1. Magnet to Model

The Ising Model is the next study along our trajectory to simulating and understanding the ϕ_2^4 model. In this chapter we will explore how a physical system is made tractable to simulation techniques, the algorithms designed to drive the dynamics, and the many statistical techniques used to extract the tantalizing physics behind the bits.

1.1. **Ferromagnetism and the Ising Model.**

In the previous chapter we introduced the random-walk as a brief outline for the type of thinking involved in our physical simulation work. The Ising Model is an expansion of the random-walk with physics embedded in the definition. As our motivation for the random-walks, we considered Charlie the Drunkard. We begin here with a material: the ferromagnet. These are substances we are familiar with that exert macroscopic magnetic force: iron, nickel, cobalt, molecules involving these elements, and others. A magnetic bar is all fun and games until it heats up.

A ferromagnet will lose its magnetism at a critical temperature T_c, above which the thermal energy exceeds the interaction energy and the electrons lose their orientation. To quantify the two phases of the model the mean magnetization is used

$$\langle M \rangle = \frac{1}{N^2} \sum_{i}^{N^2} s_i \tag{1}$$

the magnetization fully characterizes the phases and is used to find the phase transition. For temperatures below T_c, $\langle m \rangle \neq 0$ and settles in to a value dependent on T. The Ising model exhibits long range order of aligned spins. However, for temperatures above T_c, $\langle m \rangle = 0$ defining a phase totally disordered. The different states can be seen in Figure 1 The duality of ordered vs disordered is created by the magnetization make it the model's order parameter. Conversely, the phase at which the mean magnetization is 0 can be thought of as symmetric, whereas nonzero mean magnetization values are asymmetric. The phase transition can also be thought of as a point where symmetry is broken. An ordered state has broken symmetry and a disordered state has spin symmetry.

Knowing at what temperature this phase transition occurs is important to know if you are playing with your ferromagnet in extreme temperatures, but also to acquire an accurate description of the model. How do we probe this question by using a model similar to the random-walk?

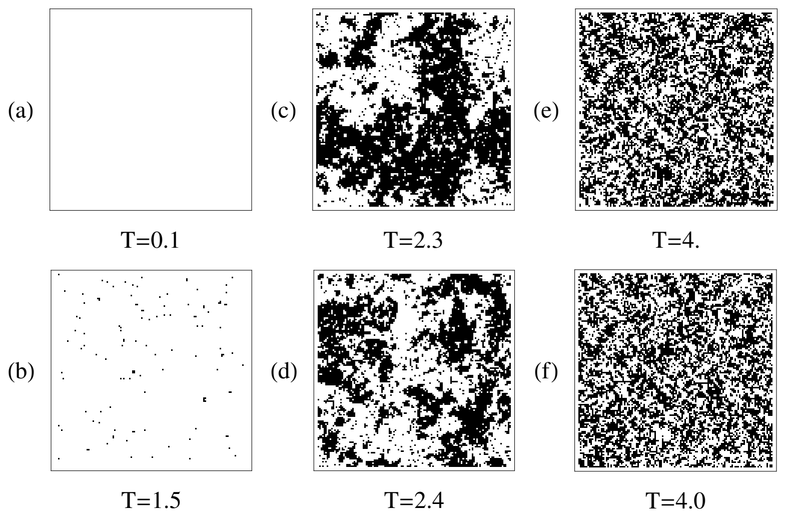

FIGURE 1. The evolution of macro state configurations for an Ising Model on a 128x128 lattice: it begins as a maximally-ordered state for low temperatures (a); a few anti-ordered spins appear (b); large domains form near the critical temperature (c & d), and the system loses all order at high temperatures (e). (f) is the inverse of (e) with no difference perceived by eye, a hallmark of a randomly disordered state.

In the random-walk, the degree of freedom we were interested in was the step direction. In the ferromagnet the only degree of freedom is the spin of the electron. There are physical effects from the charged nucleus and the orbital angular momentum of the electron. These can be neglected in our model as they have little effect on the phase transition. The physics we do have to preserve is the interaction between electrons. With further simplification, here is what we have:

(1) N x N lattice

(2) Spin 1/2 at each site

(3) Interaction energy between neighboring electrons

(4) External magnetic field

(5) An external temperature

(6) Torus boundary conditions

Boundary conditions are necessary to eliminate edge effects, resulting in a lattice that is perfectly symmetrical. The Ising Model is simulated at a range of temperatures. The dynamics of the system is driven by the Hamiltonian, defined as

$$\mathcal{H} = -J \sum_{i,j=<nn>} s_i s_j - B \sum_{i,j}^{N} s_{ij} \tag{2}$$

where J is the interaction energy between nearest neighbors and B is a parameter for an external magnetic field. In the following sections, we will consider the two-dimensional Ising Model, setting $J = 1$ and $B = 0$. The positive value we have set for J is the ferromagnetic interaction, resulting in an ordered state of aligned spins. A negative value is antiferromagnetic, giving a state with nearest spins anti-aligned. We choose a ferromagnetic interaction energy so we have a well ordered state that is easily distinguishable because of all its aligned spins and thus a higher magnetization

This is a comfortable starting point to begin our simulations because the Ising Model in two dimensions has been explicitly solved for, providing a check for our data. It also will provide an excellent cornerstone for our experimentation with the ϕ_2^4 model because it shares the same universality class, meaning the behavior close to the phase transition of both models are equivalent.

1.2. **Statistical Mechanical Descriptors.** The genesis of a statistical-mechanical analysis lies in the partition function

$$Z = \sum_s e^{-\beta E_s}, \qquad \beta = \frac{1}{k_B T} \tag{3}$$

where the sum is over all microstates of the system. This is the canonical partition function, defined for a system kept at constant temperature, volume, and number of particles, but allowed to exchange heat. Other partition functions allow for additional changes, but we will be working with our Ising Model in the canonical ensemble.

The partition function bears many fruit. Pertinent thermodynamic quantities can be extracted from it along with a definition for the dynamics of the system. The power of Z is obvious by the simplicity of the equations you can write for the ensemble average energy, heat capacity, Helmholtz free energy, magnetization, and susceptibility

$$\langle E\rangle = -\frac{\partial lnZ}{\partial\beta} \tag{4}$$

$$C_v = \frac{1}{k_BT^2}\frac{\partial^2 lnZ}{\partial\beta^2} \tag{5}$$

$$A = -k_BTlnZ \tag{6}$$

$$M = \frac{\partial A}{\partial B} \tag{7}$$

$$\chi = \frac{\partial M}{\partial B} \tag{8}$$

A more powerful use of the partition functions yields the probability of our system existing as one configuration in a set of microstates. The expression

$$P_s = \frac{1}{Z}e^{-\beta E_s} \tag{9}$$

will later guide our approach to experimenting with the Ising Model by Monte Carlo simulation.

2. Explicit Solutions of the Ising Model

The one-, and two-dimensional Ising Model have both been explicitly solved for. We will tour both solutions as one builds into the other and to juxtapose the idealized approach of finding an explicit solution with the experimental Monte Carlo method for messier problems. It will be seen that data from the Monte Carlo simulations reproduce the explicit functions, thus providing a powerful method to obtain behavior of systems that do not have accompanying explicit solutions

2.1. **The Ising Model in One dimension.** The Ising Model in a single dimension is simply a string of spins interacting with their two nearest neighbors. It was first solved by Ising [7], hence gaining his name. We again have to deal with the two endpoints by connecting them, resulting in the topology of a loop. The Hamiltonian for the one-dimensional Ising Model is simply (2) reduced by one dimension. Taking the coupling strength to be constant we have

$$\mathcal{H} = -J\sum_i^N s_is_i + 1 - B\sum_i^N s_i \tag{10}$$

The partition function (3) is then

$$Z = \sum_{s_1} ... \sum_{s_N} e^{-\beta \mathcal{H}} \tag{11}$$

where we sum over all possible configurations of spins s_i. To progress to a solution [6] we resort to the transfer-matrix method, a reexpression of the Boltzmann factor weighting. The transfer matrix P is defined with elements

$$\langle s|P|s'\rangle = e^{-\beta(Jss'+Bs)} \tag{12}$$

where s and s' go over the two possible spin values. P is then

$$P = \begin{pmatrix} e^{\beta(J+B)} & e^{\beta(-J+B)} \\ e^{\beta(-J-B)} & e^{\beta(J-B)} \end{pmatrix} \tag{13}$$

We can now rewrite (11) as

$$\begin{aligned} Z &= \sum_{s_1} ... \sum_{s_N} \langle s_1|P|s_2\rangle\langle s_2|P|s_3\rangle ... \langle s_N - 1|P|s_N\rangle\langle s_N|P|s_1\rangle \\ &= \sum_{s_1} \langle s_1|P^N|s_1\rangle \\ &= Tr(P^N) \end{aligned} \tag{14}$$

Now we have an expression for the partition function that is tractable to solution. We can calculate the trace in (14) by diagonalizing P. This is done by first finding the eigenvalues of P by solving the characteristic equation

$$det(P - \lambda I) = 0$$

which gives us

$$\begin{aligned} &\lambda^2 - \lambda e^{\beta J}\cosh(\beta B) + 2\sinh(2\beta J) = 0 \quad \text{where} \\ &\lambda_{\pm} = e^{\beta J}(\cosh(\beta J) \pm \sqrt{\sinh^2(\beta B) + e^{-4\beta J}}) \end{aligned} \tag{15}$$

That total expression for (14) can then be rewritten in terms of (15). We are interested in finding the solution of the Ising Model in the thermodynamic limit where $N \to \infty$. In this limit, λ_+ dominates. In short

$$\begin{aligned} Tr(P^N) &= \lambda_+^N + \lambda_-^N \\ Z &= \lambda_+^N, \qquad \text{as} \quad N \to \infty \end{aligned} \tag{16}$$

We can now utilize (5) and (6) to obtain an expression for the magnetization.

$$M = \frac{\sinh(\beta B) + \frac{\sinh(\beta B)\cosh(\beta h)}{\sqrt{\sinh^2(\beta B)+e^{-4\beta J}}}}{\cosh(\beta B) + \sqrt{\sinh^2(\beta B) + e^{-4\beta J}}} \tag{17}$$

We are interested in the possibility of the one-dimensional Ising model to exhibit a phase transition and at what temperature it occurs. Remember, a phase transition occurs when the Ising Model changes from a symmetric state, to a broken symmetry state. The equation we find for the magnetization must reflect this transition. There must be some critical temperature at which the magnetization goes from nonzero to zero.

To look for this point, we turn off the external magnetic field B, as it is not an intrinsic feature of our model. But at $B = 0$, the magnetization also vanishes as $\cosh(\beta B) \to 1$ and $\sinh(\beta B) \to 0$. This is true for any value of β or J. We are left with a boring model that does not feature a phase transition at any physical temperature. There is no point to start our Monte Carlo simulations here, unless a result of nothing excites you.

2.2. **Solution for the Two Dimensional Ising Model.** The solution to the two-dimension model follows the methodology of the previous solution, until the greatest eigenvalue of the transfer matrix is sought. The solution was first completed by Onsager[12] and will not be repeated here due to the lengthy calculation of the eigenvalue. I will instead outline the differences in the procedure due to the increase in dimensionality and restate the results. My summary will follow Kaufman's [8] "short and sweet" updated solution as presented by Haung [6].

We again begin with the Hamiltonian (2) which is now obviously summed over two indices. The transfer matrix P is constructed by considering columns of spins, instead of a singular spin. This results in a 2^n x 2^n matrix. The partition function is again given by the trace, but here we are left with a difficult task. Finding the largest eigenvalue of P requires many pages of calculation and manipulation of operators. It involves defining a handful of operators, most of which are familiar to the physicist: Pauli spin matrices, a related set of specific high-dimensional gamma matrix, rotation operators, and a spin representative. Other operators pop-up throughout the solution, but all are in terms of the gamma matrices.

The solution begins by noting that the transfer matrix can be rewritten in terms of three other matrices, which are then recast in terms of gamma matrices. P is then diagonalized in a specific form through a head-banging procedure utilizing the properties of the rotation matrix and the spin representative. It turns out that the largest eigenvalue of the matrix is found in the $n \to \infty$ limit. Taking this limit involves simplifying an ugly integral, but then we reach the explicit solutions in an acceptable form by remembering that the largest eigenvalue is equivalent to the partition function.

Onsager [12] finds the explicit form for the energy

$$\frac{U}{N} = \coth 2\beta (1 + \frac{2}{\pi} k_1'' K_1) \tag{18}$$

and uses that to obtain the specific heat

$$\frac{C}{Nk_B} = \frac{2}{\pi} (\beta \coth 2\beta)^2 [2K_1 - 2E_1 - (1 - k_1'')(\frac{\pi}{2} + k_1'' K_1)] \tag{19}$$

where K_1 and E_1 are the elliptical integrals

$$K_1 = K(k_1) = \int_0^{\pi/2} (1 - k_1^2 \sin^2\phi)^{-1/2} d\phi \tag{20a}$$

$$E_1 = E(k_1) = \int_0^{\pi/2} (1 - k_1^2 \sin^2\phi)^{1/2} d\phi \tag{20b}$$

where the specific heat and the elliptical integrals are written in terms of

$$k_1 = 2\sinh 2\beta / \cosh^2 2\beta$$
$$k_1'' = 2\tanh^2 2\beta - 1$$

A singularity in the specific heat occurs for $\beta = \frac{1}{2}\log(\coth(\pi/8))$ for which $k_1 = 1$, $k_1'' = 0$, $K_1 = \infty$, and $E_1 = 1$. For these values the specific heat (19) becomes infinite as K_1 dominates and the energy (18) is continuous. The critical temperature is easily calculated as $T_c k_B = 2.269$.

To check that a phase transition does indeed occur at T_c, we turn to the explicit order parameter. For temperatures above the critical temperature, the magnetization [14] is indeed non-zero

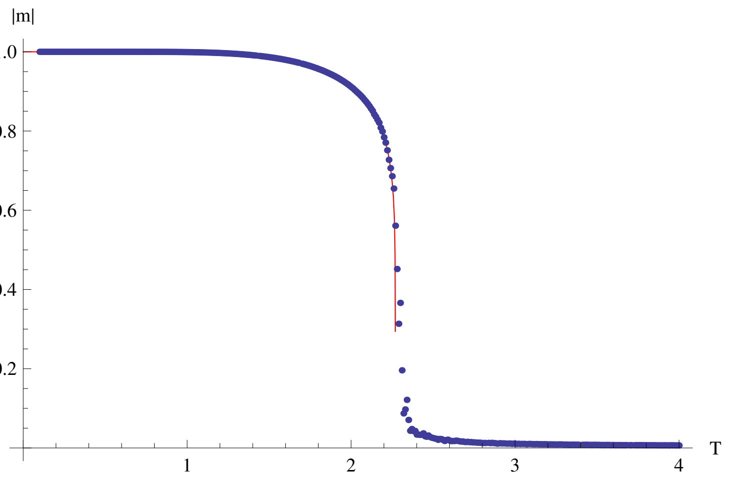

FIGURE 2. Absolute value of the magnetization vs temperature from Monte Carlo runs of a 256x256 lattice. Onsager's solution (red) abruptly ends at T_c.

$$M = \{1 - [\sinh(2\beta J)]^{-4}\}^{\frac{1}{8}} \tag{22}$$

but vanishes for smaller temperatures. This can be seen in Figure ??. Ladies and gentlemen, we have found our phase transition.

3. Simulating the Model

The simulation of the Ising model was written in C++ from scratch. I used multiple sources [5, 9] to assist me in the course of programming and data analysis, using them to check my data against. Simulations were ran with the objective of obtaining a qualitative understanding of the two-dimensional Ising model to provide a background for the ϕ_2^4 model. I will first go over the theory of Monte Carlo simulations, describe the algorithms used, and explain the data needed to extract the physics. All data was plotted using Mathematica.

3.1. Markov Chains and Detailed Balance.

A Markov process is a type of dynam[illegible]where the probability of a state is only dependent on the state immediately preceding it. Consider the current state of the system X_{t_n} as one element in the set of all possible configuration of the system $\{S_i\}$. The probability of the system being in state X_{t_n} as dictated by a Markov process [9] is written as

$$P(X_{t_n}) = P(X_{t_n} = S_{i_n} | X_{t_{n-1}} = S_{i_{n-1}}) \tag{23}$$

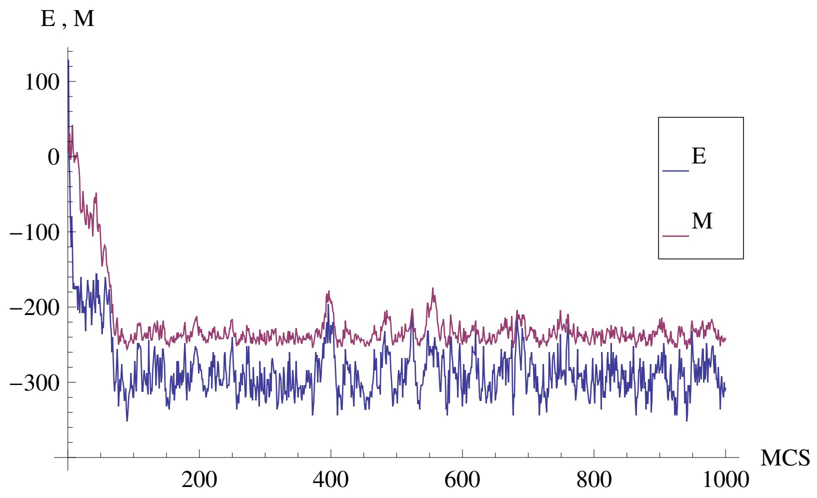

FIGURE 3. Time series plot of energy and magnetization. It takes about 100 Monte Carlo steps to reach an equilibrium state. Measurements must be taken of equilibrium states to obtain good and accurate statistics.

The set of states $\{X_{t_n}\}$ is a Markov Chain. Thinking of $\{X_{t_n}\}$ as an evolving system, (23) can be interpreted as a transition probability W_{ij} for the system to evolve from state S_i to state S_j

$$W_{ij} = P(X_{t_n} = S_j | X_{t_{n-1}} = S_i) \tag{24}$$

W_{ij} must satisfy the typical requirements for probabilities,

$$W_{ij} \geq 0, \qquad \sum_j W_{ij} = 1$$

Monte Carlo simulations produce a Markov Chain of system states at equilibrium. We expect that the probability of our system to be in a state of equilibrium to be high at some time down the chain. The total probability of $X_{t_n} = S_{i_n}$ is (23) multiplied by the probability of the state preceding it. At time t

$$P(S_j, t) = P(X_{t_n} = S_j | X_{t_{n-1}} = S_i) P(X_{t_{n-1}} = S_i)$$

Noting that the first factor is the transition probability, and taking the time derivative we have a continuity equation

$$\frac{dP(S_j,t)}{dt} = -\sum_i W_{ji}P(S_j,t) + \sum_j W_{ij}P(S_i,t) \tag{25}$$

This equation shows us that the change in probability of our system being in S_j in a Markov Chain is a balance between the total probability of moving to and away from S_j.

Importance sampling Monte Carlo simulations sets a requirement on (25) by considering a system in equilibrium. The Principle of Detailed Balance requires

$$W_{ji}P_{eq}(S_j) = W_{ij}P_{eq}(S_i) \tag{26}$$

and the continuity equation becomes

$$\frac{dP_{eq}(S_j,t)}{dt} = 0 \tag{27}$$

Thus, at a time t such that the system has reached equilibrium, (27) shows that system is not evolving towards a state that is becoming more favorable. Instead, X_{t_n} is limited to S_{t_n} that have a non-zero transition probability. This also ensure ergodicity. The physics that dictate a Markov Chain dynamic and the Principle of Detailed Balance become clear in the next section.

Figure 3 shows time series data for energy and magnetization. To obtain accurate results for thermodynamic variables, measurements must be taken from equilibrium states. Measurements from non-equilibrium states are outliers in the averages of energy. This results in specific heat and susceptibility plots that do not show a peak, noisy magnetization plots, and negative or noisy values for the Binder cumulant. The number of steps it takes for the lattice system to reach equilibrium is dependent on the initial state of the lattice. It is a good practice to run thermalization steps equal to the number of steps you intend to take measurements of. This essentially guarantees that you are only taking measurements at equilibrium. A sloppier approach ignores the extra thermalization steps, and instead takes the Monte Carlo steps to be much greater than steps needed to reach equilibrium, so nonequilbrium measurements have virtually no impact on the averages.

3.2. **The Metropolis Algorithm.** Now that we understand the dynamics involved in our simulation, we must develop a method to achieve them. We know that the Ising Model is composed of many 1/2-spins and that they will all align in the broken symmetry phase or be in random alignment in the symmetric phase.

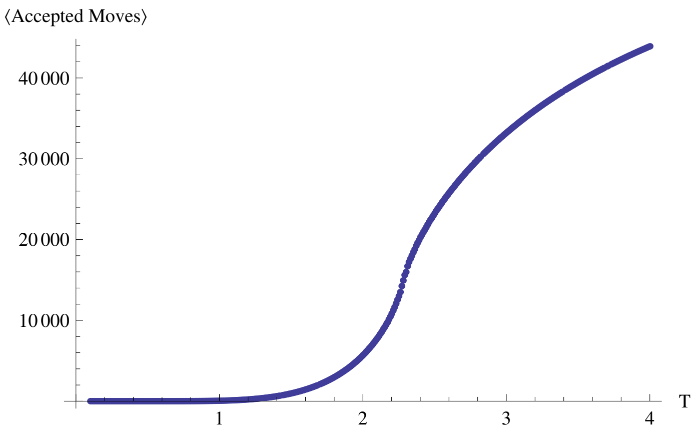

FIGURE 4. Average number of accepted Metropolis moves on a 256x256 lattice. More steps are accepted as the model becomes disordered for higher temperatures. It may be possible to estimate T_c by finding the inflection point.

The intrinsic determinant of the critical temperature is the form of the Hamiltonian. We can thus consider a specific site on the lattice, flip its spin, and re-calculate the Hamiltonian. A flip that decreases the Hamiltonian is a move closer to equilibrium. At equilibrium we must form a method for numerically calculating (26). The probability of a microstate is given by the classical Boltzmann factor and partition function (9).

By rearranging (26) and using (9), we can write

$$\frac{P_i(t)}{P_j(t)} = \frac{W_{ji}}{W_{ij}} = e^{\beta(E_j - E_i)} \tag{28}$$

Any transition probability satisfying (28) will ensure detailed balance. Simplifying $E_j - E_i$ to ΔE, we can make the simple choice of letting W_{ij} be the probability ratio, and W_{ji} to be unity. This is the Metropolis [11] choice and is defined as

$$W_{ij} = \begin{cases} e^{-\beta \Delta E} & \text{if } \Delta E > 0 \\ 1 & \text{if } \Delta E \leq 0 \end{cases} \tag{29}$$

which clearly satisfies (28). We implement this choice in the Metropolis Algorithm by following the below procedure:

(1) Randomize the lattice to form an initial state.

(2) Randomly choose a site i

(3) Calculate ΔE resulting from flipping s_i.

If $\Delta E \leq 0$, accept the flip.

If $\Delta E > 0$, generate a random number $r \in (0, 1)$. If $r < e^{-\beta \Delta E}$, accept the flip.

(4) Return to step 2 until iterations equal to the size of the lattice have been completed

Completing the algorithm is defined as one Monte Carlo step, a non-deterministic time scale for the dynamics of the Ising Model. The behavior of the Metropolis algorithm for different temperatures is seen in Figure 4. More moves are accepted as the temperature increases and the state becomes disordered, ultimately a result of the large number of microstates that can compose a disordered state.

3.3. **Fluctuations in states.** No longer being in the realm of explicit solutions, we must consider how to extract macroscopic thermodynamic variables from our set of configurations in Monte Carlo space. We are interested in finding the phase transition of the Ising Model by running lattice simulations at various temperatures and determining at what temperature the transition occurs. Two thermodynamic quantities that indicate a phase transition are the specific heat C_v and the susceptibility χ. Both quantities diverge as $T \to T_c$, see Figures **??** and **??**.

At equilibrium, our simulations fluctuate [9] they explore configurations about the equilibrium energy. To characterize the internal energy at a given temperature, we simply take the average energy, which can be represented as a weighted average in terms of Boltzmann factors.

$$\overline{U} = \langle \mathcal{H}(\mu) \rangle \tag{30}$$

$$= \sum_{\mu} \mathcal{H}(\mu) e^{-\beta \mathcal{H}(\mu)} / \sum_{\mu} e^{-\beta \mathcal{H}(\mu)} \tag{31}$$

and using the definition (5)

$$C_v = \frac{1}{k_B T^2} \langle \mathcal{H}^2 \rangle - \langle \mathcal{H} \rangle^2 \tag{32}$$

Following the same procedure, we also have an expression for the susceptibility

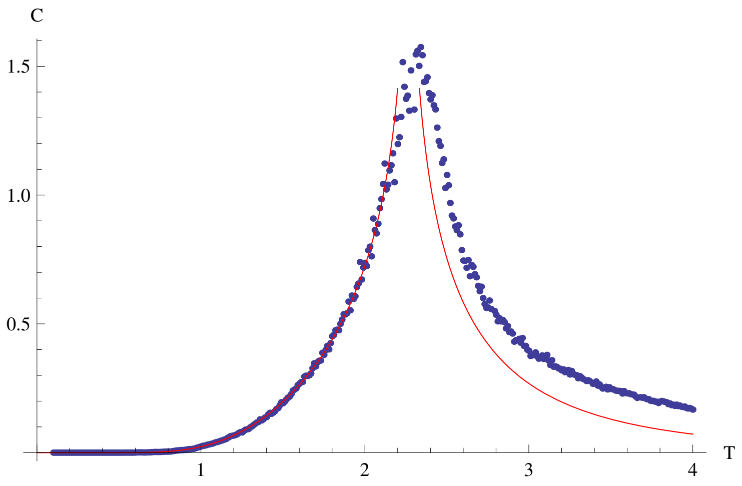

FIGURE 5. Specific Heat for a 128x128 lattice. C diverges as $T \to T_c$ and is fit well by Onsager's solution(red) (19). The tail of the plot will approach the explicit solution as $L \to \infty$ and as Monte Carlo

$$\chi = \frac{1}{k_B T} \langle M^2 \rangle - \langle M \rangle^2 \tag{33}$$

3.4. **Correlation Time and Critical Slowing Down.** Once the Ising Model reaches equilibrium, the magnetization and energy settle in to a temperature-dependent expected value. These quantities do not stay flat, but periodically fluctuate about these expected values (see Figure 3). To characterize this periodicit nd thus quantify how often a state at t_o is correlated with a state measured at $t + t_o$ we define a normalized autocorrelation function

$$C_A(t) = \frac{\langle A(t_o) A(t + t_o) \rangle - \langle A \rangle^2}{\langle A^2 \rangle - \langle A \rangle^2} \tag{34}$$

where A is the quantity we are interested in measuring. Because the origin of our Monte Carlo system is stochastic and time is non-deterministi ny measurement can be taken as the time origin. The term is then summed over all possible time intervals of size t and averaged.

The autocorrelation function is normalized such that at $t = 0$ a value of 1 represents full correlation and as $t \to \infty$ that function decays to an uncorrelated zero. The asymptotic behavior of (34) is exponential deca

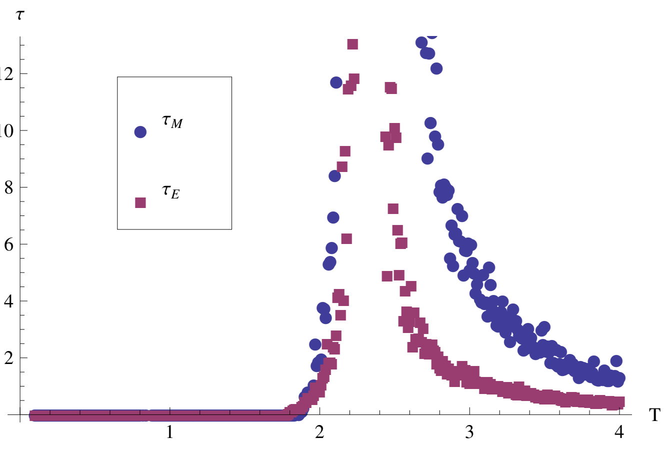

FIGURE 6. Correlation times for energy and magnetization for a 32x32 lattice. Correlation times diverge as $T \to T_c$, requiring more runs to attain quality statistics. τ_E is not equivalent to τ_M due to greater fluctuation of the magnetization at high temperatures.

$$C_A(t) \to e^{-t/\tau_A} \tag{35}$$

This can be integrated over all t and the correlation time [5] τ is given as

$$\tau_A = \int_0^\infty C_A(t) \tag{36}$$

The correlation time is a measure of many Monte Carlo steps we have to iterate through before we produce an independent measurement. Our simulations are not continuous, so to obtain τ we simply rewrite (36) as a discrete sum over our simulation time. Knowing that our measurements are correlated in time, we must include the correlation time in our error estimates. The error [9] is redefined in terms of the typical standard deviation and τ

$$(error)^2 = \frac{\sigma^2}{n}(1 + 2\tau/\partial t) \tag{37}$$

It is not immediately obvious from (36) and (34) how τ behaves as our system approaches the critical temperature. We can construct a good description of the behavior of τ by considering the critical dynamics we observed above. Remember that the observables susceptibility and specific heat both diverge as T_C is approached, resulting in an observed maximum. The magnetization also has diverging behavior as $T \to T_C$ as our systems transitions from a ordered to disorder state. These critical dynamics can be expressed as power laws [9].in terms of the Ising Model's critical exponents: those found by Onsager that define the universality class.

$$\chi = \chi_o \epsilon^{-\gamma}, \qquad C_V = C_{Vo} \epsilon^{-\alpha}, \qquad M = M_o \epsilon^{\beta} \tag{38}$$

where ϵ is the reduced temperature $|1 - T/T_C|$. These expression represent asymptotic behavior of the observables only as $\epsilon \to 0$. We can now consider (34) dependency on temperature. As the critical temperature is approached, our observable A diverges and so does τ. Figure **??** shows the divergence of the correlation time for the energy and magnetization. This is the problem of critical slowing down. To ensure that we can find the peaks of the susceptibility and specific heat with statistical accuracy, we must go back to our algorithm and consider how we might stop the diverging behavior of τ and thus reduce the Monte Carlo error (37).

3.5. **The Wolff Cluster Algorithm.** To battle the problem of critical slowing down, we turn to a more powerful algorithm. The Wolff Cluster Algorithm [13] extends the approach of Metropolis to multiple spins. Instead of considering one spin for a flip, the Wolff Algorithm considers a cluster. The cluster is formed by choosing a site as the cluster seed. Bonds are formed between it and nearest neighbors by a probability that satisfies detailed balance. One of the nearest neighbors becomes the seed and the cluster continues to be built until all possible bonds have been built. The whole cluster is then flipped.

The Wolff algorithm significantly decreases the correlation between successive states by its dramatic construction of a new state. Thus, at least for the Ising Model, the Wolff Algorithm completely vanquishes the problem of critical slowing down. The Algorithm goes as follows

(1) Randomly choose a site i to be the cluster seed
(2) Add bonds to all nearest neighbors j with probability $P = 1 - e^{J\beta\delta_{s_i s_j}}$
(3) If all bonds have been tried, pick a nearest neighbor site to be the new seed
(4) Continue steps 2-3 until no more bonds are created
(5) Flip the cluster
(6) Repeat from 1

Ergodicity is ensured in the Wolff Algorithm because there is a non-zero probability that the cluster consists of only one spin and a flip is always made. The autocorrelation time is modified [13] as

$$\tau_\chi = \overline{\tau_\chi} m \langle c \rangle L^{-2} \tag{39}$$

where m is the number of updates between each measurement and c is cluster size. Away from the phase transition for $T > T_c$ the cluster size is large as the system is in an ordered state. But at these temperatures, the correlation time is small anyway, so the Wolff correlation time is similar to the Metropolis. However, as the $T \to T_c$, the cluster size becomes smaller relative to L, and thus the correlation time is significantly reduced.

3.6. **To the Thermodynamic Limit: Finite Size Scaling.** We have quickly found the critical temperature of the two-dimensional Ising Model with fairly intuitive algorithms while accumulating key thermodynamic averages. But we have not yet reached our goal. As we simulate the Ising Model on lattices of different size, the observed critical temperature shifts. To find the critical temperature at the thermodynamic limit as $N \to \infty$ we need a method to qualitatively model the systematic dependence of critical temperature with lattice size. One such method is finite size scaling.

The behavior of our thermodynamic observables near the critical temperature is described by (38). To recast the critical behavior in terms of a length to relate to the lattice size, we define the correlation length

$$\xi = \xi_o \epsilon^{-\nu} \qquad \text{as } \epsilon \to 0 \tag{40}$$

which is similar to the correlation time: expressing how similar the state of the Ising Model is across the lattice. (40) features the same divergence as the thermodynamic variables. we can rewrite the critical behavior of the susceptibility in terms of the correlation length

$$\chi = \xi^{\gamma/\nu}$$

The behavior of the susceptibility runs into a block for finite lattices because the similar critical behavior of the correlation length cannot exceed the size of the lattice L. As we increase L in the simulations, the divergent of behavior of ξ also grows. To obtain the true susceptibility at the thermodynamic limit, we need to introduce a scaling function to the above expression for χ.

$$\chi = \xi^{\gamma/\nu} \chi^o(L/\xi)$$

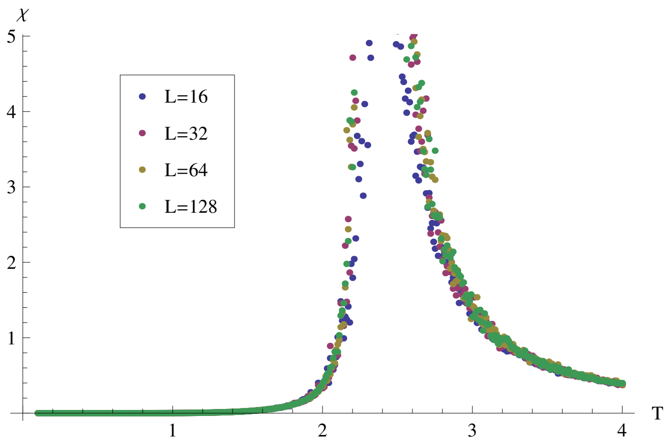

FIGURE 7. Susceptibility for a lattices of varying sizes. The susceptibility peaks shift closer to T_c as $L \to \infty$

The above expression has an unnecessary double dependence on L. This can be eliminated by rewriting it in terms of the reduced temperature $\epsilon = (T - T_c)/T_c$, giving us an explicit dependence on T. We then have for the susceptibility and specific heat

$$\chi = L^{\gamma/\nu}\chi^o(\epsilon L^{1/\nu}), \tag{41a}$$

$$C = L^{\alpha/\nu}C^o(\epsilon L^{1/\nu}) \tag{41b}$$

From these scaling forms, we can consider plotting $\chi L^{-\gamma/\nu}$ against $\epsilon L^{1/\nu}$ for various lattice sizes. Without *a priori* knowledge of the critical exponents γ and ν, we must fit for those along with T_c in ϵ. Once we have found a good fit for these three parameters, the fitted susceptibility curves should all merge, thus giving us T_c. This method known as data collapse can naively be done by eye or rigorously by minimizing statistical quantities of the fit. But there is a more well reasoned method.

We have previously found the peak of the susceptibility per lattice size. The peaks of (41a) occur at the same point, when χ^o is maximum. This point $T_c(L)$ is defined [1] asymptotically as

$$T_c(L) = T_c(1 + x_t^* L^{-1/\nu}), \qquad x_t^* = tL^{1/\nu} \tag{42}$$

where x_t^* is the location of the peak of the scaling function and T_c is the critical temperature in the thermodynamic limit. The above equation holds only for large

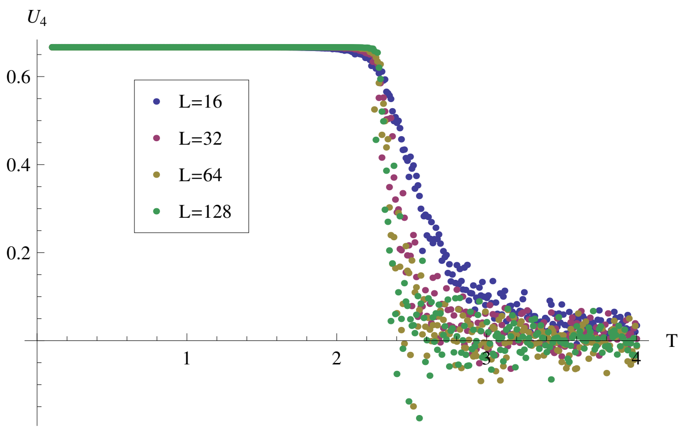

FIGURE 8. Fourth order cumulant for a 128x128 lattice. The intersection of all four plots is difficult to locate visually and may not be an accurate estimate of T_c for small lattice sizes. $U_4 \to 2/3$ for $T < T_c$ and $U_4 \to 0$ for $T > T_c$. Negative values are a result of noise. The plot can be resolved about the temperature at which U_4 diverges to obtain a better estimate of the intersection point.

lattices and temperatures close to T_c. For smaller lattices, further corrections must be taken into account. Thermodynamic variables will be scaled by a power-law with an exponent $-w$ that is unique for each variable. For example, the magnetization at T_c would scale with lattice size like $L^{-\beta/\nu}(1 + cL^{-w})$.

The temperatures at which we find our thermodynamic maxima per lattice size varies proportionally from T_c by $L^{1/\nu}$, as seen in (42). Power-law corrections to asymptotic behavior take the form $a_1 L^{-\theta/\nu} + \ldots$ and $b_1 L^{-1/\nu} + \ldots$ [1]. However, taking into consideration all of these correction terms leads too many parameters to be successfully fit. We can simplify the corrections by using only one term L^{-w} with the hopes that our lattice sizes are large enough to exhibit asymptotic behavior and any remaining corrections can be well approximated by the single term. The new estimate [4] for $T_c(L)$ becomes

$$T_c(L) = T_c + \lambda' L^{-1/\nu}(1 + b' L^{-w}) \tag{43}$$

The above equation has four fitting parameters, introducing a possible difficulty in the fitting procedure. It is important to note that λ' is dependent on the thermodynamic variable chosen to locate $T_c(L)$. Each thermodynamic variable has a different scaling function, so both $T_c(L)$ and λ' will differ between susceptibility and specific heat. To obtain a good fit from (43) it is important to obtain accurate values of $T_c(L)$ and ν. Remember for the two-dimensional Ising Model, we know the value of ν from Onsager's explicit solution. However, we can obtain a value of ν by examining another variable.

An estimate of ν can be extracted from the fourth-order magnetization cumulant defined as

$$U_4 = 1 - \frac{\langle m^4 \rangle}{3\langle m^2 \rangle^2} \tag{44}$$

The maximum of the slope of the cumulant scales like $L^{1/\nu}$. This occurs near T_c and be expressed by first introducing the quantity $K = J/k_B T$ and looking at the derivative of U_4

$$\frac{\partial U_4}{\partial K}|_{max} = aL^{1/\nu}(1 + bL^{-w}) \tag{45}$$

But the fourth order cumulant can also be used as a more direct method to finding T_c in the thermodynamic limit. If we plot U_4 for many different lattice sizes, we will find that they will intersect at a given temperature which turns out to be T_c [3]. However, this is again valid only for large lattices. If smaller lattices are included, finite-size correction terms will obfuscate the shared intersection point (see Figure 8).

4. Physical Applications and Sibling Models

The Ising model is an abstraction of a magnetic behavior. And yet despite its simplicity, it is able to capture with impressive accuracy the physics of phase transitions. This is because a phase transition is an emergent phenomena that results from the interacting behavior of each spin: the microscopic physics of atoms can be simplified and reduced. The Ising model can be adapted to other physical systems with small changes to the Hamiltonian along with a different interpretation. Accompanying the changed Hamiltonian, subtle and non-intuitive changes in the phases and the transition emerge. One must reconsider how to interpret the simulation, quantify the phase transition, and incorporate time-scale behavior.

4.1. **Lattice Gases.**

The lattice gas is the simplest adaptation of the Ising model requiring only a reinterpretation of the model. A lattice gas describes a gas with a finite number N_a of particles on a discrete NxN lattice such that $N_a < N$. Each particle takes on a discrete position on the lattice, with only one particle fitting at each site. Each site is either occupied or not, corresponding to an Ising spin up or down. The particles of the gas only interact with nearest neighbor. The Ising magnetization maps to the specific volume, and the difference between the Ising free energy per spin and magnetic field map to the pressure.[10] Because of the number of particles is conserved, the Metropolis algorithm has to modified to utilize spin-exchange dynamics in place of spin flips. The lattice gas could alternatively be interpreted as a binary alloy, where spin up are particle of type A and spin down are particles of type B.

4.2. **Monte Carlo on Spin Glasses.** Spin glasses are magnetic systems with spin interactions that compete on the nearest neighbor level to evolve towards an ordered state. Competing interactions are due to frozen-in disorder and prevent long-range order from emerging. A phase transition occurs for spin glasses at a freezing temperature where a redefined order manifests in which spins are aligned in random directions. Many materials share the characteristic phenomena of spin glasses when they are randomly diluted or are noncrystalline. The Ising model is used to model spin glasses for Monte Carlo simulations. The Hamiltonian with non-zero magnetic field (2) is used, but the coupling strength–now called exchange constant-is distributed on a Gaussian by the Edwards-Anderson [2] form

$$P(J_{ij}) = \frac{1}{\sqrt{2\pi\sigma_J^2}} e^{-\frac{N}{2\sigma_J^2}(J_{ij}-\mu_J/N)^2} \tag{46}$$

Anitferromagnetic and ferromagnetic exchange constants occur with equal probability, resulting in the characteristic frustrated interactions. This gives rise to a time-dependent feature when the spin glass is cooled below T_f while a magnetic field is on. The magnetic field is then switched off. The magnetization of the spin glass was as would be expected for a paramagnetic material. But once the magnetic field is gone and $T < T_f$, the magnetization drops to a non-zero value and non-exponentially decays toward zero.

4.3. **n-Vector Model.** The n-vector model is an expansion on the Ising model where spins are allowed to point in any direction within the dimensionality of the model. The basis components of the spin vector are coupled. The Hamiltonian

$$\mathcal{H} = -J \sum_{\langle nn \rangle} (S_i \dot{S}_j) \tag{47}$$

Where S_i is an n-dimensional unit spin vector. The Ising model is the special case where $n = 1$; the $n = 2$, $n = 3$, and $n = 4$ cases are known as the XY, Heisenberg, and Higgs Toy models respectively. To simulate the model in $n > 1$, one or more spin angles are randomly chosen for each site in an adapted Metropolis algorithm.

Long range order does not appear for low temperatures, but a phase is defined by the presence of topological excitations. These vortex-antivortex pairs unbind at the transition temperature T_{KT} [9].

4.4. **q-state Potts Model.** The q-state Potts Model [9] is yet another extension of the Ising Model. Instead of having only two-discrete states for spin, there are q possible states. Coupling occurs only for spins in the same state, represented in the Hamiltonian

$$\mathcal{H} = -J \sum_{i,j} \delta_{s_i s_j} \tag{48}$$

where $s_i = 1, 2, ..., q$. The Ising model is the special case of $q = 2$.

5. Summary

The Ising model has been an adequate introduction to Monte Carlo techniques, programming skills, and data analysis. The Markov theory behind Monte Carlo has been presented, giving a good understanding of why this simulation technique is so powerful. Multiple avenues of finding the critical temperature have been presented. These will be used in our experimentation with ϕ_2^4 theory and rigorous error analysis will be introduced. The Ising model provides a comfortable foundation for learning and experimenting with numerical modeling techniques and provides many rich models that have spawned from it.

References

[1] K. Binder. Finite size scaling analysis of Ising model block distribution functions. *Z.Phys.*, B43:119–140, 1981.

[2] K. Binder and A. P. Young. Spin glasses: Experimental facts, theoretical concepts, and open questions. *Rev. Mod. Phys.*, 58:801–976, Oct 1986.

[3] Kurt Binder. Finite size scaling analysis of ising model block distribution functions. *Zeitschrift für Physik B Condensed Matter*, 43(2):119–140, 1981.

[4] Alan M. Ferrenberg and D. P. Landau. Critical behavior of the three-dimensional ising model: A high-resolution monte carlo study. *Phys. Rev. B*, 44:5081–5091, Sep 1991.

[5] Harvey Gould and Jan Tobochnik. *An Introduction to Computer Simulation Methods: Applications to Physical Systems.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1995.

[6] Kerson Huang. *Introduction to statistical physics.* CRC, 2001.

[7] Ernst Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik A Hadrons and Nuclei*, 31(1):253–258, 1925.

[8] Bruria Kaufman. Crystal statistics. ii. partition function evaluated by spinor analysis. *Phys. Rev.*, 76:1232–1243, Oct 1949.

[9] D. P. Landau and K. Binder. *A Guide to Monte Carlo Simulations in Statistical Physics - 2nd Edition*. Cambridge University Press, September 2005.

[10] T. D. Lee and C. N. Yang. Statistical theory of equations of state and phase transitions. ii. lattice gas and ising model. *Phys. Rev.*, 87:410–419, Aug 1952.

[11] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[12] Lars Onsager. Crystal statistics. i. a two-dimensional model with an order-disorder transition. *Phys. Rev.*, 65:117–149, Feb 1944.

[13] Ulli Wolff. Collective monte carlo updating for spin systems. *Phys. Rev. Lett.*, 62:361–364, Jan 1989.

[14] C. N. Yang. The spontaneous magnetization of a two-dimensional ising model. *Phys. Rev.*, 85:808–816, Mar 1952.

```
//
//  Ising_mine.cpp
//
//
//  Created by Tyler Ogden on 12/15/12.
//  Copyright (c) 2013 Hampshire College. All rights reserved.
//

#include <iostream>
#include <cstdlib>
#include <cstddef>
#include <cmath>
#include <algorithm>
#include <trng/mt19937_64.hpp>
#include <trng/uniform_int_dist.hpp>
#include <trng/uniform01_dist.hpp>
#include <vector>
#include <fstream>

using std::vector;      using std::cout;
using std::endl;

//spin lattice of 16x16
const size_t L = 128;
int iL = int(L);
int N = iL*iL;
static int spin[L][L];
static double temperatureCritical = 2/log(1+sqrt(2));
double *cE, *cM, *Etseries, *Mtseries;
bool **ClusterSpin;

// Nearest-neighbour index array
static int nnup[L],nndn[L];

//time series data structure
struct TimeSeriesData {
    int magnetization, energy;
};
struct ThermalQuant{
    double emean, e2mean, mmean, m2mean, m4mean, mabsmean, susceptibility,
        specificHeat, U4, \
    tauE, tauM;
};

trng::mt19937_64 rng;


//Metropilis Alogorithm with random site choice
void Metropolis_rand(int& e, int& aM, int& m, const int& l,vector<float>& w,
    const int& cc){
    int x,y,dE;
    trng::uniform_int_dist uniform(0,l);
    trng::uniform01_dist<> uni;


    for (int i=0; i<N; i++) {
        x = uniform(rng);
        y = uniform(rng);
        dE = 2*spin[x][y]*(spin[x][nnup[y]] +spin[x][nndn[y]]
                           + spin[nnup[x]][y] + spin[nndn[x]][y]);
        //double mag_sum = b(spin[x][y]);

        if (dE <= 0 || uni(rng) < w[dE]) {
            spin[x][y] = -1 * spin[x][y];
            e += dE;
            aM++;
            m += 2*spin[x][y];  //factor of 2 for change and cancel
        }
    }

    Mtseries[cc] = m;
    Etseries[cc] = e;

}

//function to calculate thermodynamic quantitites
ThermalQuant calculate(const vector<TimeSeriesData>& ts, const int& s,const long&
    step, const double& t){
    ThermalQuant output;
    double esum = 0 , e2sum =0 , msum =0 , m2sum =0 , mabssum =0 , m4sum =0,
        Enorm =0, Mnorm = 0, mm=0, ee=0;
    int ncorr;
    bool checkE = false, checkM = false;

    //get sums
    for (vector<TimeSeriesData>::const_iterator iter = ts.begin(); iter != ts.end
        (); iter++){
        mm = (*iter).magnetization;
        ee = (*iter).energy;
        esum += ee;
        e2sum += (ee*ee);
        msum += mm;
        m2sum += fabs(mm*mm);
        mabssum += fabs(mm);
        m4sum += fabs(mm*mm*mm*mm);
    }
    output.emean = esum/(step);
    output.e2mean = e2sum/(step);
    output.mmean = msum/step;
    output.m2mean = m2sum/step;
    output.mabsmean = mabssum/step;
    output.m4mean = m4sum/step;
    output.susceptibility = (1/t)*(output.m2mean - output.mabsmean*output.
        mabsmean);
    output.susceptibility /= N;
    output.specificHeat = (1/(t*t))*(output.e2mean - output.emean*output.emean);
    output.specificHeat /= N;
    output.U4 = output.m4mean;
    output.U4 /= output.m2mean*output.m2mean;
    output.U4 /= -3;
    output.U4 += 1;
    //calculate correlation times
    Enorm = output.e2mean - output.emean*output.emean;
    Mnorm = output.m2mean - output.mabsmean*output.mabsmean;

    //build double length time series for autocorellation calculation
    for (int i =0; i < step;i++){
        Etseries[step + i] = Etseries[i];
        Mtseries[step + i] = Mtseries[i];
    }

    //calculate autocorrelation function
    for (int t = 1; t < step; t++) {
        ncorr = 0;
        for (int i = 0; i < step; i++) {
            cE[t] += Etseries[i]*Etseries[i+t];
            cM[t] += fabs(Mtseries[i]*Mtseries[i+t]);
            ++ncorr;
        }
        cE[t] /= ncorr;
        cM[t] /= ncorr;
        cE[t] -= output.emean*output.emean;
        cM[t] -= output.mabsmean*output.mabsmean;
        cE[t] /= Enorm;
        cM[t] /= Mnorm;


        //add to sum if function is greater than zero
        if (cE[t] > 0 && !checkE){
            output.tauE += cE[t];
        }
        else{
            checkE = true;
        }

        if (cM[t] > 0 && !checkM) {
            output.tauM += cM[t];

        }
        else {
            checkM = true;
        }
        //break loop when both functions are negative
        if (checkE && checkM) {
            break;
        }
    }
    return output;
}

//construct initial state of lattice
void initialize(const int& in, const size_t& l, const int& n, int& e, int& m,
    const long& step){
    trng::uniform_int_dist uni(0,2);
    e = 0;
    m = 0;
```

```
        int dE;

        ClusterSpin = new bool* [L];
        for (int i = 0; i < L; i++) {
            ClusterSpin[i] = new bool [L];
        }

        cE = new double [2*step + 1];
        cM = new double [2*step + 1];
        Etseries = new double [2*step];
        Mtseries = new double [2*step];
        for (int i = 0; i <= 2*step; i++) {
            cE[i]=0;
            cM[i]=0;
            Mtseries[i]=0;
            Etseries[i]=0;
        }

        if (in == 1) {
            for(int x=0;x<l;x++) for(int y=0;y<l;y++) spin[x][y] = 1;
            m = n;
            e = -2*n;
        }
        else if (in == -1){
            for(int x=0;x<L;x++) for(int y=0;y<L;y++) spin[x][y] = -1;
            m = -1*n;
            e = 2*n;
        }
        else {
            for(int x=0;x<L;x++) for(int y=0;y<L;y++){
                if (uni(rng) == 0){
                    spin[x][y] = -1;
                    --m;
                }
                else{
                    spin[x][y] = 1;
                    ++m;
                }
            }
                for(int x=0;x<L;x++) for(int y=0;y<L;y++){
                    dE = 2*spin[x][y]*(spin[x][nnup[y]] +spin[x][nndn[y]]
                                        + spin[nnup[x]][y] + spin[nndn[x]][y]);
                    e += dE;
                }
        }

    }

    //outputs lattice file
    void snapshot(const double& m, const size_t& l){
        char fname[128];
        sprintf(fname, "snapshot_T%f.dat",m);

        std::ofstream snap(fname);

        for (int i =0; i < l; i++) {
```

```
        for (int j=0; j < l; j++) {
            snap << spin[i][j] << "\t";
        }
        snap << endl;
    }
    snap.close();
}

//Wolff Cluster Algorithm
//pick random cluster seed; attach bonds to like-spin sites with probability p =
    1 - exp[-2*spin(c)*spin(nn)]
int Wolff_simple(const size_t& l, double e){
    int x,y;
    double cluster = 0;
    double core;
    trng::uniform_int_dist uniform(0,l);

    void tryBondAdd (int& aa, int& bb, double& spn, double cre);
    void growCluster (int& a, int& b, double& cre);


    //initialize ClusterSpin value
    for (int i = 0; i < l; i++) {
        for (int j = 0; j<l; j++) {
            ClusterSpin[i][j] = false;
        }
    }

    //get random lattice site cluster core
    x = uniform(rng);
    y = uniform(rng);
    core = spin[x][y];

    growCluster(x,y,core);

    for (int i = 0; i < l; i++) {
        for (int j = 0; j<l; j++) {
            if (ClusterSpin[i][j] == true) {
                cluster++;
            }
        }
    }



    return(cluster);
}

void tryBondAdd (int& aa, int& bb, double& spn, double cre){
    double p;
    trng::uniform01_dist<> uni;

    void growCluster (int& a, int& b, double& cre);

    if (cre*spn >= 0){
        p = (1-exp(-2*cre*spn));
```

```
            if (uni(rng) < p){
                cre = spn;
                growCluster(aa, bb, cre);
            }
        }
    }

    void growCluster (int& a, int& b, double& cre){
        double nbor;
        ClusterSpin[a][b] = true;
        spin[a][b] = -1*spin[a][b];

        if (!ClusterSpin[nnup[a]][b]) {
            nbor = spin[nnup[a]][b];
            tryBondAdd(nnup[a], b, nbor, cre);
        }

        if (!ClusterSpin[nndn[a]][b]) {
            nbor = spin[nndn[a]][b];
            tryBondAdd(nndn[a], b, nbor, cre);
        }

        if (!ClusterSpin[a][nnup[b]]) {
            nbor = spin[a][nnup[b]];
            tryBondAdd(a, nnup[b], nbor, cre);
        }

        if (!ClusterSpin[a][nndn[b]]) {
            nbor = spin[a][nndn[b]];
            tryBondAdd(a, nndn[b], nbor, cre);
        }
    }

    int main(int argc, char** argv)
    {
        long mcs;
        int acceptedMoves,E,M,count;
        double temperature = temperatureCritical;
        vector<float> boltz(9);
        vector<int> Etseries, Mtseries;
        char fname[200];
        std::ofstream outfile;
        ThermalQuant thermdata;
        //take input
        if (argc == 3){
            mcs = atol(argv[1]);
            temperature = atof(argv[2]);
        }
        else{
            cout << "Wrong input number. QUITTING....." << endl;
            return(1);
        }
        vector<TimeSeriesData> tseries(mcs);

        // fill up the nearest neighbor array
        for (int i=0; i<L; i++) {
```

```
        nnup[i] = (i+1) % L;
        nndn[i] = (i-1+L) % L;
    }

    //possible Boltzmann factors for 2-D Ising
    boltz[4] = exp(-4/temperature);
    boltz[8] = exp(-8/temperature);



    //Initial spin microstate - all spins up
    initialize(0, L, N, E, M,mcs);
    count =0;

    sprintf(fname,"tseries.dat");
    outfile.open(fname, std::ios::out | std::ios::app);

    //Run Metropolis Algorithm
    for (vector<TimeSeriesData>::iterator iter = tseries.begin(); iter != tseries
        .end(); iter++) {
        count++;
        (*iter).energy = E;
        (*iter).magnetization = M;

        outfile << E << "\t" << M << endl;

        //*iter.microstate = spin;
        Metropolis_rand(E, acceptedMoves, M, iL, boltz,count);
    }

    //calculate averages, and thermodynamic quantities
    thermdata = calculate(tseries, N, mcs, temperature);




    /*outfile << temperature << std::setprecision(9) << "\t" << thermdata.emean
        << "\t" << thermdata.mabsmean << "\t" << thermdata.susceptibility <<"\t"
    << thermdata.specificHeat << "\t" << thermdata.Uls
     4 << "\t" << thermdata.tauE << "\t" << thermdata.tauM << "\t" <<
         acceptedMoves << endl;*/

    outfile.close();

    cout << thermdata.m4mean << "\t" << (thermdata.m2mean)*(thermdata.m2mean) <<
        "\t" << thermdata.U4 << endl;

    //snapshot(temperature,L);

}
```