factor of $1/2$ is needed because a move attempt to the left or right is chosen at random. If the walker is at lattice site 0, then p_{left}=0 and p_{right}=p_{move}. Since the walker is in the current state, the initial state vector is $\vec{v}_{\text{I}}^{\text{T}} = (\,0 \quad 0 \quad 1\,)$.

Each algorithmic step again uses two uniformly distributed random numbers, $\tilde{r}$ and $\bar{r}$. The algorithm to repeat for K escapes is:

- Place the walker at lattice site 0.
- Set the time clock to t=0.
- Perform the following steps until the walker is absorbed at lattice site 16.
  - Calculate two random numbers, $\tilde{r}$ and $\bar{r}$.
  - Calculate (or look up from tables) the two probabilities p_{right} and p_{left}.
  - Calculate the time m to exit from the current state m=$\left\lfloor \frac{\ln(\bar{r})}{\ln(1-p_{\text{left}}-p_{\text{right}})} \right\rfloor + 1$, where $\lfloor z \rfloor$ is the integer part of z.
  - Move the walker to the right if $\tilde{r} \leq \frac{p_{\text{right}}}{p_{\text{right}}+p_{\text{left}}}$. Otherwise move the walker to the left.
  - Increment the time to t+m.

In a primitive way, if the probability to move is very small, say 0.001 per time step, then instead of increasing the time by one unit for each of the mostly unsuccessful attempts to move, we always move the particle and increase the time by on average 1000 units, thus saving computing effort by a factor 1000.

A number of remarks are required. The number of time steps m to exit the current state is given by Eq. (7). For this simple case where there is only one transient state (s=1), the matrix $\mathbf{T}$=T_{11}=1−p_{left}−p_{right} is a scalar, and Eq. (7) gives the time of exit from the transient state to be $T_{11}^m < \bar{r} \leq T_{11}^{m-1}$. This reduces to the equation for m in terms of the natural logarithms. Also, since $\mathbf{T}$ is a scalar, there is no m dependence on the probabilities to exit to the left or right state since from Eq. (9) the factor of T_{11}^{m-1} cancels due to the normalization.

Using the event-driven simulation, the time to measure K=1000 escapes of the algorithm at T=0.05 would be 10^3 minutes, or about 17 hours of Cray SV1 time. For this temperature the event-driven simulation is about one million times faster than the standard Monte Carlo algorithm!

4.3 Ion Channel s=2 MCAMC

At low temperatures, even the event-driven simulation can take a large amount of computer time. This is principally due to two portions of the energy landscape, namely the lattice site pairs (5,6) and (8,9) in Fig. 1. In these situations it is advantageous to use more than one state in the transient subspace. We will describe the case of two transient states, so a s=2 Monte Carlo with Absorbing Markov Chains (MCAMC) will be used whenever a walker steps into the pair (5,6) or the pair (8,9) in Fig. 1. At low temperature the probability of stepping out of this pair is exponentially small compared to the probability of stepping to the other site of the pair. Consequently, the walker performs many steps within the pair before it eventually steps out of the pair of lattice sites. We have chosen the energies of these lattice pairs to make the s=2 MCAMC algorithm easy to describe. The general case can easily be constructed using the absorbing Markov chain methodology of Sec. 3.0. In our case, the sites of the pair (5,6) or (8,9) have the same energy (chosen to be zero). Also, the bordering lattice sites 4 and 7 (or 7 and 10) have the same energy. We will use the s=1 MCAMC, the event-driven simulation of Sec. 4.2, unless the walker has just stepped onto a lattice site pair (5,6) or (8,9). Let x be the probability of stepping from site 5 to 4 in one time step, so

$$x = \frac{1}{2}\frac{\exp(-E_4/k_{\rm B}T)}{\exp(-E_5/k_{\rm B}T)+\exp(-E_4/k_{\rm B}T)} \; . \tag{20}$$

The probabilities of stepping from 6 to 7 or from 8 to 7 or from 9 to 10 in one time step are also equal to x. The probability of stepping from 5 to 6 in one time step is $\frac{1}{4}$, which is also the probability of stepping from 6 to 5, 8 to 9, or 9 to 8.

The Markov matrix for this s=2 transition is

$$\mathbf{M}_{\text{ion 2}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ x & 0 & 1-x-\frac{1}{4} & \frac{1}{4} \\ 0 & x & \frac{1}{4} & 1-x-\frac{1}{4} \end{pmatrix} \; . \tag{21}$$

In our case we have

$$\mathbf{T}_2 \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1-x-\frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & 1-x-\frac{1}{4} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = [1-x] \begin{pmatrix} 1 \\ 1 \end{pmatrix} \; . \tag{22}$$

Thus the time m to exit from the pair of lattice sites is given by

$$m = \left\lfloor \frac{\ln(\bar{r})}{\ln(1-x)} \right\rfloor + 1 \; . \tag{23}$$

To calculate the probability of exiting to either side from a lattice site pair, for the normalization use

$$\vec{v}_{\rm I}^{\rm T} \mathbf{T}_2^{m-1} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = (1-x)^{m-1} \; . \tag{24}$$

Also use the spectral decomposition of $\mathbf{T}_2$, which is

$$\mathbf{T}_2^{m-1} = \frac{1}{2}(1-x)^{m-1} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \end{pmatrix} + \frac{1}{2}\left(\frac{1}{2}-x\right)^{m-1} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \begin{pmatrix} 1 & -1 \end{pmatrix} \; . \tag{25}$$

This gives from Eq. (9) that the probability of exiting the lattice pair back to the same lattice point the walker entered from is

$$p_{\text{exit back}} = \frac{1}{2}\left[1 + \left(\frac{\frac{1}{2}-x}{1-x}\right)^{m-1}\right] , \tag{26}$$

and the probability of exiting to the lattice site opposite to the site it entered the lattice pair from is

$$p_{\text{exit other}} = \frac{1}{2}\left[1 - \left(\frac{\frac{1}{2}-x}{1-x}\right)^{m-1}\right] . \tag{27}$$

The algorithm to use s=2 MCAMC for these two lattice pairs is:

- Place the walker at lattice site 0.
- Set the time clock to t=0.
- Perform the following steps until the walker is absorbed at lattice site 16.
  - Calculate two random numbers, $\tilde{r}$ and $\bar{r}$.
  - If the walker has not just stepped onto lattice site 5, 6, 8, or 9 then use the rejection-free (s=1 MCAMC) algorithm described in Sec. 4.2:
    - Calculate (or look up from tables) the two probabilities p_{right} and p_{left}.
    - Calculate the time m to exit from the current state $m=\left\lfloor \frac{\ln(\bar{r})}{\ln(1-p_{\text{left}}-p_{\text{right}})} \right\rfloor + 1$.
    - Move the walker to the right if $\tilde{r} \leq \frac{p_{\text{right}}}{p_{\text{right}}+p_{\text{left}}}$, otherwise move the walker to the left.
  - If the walker has just stepped onto lattice site 5, 6, 8, or 9 then use s=2 MCAMC:
    - Calculate (or look up from tables) the probability x from Eq. (20).
    - Calculate the time m to exit the pair of lattice points using Eq. (23).
    - Calculate (or look up from tables) $p_{\text{exit back}}$ from Eq. (26).
    - If $p_{\text{exit back}} < \bar{r}$ then move the walker to the site it entered the lattice pair (5,6) or (8,9) from. Otherwise move the walker to the other exit lattice site of the pair.
  - Increment the time to $t+m$.

Using this s=2 MCAMC algorithm at T=0.05 for K=1000 escapes takes about 0.35 minutes of CPU time. This is about 5000 times less than the s=1 MCAMC (n-fold way) algorithm, and about 10^{10} times faster than the traditional Monte Carlo algorithm!

The timings for all three algorithms are shown in Fig. 3. The timing for the standard Monte Carlo algorithm is proportional to the average lifetime shown in Fig. 2. Figure 3(b) shows the crossovers between the timings for the various algorithms. The s=2 MCAMC turns out to be always at least as fast as the s=1 MCAMC. This is not the usual case, and is due to the very simple form for $\mathbf{T}_2$. The standard Monte Carlo is faster at high temperatures, with the s=2 MCAMC being faster at temperatures below about $T=\frac{1}{2}$.

4.4 Ion Channel Projective Dynamics

This simple model can actually be solved analytically by noting that the transient matrix is tridiagonal and has the form

$$\mathbf{T}_{16} = \begin{pmatrix} 1-s_{15}-g_{15} & s_{15} & 0 & 0 & 0 & 0 \\ g_{14} & 1-s_{14}-g_{14} & s_{14} & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & g_2 & 1-s_2-g_2 & s_2 & 0 \\ 0 & 0 & 0 & g_1 & 1-s_1-g_1 & s_1 \\ 0 & 0 & 0 & 0 & g_0 & 1-g_0 \end{pmatrix} \tag{28}$$

where g_i is the single-time-step transition probability from i to $i{+}1$ (the 'growning probability') and s_i is the single-time-step transition probability from i to $i{-}1$ (the shrinking probability). In other words, we have changed notation slightly and g_i are the appropriate p_{right} and s_i are the appropriate p_{left}. The walker starts in state 0, so the initial vector is $\vec{v}_{\text{I}}^{\text{T}} = (\,0 \quad 0 \quad \cdots \quad 0 \quad 1\,)$. The recurrent matrix is given by

$$\mathbf{R}_{16} = \begin{pmatrix} g_{15} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{29}$$

since the walker is absorbed when it steps onto state 16 from state 15.

Introduce the average time $h(i)$ spent in state i during this absorbing Markov process. Then the average lifetime is the sum of these residence times,

$$\langle \tau \rangle = \sum_{i=0}^{15} h(i) \; . \tag{30}$$

An iterative solution for the $h(i)$ terms can be found by considering the number of times during K escapes that a walker passes an imaginary midpoint between state i and $i{-}1$. Since there are K escapes and each escape must pass through each mid-point, there will be K more walks to lattice site i than to lattice site $i{-}1$. During K walks the average time spent in state i is $h(i)\,K$. Thus

$$g_{i-1} h(i-1) K = K + s_i h(i) K \; . \tag{31}$$

For site 16 there is no shrinking probability and $g_{15} h(15) K = K$ or

$$h(15) = \frac{1}{g_{15}} \; . \tag{32}$$

Solving for the other $h(i)$ values gives

$$h(i-1) = \frac{1 + s_i h(i)}{g_{i-1}} \; . \tag{33}$$

Thus we have a closed form for the lifetime for this simple process. This closed form is shown as the solid line in Fig. 2. Note that the time unit is clock ticks, i.e. mcs (Monte Carlo trial steps).

The model we used for the ion channel has only one entrance of the ion (from the left). A more realistic ion channel model has the possibility of entering and exiting from both ends of the channel [28]. For our simple model this only means that the state with no ions in the channel needs to be included together with the possibility of entering either end of the channel from this state and being absorbed by exiting from either end. Again, the absorbing Markov chain is tridiagonal. It now has two absorbing states. In addition to τ, other quantities of interest include the average time to exit one end, given that it entered from a particular end. These times can again be calculated using the methodology of absorbing Markov chains.

In the sections below, we will utilize this formulation in the projective dynamics method for the escape of more complicated models from the metastable state [55]. In the majority of situations there is no analytic expression, but the same formulation can be used, see Sec. 5.4.

5.0 Dynamic Ising Simulations

The simple model for ion channels introduced above illustrates many of the advanced Monte Carlo methods, but it is too simple to completely illustrate their use. We now describe advanced dynamic Monte Carlo simulations for a more complicated but still simple model, the square-lattice Ising ferromagnet described in Sec. 2.0. This model has N Ising spins σ which can be either up $(+1)$ or down (-1) at each lattice site. We will consider the model with periodic boundary conditions on an $L\times L$ lattice, so $N=L^2$. We again are interested in the average lifetime $\langle\tau\rangle$ of the metastable state. The time units that correspond to physical time are Monte Carlo steps per spin, MCSS, as described in Sec. 2.0. The system starts with all spins up, and has an applied magnetic field H which is down ($H\leq 0$). We consider only ferromagnetic nearest-neighbor interactions of strength J. The energy is given by Eq. (1). The total magnetization of the system is

$$M = \sum_{i}^{L^2} \sigma_{i=1} \; . \tag{34}$$

This model has a critical temperature $k_{\rm B}T_{\rm c}= 2J/\ln(1+\sqrt{2})\approx 2.269\cdots J$. We want to measure the average lifetime $\langle\tau\rangle$ for $T<T_{\rm c}$ for the system to progress from the initial state with all spins up to a state with an equal number of spins up and down, i.e. to $M=0$. In this way we can study the dynamics of the model below the critical temperature, where the metastable decay of the model is dominated by nucleation and growth [20, 21, 56].

Even for this simple model the understanding of $\langle\tau\rangle$ is complicated. Two recent reviews of nucleation and growth analysis of metastable decay in the Ising model are presented in [24, 57]. A review of earlier works on metastable decay is given in Ref. [58]. More complicated Ising models, including other boundary conditions [59], heterogeneous nucleation [60], and an approximate demagnetizing field [61] have been studied as well. The Ising model with periodic boundary conditions has four length scales [23, 24]. The lattice spacing a and the lattice size L are two of them. Another length scale is the radius of the critical droplet, $R_{\rm c}(T,H)$, which is a function of temperature and applied field. Droplets with radii smaller than $R_{\rm c}$ are subcritical and will most likely shrink,

while droplets with radii larger than R_c are supercritical and will most likely grow. The remaining length scale has been called $R_0(T,H)$ [23], and it also depends on T and the strength of the applied field. This can be regarded as the average size to which a supercritical droplet will grow before it encounters another supercritical droplet. Note that $R_0 > R_c$ always.

These four length scales give rise to different types of decay mechanisms for the metastable state, depending on their inter-relationship. Only if R_c and R_0 are larger than a is the droplet nucleation picture applicable, so when $R_c < a$ the system is in a strong-field (SF) regime where there is no long-lived metastable state. For weaker fields and for large lattices, $a \ll R_c < R_0 \ll L$. Then many supercritical droplets form during the decay to $M=0$. This is called the multi-droplet (MD) regime, and it is well described by the well-known Kolmogorov, Johnson-Mehl, Avrami theory [62, 63]. For smaller lattices, where $a < R_c \ll L \ll R_0$, only one supercritical droplet forms and grows to take the system to $M=0$, so this regime is called the single-droplet (SD) regime. For the square-lattice Ising model the crossovers between these various regimes are shown in Fig. 4 for different values of L. Another regime, where $L < R_c$ is called the coexistence regime. It is not shown in the figure.

The cross-overs between the different decay regimes and how they depend on H and T are of central importance [23, 24]. A conservative estimate for the cross-over between the SF and MD regime is where $R_c = a/2$. This is shown as the heavy dashed curve in Fig. 4, and is independent of L. The cross-over between the MD and SD regime can be estimated by noting that the standard deviation of the lifetime, $\Delta\tau = \sqrt{\langle\tau^2\rangle - \langle\tau\rangle^2}$, approximately equals $\langle\tau\rangle$ in the SD regime and is much smaller than τ in the MD regime. One consequence of this is that the measurement of $\langle\tau\rangle$ is self-averaging in the MD regime, but is very far from self-averaging in the SD regime. Hence, to measure $\langle\tau\rangle$ in the MD regime only a few realizations are required, and the $K=1000$ escapes used here are more than adequate. Conversely, in the SD regime the $K=1000$ escapes we average over are barely sufficient since the distribution of τ has an extremely long tail. The estimate for the cross-over between the MD and SD regimes, called the dynamic spinodal [23, 64], is given by the point where $\Delta\tau = \langle\tau\rangle/2$. The location of the dynamic spinodal depends on L, and is shown for various values of L in Fig. 4.

For low temperatures and $|H| > 2J$, a single overturned spin is the nucleating droplet [65] with an energy barrier given by $\Gamma(H,J) = 8J - 2|H|$. The dynamic spinodal (DSp) for the random-update dynamic in Sec. 5.1 can be found [66] by equating the average growth time to the average nucleation time to give

$$\frac{(4 - H_{\mathrm{DSp}})}{k_{\mathrm{B}}T} = \frac{3}{2}\ln(L) - 0.82 \; , \tag{35}$$

where the last term is the only fitting parameter and was chosen to provide agreement for values of L between 8 and 240. The factor of $\frac{3}{2}$ changes if the dynamic changes, for example it is 1 for sequential updates [66]. For a fixed field the dynamic spinodal approaches the $T=0$ axis as

$$k_{\mathrm{B}}T_{\mathrm{DSp}} = \frac{4 - |H|}{\frac{3}{2}\ln(L) - 0.82} \; , \tag{36}$$

and without adequate finite-size scaling such a slow approach toward T=0 as a function of L could easily be misinterpreted as a phase transition.

For low temperatures, as seen in Fig. 4, the single-droplet regime is dominant. However for strong fields the discreteness of the lattice becomes important and the nucleating droplets are no longer circular. For arbitrarily low temperatures and $|H|<2J$ with a fixed lattice size, the metastable decay is dominated by the SD regime, and the predicted lifetime is given by [65]

$$k_{\rm B}T\ln(\langle\tau_{\rm low}\rangle) = \Gamma(H,J) = 8J\ell_c - 2|H|(\ell_c^2 - \ell_c + 1)\ , \tag{37}$$

where

$$\ell_c = \left\lceil \frac{2J}{|H|} \right\rceil\ , \tag{38}$$

and $\lceil x \rceil$ denotes the smallest integer not less than x. The nucleating droplet is a cluster of overturned spins of the stable phase which is a $\ell_c\times(\ell_c-1)$ rectangle with one additional overturned spin on one of the longest sides of the rectangle [65]. There is a non-trivial prefactor A that enters the average lifetime [67] as

$$\langle\tau_{\rm low\ SD}\rangle = A_{\rm SD}\exp\left[\Gamma(H,J)/k_{\rm B}T\right]\ . \tag{39}$$

The prefactor $A_{\rm SD}$=5/4 for ℓ_c=1 and $A_{\rm SD}$=3/8 for ℓ_c=2 [67]. Closed forms for A_{SD} for larger values of ℓ_c have not yet been calculated.

The three limits, $L\to\infty$, $T\to 0$, and $|H|\to 0$ do not commute for the decay of the metastable state for the Ising model. Taking first $L\to\infty$ places the system always in the MD regime for any finite temperature [see Fig. 4 and Eq. (36)]. In this case the average lifetime is given by [68]

$$\langle\tau_{\rm low\ MD}\rangle = A_{\rm MD}\exp\left[\frac{\Gamma(H,J) + (4J - 2|H|)}{3k_{\rm B}T}\right] \tag{40}$$

whenever $\ell_c>1$. The factor of 3 (which is d+1 in d dimensions) in the denominator is due to the multi-droplet nature of the decay and is the same as that for the MD regime for circular droplets [62, 63]. The additional term is due to the fact that at low temperatures for $|H|<2J$ the growth of supercritical droplets is dominated by nucleation events on the surface of these droplets [68].

The advanced algorithms to be described below perform differently in the different decay regimes.

5.1 Standard Monte Carlo for the Ising Model

The Ising energies of Eq. (1) are sufficient to obtain the statics of the model, such as the specific heat or magnetic susceptibility. However, this model does not have any dynamic since it is a classical model. A class of dynamics for this model can be derived using a Master equation approach [19, 20, 21]. Martin in 1977 [22] showed that it is possible to start with quantum spin $\frac{1}{2}$ particles each coupled to their own particular heat bath, and to obtain the dynamic described in Sec. 2.0.

Note that there are an infinite number of possible dynamics that could obtain the correct statics with the energy of Eq. (1). For example, it is possible to use a Metropolis spin flip probability [1, 4] given by Eq. (3) rather than the Glauber spin flip probability Eq. (2). It would also be possible to choose the spin in a particular order, perhaps sequentially. However, using one of these dynamics would lead to different values of $\langle\tau\rangle$ compared with the physically justified dynamic. For example, it has been shown [23, 24] that the power of the H prefactor of the nucleation rate differs between the random update and sequential update dynamics, and the random update dynamic has an H prefactor consistent with the value expected from field-theory arguments [69, 70]. Thus the physically motivated dynamic of Sec. 2.0 should be used in cases where the dynamics is given a physical meaning. Furthermore, some of the advanced algorithms such as the n-fold way and MCAMC algorithms work only for the random update dynamic — or would have to be modified significantly for the sequential update dynamic.

The standard dynamic Monte Carlo for metastable decay of the Ising model is:

- Start the lattice with all N spins up.
- Set the time t to zero.
- Perform the following steps until the system reaches a magnetization M=0:
  - Increment the time t to t+1.
  - Calculate two random numbers, $\tilde{r}$ and $\bar{r}$.
  - Randomly choose one of the N spins using $\tilde{r}$.
  - Calculate (or look up) the current energy $E_{\rm old}$ and the new energy $E_{\rm new}$ if the chosen spin were to flip.
  - Calculate (or look up from a table) the Glauber flip probability.
  - Use Eq. (2) with $\bar{r}$ and either flip the spin or retain the old spin configuration.

This algorithm is executed K times, and the statistics for the mean lifetime are accumulated. A single algorithmic step is a Monte Carlo step (mcs), while the time that should be proportional to the physical time is a Monte Carlo step per spin (MCSS). The average lifetime, calculated for K=1000 escapes, is shown for two different fields in Fig. 5(a) and Fig. 6.

5.2 n-fold Way for the Ising Model

The n-fold way algorithm introduced by Bortz, Kalos, and Lebowitz [33] uses spin classes for the Ising model. For the isotropic square-lattice Ising model with periodic boundary conditions in finite field, every spin belongs to one of n=10 different classes. These classes are organized by the spin orientation (up or down) and the number of nearest-neighbor spins that are up (which can be 0, 1, 2, 3, or 4). The energy of every spin in a particular class is the same, so the probability of flipping any spin in a given class is the same. The 10 spin classes are listed in Table 1. Note that for escape from the metastable state we start with all spins up so all spins are initially in class 1. When an up (down) spin flips, it changes its spin class by +5 (−5). Furthermore, when an up (down) spin flips, all four of

Class Number	Spin Orientation	Number of nearest-neighbor spins up	$-E_{\rm old}$	$-E_{\rm new}$
1	$\uparrow$	4	$4J+H$	$-4J-H$
2	$\uparrow$	3	$2J+H$	$-2J-H$
3	$\uparrow$	2	$+H$	$-H$
4	$\uparrow$	1	$-2J+H$	$2J-H$
5	$\uparrow$	0	$-4J+H$	$4J-H$
6	$\downarrow$	4	$-4J-H$	$4J+H$
7	$\downarrow$	3	$-2J-H$	$2J+H$
8	$\downarrow$	2	$-H$	$+H$
9	$\downarrow$	1	$2J-H$	$-2J+H$
10	$\downarrow$	0	$4J-H$	$-4J+H$

Table 1: The 10 classes of Ising spins for the square-lattice nearest-neighbor model.

its nearest-neighbor spins change their spin classes by $+1$ (-1). Also shown in Table 1 are the new and old local energies of the spins — remember $J>0$ while $H<0$ for our escape from the metastable state.

Let n_i be the number of spins in class i. One always has

$$N = \sum_{i=1}^{10} n_i \; . \tag{41}$$

Let p_i be the probability of flipping a spin in class i using the dynamic such as the Glauber dynamic of Eq. (2), given that the spin was chosen during the random choice portion of the dynamic. Then the probability in one time step of flipping any spin in class i is $n_i p_i/N$ since n_i/N is the probability of randomly choosing a spin in class i and p_i is the flip probability once a spin in class i has been chosen. In order to exit from the current spin configuration, a spin in one of the 10 classes must flip. Consequently, the absorbing Markov chain has one transient state and 10 absorbing states. In general, the n-fold way has n classes for a particular discrete spin model, and there are n absorbing states and one transient state, the current spin configuration.

The original n-fold way algorithm was written in continuous time [33]. It was recast in a form for discrete time in 1995 [34]. We will use the discrete-time version because it allows for a much easier generalization to more than one transient state. Introduce

$$Q_i = \frac{1}{N} \sum_{j=1}^{i} n_j p_j \qquad 1 \le i \le n \tag{42}$$

and define Q_0=0. The absorbing Markov matrix for exiting from the current spin config-

uration is

$$\mathbf{M}_{n+1} = \frac{1}{N} \begin{pmatrix} N & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & N & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & N & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & N & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & N & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & N & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & N & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & N & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & N & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & N & 0 \\ n_1p_1 & n_2p_2 & n_3p_3 & n_4p_4 & n_5p_5 & n_6p_6 & n_7p_7 & n_8p_8 & n_9p_9 & n_{10}p_{10} & \mathbf{T}_{1,1} \end{pmatrix} \tag{43}$$

where $\mathbf{T}_{1,1}$=$N(1-Q_{10})$ is the transient matrix, here with a single element.

The algorithm to perform the n-fold way for escape from the metastable state is:

- Start the lattice with all N spins up. Set n_1=N and n_i=0 for i>1 and also set M=N.
- Set the time t to zero.
- Perform the following steps until the system reaches a magnetization M=0:
  - Calculate three random numbers, $\tilde{r}$, $\bar{r}$, and r.
  - Calculate the 10 values for Q_i using Eq. (42).
  - Calculate the time m (in units of Monte Carlo spin flip attempts) to exit from the current spin configuration (state) using

$$m = \left\lfloor \frac{\ln(\bar{r})}{\ln(1-Q_{10})} \right\rfloor + 1 \; . \tag{44}$$

  - Using the random number $\tilde{r}$, find the spin class k that satisfies

$$Q_{k-1} \le \tilde{r} Q_{10} < Q_k \; . \tag{45}$$

  - Use the random number r to pick one of the n_k spins from spin class k, and flip the chosen spin.
  - Change the number of spins in class k to n_k−1.
  - If the flipped spin was up (down), change the number of spins in class n_{k+5} (n_{k-5}) by +1.
  - If the flipped spin was up (down), for each of the four nearest-neighbor spins initially in class j change the number of spins in class j to be n_j−1 and the number of spins in class n_{j+1} (n_{j-1}) to be n_{j+1}+1 (n_{j-1}+1).
  - Set the time t to t=t+m.

A number of remarks are in order. First, just as in the n-fold way for the simple ion channel, the time to exit the current state comes from using Eq. (7) with the Markov matrix of Eq. (43). Since there is only one transient state, the equation

$$(1-Q_{10})^m < \bar{r} \le (1-Q_{10})^{m-1} \tag{46}$$

reduces to Eq. (44). Second, the state to which the current spin configuration changes comes from using Eq. (9) with the Markov matrix of Eq. (43). Since there is only one transient state, the factors of $\mathbf{T}^{m-1}$ in Eq. (9) are scalars and cancel, and the normalization in Eq. (9) becomes $\vec{v}_{\rm I}^{\rm T}\mathbf{R}\vec{e}{=}Q_{10}$.

To recover the original [33] continuous-time result, note that $Q_{10}{<}1$ always, and usually $Q_{10}{\ll}1$ for most spin configurations near a minimum of the free energy (i.e. near a metastable or stable state). Expanding Eq. (44) gives

$$\begin{aligned} m &= \left\lfloor \ln(\bar{r}) \left(\frac{1}{\ln(1-Q_{10})} \right) \right\rfloor + 1 \\ &\approx \left\lfloor \ln(\bar{r}) \left[-\frac{1}{Q_{10}} + \frac{1}{2} + \frac{1}{12}Q_{10} + \frac{1}{24}Q_{10}^2 + \frac{19}{720}Q_{10}^3 + \cdots \right] \right\rfloor + 1 \, . \end{aligned} \tag{47}$$

$$\tag{48}$$

Keeping only the first term in Eq. (47) gives the time increment

$$m \approx \Delta t = -\frac{\ln(\bar{r})}{Q_{10}} \, , \tag{49}$$

with the time units in Monte Carlo steps (mcs). To change the time units to physical time units (MCSS) for Δt, one needs to divide Eq. (49) by N. The time unit is trivial to change to MCSS in the continuum n-fold way since it just cancels the factor of N in the definition for the Q_i, Eq. (42), and gives the result in Ref. [33] (where the authors define their values of Q_i without the $1/N$ normalization). However, no such fortuitous cancelation between m and N during a time unit change occurs in the discrete version of the n-fold way, Eq. (44).

There is an efficient way [33] to keep track of the classes of all the spins that uses four arrays. (See Ref. [33] for the use of three arrays and the additional packing of information). The array `NNCLS(10)` contains the number of spins n_i in each of the 10 classes. The two-dimensional array `LOC(`N`,10)` contains the `LOC`ation of the spins in the lattice for each of the 10 classes of spins, where the $L{\times}L$ lattice is considered as a one-dimensional array of spins. There is no particular order within a class of spins, and only the `NNCLS(i)` spins in `LOC(j,i)` contain spin locations in current use. (Adjustable partitions in a single array of length N could be used for `LOC` [33], but in `FORTRAN` a price is paid in terms of the time required to do the bookkeeping involved in changing the adjustable partitions after every time step. Here `C` and `C++` programmers have an advantage.) Array `LOOK(`N`)` determines the index of the first dimension in LOC for each of the N spins, and the array `LOOC(`N`)` is used to determine the class of a spin in `LOC`, given its position in the lattice. Hence `LOC(LOOK(i),LOOC(i))` = `i`.

Suppose we have chosen class k using $Q_{k-1}{\le}\tilde{r}Q_{10}{<}Q_k$ and want to flip one of the n_k spins in this class. We first calculate a uniform random integer j between 1 and `NNCLS(k)`

and find the spin location $i = \texttt{LOC}(j,k)$. Then we need to update all four arrays for the change in classes of the ith spin and its four nearest neighbors. This indexing provides a fast method for changing the classes of spins in the n-fold way algorithm [33, 71].

The CPU time required for the data presented in Fig. 5(a) is presented in Fig. 5(b) and the CPU time required for the data presented in Fig. 6 are shown in Fig. 7. Note that the n-fold way algorithm is significantly faster at low temperatures (by orders of magnitude!) than the standard dynamic Monte Carlo algorithm. For $H{=}{-}3J$ the CPU time required for the n-fold way simulation is roughly independent of the temperature for any value of L. This is because for $2J{<}|H|{\leq}4J$ a single overturned spin is the nucleating droplet [65]. For $|H|{<}2J$, as in Fig. 6, the CPU time required for the n-fold way simulation increases as the temperature is lowered since the nucleating droplet includes more than one overturned spin [65]. See Fig. 7.

5.3 MCAMC for the Ising Model

Although the n-fold way algorithm is extremely fast for $2J{<}|H|$, it is possible to further decrease the amount of CPU time required for a given simulation with $|H|{<}2J$ by using more states in the transient subspace. In particular, at low temperatures where the energy of the nucleating droplet is known [65] to be given by Eq. (37), it is possible to use higher s MCAMC only for the exit from the state with all spins up and use $s{=}1$ MCAMC (n-fold way) for the other moves. This is similar to the strategy used in the MCAMC for the ion-channel model of Sec. 4.3.

For $s{=}2$ MCAMC the transient matrix is

$$\mathbf{T}_{s=2} = \begin{pmatrix} 1 - \frac{[p_6+4p_2+(N-5)p_1]}{N} & \frac{p_6}{N} \\ p_1 & 1-p_1 \end{pmatrix} \tag{50}$$

and the recurrent matrix is

$$\mathbf{R}_{s=2} = \frac{1}{N} \begin{pmatrix} 4p_2 & (N-5)p_1 \\ 0 & 0 \end{pmatrix} . \tag{51}$$

The first transient state (in the lower right-hand corner of the matrix) is the initial state (all spins up), and the second state is all the N spin configurations with one overturned spin. The first absorbing state corresponds to all $2N$ spin configurations with two nearest-neighbor overturned spins, and the second to all $N(N{-}5)/2$ spin configurations with two overturned spins that are not nearest-neighbor spins. In the MCAMC algorithm, if the spin configuration has two or more overturned spins, the normal n-fold way algorithm is used. If the system returns to the initial state (all spins up), the absorbing Markov chain part of the algorithm using Eqs. (50) and (51) is performed.

For $s{=}3$ the transient matrix for starting from the state with all spins up is

$$\mathbf{T}_{s=3} = \frac{1}{N} \begin{pmatrix} N-2p_7-6p_2-(N-8)p_1 & 2p_7 & 0 \\ 4p_2 & N-p_6-4p_2-(N-5)p_1 & p_6 \\ 0 & Np_1 & N(1-p_1) \end{pmatrix} \tag{52}$$

and the recurrent matrix is

$$\mathbf{R}_{s=3} = \frac{1}{N} \begin{pmatrix} (N-8)p_1 & 2p_2 & 4p_2 & 0 \\ 0 & 0 & 0 & (N-5)p_1 \\ 0 & 0 & 0 & 0 \end{pmatrix} . \tag{53}$$

Here the transient state added to the s=2 MCAMC is all $2N$ spin configurations with two overturned nearest-neighbor spins. The absorbing states correspond (left to right) to all the spin configurations with: 1) two nearest-neighbor overturned spins and one not-connected overturned single spin; 2) a linear chain of three overturned spins; 3) an L-shaped arrangement with three overturned spins — which corresponds to the nucleating droplet at low temperatures for $J<|H|<2J$ which corresponds to ℓ_c=2 of Eq. (38); 4) two not-nearest-neighbor overturned spins.

After exiting from the absorbing Markov Chain portion of the algorithm, the spin configuration on exit must be decided on since each of the absorbing states corresponds to a large number of spin configurations with equal absorption probabilities. The method used in the n-fold way algorithm to decide on which particular spin within a chosen spin class is generalized to accomplish this. For example, the first absorbing state for the s=3 MCAMC uses three random numbers, r_i, to construct the exiting spin configuration. First r_1 is used to pick uniformly one of the N spins and this spin is flipped to down. (With periodic boundary conditions this step is not strictly necessary, but is advisable to avoid any correlations between the random number and the algorithm [72] — particularly if the exit probability is low.) Then r_2 is used to pick uniformly one of the 4 nearest-neighbor spins of the flipped spin and this spin is flipped. Finally, r_3 is used to pick uniformly one of the $N-8$ spins, not including the two flipped spins or their 6 nearest-neighbor spins, and this spin is flipped. This gives the exiting spin configuration for this particular absorbing state. A similar procedure, using as many random numbers as required, is performed if a different state is the absorbing state.

The time m to exit the absorbing Markov chain is given by using a random number $\bar{r}$ and Eq. (7) with m satisfying

$$\vec{v}_{\rm I}^{\rm T}\mathbf{T}^m\vec{e} < \bar{r} \le \vec{v}_{\rm I}^{\rm T}\mathbf{T}^{m-1}\vec{e}\,. \tag{54}$$

Remember that m is in units of Monte Carlo steps (mcs), while the physical time is in units of Monte Carlo steps per spin (MCSS). This equation does not simplify to an equation in closed form as it did for the s=1 case, Eq. (44). However, it can be solved iteratively using for example a bracketing and bisection method [2]. It is critical for issues of algorithmic speed to obtain a reasonable guess for m and brackets for m for a given absorbing Markov chain and a given $\bar{r}$. This can be done by assuming a form such as Eq. (44) with upper and lower bounds [73] for the eigenvalues of the transient matrix.

The choice of which state the MCAMC algorithm absorbs to is given using a random number $\tilde{r}$ and the vector $\vec{q}$ of Eq. (9). If the elements of $\vec{q}$ are given by q_i, then the absorbing state j is the solution of the equation

$$\sum_{i=1}^{j-1} q_i \le \tilde{r} < \sum_{i=1}^{j} q_i \tag{55}$$

with the first sum being set to zero if j=1. This corresponds to Eq. (45) for the n-fold way algorithm. For the particular s=2 MCAMC of Eqs. (50) and (51), the order of the calculation of m and the exit state is not important, just as the order is not important for the n-fold way algorithm. This is because there is only one transient state that the absorbing Markov chain can exit from. In this case the explicit m dependence in Eq. (9)

Algorithm	Γ_0	$\Gamma-\Gamma_0$	A	$J/k_{\rm B}T$ of fit point
Monte Carlo	0	$24J$-$14\|H\|$	1.21×10^{-8}	1.231
n-fold way (s=1)	$8J$-$2\|H\|$	$16J$-$12\|H\|$	1.32×10^{-7}	2.1
s=2 MCAMC	$12J$-$4\|H\|$	$12J$-$10\|H\|$	1.44×10^{-6}	2.857

Table 2: The values used to fit the required CPU time for different MCAMC algorithms in Fig. 7. The fit functions are valid for ℓ_c=3, so $\frac{2}{3}J<|H|<J$.

cancels in the numerator and denominator. However, in general this is not the case and Eq. (9) and Eq. (55) must be used to calculate the absorbing state one must exit to.

If a MCAMC algorithm for fixed s is not performing very well, it can always be improved upon by adding more states to the transient subspace. A natural question is which state(s) should be added first. The more transient states there are, the longer calculations using Eq. (7) and Eq. (9) will take — as well as there being a higher probability that a programming error is made in constructing either the Markov matrix or the exiting spin configuration. So the states should be added to the transient subspace in an intelligent fashion. The prescription for finding which absorbing state(s) should be added to the transient subspace is found using Eq. (11). Whichever state(s) have the largest probabilities of absorption should be added to the transient subspace. Of course this will depend on the model and the model parameters — for example in metastable decay for the Ising model the absorbing state with the highest probability in Eq. (11) depends on L, $|H|$, and T.

For very low temperatures, the value of m for each exit can be very large. It is then important not to let the discreteness of the random number used to obtain m affect the final answer for m. This is important for both low-temperature standard Monte Carlo in which the p_i can be very small, as well as for all s-state MCAMC algorithms. In practice, I accomplished this by using another random number distributed uniformly between 0 and 0.01 whenever the first chosen random number was less than 0.01. The factor of 0.01 is arbitrary and will depend on the number of bits of the random number generator in use. Furthermore, to go to very low temperatures (and hence large lifetimes) I found it useful to code using routines that can be set to a large precision. I used the package MPFUN [74].

Fits to the required CPU times can be obtained at low temperatures by using the energy difference between the nucleating droplet and the most probable absorbing state. For the data in Fig. 7, these are listed in Table 2.

Average lifetimes using s=2 and s=3 MCAMC are presented in Figs. 5(a) and 6. The CPU time required for these MCAMC algorithms for $|H|=0.75J$ is shown in Fig. 7. For $|H|<2J$ and low temperatures, the higher s MCAMC algorithms can perform many orders of magnitude faster than the s=1 MCAMC algorithm, which is itself many orders of magnitude faster than the standard dynamic Monte Carlo algorithm! For example, at $k_{\rm B}T=0.25J$ the s=3 MCAMC algorithm is about 10^{16} times faster than the standard algorithm!

5.4 Projective Dynamics for the Ising Model

The MCAMC algorithm performs extremely well in the low-temperature and strong-field regime, i.e. in the single-droplet (SD) and coexistence regimes at low temperatures. However, MCAMC with small s does not perform very well at higher temperatures or for large system sizes. This is because the average time increment m given by Eq. (54) is relatively small for large T or large L. In principle it would be possible to construct higher s MCAMC that would work in this regime, but this would require a lot of bookkeeping of the large number of transient and absorbing states. Is there another advanced simulation method that can be used in the multi-droplet regime? For the simple ion-channel model of Sec. 4.4, the solution was found by using the growing and shrinking probabilities. However, for an N-spin Ising model the underlying Markov matrix is $2^N \times 2^N$ with complicated transition probabilities between the 2^N states. In continuous time the probability $P(\{\sigma\}, t)$ of the N Ising spins being in state $\{\sigma\}$ at time t is given by

$$\frac{\partial P(\{\sigma\}, t)}{\partial t} = \sum_{\{\sigma'\}} \Gamma(\{\sigma\}|\{\sigma'\}) P(\{\sigma'\}, t) \; , \tag{56}$$

where $\Gamma(\{\sigma\}|\{\sigma'\})$ is the probability per unit time of moving to spin configuration $\{\sigma\}$, given that the current spin configuration is $\{\sigma'\}$.

The idea behind the Projective Dynamics (PD) method is that one still expects that a simple one-dimensional physical picture of the free energy, Fig. 8, might hold in this complicated situation. In particular, one considers lumping together all states with a total magnetization M to form a $(N+1)\times(N+1)$ lumped Markov matrix. Mathematically, the Markov matrix with 2^N states is not even weakly lumpable [75], but is most likely almost weakly lumpable with respect to the states along the escape from metastability. Thus for the Master equation on the $N+1$ states of a particular magnetization we get

$$\frac{\partial P(M, t)}{\partial t} = \sum_{\{M'\}} W(M|M') P(M', t) \; . \tag{57}$$

Note that due to the single-spin flip dynamic there will only be non-zero transition probabilities between states that have magnetizations that differ by one overturned spin. Then the question is how to define the lumped transition probabilities $W(M|M')$. One possibility is to assume an Arrhenius form [76] for the transition probabilities for the lumped states [66, 77]. This will give the correct free energy for the static model, and actually gives a reasonable approximation for $\langle \tau \rangle$ in the single-droplet regime [66, 77]. It has been extended to projection onto two dimensions, magnetization and energy [78, 79]. However, in the multidroplet regime it is important to also get the growth phase correctly, and this is a non-equilibrium phenomenon. This leads to poor agreement between the average lifetime obtained from standard simulations and by using the Arrhenius form for the shrinking and growing probabilities on the lumped states [66].

However, it is possible to actually measure the lumped transition probabilities during a sequence of K simulated escapes from the initial all-spin-up configuration. The lumped matrix is tridiagonal due to the single-spin-flip character of the dynamic. Define $\langle c_i \rangle_M$ to be the average number of spins in class i during the simulated escape, given that the configuration has magnetization M. In terms of the flipping probability of a spin in class

i, the growing rate of the stable phase (spins down) is

$$g(M) = \sum_{i=1}^{5} \langle c_i \rangle_M p_i \ , \tag{58}$$

and the shrinking rate of the stable phase is

$$s(M) = \sum_{i=6}^{10} \langle c_i \rangle_M p_i \ . \tag{59}$$

Note that the $\langle c_i \rangle_M$ should be measured during the actual escape from the initial spin configuration. Then the 'average' transition probability for the lumped matrix is

$$W(M|M') = s(M')\delta_{M+2,M'} + g(M')\delta_{M-2,M'} \ . \tag{60}$$

In particular, the probability of going from lumped state M to lumped state M' is given by

$$W(M|M') = \frac{1}{n(M')} \sum_{\sigma \in \{\sigma|M\}} \sum_{\sigma' \in \{\sigma|M'\}} \Gamma(\{\sigma\}|\{\sigma'\}) \ , \tag{61}$$

where $n(M)$ is the number of states with magnetization M.

The random walk starts at $M{=}L^2$ and terminates at $M{=}0$. We define $h(M)$ as the total time spent in M. If L is even, $h(M){=}0$ for M odd, since these magnetizations are not possible. Then, in analogy with Eqs. (32) and (33), one has

$$\langle \tau \rangle = \sum_{M=2}^{L^2} h(M), \quad h(M) = \frac{1 + s(M-2)h(M-2)}{g(M)}, \quad h(2) = \frac{1}{g(2)} \ . \tag{62}$$

An error estimate for $\langle \tau \rangle$ can be obtained from error estimates of the $h(M)$. Assume error estimates for $g(M)$ and $s(M)$ have been obtained, for example by using a jack-knife procedure [80]. Then using standard error propagation [81], the error in the lifetime is

$$(\delta\tau)^2 = \sum_{M=2}^{L^2} \left[\delta h(M)\right]^2 \tag{63}$$

with

$$\delta h(2) = \frac{\delta g(2)}{g(2)^2} \tag{64}$$

and iteratively

$$\begin{aligned} \delta h(M) \ = \ & \frac{1}{g(M)^2} \Big\{ \left[1 - s(M-2)h(M-2)\right]^2 \left(\delta g(M)\right)^2 \\ & + \left[h(M-2)\right]^2 \left(\delta s(M-2)\right)^2 \\ & + \left[s(M-2)\right]^2 \left(\delta h(M-2)\right)^2 \Big\}^{\frac{1}{2}} \ . \end{aligned} \tag{65}$$

How close to the actual lifetime is the average lifetime measured by using Eq. (62)? The answer is that they are *identical* to within statistical errors! This is true even though

the original Markov matrix on 2^N states is not lumpable. For the Ising model this can be proven using class populations [82]. However, there is a more transparent proof [83]. Assume that all the $g(M)$ and $s(M)$ were correct. Since $g(2)$ is correct, from Eq. (62) $h(2)$ is correct. Since $h(2)$, $s(2)$ and $g(4)$ are correct, from Eq. (62) $h(4)$ is also correct. This can then be iterated to obtain the exact value of $\langle\tau\rangle$. Note that although the exact value of $\langle\tau\rangle$ is obtained, only approximations to the higher moments of τ are obtained.

The lumped growing and shrinking probabilities also allow one to numerically investigate the 'free-energy' landscape during the escape from the metastable state [84, 85]. Figure 9 shows these growing and shrinking probabilities for a chosen set of parameters for the square-lattice Ising model. Wherever $g(M)$=$s(M)$, the probability of growing is equal to the probability of shrinking, and that value of M must correspond to an extremum during the motion. In terms of the coarse-grained 'free energy' during this escape, these correspond to the metastable state, the saddle point, and the stable state. In more complicated models the locations where $g(M)$=$s(M)$ also represent extrema. This fact allowed [85] both the Barkhausen volumes and activation volumes during thermally assisted domain-wall motion in a model for magnetization reversal in Fe sesquilayers on W(110) to be obtained [86]. Using Eq. (62) to obtain $h(M)$ and assuming that just as in equilibrium the 'free energy' during the escape from the metastable state is

$$\mathcal{F}(M) \propto -\ln\left[h(M)\right] \tag{66}$$

gives a way to measure the 'constrained free energy during escape from the metastable state,' Fig. 10, similar to Fig. 8.

It is also possible to introduce a moving wall in the magnetization to force the system out of the metastable state, and to measure $g(M)$ and $s(M)$ during this (hopefully quasi-static) process [87]. The wall position is given by

$$M_{\rm wall}(t) = (L^2{+}1) - v_{\rm wall}t\ . \tag{67}$$

For a soft wall the normal Monte Carlo flip probability is multiplied by

$$p_{\rm wall} = \exp\left[-c\left(M_{\rm new} - M_{\rm wall}\right)\right] \tag{68}$$

if the wall at position $M_{\rm wall}$ is past the magnetization of the Monte Carlo new trial configuration [88]. Otherwise there is no change in the flip probability. Another possibility is to introduce a hard wall, c=∞, that always flips a chosen up spin if $M_{\rm wall}$ is past the current magnetization. Fig. 11 shows an example of lifetimes for a hard forcing wall as a function of the wall speed. The difference between the lifetimes measured from Eq. (62) with the number of Monte Carlo Steps per spin before escape with the wall gives an estimate for the speed-up due to incorporating the forcing wall. Of course, for a wall that moves too fast the process is not quasi-static, and the lifetimes are not accurate. However, good results are obtained even for relatively fast walls. For example, for the hard wall of Fig. 11 a velocity of $3{\times}10^{-6}{\rm M}/L^2{\rm MCSS}$ gives a lifetime comparable to an actual escape, but requires almost five orders of magnitude less computer time. The soft walls do not seem to help the convergence, and additional computations are needed at each step to calculate $p_{\rm wall}$. This often makes the hard wall more computationally efficient than the soft walls [88]. For example, for the same parameters as in Fig. 11 with c=0.01

with a speed of $1.0\times10^{-5}\mathrm{M}/L^2\mathrm{MCSS}$ the lifetime was 2.6×10^9 MCSS, had not yet become independent of the forcing speed, and required about 2.1 minutes of Cray SV1 CPU time for each escape. This should be compared with the same forcing speed in Fig. 11 for the hard wall, which required only about 0.06 Cray SV1 CPU minutes for each escape.

In some cases the class populations $\langle c_i \rangle$ during escape from the metastable state can be approximated by the class populations in equilibrium, $\langle c_i \rangle^{\mathrm{equ}}$ [84]. These equilibrium class populations can be measured during an equilibrium Monte Carlo simulation [66]. Then using Eq. (62) with this assumption the lifetime can be obtained at *all* temperatures and fields since the dependence of the flip probabilities on T and H are known. The approximation for the $\langle c_i \rangle$ actually becomes better as $|H|$ becomes smaller since then the nonequilibrium configurations play less of a role in the metastable decay. In this way lifetimes of about 10^{30} Monte Carlo Steps per spin have been obtained for Ising ferromagnets in two and three dimensions for small H [84]. In addition, with the assumption that the class populations do not change with a change of dynamics, values of $\langle c_i \rangle$ obtained using one dynamic [say Eq. (2)] can be used to obtain lifetimes using a different dynamic [say Eq. (3)].

As seen in Eqs. (58) and (59), the growing and shrinking rates are functions of both the average of the spin classes, $\langle c_i \rangle$, and the single-spin flip probabilities, p_i. The functional form for p_i is given by Eq. (2) using the energies in Table 1. Hence the functional form for the p_i are known for all T and H values. One could assume that the average class populations do not change too much with changing field or temperature. Then the class populations at one simulated value of T and H can be used to obtain growing and shrinking probabilities at other values of H and T. For the parameters in Fig. 12 this assumption produces the wrong functional form for $\langle \tau \rangle$. This is because for a growing interface the class populations are not constant [89]. An alternative assumption is that the average class populations can be given by the simulation results at $k_{\mathrm{B}}T{=}J$ except for magnetizations near the metastable well, where class populations from equilibrium Monte Carlo simulations [66, 84] can be used. As seen in Fig. 12 this combination of assumptions works well for the chosen parameters over almost the entire temperature range. This is because at low temperatures the growth phase contributes very little to the average lifetime. Furthermore, for our fixed L and H, the most probable equilibrium configuration is nearly the same as the most probable path of the nucleating droplet at fixed M as T becomes small. Such techniques must be used in conjunction with knowledge about the physics of the escape from the metastable state to be accurate, as Fig. 12 demonstrates. Simple approximate class populations during the growth phase [89] could alternatively be used for the growth portion of these types of calculations.

It is also possible to approximate the growing and shrinking probabilities of a system of size $2N$ from a system of size N [84]. Since the relevant configurations typically contain many small droplets, to a good approximation the larger system can be viewed as consisting of two 'independent' copies of the smaller system. In this approximation one can write that

$$g(2N,M) \approx \frac{\sum_{M'=M}^{2N} h(N,N+M-M')h(N,M')\left[g(N,N+M-M')+g(N,M'-N)\right]}{\sum_{M'=M}^{2N} h(N,N+M-M')h(N,M'-N)} \tag{69}$$

and

$$s(2N,M) \approx \frac{\sum_{M'=M}^{N} h(N,N+M-M')h(N,M')\left[s(N,N+M-M')+s(N,M'-N)\right]}{\sum_{M'=M}^{N} h(N,N+M-M')h(N,M'-N)} \tag{70}$$

where the explicit dependence on the volume has been included. Remember that for N even, the odd magnetizations can be included since they have zero residence times. Note that the cut-off in this extrapolation does not remain at the original value of M, but rather is a constant in the number of overturned spins on a lattice of any size. Using Eqs. (69) and (70) repeatedly, one can extrapolate to very large lattices. While the lattice sizes extrapolated to are nowhere near world record sizes [90], much additional information has been obtained from the small-lattice simulations. This is illustrated in Fig. 13.

These types of projective dynamics techniques with a forcing wall can also be used on other discrete spin models and on other lattices to obtain extremely long lifetimes [91]. For example, Figure 14 shows results for the simple-cubic Ising ferromagnet on a 32^3 lattice. The temperature and field are such that $\langle\tau\rangle$=7.3×10^{15} MCSS. This lifetime is more than 10^6 times longer than a recent large-scale simulation of the model using standard Monte Carlo simulations [92]. Here the inverse speed of the hard forcing wall was 5000 n-fold way updates per spin per overturned spin. For the square lattice at low temperatures at fixed L, the nucleating droplet has neither a square nor a circular shape [66, 93, 94]. Similarly, for the simple cubic lattice the nucleating droplet at low temperatures for fixed L has a particular shape which is neither a cube nor a sphere [95, 96]. These discrete lattice effects cause the oscillations in the quantities seen in Fig. 14.

5.5 Parallelization of Dynamic Monte Carlo

With the widespread availability of massively parallel computers, with the number of processing elements (PEs) on the order of hundreds to thousands, the question arises as to how dynamic Monte Carlo simulations can be implemented efficiently on these parallel computers. Often the parallelization can be accomplished in a trivial fashion, just by letting each PE run its own copy of one of the K escapes. Then no communication between PEs must be performed until the escape is finished and statistics are collected. However, sometimes the lattice size to be simulated is too large to fit onto a single PE, or fitting the simulation on a PE will use too much swapping of virtual memory, leading to long execution times. In such cases a non-trivial parallelization scheme is needed.

The initial impression might be that dynamic Monte Carlo is a scalar algorithm. However, in 1988 Lubachevsky [97] showed that the dynamic Monte Carlo simulation for the Ising model belongs to a class of problems called discrete-event simulations and can be parallelized. As mentioned in Sec. 4.2, a discrete-event simulation [54] is a problem in which the state vector changes discontinuously at certain (perhaps random) times. Discrete-event simulations are used in an extremely wide range of applications in science, technology, manufacturing, and sociology. Just a few recent examples include preventative veterinary medicine [98], resource allocation following a terrorist bombing [99], modeling concrete supply and delivery [100], aircraft design [101], ecological models [102, 103], design of logical circuits [104, 105], manufacturing systems [106], battlefield simulations [107], and simulation of wireless services [108]. For the Ising model the method of paral-

lelization is to place a block of spins, say $\ell \times \ell$ in two dimensions, on a processing element. Then the total number of spins on the lattice is $N=L^2=\ell^2 N_{\rm PE}$ where $N_{\rm PE}$ is the number of PEs used in the simulation. A block of spins and its neighboring spins from nearest-neighbor blocks on a square lattice is shown in Fig. 15, shown for $\ell=6$. For $\ell=6$ there are 16 interior spins and 20 boundary spins on each PE.

To parallelize the dynamic Monte Carlo algorithm, each PE has its own time clock with its own simulation time. This local time is called the 'virtual time' of the block of spins [109]. Remember that the entire system has a single time — but for parallel implementation each PE has its own clock with its own virtual time. To calculate some quantity at a particular time, for example the spin configuration at time t, the simulation must evolve until the clocks on all the PEs are at time t. However, if one is interested only in advancing the spin configuration over time (as opposed to measuring the spin configuration at a particular time) each PE can independently advance the spins in its block and update its time.

Of course the blocks of spins on nearest-neighbor PEs are not independent. Consequently the parallelization must ensure that causality is not violated as the simulation progresses. The conservative approach [97] to avoid causality violations is to have a PE idle (wait) until it is guaranteed that an update will not violate causality. If the time on a PE's clock is behind that of all of its nearest-neighbor PEs, then there is no possibility that causality is violated, and the PE can advance its spin configuration and time. Another approach to avoiding causality violations, called the speculative or optimistic approach, is to assume that no causality violations occur and have every PE always advancing its spin configuration [109, 110]. Of course at some point a causality violation will occur for a particular PE. Then it must roll back to a previous time and spin configuration before causality was violated. This rollback mechanism can then cause causality violations and hence rollbacks on the nearest-neighbor PEs. Hence such rollbacks can cascade [11–114] over the lattice.

We will only describe parallelization of the dynamic Monte Carlo for the Ising model using the conservative approach [97]. The same approach can easily be applied to other dynamic models, including dynamic classical models with continuous degrees of freedom. Each PE advances both its clock and its $\ell \times \ell$ block of spins *if and only if* its virtual time is not greater than the virtual times of *all* its nearest-neighbor blocks. Otherwise the PE idles until its virtual time is not greater than those of its nearest-neighbor PEs. In other words, if the virtual time for a PE is not a local minimum of the 'virtual time' surface then it executes a `wait until` directive until its virtual time is a local minimum. The 'virtual time' surface is the surface of 'virtual times' on all the PEs. This algorithm is free of deadlocks, because at worst the PE with the global minimum virtual time can advance its virtual time.

The checking restriction of virtual times can be relaxed for $\ell>1$ so that once a PE chooses a boundary spin to try to flip it checks the virtual time only of the single PE (or the two PEs for a corner spin) that the chosen boundary spin has its nearest-neighbor spin on. If this neighboring PE's virtual time is less than the updating PE's virtual time, then the updating PE `waits until` its virtual time is less than this neighboring PE's virtual time before the update attempt is made.

Figure 16 shows the update rate in Monte Carlo steps (mcs) per second on a Cray T3E as a function of the number of processing elements ($N_{\rm PE}$) used. This is based on work

reported in Refs. [115, 116]. The standard dynamic Monte Carlo algorithm of Sec. 5.1 was run using $N_{\rm PE}$ values of 8, 16, 64, 128, and 400, a fixed block size with ℓ=128, T=0.7$T_{\rm c}$, and $|H|$=0.2857J. The size of the simulated lattice was L=$\ell\sqrt{N_{\rm PE}}$, so the largest lattice simulated had L=2560. Figure 16 shows that the algorithm seems to be scalable — in other words, the update rate increases linearly with $N_{\rm PE}$ as the number of PEs and the lattice size L increases for a fixed T, H, and ℓ.

One important question in parallel discrete-event simulations using the conservative approach is the utilization of the PEs. In other words, what fraction of the PEs are not idling at a particular time? This can be answered by using non-equilibrium surface science methodologies [117, 118] to study the average utilization, $\langle u\rangle$, of the parallel algorithm. The value of $\langle u\rangle$ for the standard dynamic Monte Carlo simulations of Fig. 16 are shown in Fig. 17.

For the worst-case scenario for the average utilization, ℓ=1 (one spin on each block), it is possible to study the average utilization using finite-size scaling of simulations of the virtual time to obtain information about $\langle u\rangle$ as $N_{\rm PE}\rightarrow\infty$ [117, 119]. Fig. 17 shows the result obtained for a square lattice [119] from finite-size scaling of simulation data. Much more information about the virtual time surface can be obtained for a one-dimensional lattice (with periodic boundary conditions). In Ref. [117] it was shown that the virtual time surface for the conservative implementation of parallel discrete-event simulations is governed by the Edwards-Wilkinson Hamiltonian [120, 121]. Universality arguments then prove that $\langle u\rangle$ must be bounded away from zero as $N_{\rm PE}\rightarrow\infty$. This *proves* that in one dimension the conservative implementation of parallel discrete-event simulations are scalable in the worst-case scenario! This is a very general result for all one-dimensional lattices, *regardless* of the actual physical problem being simulated. In one dimension precise finite-size scaling can be used to show that $\langle u\rangle$ is just slightly less than $\frac{1}{4}$ in the worst-case scenario. This value is also shown in Fig. 17. The fact that $\langle u\rangle$ is finite, and hence the algorithm is scalable, is a non-trivial result. It has been shown, including exact analytic results, that there are other surfaces from reasonable growth models where the density of local minima approaches zero [122]. A finite-size scaling analysis of the approach to the $N_{\rm PE}\rightarrow\infty$ limit in one dimension shows that

$$\langle u\rangle_{N_{\rm PE}} - \langle u\rangle_{\infty} \propto \frac{1}{N_{\rm PE}} \tag{71}$$

for the ℓ=1 case [117]. This is the scaling form for the density of minima for this type of nonequilibrium surface [123]. In addition it was shown [117] that the fluctuations in the density of local minima is

$$\sigma^2_{N_{\rm PE}} = \langle u^2\rangle_{N_{\rm PE}} - \langle u\rangle^2_{N_{\rm PE}} \propto \frac{1}{\sqrt{N_{\rm PE}}} \tag{72}$$

for the one dimensional ℓ=1 case [117].

Another intrinsic property of the virtual time surface is its width as $N_{\rm PE}\rightarrow\infty$. This is important because, in order to measure a spin configuration at a particular time t, all PEs must wait (idle) once they reach time t until *all* the PEs have caught up. If the virtual time surface has a width which diverges as $N_{\rm PE}\rightarrow\infty$, then this measurement process will not scale. In Ref. [117] it was shown that indeed the interface width diverges in one

dimension since the virtual time horizon belongs to the Edwards-Wilkinson universality class. In higher dimensions numerical studies also indicate a divergence of the width of the virtual time surface [119]. It has, however, been shown that a slight change in the algorithm [118] can produce both a scalable algorithm and a virtual time surface with a finite surface width.

Another way to handle measurements is with a buffer of spin configurations. These spin configurations are stored once the virtual time on a PE is equal to a particular time at which a measurement is to be made. As long as the buffer is finite and the virtual time surface width diverges as $N_{\rm PE}\to\infty$, this measurement procedure will also not scale.

The inner loop of the parallel dynamic Monte Carlo algorithm is executed on each PE in parallel. Each PE updates an $\ell\times\ell$ block of spins, and each PE has its own clock to keep its virtual time. The parallelization is much easier in continuous time since in this case no synchronization is required because the virtual times on neighboring PEs cannot be equal. The simulation is started with all spins up and each PE has its virtual time set to zero. Then each PE determines the time $\Delta t=-\ln(\bar{r})$, where $\bar{r}$ is a uniformly distributed random number, of its first update attempt. The inner loop (written here specifically for a square lattice) is then executed repeatedly:

- Select a spin from the ℓ^2 spins on the block with equal probability.
- If the chosen spin is:
  - One of the $(\ell-2)^2$ spins in the kernel of the block proceed to the next step.
  - One of the $4\ell-4$ spins on the boundary, then check the virtual time of the neighboring PE or PEs that the chosen spin interacts with. If at least one of the neighboring PEs has a virtual time less than the updating PE's virtual time of the next update, then `wait until` this is no longer the case. (For the square lattice, this checking and wait-until condition requires 2 PEs for corner spins and one PE otherwise.)
- Update the spin using the same probabilities as in the standard dynamic Monte Carlo serial algorithm, i.e. with Eq. (2) or Eq. (3).
- Add Δt to the local virtual time.
- Determine the virtual time of the new next update by using the *local* time increment $\Delta t=-\ln(\bar{r})$, where $\bar{r}$ is a uniformly distributed random number.

The performance of this algorithm is shown in Fig. 16, and its utilization is shown in Fig. 17.

Can the advanced dynamic Monte Carlo algorithms also be parallelized? Lubachevsky in 1988 [97] proposed a method to parallelize in a conservative fashion the n-fold way algorithm. This was implemented and tested for the first time in 1999 [115, 116]. The idea is to have each PE using the n-fold algorithm with equations similar to Eq. (44) in discrete time or Eq. (49) in continuous time to update its virtual time. However, to ensure causality is not violated a 'shield' of spins at each surface acts as an absorbing state. In each block an additional absorbing class is defined which contains the spins on the boundary. There are $N_{\rm b}=4(\ell-1)$ of these spins. The original n-fold way tabulation

of the 10 spin classes for the square lattice are only performed for the $N_{\rm k}=(\ell-2)^2$ spins in the kernel. Hence $N_{\rm k}=\sum_{i=1}^{10} n_i$ and $N_{\rm b}+N_{\rm k}=\ell^2$. Note that each PE keeps its own class populations and its own virtual time. For each escape the simulation is started with all spins up and each PE has its virtual time set to zero. Then each PE determines the time Δt from Eq. (73) of its first spin update. The inner loop of the continuous time algorithm, performed by every PE in parallel, is:

- Use a uniformly distributed random number $\tilde{r}$ to select a class according to the relative weights $\{N_{\rm b}, n_1 p_1, n_2 p_2, \cdots, n_{10} p_{10}\}$, then use another random number r_1 to select a spin from the chosen class with equal probabilities.
- If the spin is:
  - One of the $(\ell-2)^2$ spins in the kernel, then flip this spin (with probability one — no rejection) and proceed to the next step.
  - One of the $4\ell-4$ spins on the boundary, then `wait until` the *local* simulated virtual time of the next update is not greater than the virtual time on the neighboring PE(s). This spin *may or may not* flip: flip it using the flip probability of Eq. (2) or Eq. (3) and proceed to the next step.
- If the chosen spin was flipped, update the 10 spin classes n_i.
- Add Δt to the local virtual time.
- Use a random number $\bar{r}$ to determine the time of the new next update (in units of Monte Carlo steps — mcs) by using the *local* time increment

$$\Delta t = -\frac{\ell^2 \ln(\bar{r})}{N_{\rm b} + \sum_{i=1}^{10} n_i p_i} \; . \tag{73}$$

The performance of this algorithm with fixed ℓ as a function of $N_{\rm PE}$ is shown in Fig. 16. For the parameters of the figure, the parallel n-fold way algorithm is almost twice as fast as the parallel standard dynamic Monte Carlo algorithm. For the parallel n-fold way algorithm it was found that the average utilization $\langle u \rangle$ was about $\frac{1}{2}$ (Fig. 17). This utilization is lower than the utilization of the parallel standard Monte Carlo simulation (Fig. 17). Nevertheless, as seen in Fig. 16 the n-fold way performs better in the number of spin flip trials per second per PE than does the parallel standard Monte Carlo algorithm. The introduction of the shield spins as an absorbing class limits the performance of the parallel n-fold way algorithm at low temperatures [115]. This is because at low temperatures the flip probabilities for each class, p_i, are small and the absorption almost always will take place in the class with the $N_{\rm b}$ shield spins. In this case the average time step is limited by the block size, and is approximately $\langle \Delta t \rangle \approx \ell/4$ [115].

6.0 Summary and Discussion

Several advanced dynamic Monte Carlo algorithms have been introduced. All these algorithms preserve the underlying dynamic of the model — they just implement it in a more

intelligent fashion on the computer. One algorithm is the Monte Carlo with Absorbing Markov Chains (MCAMC) algorithm. The lowest-order, s=1, MCAMC algorithm corresponds to a discrete-time version of the n-fold way algorithm [33, 34]. The other algorithm is the projective dynamics algorithm. We have shown that these advanced algorithms can be applied to models to study escape from metastable states. In all cases, the advanced algorithms do *not* alter the underlying dynamic of the model, but only implement the underlying dynamic in a different fashion. These algorithms can be *many* orders of magnitude faster than the standard Monte Carlo simulations. Consequently, they allow simulations to be performed in parameter regimes it is impossible to study with standard algorithms.

The MCAMC method was applied to a simple lattice model for motion of a random walker through a one-dimensional energy landscape in Sec. 4.0. This is a preliminary problem to using such methods to simulate motion of a particle through a free-energy landscape caused by a pore, such as ionic motion through a biological ion channel. It was shown that the n-fold way can be much faster than standard Monte Carlo. The n-fold way algorithm, sometimes called event-driven simulation or a rejection-free algorithm, corresponds to s=1 MCAMC. In this model the higher s MCAMC algorithms were introduced and found to sometimes be orders of magnitude faster than the n-fold simulations. For this simple model, the projective dynamics algorithm was found to be solvable for a finite lattice. For more realistic simulations of motion through pores this will no longer be the case, but the projective dynamics algorithm is expected to again be applicable.

Metastable decay of the Ising ferromagnet was illustrated using the MCAMC and projective dynamics algorithms. Again, the higher s MCAMC algorithms were shown to often be many, many orders of magnitude faster than lower s MCAMC algorithms or standard Monte Carlo. The disadvantage of the MCAMC algorithms is that they require a substantial amount of bookkeeping. On the other hand, the projective dynamics algorithm can also be many orders of magnitude faster than conventional simulations, and it does not require very much bookkeeping. In addition, the projective dynamics method can provide the magnetization of the metastable and stable states as well as the intervening saddle point. However, if higher moments of the distribution of escape times are desired, the projective dynamics algorithm only provides approximations. The MCAMC algorithms were found to work best at low temperatures in reasonably strong fields: near the single-droplet regime of the 'metastability phase diagram'. The projective dynamics algorithm was shown to also work in the multi-droplet regime.

Although both the MCAMC and projective dynamics algorithms were initially designed for dynamic Monte Carlo simulations of models with discrete states, they have recently started to be generalized to spin models with continuous degrees of freedom. Here although there is an underlying spin dynamic [124] which is used for example in micromagnetic calculations [125, 126], in some cases it is still possible to ascribe a physical meaning to a Monte Carlo move [127]. For the classical Heisenberg model recent exploratory efforts to apply the n-fold way [129, 130] and the projective dynamics methods [128] have been encouraging.

The advanced dynamic Monte Carlo algorithms presented here have the goal of accelerating the simulation of dynamics for physical systems. Methods with the same goal for molecular dynamics simulations have also recently been advanced by Voter. These include the hyperdynamics [131], parallel replica dynamics [132], and temperature-extrapolated

dynamics [133]. These three methods can be used either individually or in combination for molecular dynamics simulations. They all assume that transition-state theory [134, 135] is satisfied. This assumption is related to the assumption in the projected dynamics that the ratio of growing and shrinking probabilities is given by Boltzmann weights. As shown in [66] for the Ising model, such an assumption works reasonably well in the single-droplet regime but fails in the multidroplet regime. Furthermore, in Ref. [66] the prefactor of the nucleation rate was shown to depend on the explicit dynamic used, even though the dynamics all satisfied assumptions similar to transition-state theory. It can be anticipated that only in the regimes where the growth phase is unimportant will the transition-state theory assumptions be satisfied, and only there will Voter's advanced molecular dynamics algorithms give correct results. Conversely, neither the MCAMC nor the projective dynamics methods presented here depend on any assumptions related to transition-state theory. They are just different ways to implement the original dynamic. Hence these advanced dynamic Monte Carlo algorithms give the correct result.

Another recent dynamic Monte Carlo algorithm called Transition Matrix Monte Carlo (TMMC) has been proposed by Wang, Tay, and Swendsen [136]. The idea is very similar to that of projective dynamics [84, 87]. The underlying $2^N \times 2^N$ Markov matrix of the N spin Ising model is lumped using equations identical to Eq. (57) and (61) but with energy rather than magnetization as the lumping variable. Wang et al. study the square-lattice Ising model at T_c in zero field. The lumped Markov matrix they obtain is not tridiagonal like the projective dynamics Markov matrix in Sec. 5.4, so our proof that the dynamic is unaffected by the lumping is no longer applicable. In fact, they obtain dynamic relaxation times different from those of the unlumped Markov matrix. Consequently, the TMMC method must alter the underlying dynamics of the model. It would be interesting to project onto both slow variables, M and energy, for the Ising model [78, 79]. This can be done [66] using the formal relationship between the projective dynamics and the Nakjima-Zwanzig [137, 138] projection-operator formalism for the master equation, which is equivalent to Mori's [139] projection-operator formalism for the equations of motion of observables [140, 141]. Projecting onto both slow variables should make the underlying Markov matrix closer to weakly lumpable.

In summary, the projective dynamics and MCAMC algorithms allow dynamic Monte Carlo simulations to be performed cheaper, better, faster — without changing the underlying dynamic. They should be used in all situations where long time dynamic Monte Carlo simulations are required to understand the underlying physics of the discrete-state-space model under investigation.

Acknowledgments

The author gratefully acknowledges discussions with M. Kolesik, G. Korniss, J. Lee, S. J. Mitchell, J. D. Muñoz, R. A. Ramos, H. L. Richards, and P. A. Rikvold. This work was supported by the U.S. National Science Foundation Grant No. DMR-9871455. This research used the resources of the National Energy Research Scientific Computer Center (NERSC), which is supported by the Office of Energy Research of the US DOE under Contract No. DE-AC03-76SF00098.

References

[1] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, J. Chem. Phys. **21**, 1087 (1953).

[2] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Computing* (Cambridge Univ. Press, Cambridge, 1986).

[3] H. E. Stanley, *Introduction to Phase Transitions and Critical Phenomena* (Oxford Univ. Press, New York, 1971).

[4] K. Binder, in *Monte Carlo Methods in Statistical Physics*, Second Edition, edited by K. Binder (Springer-Verlag, Berlin, 1986).

[5] K. Binder and D. W. Heermann, *Monte Carlo Simulation in Statistical Physics*, Third Edition, (Springer-Verlag, New York, 1997).

[6] R. H. Swendsen and J.-S. Wang, Phys. Rev. Lett. **58**, 86 (1987).

[7] U. Wolff, Phys. Rev. Lett. **62**, 361 (1989).

[8] L. Chayes and J. Machta, Physica A **239**, 542 (1997); Physica A **254**, 477 (1998).

[9] B. Berg and T. Neuhaus, Phys. Lett. B **267**, 249 (1991).

[10] B. Berg and T. Neuhaus, Phys. Rev. Lett. **68**, 9 (1992).

[11] B. A. Berg, U. Hansmann, and T. Neuhaus, Phys. Rev. B **47**, 497 (1993).

[12] B. A. Berg, U. Hansmann, and T. Neuhaus, Z. Phys. B **90**, 229 (1993).

[13] J. Lee, Phys. Rev. Lett. **71**, 211 (1993); erratum **71**, 2353 (1993).

[14] E. Marinari and G. Parisi, Europhys. Lett. **19**, 451 (1992).

[15] W. Kerler and P. Rehberg, Phys. Rev. E **50**, 4220 (1994).

[16] J. P. Valleau and D. N. Card, J. Chem. Phys. **57**, 5457 (1972).

[17] E. P. Münger and M. A. Novotny, Phys. Rev. B **43**, 5773 (1991).

[18] A. M. Ferrenberg and R. H. Swendsen, Phys. Rev. Lett. **61**, 2635 (1988).

[19] R. J. Glauber, J. Math. Phys. **4**, 294 (1963).

[20] K. Binder and E. Stoll, Phys. Rev. Lett. **31**, 47 (1973).

[21] K. Binder and H. Müller-Krumbhaar, Phys. Rev. B **9**, 2328 (1974).

[22] Ph. A. Martin, J. Stat. Phys. **16**, 149 (1977).

[23] P. A. Rikvold, H. Tomita, S. Miyashita, and S. W. Sides, Phys. Rev. E **49**, 5080 (1994).

[24] P. A. Rikvold and B. M. Gorman, in *Annual Reviews of Computational Physics I*, edited by D. Stauffer (World Scientific, Singapore, 1994), p. 149.

[25] M. Iosifescu, *Finite Markov Processes and their Applications* (Wiley, New York, 1980), Chapter 3.

[26] W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains* (Princeton Univ. Press, Princeton, New Jersey, 1994).

[27] D. P. Chen, V. Barcilon, and R. S. Eisenberg, SIAM J. Appl. Math **52**, 1405 (1992).

[28] V. Barcilon, D. P. Chen, R. S. Eisenberg, and M. Ratner, J. Chemical Physics **98**, 1193 (1993).

[29] E. Barkai, R. S. Eisenberg, and Z. Schuss, Phys. Rev. E **54**, 1161 (1995).

[30] S. Bergling, Biophys. Chem. **56**, 227 (1995).

[31] D. P. Chen, J. Lear, and R. S. Eisenberg, Biophys. Journal **72**, 91 (1997).

[32] C. Hartnig, W. Witschel, and E. Spohr, J. Phys. Chem. B **107**, 1241 (1998).

[33] A. B. Bortz, M. H. Kalos, and J. L. Lebowitz, J. Comput. Phys. **17**, 10 (1975).

[34] M. A. Novotny, Computers in Physics **9**, 46 (1995).

[35] G. H. Gilmer, J. Crystal Growth **35**, 15 (1976).

[36] P. A. Maksym, Semicond. Science Technol. **3**, 594 (1988).

[37] S.-k. Ma, Phys. Rev. Lett. **37**, 461 (1976).

[38] P. Jensen, N. Combe, H. Larralde, J. L. Barrat, C. Misbah, and A. Pimpinelli, Euro. Phys. J. B **11**, 497 (1999).

[39] P. Jensen, Rev. Mod. Phys. **71**, 1695 (1999).

[40] M. E. J. Newman and C. Moore, Phys. Rev. E **60**, 5068 (1999).

[41] P. Sollich and M. R. Evans, Phys. Rev. Lett. **83**, 3238 (1999).

[42] R. Melin, H. Li, N. S. Wingreen, and C. Tang, J. Chem. Phys. **110**, 1252 (1999).

[43] S. H. Charap, P.-L. Lu, and Y. He, IEEE Trans. Magn. **33**, 978 (1997).

[44] P. S. Sahni, G. S. Grest, M. P. Anderson, and D. J. Srolovitz, Phys. Rev. Lett. **50**, 263 (1983); G. S. Grest, M. P. Anderson, and D. J. Srolovitz, Phys. Rev. B **38**, 4752 (1988).

[45] H. C. Kang and W. H. Weinberg, Chemical Rev. **95**, 667 (1995); J. Chem. Phys. **90**, 2824 (1989).

[46] J. J. Lukkien, J. P. L. Segers, P. A. J. Hilbers, R. J. Gelten, and A. P. J. Jansen, Phys. Rev. E **58**, 2598 (1998).

[47] W. Krauth and M. Mezard, Z. Phys. B **97**, 127 (1995).

[48] G. S. Grest, C. M. Soukoulis, and K. Levin, Phys. Rev. Lett. **56**, 1148 (1986); Phys. Rev. B **33**, 7659 (1986).

[49] T. Graim and D. P. Landau, Phys. Rev. B **24**, 5156 (1981).

[50] S. Ozawa, T. Kobayashi, S. Kamata, and T. Haseda, J. Magn. and Magn. Mater. **90**&**91**, 287 (1990).

[51] J. W. Greene and K. J. Supowit, IEEE Trans. Computer-Aided Design **CAD-5**, 221 (1986).

[52] P. Peczak and M. J. Luton, Acta Metall. Mater. **41**, 59 (1992).

[53] F. M. Bulnes, V. D. Pereyra, and J. L. Riccardo, Phys. Rev. E **58**, 86 (1998).

[54] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*, Second Edition (McGraw-Hill, New York, 1991).

[55] J. D. Gunton and M. Droz, *Introduction to the Theory of Metastable and Unstable States* (Springer-Verlag, Berlin, 1983).

[56] D. Stauffer, A. Coniglio, and D. W. Heermann, Phys. Rev. Lett. **49**, 1299 (1982).

[57] P. A. Rikvold, M. A. Novotny, M. Kolesik, and H. L. Richards, in *Dynamical Properties of Unconventional Magnetic Systems*, edited by A. T. Skjeltorp and D. Sherrington (Kluwer, Dordrecht, 1998), p. 307.

[58] A review of early Monte Carlo work on metastable decay is given by K. Binder, in *Phase Transitions and Critical Phenomena*, edited by C. Domb and M. Green (Academic, London, 1976), Vol. 5B.

[59] H. L. Richards, M. Kolesik, P.-A. Lindgård, P. A. Rikvold, and M. A. Novotny, Phys. Rev. B **55**, 11521 (1997).

[60] M. Kolesik, H. L. Richards, M. A. Novotny, P. A. Rikvold, and P.-A. Lindgård, J. Appl. Phys. **81**, 5600 (1997).

[61] H. L. Richards, M. A. Novotny, and P. A. Rikvold, Phys. Rev. B **54**, 4113 (1996).

[62] A. N. Kolmogorov, Bull. Acad. Sci. USSR, Phys. Ser. **1**, 355 (1937); W. A. Johnson and R. F. Mehl, Trans. Am. Inst. Min. Metall. Pet. Eng. **135**, 416 (1939); M. Avrami, J. Chem. Phys. **7**, 1103 (1939); M. Avrami, J. Chem. Phys. **8**, 212 (1940); M. Avrami, J. Chem. Phys. **9**, 177 (1941).

[63] R. A. Ramos, P. A. Rikvold, and M. A. Novotny, Phys. Rev. B **59**, 9053 (1999).

[64] H. Tomita and S. Miyashita, Phys. Rev. B **46**, 8886 (1992).

[65] E. Jordão Neves and R. H. Schonmann, Commun. Math. Phys. **137**, 209 (1991).

[66] J. Lee, M. A. Novotny, and P. A. Rikvold, Phys. Rev. E **52**, 356 (1995).

[67] M. A. Novotny in *Computer Simulation Studies in Condensed Matter Physics IX*, edited by D. P. Landau, K. K. Mon, and H.-B. Schüttler, (Springer-Verlag, Heidelberg, 1997), p. 182.

[68] P. Dehghanpour and R. H. Schonmann, Commun. Math. Phys. **188**, 89 (1997).

[69] J. S. Langer, Ann. Phys. (N.Y.) **41**, 108 (1967); **54**, 258 (1969).

[70] N. J. Günther, D. A. Nicole, and D. J. Wallace, J. Phys. A **13**, 1755 (1980).

[71] There is a misprint near the bottom of page 13 in the original n-fold way paper of Ref. [33]. The randomly chosen integer should be in the range $[1, n_i]$ to find LCH, not $[1, m_i]$. I thank B. M. Gorman for pointing out this misprint.

[72] A. M. Ferrenberg, D. P. Landau, and Y. J. Wong, Phys. Rev. Lett. **69**, 3382 (1992).

[73] M. Marcus and H. Minc, *A Survey of Matrix Theory and Matrix Inequalities* (Allyn and Bacon, Boston, 1964).

[74] D. H. Bailey, ACM Trans. Math. Software **21**, 379 (1995). The MPFUN package, written by D. H. Bailey, can be found at http://netlib.uow.edu.au/mpfun/

[75] A. M. Abdel-Moneim and F. W. Leysieffer, J. Appl. Prob. **19**, 685 (1982).

[76] S. Arrhenius, Z. Phys. Chem. (Leipzig) **4**, 226 (1889).

[77] L. S. Schulman, J. Phys. A **13**, 237 (1980).

[78] I. Shteto, J. Linares, and F. Varret, Phys. Rev. E **56** 5128 (1997).

[79] I. Shteto, K. Boukheddaden, and F. Varret, Phys. Rev. E **60** 5139 (1999).

[80] B. A. Berg, Computer Phys. Commun. **69**, 7 (1992).

[81] J. Topping, *Errors of Observation and Their Treatment*, 4^{th} edition, (Chapman and Hall, London, 1972).

[82] M. A. Novotny, M. Kolesik, and P. A. Rikvold, Computer Phys. Commun. **121–122**, 330 (1999).

[83] I thank S. J. Mitchell for this proof.

[84] M. Kolesik, M. A. Novotny, P. A. Rikvold, and D. M. Townsley, in *Computer Simulation Studies in Condensed Matter Physics X*, edited by D. P. Landau, K. K. Mon, and H.-B. Schüttler, (Springer-Verlag, Heidelberg, 1998), p. 246.

[85] M. Kolesik, M. A. Novotny, and P. A. Rikvold, in *Microscopic Simulation of Interfacial Phenomena in Solids and Liquids*, edited by S. R. Phillpot, P. D. Bristowe, D. G. Stroud, and J. R. Smith, Materials Research Society Symp. Proc. **492**, 313 (1998).

[86] M. Kolesik, M. A. Novotny, and P. A. Rikvold, Phys. Rev. B **56**, 11791 (1997).

[87] M. Kolesik, M. A. Novotny, and P. A. Rikvold, Phys. Rev. Lett. **80**, 3384 (1998).

[88] M. A. Novotny, Int. J. Mod. Phys. C **10**, 1483 (2000).

[89] P. A. Rikvold and M. Kolesik, J. Stat. Phys. **100**, 377 (2000).

[90] D. Stauffer and R. Knecht, Int. J. Mod. Phys. C **7**, 893 (1996).

[91] M. A. Novotny, M. Kolesik, D. M. Townsley, and P. A. Rikvold, unpublished.

[92] D. Stauffer, Int. J. Mod. Phys. C **10**, 809 (1999).

[93] C. C. A. Günther, P. A. Rikvold, and M. A. Novotny, Phys. Rev. Lett. **71**, 3898 (1993).

[94] C. C. A. Günther, P. A. Rikvold, and M. A. Novotny, Physica A **212** 194 (1994).

[95] D. Chen, J. Feng, and M. Qian, Science in China **40**, 832 (1997).

[96] D. Chen, J. Feng, and M. Qian, Science in China **40**, 1129 (1997).

[97] B. D. Lubachevsky, J. Comput. Phys. **75**, 103 (1988).

[98] H. G. Allore and H. N. Erb, Preventive Veterinary Medicine **39**, 279 (1999).

[99] A. Hirshberg, M. Stein, and R. Walden, J. of Trauma-Injury Infection and Critical Care **47**, 545 (1999).

[100] S. D. Smith, Civil Engineering and Environmental Systems **16**, 93 (1999).

[101] E. Powell, Aerospace Engineering **19**, 25 (1999).

[102] G. Nicolis and I. Prigogine, *Self-Organization in Nonequilibrium Systems* (Wiley, New York, 1977).

[103] K. Glass, M. Livingston, and J. Conery, Proc. of the 11^{th} Workshop on Parallel and Distributed Simulation, PADS '97 (IEEE Computer Society, Los Alamitos, California, 1997), p. 60.

[104] N. Fröhlich, R. Schlangenhaft, and J. Fleischmann, Proc. of the 11^{th} Workshop on Parallel and Distributed Simulation, PADS '97 (IEEE Computer Society, Los Alamitos, California, 1997), p. 64.

[105] J. Keller, T. Rauber, and B. Rederlechner, VLSI Design **9**, 219 (1999).

[106] C. Peng and F. F. Chen, Computers & Industrial Engineering **31**, 327 (1996).

[107] J. B. Hiller and T. C. Hartrum, Proc. of the 11^{th} Workshop on Parallel and Distributed Simulation, PADS '97 (IEEE Computer Society, Los Alamitos, California, 1997), p. 12.

[108] S. C. Borst, S. A. Grandhi, C. L. Kahn, K. Kumaran, B. D. Lubachevsky, and D. M. Sand, Bell Labs Technical Journal **2**, 81 (1997).

[109] D. R. Jefferson, Assoc. Comput. Mach. Trans. Programming Languages and Systems **7**, 404 (1985).

[110] R. Fujimoto, Commun. of the ACM **33**, 30 (1990).

[111] B. Lubachevsky, A. Shwartz, and A. Weiss in *Proceedings of the 1989 Winter Simulation Conference*, edited by E. A. MacNair, K. J. Musselman, and P. Heidelberger (Association for Computing Machinery, New York, 1989), p. 630.

[112] B. D. Lubachevsky, A. Weiss, and A. Shwartz, ACM Trans. on Modeling and Computer Simulations **1**, 154 (1991).

[113] C. N. Tay, Y. M. Teo, and S. T. Kong, Proc. of the 11^{th} Workshop on Parallel and Distributed Simulation, PADS '97 (IEEE Computer Society, Los Alamitos, California, 1997), p. 116.

[114] S. C. Tay, Y. M. Teo, and R. A. Ayani, Proc. of the 12^{th} Workshop on Parallel and Distributed Simulation, PADS '98 (IEEE Computer Society, Los Alamitos, California, 1998), p. 30.

[115] G. Korniss, M. A. Novotny, and P. A. Rikvold, J. Comput. Phys. **153**, 488 (1999).

[116] G. Korniss, G. Brown, M. A. Novotny, and P. A. Rikvold, in *Computer Simulation Studies in Condensed Matter Physics XI*, edited by D. P. Landau and H.-B. Schüttler, (Springer-Verlag, Heidelberg, 1999), p. 134.

[117] G. Korniss, Z. Toroczkai, M. A. Novotny, and P. A. Rikvold, Phys. Rev. Lett. **84**, 1351 (2000).

[118] G. Korniss, M. A. Novotny, and P. A. Rikvold, unpublished.

[119] G. Korniss, M. A. Novotny, Z. Toroczkai, and P. A. Rikvold, in *Computer Simulation Studies in Condensed Matter Physics XIII* edited by D. P. Landau, S. P. Lewis, and H.-B. Schüttler, (Springer-Verlag, Heidelberg, 2001), p. 119.

[120] S. F. Edwards and D. R. Wilkinson, Proc. R. Soc. London, Ser. A **381**, 17 (1982).

[121] A.-L. Barabási and H. E. Stanley, *Fractal Concepts in Surface Growth* (Cambridge Univ. Press, Cambridge, 1995). Detailed discussion and copious references to the theoretical and experimental literature of non-linear surface growth can be found here.

[122] Z. Toroczkai, G. Korniss, S. Das Sarma, and R. K. P. Zia, Phys. Rev. E, **62**, 276 (2000).

[123] J. Krug and P. Meakin, J. Phys. A: Math. & General **23**, L987 (1990).

[124] M. Krech and D. P. Landau, Phys. Rev. B **60**, 3375 (1999); S. H. Tsai, A. Bunker, and D. P. Landau, Phys. Rev. B **61**, 333 (2000).

[125] A. Aharoni, *Introduction to the Theory of Ferromagnetism*, International Series of Monographs on Physics (Oxford Univ. Press, Oxford, 1996).

[126] G. Brown, M. A. Novotny, and P. A. Rikvold, J. Appl. Phys. **87**, 4792 (2000).

[127] U. Nowak, R. W. Chantrell, and E. C. Kennedy, Phys. Rev. Lett. **84**, 163 (2000); U. Nowak in *Annual Reviews of Computational Physics IX*, edited by D. Stauffer, (World Scientific, Singapore, 2001), p. 105.

[128] S. J. Mitchell, M. A. Novotny, and J. D. Muñoz, Int. J. Mod. Phys. C **10**, 1503 (2000).

[129] J. D. Muñoz, M. A. Novotny and S. J. Mitchell, in *Computer Simulation Studies in Condensed Matter Physics XIII*, edited by D. P. Landau, S. P. Lewis, and H.-B. Schüttler, (Springer-Verlag, Heidelberg, 2001), p. 92.

[130] J. D. Muñoz, M. A. Novotny and S. J. Mitchell, unpublished.

[131] A. F. Voter, Phys. Rev. Lett. **78**, 3908 (1997).

[132] A. F. Voter, Phys. Rev. B **57**, R13985 (1998).

[133] M. R. Sorensen and A. Voter, J. Chem. Phys., **112**, 9599 (2000).

[134] H. S. Johnston, *Gas Phase Reaction Rate Theory* (Ronald Press Company, New York, 1966).

[135] T. Baer and W. L. Hase, *Unimolecular Reaction Dynamics*, International Series of Monographs on Chemistry, (Oxford Univ. Press, New York, 1996).

[136] J.-S. Wang, T. K. Tay, and R. H. Swendsen, Phys. Rev. Lett. **82**, 476 (1999).

[137] S. Nakajima, Prog. Theor. Phys. **20**, 948 (1958).

[138] R. Zwanzig, J. Chem. Phys. **33**, 1338 (1960).

[139] H. Mori, Prog. Theor. Phys. **33**, 423 (1965).

[140] H. Grabert, Z. Phys. B **26**, 79 (1977).

[141] H. Grabert, *Projection Operator Techniques in Nonequilibrium Statistical Mechanics* (Springer-Verlag, Berlin, 1982), and references therein.

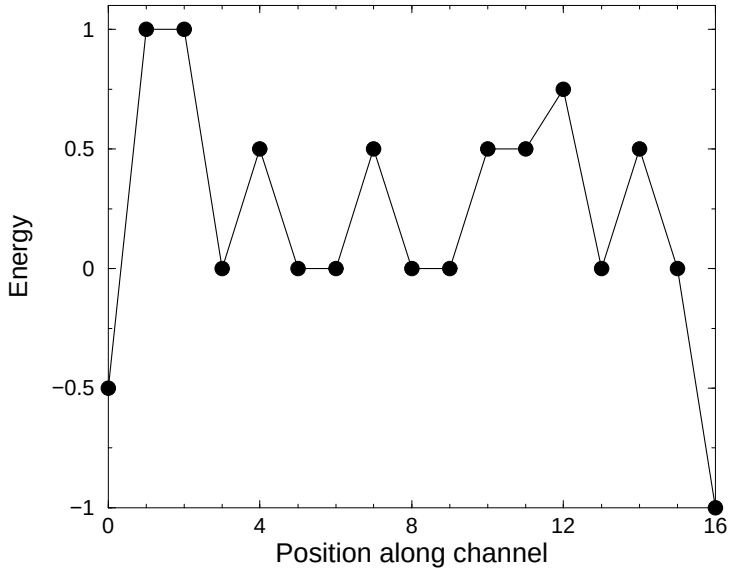


Figure 1: The dimensionless energy of each of the sites of a simple model for motion of a particle through a pore.

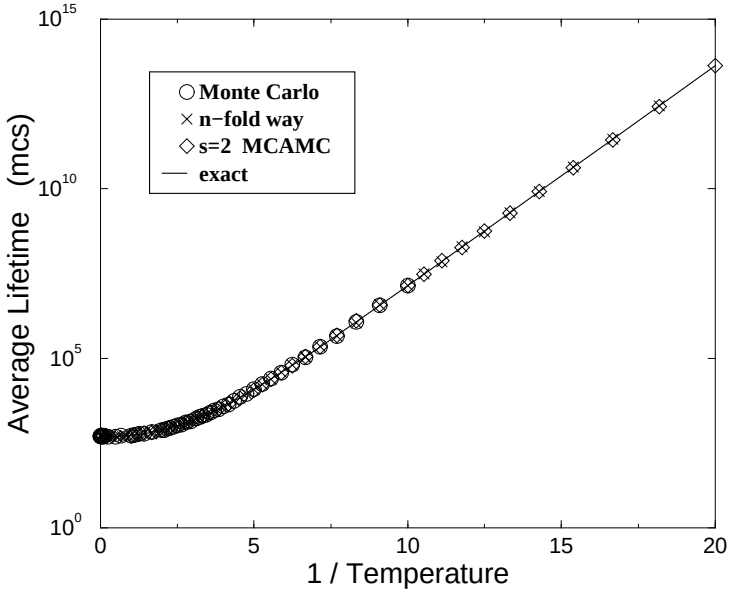
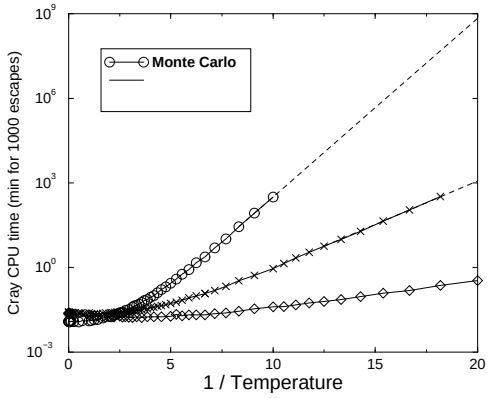


Figure 2: The average lifetime, in Monte Carlo steps (mcs), as a function of inverse temperature for particle motion in the energy arrangement of Fig. 1. See Fig. 3 for an explanation of the plotting symbols. The solid line is the exact result using Eqs. (30), (32), and (33).


Monte Carlo
Cray CPU time (min for 1000 escapes)
1 / Temperature
10^{9}
10^{6}
10^{3}
10^{0}
10^{-3}
0
5
10
15
20

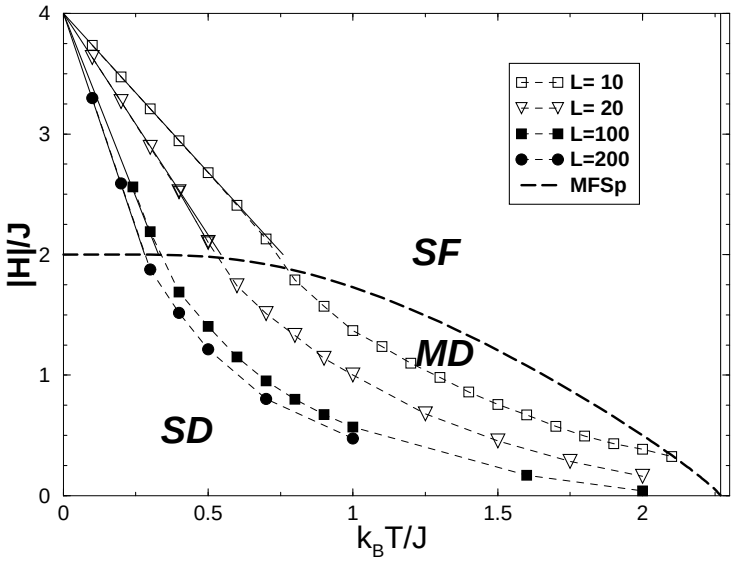


Figure 4: The cross-over 'phase diagram for metastable decay' in the square-lattice Ising ferromagnet is shown. The strong-field (SF) regime, the multi-droplet (MD) regime, and the single-droplet (SD) regime are shown. The heavy dashed line separates the SF and MD regimes and is estimated using $R_{\rm c}=a/2$. The cross-overs between the SD and MD regimes depend on the system size and are shown for L=10, 20, 100, and 200. The solid lines starting at $|H|=4J$ and T=0 are the single overturned spin cross-over between the SF and SD regimes, as described in the text. The vertical solid line shows the location of the critical temperature, $T_{\rm c}$.

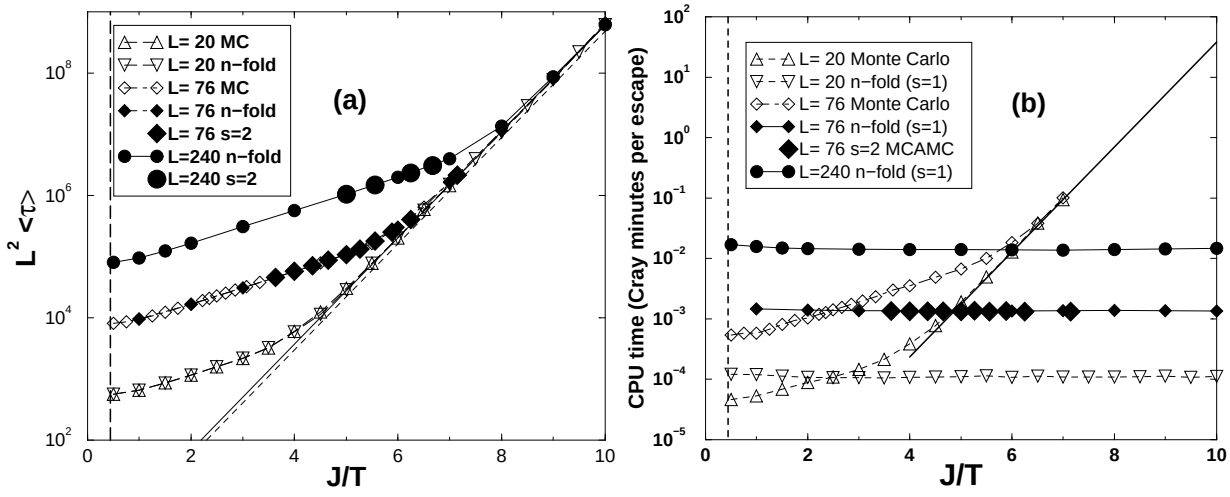


Figure 5: For $H=-3J$ with L=20, L=76, and L=240 data are shown for different dynamic Monte Carlo algorithms. (a) L^2 times the average lifetime, $L^2\langle\tau\rangle$, for the square-lattice Ising ferromagnet is shown as a function of inverse temperature, note the units of L^2 times MCSS which is Monte Carlo steps (mcs). The vertical dashed line is the critical temperature. The dashed line corresponds to the low-temperature prediction $\langle\tau\rangle$=exp$(2J/k_{\rm B}T)$ [65], while the solid line corresponds to $\langle\tau\rangle$=(5/4)exp$(2J/k_{\rm B}T)$, which includes the exact low-temperature prefactor [67]. Note that all algorithms produce the same lifetimes within statistical errors. In (b) the CRAY CPU time in minutes per escape is shown. Note that for the n-fold way algorithm (rejection free, s=1 MCAMC) the CPU time per escape is almost flat. This is because for $2J<|H|<4J$ the nucleating droplet is a single overturned spin [65]. The s=2 MCAMC algorithm, shown only for L=76, is also approximately flat. Conversely, the standard dynamic Monte Carlo algorithm requires a CPU time that grows exponentially with inverse temperature, shown as the heavy solid line.

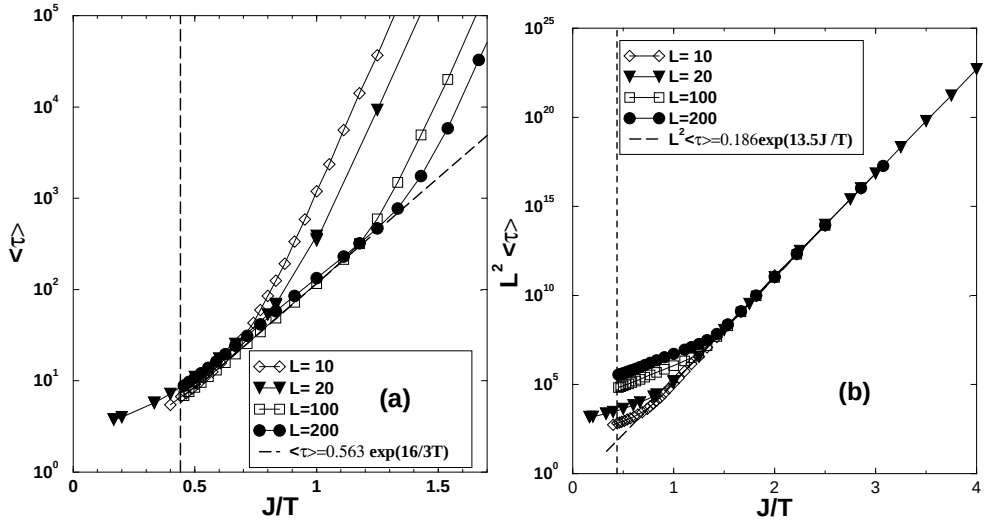


Figure 6: The average lifetime, $\langle\tau\rangle$, for the square-lattice Ising ferromagnet is shown as a function of inverse temperature for $H{=}-0.75J$ for different L. The same data are plotted in (a) and (b) in different time units. The vertical dashed line is the critical temperature. In (a) the lifetime $\langle\tau\rangle$ is shown in units of the physical time, Monte Carlo Steps per Spin (MCSS). The dashed line corresponds to $\langle\tau\rangle{=}0.563\exp(16J/3T)$ while the light solid lines connect the data points. In (b) the lifetime $\langle\tau\rangle$ is shown in units of MCSS times L^2 (or Monte Carlo steps — mcs). The heavy dashed line corresponds to $\langle\tau\rangle{=}0.186L^{-2}\exp(13.5J/T)$ and the light solid lines connect the data points. In (a) the data points lie on top of each other in the SF and MD regime since the lifetime there is independent of L. In (b) the data points in the SD regime lie on top of each other since the lifetime there is inversely proportional to the volume.

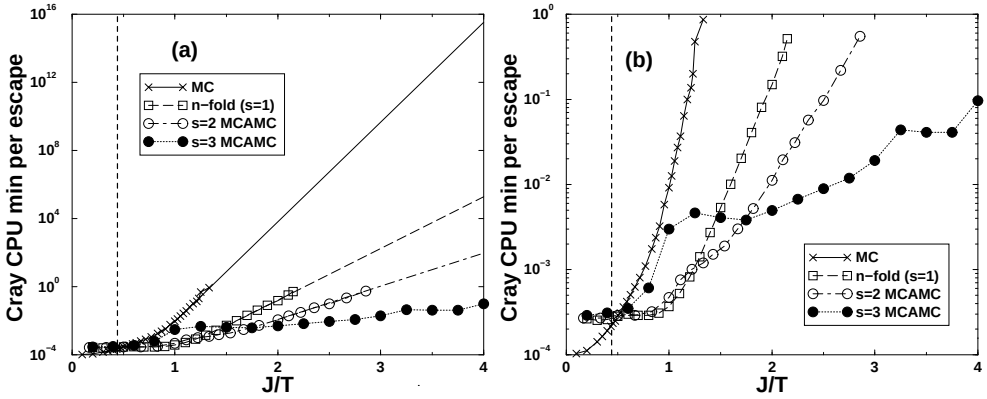


Figure 7: The CPU time required for the simulation of the 20×20 Ising ferromagnets presented in Fig. 6 is shown as a function of inverse temperature. The symbols are for standard dynamic Monte Carlo (×), n-fold way simulations (□), s=2 MCAMC (○), and s=3 MCAMC (●). (a) Uses a fit to $A_{\rm fit}\exp\left[(\Gamma-\Gamma_0)/k_{\rm B}T\right]$ to estimate the low-temperature CPU time required for the slower algorithms. This fit is given by the corresponding heavy lines. The fit parameters are listed in Table 2. (b) Shows the region where simulations were actually performed. This shows the cross-overs between the timings of the various algorithms.

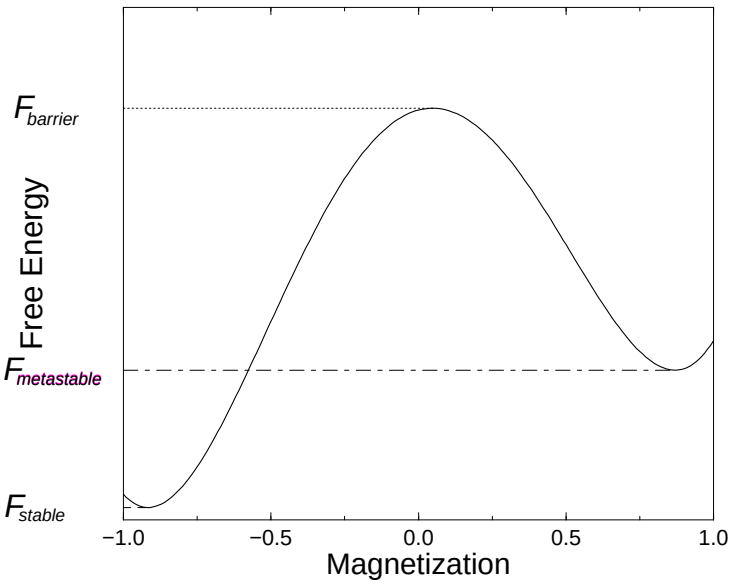


Figure 8: A schematic of the free energy per spin as a function of the total magnetization M for an Ising model. Shown are the metastable well, the stable well, and the barrier between the two wells.

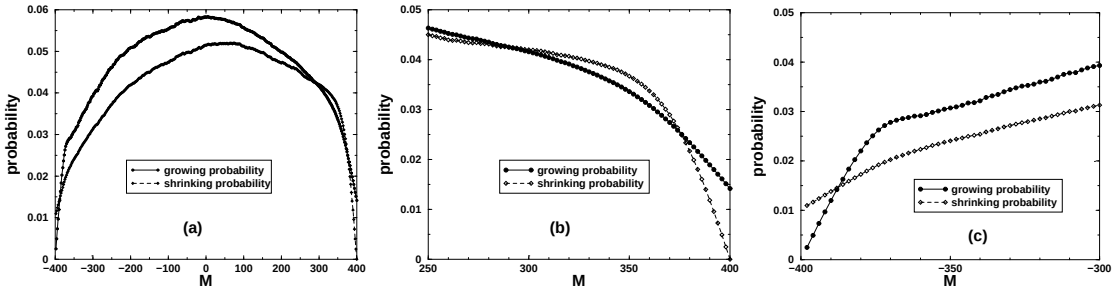


Figure 9: The growing and shrinking probabilities for escape of the square-lattice Ising model are shown as functions of the total magnetization. Wherever the shrinking and growing probabilities are equal, the system is at an extremum of the free energy. These correspond from right to left in (a) to the metastable phase, the saddle point, and the stable phase. (b) Shows the portion near the metastable phase and saddle point. (c) Shows the portion near the equilibrium phase. The parameters are L=20, T=0.8$T_{\rm c}$≈1.815J, H=−0.15J. The simulation is started with M=L^2 and is run until the first passage to M=−L^2.

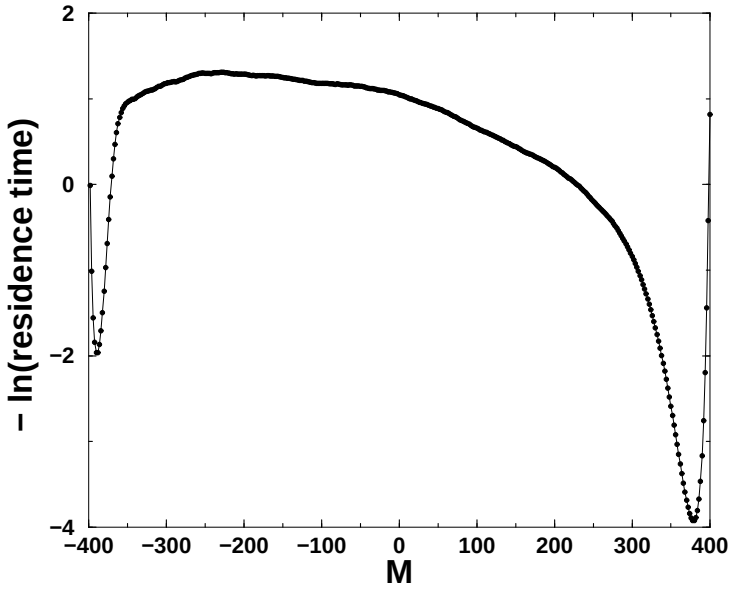


Figure 10: The negative of the natural logarithm of the residence time, $h(M)$, as a function of the system magnetization M. The parameters are the same as in Fig. 9. The integral of the residence times gives the average lifetime, which for these parameters is 1.1×10^3 MCSS. This should be compared with Fig. 8. Note that the stable phase is less probable here than in Fig. 8 since the simulation is stopped the first time the magnetization reaches M=−L^2.

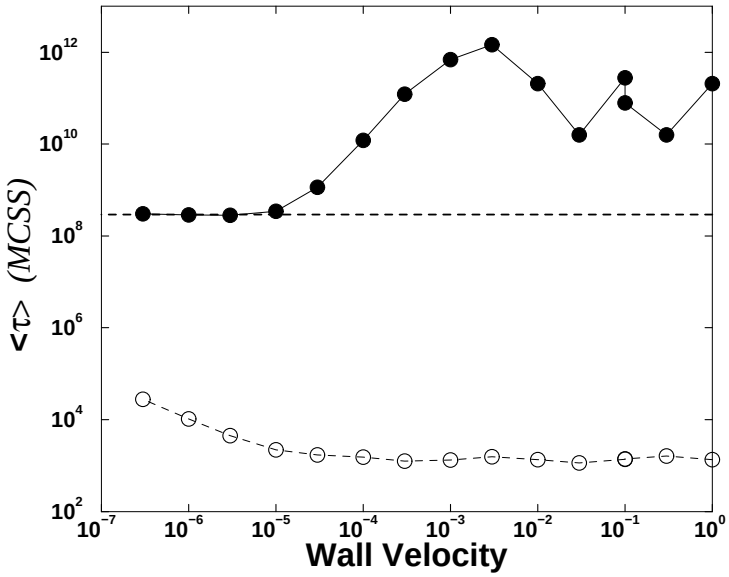


Figure 11: The average lifetime $\langle\tau\rangle$ measured from the growing and shrinking probabilities as a function of the forcing speed is shown by the filled symbols for a hard forcing wall (•). The corresponding open symbols (○) are the measured number of Monte Carlo steps per spin (MCSS) to escape from the metastable well with the forcing present. The lines connect the data points. The heavy dashed line corresponds to the result for $\langle\tau\rangle$ from standard simulations. These results are for L=20, $k_{\rm B}T$=0.5J, and $|H|$=0.75J. Note that once $\langle\tau\rangle$ is approximately independent of the forcing speed, there is a difference of about 10^5 between the results measured by the growing and shrinking probabilities and the actual escape time with the forcing wall. This 10^5 difference reflects the efficiency of the projective dynamics with a forcing wall for these parameters.

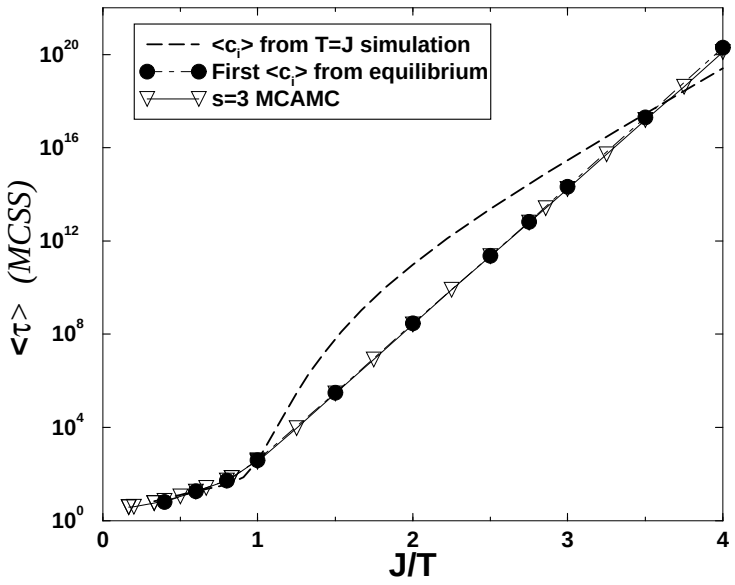


Figure 12: The average lifetime, $\langle\tau\rangle$, versus inverse temperature for L=20 and H=$-0.75J$. The s=3 MCAMC results (∇) are shown for comparison. The solid dashed curve is from a simulation at $k_{\rm B}T$=J with the assumption that the spin classes are unchanged as T changes. It has the wrong functional form. The bullets ($\bullet$) use this assumption, except the first 8 overturned spins use class populations obtained from equilibrium Monte Carlo simulations at fixed magnetization. The lines connect the data points.

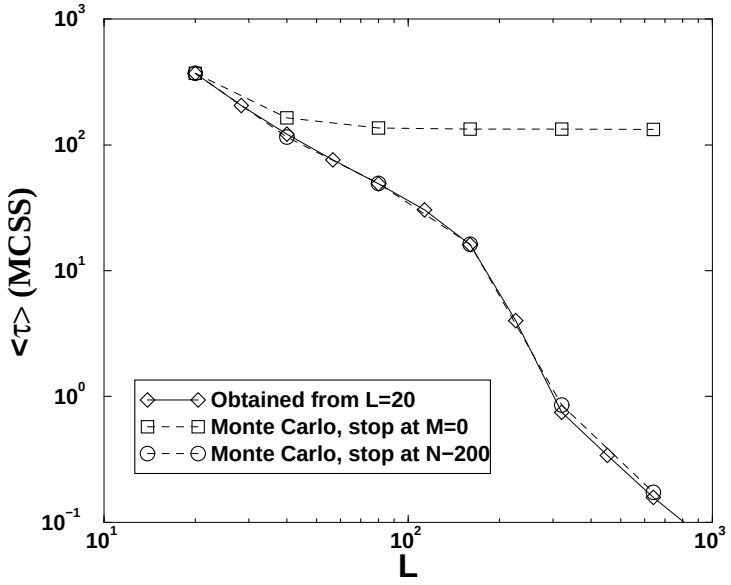
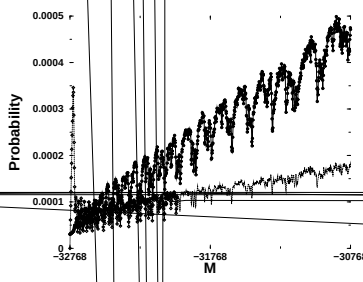


Figure 13: The lifetimes obtained from extrapolating in L using Eqs. (69) and (70) from simulations at the smallest lattice size, L=20. This is for $H=-0.75J$ and $k_{\rm B}T=J$. When the cutoff for the lifetime is kept constant at M=0, once the system is in the multi-droplet regime (for large L) the lifetime is independent of the system size (□). When the cutoff for the lifetime is not kept constant in M, but is rather constant in the number of overturned spins (here at L^2-200) for both the Monte Carlo (○) and extrapolated values (◇) the lifetime depends on L.


0.0005
0.0004
0.0003
0.0002
0.0001
Probability
0
−32768
−31768
−30768
M

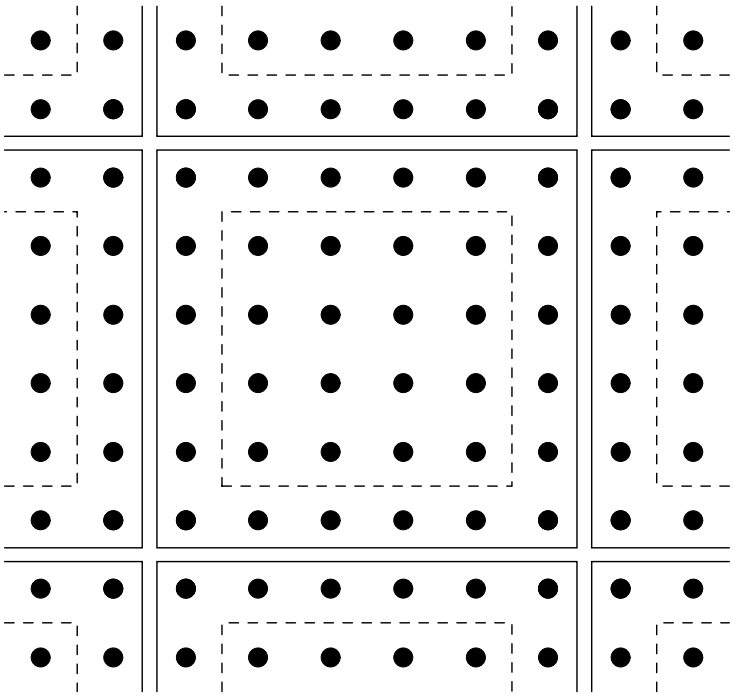


Figure 15: A portion of a lattice showing the arrangement of spins on a single processing element (PE) and its neighboring PEs. This is for ℓ=6. The solid lines represent the boundaries of spins on a single processor, while the dashed lines isolate the boundary spins from the interior spins.

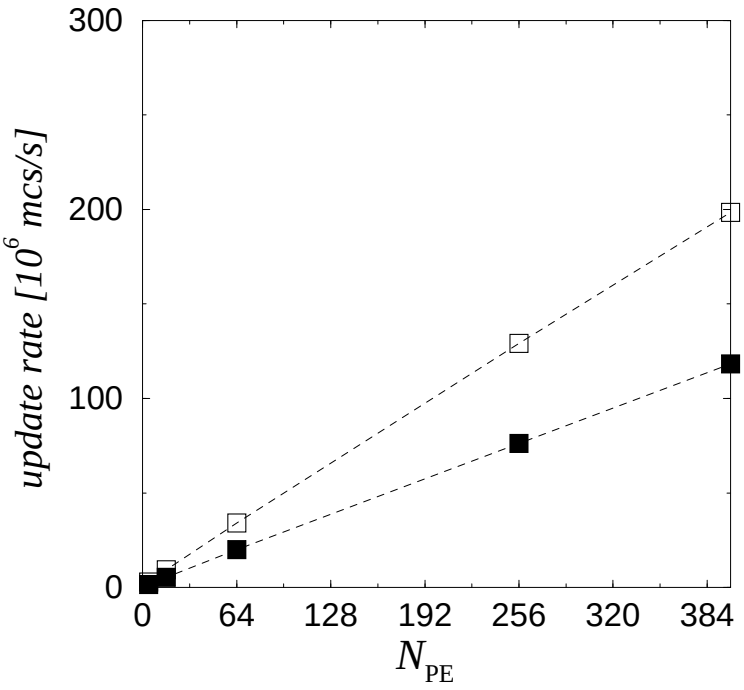


Figure 16: The update rate for the square-lattice Ising model (in units of a million Monte Carlo steps per second of wall-clock time) is shown as a function of the number of processing elements (PEs) of a Cray T3E, $N_{\rm PE}$. The algorithm is implemented using a conservative discrete-event simulation approach. The filled squares are for the standard Monte Carlo algorithm, while the open squares are for the shielded n-fold way algorithm (in continuous time). The temperature is T=0.7$T_{\rm c}$ and the applied field is $|H|$=0.2857J. Spins in blocks of $\ell\times\ell$ are placed on each PE, with ℓ=128. The size of the simulated lattice is L=$\ell\sqrt{N_{\rm PE}}$, so the largest lattice simulated has L=2560. The lines are guides for the eye. Note that a straight line fit would correspond to perfect scaling of the parallel algorithm. After Ref. [115].

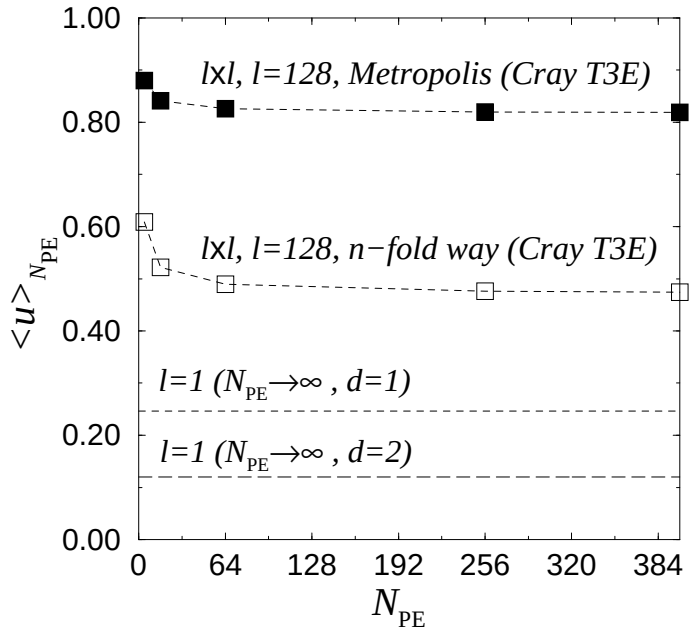


Figure 17: The average utilization, $\langle u \rangle$, is shown as a function of the number of processing elements (PEs) used. The parallelization algorithm uses a conservative discrete-event simulation approach [97]. The filled squares correspond to the standard dynamic Monte Carlo simulation for the square-lattice Ising model shown in Fig. 16, where the utilization plateaus at about 80%. Here each PE has an $\ell \times \ell$ block of spins, with ℓ=128. The average utilization for the n-fold way algorithm plateaus at about 50%. The short-dashed line corresponds to the worst-case utilization in the limit $N_{\rm PE} \to \infty$ for a one-dimensional lattice and ℓ=1 (one spin per PE). This is obtained from finite-size scaling for various lattice sizes [117]. The long-dashed line corresponds to the worst-case utilization in the limit $N_{\rm PE} \to \infty$ for a two-dimensional square lattice and ℓ=1 [119], with results obtained by running large lattice simulations.