Monte Carlo Simulation of the Spin 1/2 Ising Model in 3 Dimensions and Behavior of Spin Susceptibility Around the Critical Temperature for Ferromagnetic Transition

Baran Bodur
Department of Physics, Duke University, Durham, NC, USA, 27708

I. Introduction to the Ising Model and Objective

Ising Model is developed in order to explain the ferromagnetic behavior with the spin-spin interactions. In this model, there are fixed lattice points that can take discrete values[1] and interacting with each other based on those values. The most commonly used form of the Ising model is with spin 1/2 particles (only 2 possible values for spins) and nearest neighbour interactions between the spins. In that case, without a single spin energy term[2], the Hamiltonian of the system can be written as in equation 1.

$$H = -J \sum_{<ij>} S_i S_j \tag{1}$$

In equation 1 $< ij >$ means sum over all nearest neighbour pairs, and $S_i, S_j = \pm 1$[3]. The canonical partition function of such a system is given in equation 2, where sum over $[s]$ means summation over all possible spin configurations.

$$Q = \sum_{[s]} e^{J\beta \sum\limits_{<ij>} S_i S_j} \tag{2}$$

In this project we will study the spin susceptibility of 3 dimensional Ising Model around the critical temperature $J\beta_c = 0.2216544$ up to the lattice sizes of $L = 100$. The spin susceptibility of a given spin configuration is given as in equation 3 and susceptibility as a function of $J\beta$ and L is given by the expectation value as in equation 4.

$$\chi_s([s]) = \frac{1}{L^3} \sum_i \sum_j S_i S_j \tag{3}$$

$$\chi_s(J\beta, L) = \frac{1}{L^3 Q} \sum_{[s]} \left(\sum_i \sum_j S_i S_j \right) e^{J\beta \sum\limits_{<ij>} S_i S_j} \tag{4}$$

Analytical solution for 3D Ising Model does not exist, and exact solutions with computer requires calculating energy and susceptibility for 2^{L^3} states, which is not feasible for $L > 3$. Therefore, we will use Monte Carlo techniques developed for the Ising Model as described in the following section, apart from an exact calculation for the $2 \times 2 \times 2$ lattice. In addition we will assume periodic boundary conditions, so that the spins at the boundary will still have 6 neighbours as spins in the bulk[4].

II. Wolff's Single Cluster Algorithm

A. Application of Markov Chains and Detailed Balance

Markov chains are stochastic models in which probability of moving to a new state depends only on the current state. In steady state, the probability of being in a state $p(k)$ will depend on transition probabilities in and out of that state, the exact equation is given in equation 5, where $p(i \to k)$ is the transition probability from state i to state k.

$$p(k) = \sum_i p(i) p(i \to k) \tag{5}$$

If we have M states in total, then there will be M coupled equations like equation 5 to be solved. In our application, however, we have the inverse problem, we want to know for what transition probabilities we get correct canonical probabilities for our spin configurations. Obviously there are many ways to achieve that, but one simple case is shown below:

$$p(k) \sum_i p(k \to i) = \sum_i p(i) p(i \to k) \tag{6}$$

In equation 6, we just multiplied the LHS by 1 since the sum of transition probabilities from state k to i should be unity.

$$\sum_i p(k) p(k \to i) = \sum_i p(i) p(i \to k) \tag{7}$$

[1]For example spin-z eigenstates

[2]As in the case with zero external magnetic field, so an individual spin has no directional preference

[3]Coefficients for spins like $\hbar/2$ is absorbed in J

[4]Neighbour of a spin at (x,y,L-1) is (x,y,0) and so on.

Equation 7 is clearly satisfied when every term in both of the summations are equal. This equality (equation 8) is the so called "Detailed Balance" condition, and although not being a necessary condition, Wolff's algorithm will satisfy it.

$$p(k)p(k \to i) = p(i)p(i \to k) \tag{8}$$

B. Calculating Transition Probabilities

Our choice of algorithm is the single cluster algorithm because of its speed and simplicity. As mentioned before for large lattices in 3D, it is not practically possible to calculate the energy, hence the probability of each state. However it is possible to calculate the ratio of probabilities of two similar states in which every lattice point has the same spin in both states, apart from a single connected cluster of lattice points that have opposite spins in respective spin configurations. In such a case the energy difference of these two states are given only by the interactions at the boundary of that cluster and the rest of the lattice. That is because every spin within the cluster is -1 in one spin configuration and 1 in the other, hence energy due to all internal interactions (S_iS_j) will be the same. On the other hand all the interactions between spins that are not in the cluster will give the exact same energy for both states since the spin configurations are the same everywhere outside the cluster. Let $n_1(n_2)$ be the number of interactions (number of distinct spin pairs) at the boundary with $S_iS_j = 1(-1)$ for a state i. When the spins of every lattice point within the cluster is flipped we will go to a different state k, and now $n_1(n_2)$ will describe the number of interactions with $S_iS_j = -1(1)$ respectively. The energy difference between the two states will be as in equation 9 and the ratio of canonical probability of those states will be as in 10.

$$E_i - E_k = -2J(n_1 - n_2) \tag{9}$$

$$\frac{p(i)}{p(k)} = e^{2J\beta(n_1-n_2)} \tag{10}$$

In order to make use of the results above, we need to have a method to create these connected clusters. First, the clusters should contain only a single type of spin. Second, not every lattice point with the same spin should be included in the cluster, otherwise the cluster will grow until it reaches opposite spins in all their boundaries, then n_1 will always be 0 and after several flips we will reach a spin configuration with all spins being 1 or -1. Then every lattice point with the same spin as the starting point of the cluster should be added to the cluster probabilistically. In Wolff's algorithm, the cluster is grown by starting from a random lattice point and bonding with its neighbours with the same spin with probability p_b. This process is then repeated for every spin in the cluster, so the cluster grows until there was no bonding at the entire boundary with the rest of the spin configuration. At the end every spin that is bonded with is counted within the cluster[5]. By using the previously defined n_1 and n_2 we can find the probability of stopping at a particular boundary. Given that we are in state i this is given by equation 11, where, p_s is the probability of selecting the specific starting point, p_{cl} is the probability of getting the specific internal structure of bonds of the cluster[6], and finally $(1-p_b)^{n_1}$ is the probability of not bonding with any of the lattice points with the same spin at the boundary of the cluster.

$$p(C|i) = p_s p_{cl}(1-p_b)^{n_1} \tag{11}$$

The same probability for state k, which is only different from state i due to the opposite spin values within the cluster C, is given in equation 12.

$$p(C|k) = p_s p_{cl}(1-p_b)^{n_2} \tag{12}$$

Notice only the probability of ending the cluster at the specific boundary is changing, when the spins in the clusters are flipped. In addition, since we flip the spins within the cluster after creating it to get a new state, $p(C|i)$ is equal to $p(i \to k)$, and similarly $p(C|k)$ is equal to $p(k \to i)$. Then by dividing equations 11 and 12, we obtain the following ratio 13.

$$\frac{p(i \to k)}{p(k \to i)} = (1-p_b)^{n_1-n_2} \tag{13}$$

By using the detailed balance equation given in equation 8 we can solve for p_b as in equation 14 to finally reach the bonding probability in equation 15.

$$\frac{p(i)}{p(k)} = e^{2J\beta(n_1-n_2)} = \frac{p(k \to i)}{p(i \to k)} = (1-p_b)^{n_2-n_1} \tag{14}$$

$$p_b = 1 - e^{-2J\beta} \tag{15}$$

If we follow the procedure of bonding with same neighbouring spins with probability p_b and flipping the resultant clusters, we will obey the detailed balance between spin configurations, and at the steady state we will sample every spin configuration proportional to its canonical probability.

[5]Notice this means there are more than a single way to include a lattice point in a cluster.

[6]Actually p_{cl} is the sum of probabilities over all the internal (not boundary) bond structures that will give the same cluster, but that will not change our argument, since p_{cl} cancels out either way.

C. Measuring the Spin Susceptibility

We are now able to generate spin configurations with the probability distribution required, but we still need to measure spin susceptibility of spin configurations to average over them. We can use the expression in equation 3, but it is not practical, especially at large lattices. When one thinks the spin configuration as combination of many clusters as of the type we described previously, in average contribution to S_iS_j term between different clusters is close to zero, especially when we average over many different spin configurations as in our MC. By neglecting all inter-cluster terms, and noticing every spin in a cluster has the same value, we obtain equation 16, where $[C]$ means sum over all clusters.

$$\chi_s = \sum_{[C]} \frac{S_C^2}{L^3} \tag{16}$$

Equation 16 is easier to calculate than equation 3, however one still needs to find every cluster in a spin configuration, which is against the idea of a single cluster algorithm. When we pick a random lattice point, the probability of it being in a specific cluster is simply $^{S_C}/_{L^3}$, hence equation 16 becomes the expectation value of cluster sizes, $\langle S_C \rangle$. Then we can simply measure cluster size for a given spin configuration multiple times and average the results, or better yet we can also slowly change the spin configuration by flipping the clusters after measuring them. When we flip spins on the order of $^{L^3}/_{\langle S_C \rangle}$ [7] [8] we can average the cluster sizes and call that a measurement of spin susceptibility. By repeating this procedure many times, we will both traverse the spin configurations and measure the quantity we need. Notice that by calling a measurement as the average of cluster size distribution, we invoke the central limit theorem, and guarantee at large number of measurements we will have a gaussian distribution, which will make estimation of mean and variance straightforward.

D. Summary of Algorithm

1) Initialize the lattice
2) Select a random lattice site
3) Bond to neighbours with same spin with probability $1 - e^{-2J\beta}$
4) Repeat 3 for all lattice points added to cluster until no more lattice points are added
5) Record cluster size and flip the spins within the cluster
6) Repeat steps [2,5] $O(^{L^3}/_{\langle S_C \rangle})$ times and call the average cluster size as a measurement of spin susceptibility.
7) Repeat steps [2,6] $N + 2O(L^3)$ times, where N is chosen based on desired statistical error
8) Throw the measurements corresponding to first $O(L^3)$ cluster flips away, to get rid of measurements taken before reaching the steady state of Markov process. Use the average of second $O(L^3)$ to estimate the cluster size for finding the $O(^{L^3}/_{\langle S_C \rangle})$.
9) Estimate the mean of the distribution with sample mean and standard error of the mean with $^{\sigma}/_{\sqrt{N}}$

III. Implementation

We choose to implement single cluster algorithm in C++, mainly using standard libraries but making use of CERN ROOT libraries for random number generation[9] and plotting tools. For such MC simulations the performance of the code is critical, since low statistical errors are desired in a limited time. Because the single cluster algorithm does not include any complex calculations, the limiting factor of speed would be memory access, therefore optimization of processor cache usage will be critical.

A good place to start describing our code, is from its data structures. Below is the struct utilized to hold the value of a spin and pointers to its neighbour lattice points, which is basically a multi-linked list structure. Such a structure will enable easy and elegant access to all neighbours and also make the processor read the address of neighbours preemptively[10].

```
struct SpinStruct{
  bool value=0; //Spin Value 0 or 1
  SpinStruct *nbrs[NNBRS]={NULL,NULL,NULL,NULL,NULL,
    NULL}; //Pointer to Neighbours
};
```

The script proceeds by creating a fixed size 3D array of the SpinStruct objects and initializes all neighbours. An example for neighbour initialization for a single dimension is given below, with special attention to periodic boundary conditions.

[7]So that on average we have touched every part of the lattice before finishing a measurement.

[8]It is important to use a pre-calculated $\langle S_C \rangle$ to prevent biasing the measurements. That is to say $^{L^3}/_{\langle S_C \rangle}$ should be estimated in the beginning of the calculation and kept constant, otherwise larger cluster sizes will be favored.

[9]Specifically TRandom3 with a period of $2^{19937} - 1$ and 5 ns call time was chosen.

[10]Of course reading the values of the neighbours instead would be ideal for speed reasons, but that is much harder to implement, since the value of that particular spin should be accessible by 6 other spins.

```
if(x == 0) {    //Special Attention if x is 0
  spinsIn[x][y][z].nbrs[0] = &spinsIn[length-1][y][z];
  spinsIn[x][y][z].nbrs[1] = &spinsIn[x+1][y][z];
}
else if(x==(length-1)) { // Special Attention if x=length-1
  spinsIn[x][y][z].nbrs[0] = &spinsIn[x-1][y][z];
  spinsIn[x][y][z].nbrs[1] = &spinsIn[0][y][z];
}
else {//Default neighbours are spins at x+1 and x-1
  spinsIn[x][y][z].nbrs[0] = &spinsIn[x-1][y][z];
  spinsIn[x][y][z].nbrs[1] = &spinsIn[x+1][y][z];
}
```

All the operations performed within a single measurement is given in code segment below. As the algorithm suggests we select a random spin [11], save its value to a temporary variable and flip its spin. This early flipping prevents checking this site again without setting any flags. This results in less conditions to check and more cache space available for other purposes. “addN2Queue” function checks all the neighbours and decides to add them to the cluster and to a queue in order to check their neighbours in the future. A queue object is suitable here because we do not need to access a specific neighbour but to merely take the next one to analyze. In addition, the queue is automatically feeding the it’s front element to processor cache making it a very efficient for our purposes. We keep growing the cluster until the queue is empty, and increase cluster size by 1 for every element in the queue. A measurement is completed when number of measured clusters pass the sweep threshold and our measurement becomes average of the cluster sizes. At the end of all measurements mean and standard error of the mean is calculated and reported.

```
sumClSize = 0; //Zero counters
clSizeMsCnt = 0;
while(true) { // While loop for updates within a sweep
  curSpin=&spins[randSpin.Integer(LATLEN)][randSpin.Integer(LATLEN)][randSpin.Integer(LATLEN)]; // Select a random spin
  spinVal = curSpin->value; //Record the value
  curSpin->value = !spinVal; //Flip the spin
  addN2Queue(tempQ,*curSpin,pBond,&randSpin,spinVal); //Add nbrs to queue
  clSize = 1; //Reinitialize cluster size
  while(!tempQ.empty()) { //Keep expanding the cluster until the queue is empty (nowhere to look)
    curSpin = tempQ.front(); // Get the first element of the queue
    tempQ.pop(); //Remove the element obtained from queue
    addN2Queue(tempQ,*curSpin,pBond,&randSpin,spinVal); //Add nbrs to queue
    clSize++; //Increase cluster size
  }
  sumClSize+=clSize; //Add to total cluster size for this weep
  clSizeMsCnt++; //Increase measured cluster count
  if(clSizeMsCnt > swpCnd[2]) break; //Break when reached the number of updates
}
spinSus[swpctr] = (double)sumClSize/(double)clSizeMsCnt; //Record measurement
```

The aforementioned inline[12] function is defined below, it simply decides to add every neighbour of the current spin to the cluster, if added to the cluster value of the neighbour spin is immediately inverted and it is added to queue, so that its neigbours can be checked later.

```
inline void addN2Queue(queue<SpinStruct*> &Q, SpinStruct &curSpin, double pBond, TRandom3 *inRand, bool spinVal) { //Function to fill neighbours into queue and flip their spin
  for(unsigned int nbrcnt=0;nbrcnt<NNBRS;nbrcnt++) { //Loop over nbrs
    if(curSpin.nbrs[nbrcnt]->value == spinVal && inRand->Rndm() < pBond) { //Condition to add a bond
      curSpin.nbrs[nbrcnt]->value = !spinVal; //Flip spins
      Q.push(curSpin.nbrs[nbrcnt]); //Store neighbour pointers in queue
    }
  }
}
```

The program is configured to be able to run from command line with inputs $J\beta$, L and NSWEEP. For running the program in an automated fashion for a set of input parameters bash scripts were utilized. The results were then fitted with χ^2 method, again by using CERN ROOT libraries.

It is also worth mentioning calculation of exact distribution for $2 \times 2 \times 2$ lattice briefly. The 256 states were generated by converting integers from 0 to 256 into binary and then initializing every spin to a bit. Then energy hence the individual probability of states, partition function and the spin susceptibility of the state can be calculated through equations 1, 2 and 3. At the end a weighted average

[11]This is one part of the code where random memory access was not eliminated, fortunately this part does not limit the speed since it only happens once per cluster.

[12]The speed boost of making this function inline was not tested, but should be considerable since this function is called for every element added to a cluster.

of spin susceptibility by the probability of spin configurations were performed.

The code was profiled with "perf" command of linux and found that ratio of cache-misses to cache reference is about 0.2% for $40 \times 40 \times 40$ lattice. Although the ratio could depend on the exact input parameters, this number shows $> 99\%$ of the time the processor did not need to look at the memory for a value it needed, which will be a couple orders of magnitude slower than using the cache. For reference running 10000 measurements with $L^3/\langle S_C \rangle$ updates for $100 \times 100 \times 100$ lattice at the critical temperature takes about an hour without any parallelization on a computer with 16GB of RAM and i7-6700HQ processors, with Ubuntu 16.04 as operating system.

IV. Results

A. Comparison with Exact Solutions in $2 \times 2 \times 2$ Lattice

Before moving to larger lattices, we compared our MC results with the exact results for L=2 at multiple temperatures. Results tabulated in Table I show perfect agreement between MC and exact calculations. Another check that can be made is to compare distribution of states for both exact and MC calculations, as in Figure 1. For ease of comparison we converted each state into an 8 bit number based on the spin values. The temperature was chosen specifically high ($J\beta = 0.1$) to make distribution of states more even and easier to compare by eye, and also show agreement at higher temperatures than the results tabulated in Table I.

TABLE I: Measured spin susceptibility for L=2 at different temperatures. Number of sweeps: 10000 and number of updates: 1000. Statistical errors are given up to 2 digits only, for $J\beta = 2$ case variance was zero, hence the statistical error.

$J\beta$	Exact χ_s	MC χ_s
0.2216544	4.16888	4.1693 ± 0.0011
0.5	7.89152	7.89149 ± 0.00027
1.0	7.99982	7.99982 ± 0.00001
2.0	8.00000	8.00000 ± 0.00000

B. Lattice Size Dependence at the Critical Temperature

Expected behavior of the spin susceptibility depending on the lattice length L, at the critical temperature ($J\beta_c = 0.2216544$) from the literature is given below in equation 17, where $\eta \simeq 0.03627$. Results of our MC was given in Table II with errors less than 1%.

$$\chi_s = C_2 L^{2-\eta} \tag{17}$$

Results from Table II was fitted with equation 17 with χ^2 method, the resultant curve and parameter values can be seen in Figure 2 and also tabulated in Table III. χ^2/ndf ratio around 1 suggests the fit model was well chosen. In addition results tabulated in Table III agrees with the previous values from the literature.

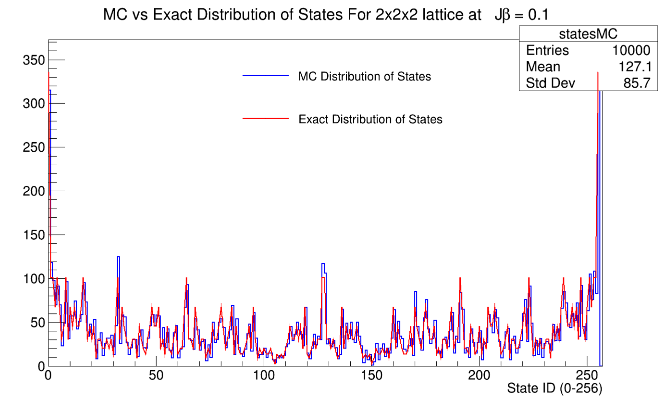

Fig. 1: Distribution of 256 possible states in MC and exact calculations. The state ID is enumerated from 0 to 255 by converting 8 bit spin information into its equivalent in base 10 (so 00000000 maps to 0 and 11111111 maps to 255).

TABLE II: Measured spin susceptibility at the critical temperature $J\beta_c = 0.2216544$ for different lattice lengths. Number of measurements is 10000 for L= 20 and 30, and 100000 for L>30 to reduce their error. Number of updates: $L^3/\langle S_C \rangle$

L	Spin Susceptibility χ_s
20	556 ± 4
30	1224 ± 8
40	2158 ± 5
50	3334 ± 7
60	4804 ± 10
70	6482 ± 14
80	8414 ± 18
90	10639 ± 23
100	13021 ± 26

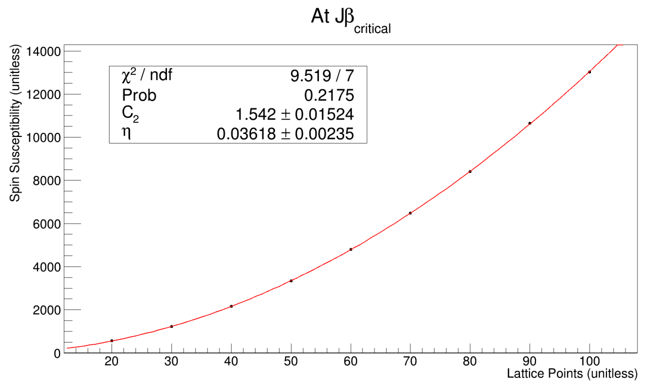

Fig. 2: Data points (black) and fit to equation 17 (red), fit results are also shown. Error bars cannot be seen because they are smaller than the size of markers.

TABLE III: Measured values of C_2 and η reported up to 2 significant digits of the statistical uncertainty. Previously calculated η falls into error range of our measurement.

η	0.0362 ± 0.0024
C_2	1.542 ± 0.015

C. Lattice Size Dependence Below Critical Temperature

Expected behavior below the critical temperature is a constant spin susceptibility with respect to L. Results of our MC was at $J\beta = 0.19948990$ $(0.9J\beta_c)$ given in Table IV with errors less than 0.1%.

TABLE IV: Measured spin susceptibility at the critical temperature $J\beta = 0.19948990$ $(0.9J\beta_c)$ for different lattice lengths. Number of measurements is 10000 for all L, with $L^3/\langle S_C\rangle$ updates.

L	Spin Susceptibility χ_s
20	18.72 ± 0.01
30	18.712 ± 0.004
40	18.715 ± 0.002
50	18.712 ± 0.002

Again the χ^2/ndf value indicates constant is an appropriate function for fit, and it can be seen that the resultant C_1 is within errorbars of all individual measurements. The fitted value of C_1 is reported in Table V.

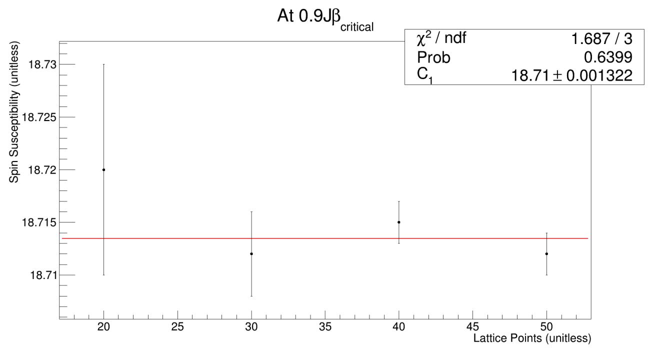

Fig. 3: Data points (black) and fit to a constant is red. The fit is passing through the errors of individual data points.

TABLE V: Measured values of C_1 reported up to 1 significant digits of the statistical uncertainty.

C_1	18.713 ± 0.001

D. Lattice Size Dependence Above Critical Temperature

Expected behavior of spin susceptibility above the critical temperature from the previous literature is given below in equation 18, where d is dimensions of the lattice (In our case: 3). Results of our MC at $J\beta = 0.2438198$ $(J\beta = 1.1J\beta_c)$ was given in Table VI with errors less than 1%.

$$\chi_s = C_0 L^d \tag{18}$$

The results and their fit to our expectations are shown in Figure 4. Again the χ^2/ndf value indicates equation 18 is an appropriate curve. Fitted values of C_0 and d is reported in Table VII.

TABLE VI: Measured spin susceptibility at the critical temperature $J\beta = 0.2438198$ $(J\beta = 1.1J\beta_c)$ for different lattice lengths. Number of measurements is 10000 for all L, with $L^3/\langle S_C\rangle$ updates

L	Spin Susceptibility (χ_s)
20	4016 ± 15
30	13613 ± 50
40	32134 ± 38
50	62758 ± 74

V. Discussion and Summary

A Monte Carlo for three dimensional Ising Model was performed by using Wolff's single cluster algorithm. With this MC the second order phase transition from ferromagnetic behavior to paramagnetic

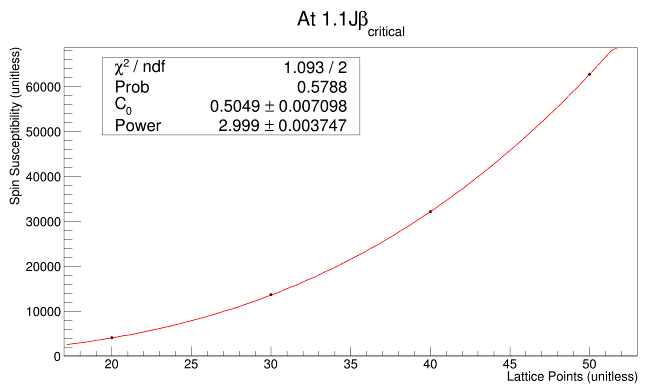

Fig. 4: Data points (black) and fit to equation 18 (red). Errorbars are smaller than marker sizes but χ^2/ndf indicates the fit is appropriate.

TABLE VII: Measured values of C_0 and d reported up to 1 significant digits of the statistical uncertainty. Note that $d = 3$ is well within error range of our result.

C_1	0.505 ± 0.007
d	2.999 ± 0.004

behavior was studied, by measuring the lattice size dependence of spin susceptibility around the critical temperature. We observed the expected behavior of susceptibility on lattice and calculated the critical exponent η, which agrees with the previous results from the literature. Beyond statistical mechanics the project was helpful because it led to considerations such as using Markov chains in Monte Carlo, and coding faster running simulation software.

VI. Guides

In this project no real physics articles were read or studied, the information guided me are listed below in a non-professional but honest way:

1) Dr. Shailesh Chandrasekharan's online lectures and project description, <https://webhome.phy.duke.edu/~sch/763/>, Access Date: 10/2018
2) École normale supérieure, Statistical Mechanics: Algorithms and Computations, online lecture 8, <https://www.coursera.org/lecture/statistical-mechanics/lecture-8-ising-model-from-enumeration-to-cluster-monte-carlo-simulations-uz6b3>, Access Date: 10/2018
3) Occasional Wikipedia
4) <http://www.cplusplus.com/>
5) <https://root.cern.ch/>