Characterising Phase Transitions using the Lee-Kosterlitz Analysis

Dhruv Pisharody and Satwik Wats

December 14, 2023

Professor: Bikram Phookun
Graduate Assistant: Philip Cherian
Course: Statistical Mechanics PHY-3610-1

Abstract

Systems that mathematically model ferromagnetism, like the Ising, q-Potts, and Classical Heisenberg models, often exhibit a phenomenon known as phase transition. This project attempts to characterise these phase transitions for our models, using the Lee-Kosterlitz algorithm. We were successfully able to determine the order of phase transition for 2D q-Potts and 2D Ising Model, and as for the 2D Classical Heisenberg Model, we were able to put up a strong argument for the absence of a phase transition using the same method. The project also explores the histogram method that allows the extrapolation of the behavior of the system in the vicinity of the critical temperature. This coupled with the Lee-Kosterlitz analysis provides a strong and more pronounced method of characterizing phase transition.

1 Introduction

Phase transition is often referred to as an abrupt change in the properties of a system as we tune a macroscopic parameter (for example, temperature) of our system. In the field of statistical mechanics we encounter many such transitions, for example Bose-Einstein Condensation, liquid-gas transition, and ferromagnetic to paramagnetic transition. In this paper, we will be discussing how to characterize the order of phase transition of different mathematical models of ferromagnetic materials using the Lee-Kosterlitz Algorithm. These transitions happen at a critical temperature and they reveal very rich and interesting physics of the material being studied at that critical point. For instance, we observe the coexistence of liquid and gas phases in Liquid-Gas Transition, and how a magnet loses its magnetization if heated above the critical temperature. Understanding these critical points not only helps us to better understand the material but also allows us to leverage their behavior near the point of transition for various physical applications.

It is worth noting that the Lee-Kosterlitz Algorithm used in this study can be applied to models beyond ferromagnets. Phase transition is a behavior that is common across multiple scientific disciplines, so the utility of the method extends far beyond the models we are considering. However, for the purposes of this study, we will limit our focus to this subset of materials that exhibit a phase transition from an ordered (ferromagnetic) phase to a disordered (paramagnetic) phase.

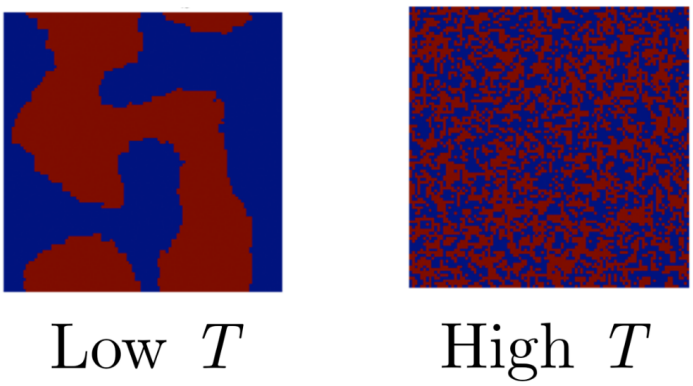

Figure 1: The above two cases represent the phase transition from a ordered to a disordered phase for a 2D Ising Model. The low temperature state shows order and a non-zero value of magnetization, while the high temperature shows random arrangements of spins, a discorded state of zero magnetization, reference 6.

An object exhibiting ferromagnetism consists of magnetic moments (spins), each of which tends to align in the same direction as its neighbouring spins, an alignment driven by exchange interactions between neighbouring spins. A common type of mathematical model used to represent this in statistical mechanics, are models with spins placed on a lattice. The different rules these spins can obey, like their possible orientations and the way in which neighbouring spins interact, lead to different such ferromagnetic models.

The phase transition is a consequence of the behavior of the spins due to the exchange interaction near the critical temperature. When the phase transition is happening between the two phases of the system (from ordered to a disordered phase), it is possible to construct an order parameter that would be compatible with the system and would explain its behavior. The order parameter generally vanishes in the disordered phase while being finite in the ordered phase.

Multiple models exhibit different types of transition, and as we are working with ferromagnetic material we find it is easier to categorize the phase transition in two types as they would help us characterize the transition: the first-order phase transition and the second-order phase transition. The most general definition of these types are based on the discontinuity of the derivatives of the free energy. If there is a discontinuity in the first derivative of the free energy (for example the magnetization or energy), then we have a first order phase transition. If the discontinuity is in the second derivative of free energy (for example the magnetic susceptibility or specific heat) then we have a second order phase transition.

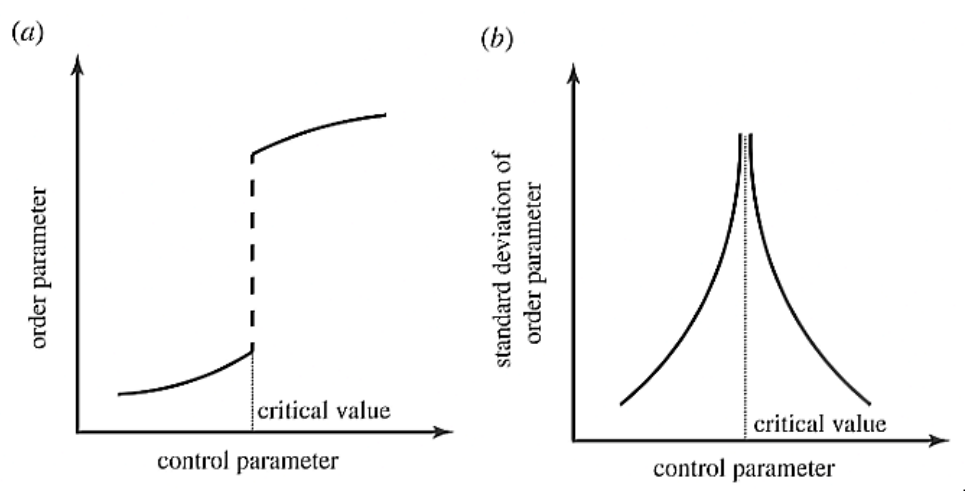

Figure 2: The plot of order parameter, which is the first derivative of free energy, against the control parameter (temperature, for example) shows a discontinuity for a first-order phase transition as seen in sub figure (a). On the other hand, a discontinuity in the second derivative of the free energy, as seen in sub figure (b) which shows a diverging behavior, indicates a second-order phase transition 8.

2 Ferromagnetic Models

2.1 2D Ising Model

One of the simplest ferromagnetic models is the 2D Ising model, where each spin on the lattice can take only two possible orientations: up (represented as +1) and down (represented as -1). The Hamiltonian of an Ising system of spins is given by

$$H = -J \sum_{\langle i,j \rangle} s_i s_j - h \sum_i s_i$$

where the sum is over nearest neighbour pairs of spins s_i and s_j, with J being a coupling constant. The second term is the energy of the system due to an external constant magnetic field, with h being the strength of this magnetic field.

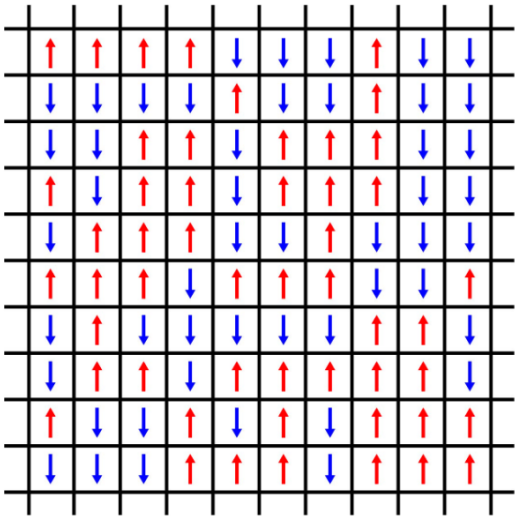

Figure 3: Lattice representation of 2D Ising Model, where the spins can point in either up or down direction. The interaction energy for each pair of spins take value either $+J$ or $-J$, depending on whether the spins in the pair have the same or different orientations.

The 2D Ising model is advantageous to work with not just due to the simplicity of its lattice, but also because it has been analytically solved. In 1944, Lars Onsager solved the 2D Ising model exactly, for the special case where there is no external magnetic field. This solution shows that the 2D model exhibits a second-order, or continuous, phase transition, with a critical temperature of 2.269 for $J = 1$.

2.2 q-Potts Model

The q-Potts model is a generalization of the Ising model. It allows the spins on the lattice to take any of the q values for a range of $(1, 2, 3, ..., q)$. To put it more rigorously, we can say that the spins can point in any of q evenly spaced directions on the unit circle, having a polar coordinate θ_s that takes the value $\theta_s = \frac{2\pi s}{q}$ (where s goes from 0 to $q - 1$ in steps of 1). Using this definition, we can express the Hamiltonian of the system as:

$$H = J \sum_{\langle i,j \rangle} \cos\left(\theta_{s_i} - \theta_{s_j}\right)$$

The sum is over the nearest neighbor pairs $\langle i, j \rangle$, and J is the coupling constant. This model is more commonly known as the vector Potts Model or the Clock Model, which transforms to the XY model in the limit $q \to \infty$. However, for our computational system, we work with something much simpler and often referred to as the Standard Potts Model. This method simplifies the interaction between the spins to the Kronecker delta function:

$$H_p = -J_p \sum_{(i,j)} \delta(s_i, s_j)$$

For example, a $q = 3$ state would allow the spins to take values of 1,2 and 3. Their nearest neighbor interaction will give 0 as the interaction energy if the spins are not in the same state, and $-J_p$ if they are in the same state (i.e., they point in the same direction).

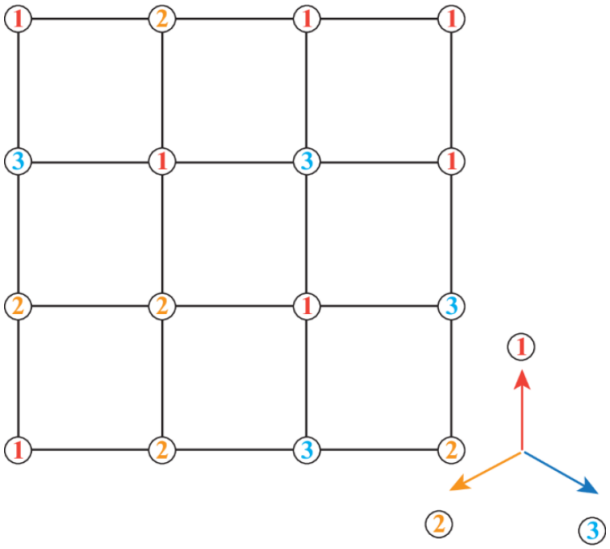

Figure 4: Lattice representation of q=3 state Potts Model. The spin on the lattice can take any of the q=1,2,3 values. If the neighbouring spins are in the same direction then that bond will have interaction energy of $-J_p$ otherwise they will have zero interaction energy 7.

The q-Potts model exhibits very intriguing behavior near the critical q-point of $q_c = 4$. For $q \leq q_c$, we have a continuous phase transition for our Potts model. However, as soon as we keep $q > q_c$, we have a first-order phase transition. For our project, we will be working above the critical q-point and primarily examining the behavior of the $q = 8$ Potts model. Analytically, the critical temperature for any q-state Potts Model is defined by:

$$T_c = \frac{1}{\ln(1+\sqrt{q})} \tag{1}$$

We can use this equation to study the phase transition of the different q-state Potts models near the critical temperature, and characterizing their phase transitions.

2.3 2D Classical Heisenberg Model

The Classical Heisenberg Model is a further extension of the q-Potts model, where our spins can now be any vector on the unit sphere. Unlike the Ising and q-Potts models though, which exhibit a discreteness in the possible values a spin can take, the Classical Heisenberg Model spins have a continuous range of values they can take. The Hamiltonian of this model is given by

$$H = -J \sum_{\langle i,j \rangle} \mathbf{s_i} \cdot \mathbf{s_j} - \mathbf{h} \cdot \sum_i \mathbf{s_i}$$

Here the first term is the interaction energy, a sum over nearest neighbouring pairs of spins s_i and s_j with J being a coupling constant. The second term is the energy of the system due to an external constant magnetic field, which for this project we simplified to being simply a field in the z-direction.

The Classical Heisenberg Model has not been analytically solved, however the global rotation symmetry it shows (the invariance of the Hamiltonian when all spins are rotated by some angle, when there is no external magnetic field) means that the Mermin-Wagner theorem is applicable to it. This theorem states that any systems (in dimensions less than or equal to 2) with sufficiently short-range interactions and exhibiting continuous symmetry, will not have this symmetry be spontaneously broken for any finite temperatures. Since a phase transition is effectively a breaking of the symmetry of the order parameter at some finite temperature, this means that the Classical Heisenberg Model does not exhibit any phase transitions.

3 Computational Methods and Algorithms

3.1 Metropolis-Hastings Algorithm

Now that we have laid the groundwork for what each of our three models are, we can begin computationally simulating them. To simulate the system at a given temperature, we use what is known as the Metropolis-Hastings algorithm. We begin with an initial system of spins on a lattice, and a temperature T we want to simulate the system at. Choosing a random spin, we perform a trial rotation of it. For the Ising model, this is a direct flip, for the q-Potts model this is a choosing of any of the q states the spin can take, and for the Classical Heisenberg model this is a choosing of any random point on the unit sphere (a possible value for a spin to have here).

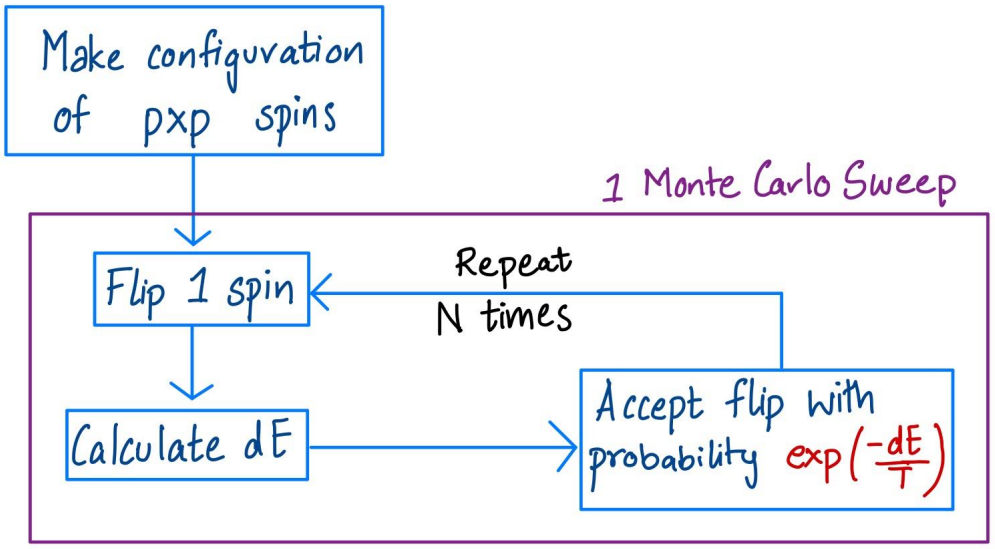

Figure 5: Flowchart depicting the steps for one Monte Carlo sweep of the Metropolis-Hastings algorithm

We then calculate the change in energy dE this trial rotation would cause, by calculating the change in energy of just the rotated spin instead of for the whole lattice. If this dE is negative, i.e. the rotation lowers the energy of the lattice, it is accepted. If the dE is positive, it is accepted with a probability $\exp(-dE/T)$, i.e. with a probability drawn from the Boltzmann distribution. Computationally, we achieve this by choosing a random number between 0 and 1, and accept the rotation only if it is less than $\exp(-dE/T)$. Whether the change is accepted or rejected, we move on, repeating these steps of choosing a random spin and performing a trial rotation N times, where N is the number of spins on the lattice, and this is referred to as one Monte Carlo sweep. In one sweep, on average, each spin is selected and rotated.

The Metropolis-Hastings algorithm allows us to simulate our system at any temperature we want, by running a large number of Monte Carlo sweeps on it at this temperature, and waiting for it to reach equilibrium. This equilibration can be observed by plotting the energy (or magnetisation) of the system after each sweep, upon doing which we observe a stabilisation in the value it takes after about 500 sweeps. Once this equilibrium has been reached, the energy and magnetisation of the system at this temperature can be determined by simply taking the mean of the values of these quantities after every sweep after equilibration.

For the purposes of identifying and characterising phase transitions therefore, the Metropolis-Hastings algorithm can be very helpful. By simulating the system over a range of temperatures, and plotting the energy and magnetisation of it against temperature, we can identify peaks and discontinuities, which as can be seen from figure 2 above, can allow us to identify the kind of phase transition of the system.

However, this method of identifying phase transitions is not very conclusive. What would seem to be a discontinuity in a graph may only be a relic of computation on a small, finite lattice (which mathematically would not have any phase transitions at all), and not an indication of a phase transition. Additionally,

obtaining the graph in the first place requires simulating the system for a large number of temperatures, and with simulation itself requiring a large number (on the order of 10,000) of Monte Carlo sweeps to be run, and above all this the system itself needing to be quite large to get significant results, this becomes a computationally impractical method of working on this problem. What we would instead like is to get data about the behaviour about the system near the critical temperature, using a method that is easier and faster to perform. This is where we turn to the Lee-Kosterlitz analysis.

3.2 Lee-Kosterlitz Analysis

The Lee-Kosterlitz analysis is a direct consequence of Landau's theory, which states that we can construct the free energy of a system near the critical temperature, given that it obeys the symmetry of the Hamiltonian and is analytic in the order parameter and its derivatives. Once we identify the order parameter for our system, we can perform a Taylor expansion of the free energy in terms of the order parameter because we know that it takes a small value near the critical temperature. A similar treatment of free energy and order parameter can be seen through the mean field approach. However, the signs of the coefficients in this expansion play a central role in characterizing phase transition. This Taylor expansion assumes that any non-trivial dependence of temperature will be found only in the lower order terms of the expanded series *.

The Taylor expansion of the free energy F for the system with one-dimensional order parameter η gives us:

$$\Delta F = -h\eta + a\eta^2 + c\eta^3 + b\eta^4 + ... \tag{2}$$

Here, $\Delta F = F - F_0(T)$, where $F_0(T)$ is a constant with respect to the order parameter (being only a smooth function of temperature), and therefore does not participate in characterizing the nature of the phase transition. The term h represents the perturbing field that couples linearly with the order parameter. The odd power terms will disappear from the expansion† if the spin system has some kind of symmetry causing its order parameter to exhibit an invariant nature under sign change ($\eta \to -\eta$), like the global rotational symmetry of the Classical Heisenberg Model.

For systems exhibiting such symmetries, we only need to keep terms up to the fourth order, as the order parameter is very small near the critical temperature. However, to ensure our system is thermodynamically stable, the highest order term present in our series must have a positive coefficient, so if the quartic term has a negative coefficient, we take the expansion till the 6th order. The sign of the quartic term plays a significant role in understanding the difference between the orders of phase transitions, being negative for a first-order phase transition, and positive for a second-order phase transition.

To get a more clear picture of how we can use this expansion of free energy to characterize phase transition, we will work with the simplest case, where we implement the Landau theory to the 2D Ising Model. The order parameter for a 2D Ising Model is the net magnetization m. The system has Z_2 symmetry, meaning the up and down state are same under the rotation by π. As we are working with a system that is invariant under $m \to -m$ transformation, the expansion of free energy near the critical temperature will be:

$$\Delta F = -hm + C_1 m^2 + C_2 m^4 \tag{3}$$

As the first linear term is dependent on an external magnetic field which may not be present, the effective lowest order term here is quadratic in m, and so (as per one of the assumptions of Landau theory) it will have the temperature dependence, changing sign below and above a critical temperature (T_c), meaning we can rewrite the above expression as:

$$\Delta F = -hm + a(T - T_c)m^2 + bm^4 \tag{4}$$

Now, if we minimize the free energy with respect to the order parameter we get:

$$\frac{\partial \Delta F}{\partial m} = -h + 2a(T - T_c)m + 4bm^3 = 0 \tag{5}$$

Which, if our system is not in an external magnetic field ($h = 0$), becomes

$$\frac{\partial \Delta F}{\partial m} = 2a(T - T_c)m + 4bm^3 = 0$$

*The constants in other higher order terms might also have temperature dependence, but they have a negligible effect near the critical point.

†Except for the first linear term $h\eta$, due to its coefficient not necessarily sharing the same symmetry.

$$\implies m = 0, +\sqrt{\frac{a(T_c - T)}{2b}}, -\sqrt{\frac{a(T_c - T)}{2b}} \tag{6}$$

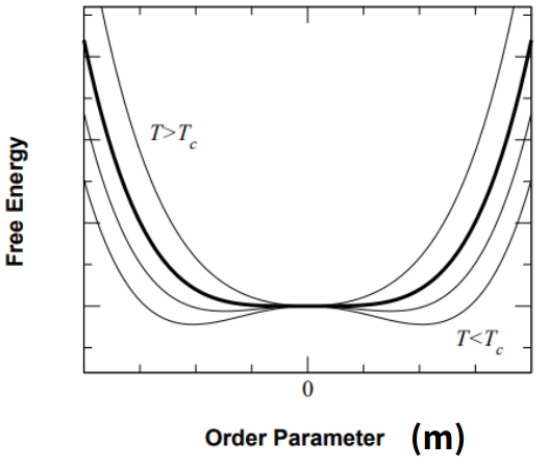

Figure 6: Free Energy against Magnetisation (the order parameter), for temperature above and below the critical temperature; (Olmsted, pg. 10, reference 8)

For $T > T_c$ we get $m = 0$ as the only real solution for magnetization, but for $T < T_c$ we get the latter two non-zero solutions as well, with $m = 0$ becoming an unstable solution as can be seen in figure 6 above. This behaviour of the magnetisation, being zero for one range of temperatures and non-zero for another, represents the two phases of the system.

Now, looking at the second derivative of the free energy with respect to the magnetisation, i.e. the magnetic susceptibility χ, also defined as

$$\chi = \frac{\partial m}{\partial h}$$

we can look at what happens to it near the critical temperature. Taking the partial derivative of both sides of equation 5 with respect to the external magnetic field,

$$\frac{\partial}{\partial h}\left(\frac{\partial \Delta F}{\partial m}\right) = -\frac{\partial h}{\partial h} + 2a(T - T_c)\frac{\partial m}{\partial h} + 4b\frac{\partial m^3}{\partial h}$$

$$0 = -1 + 2a(T - T_c)\chi + 12bm^2\chi$$

$$\chi = \frac{1}{2a(T - T_c) + 12bm^2}$$

For $T > T_c$, the magnetisation will only take on the value 0, making the above

$$\chi = \frac{1}{2a(T - T_c)}$$

However, for $T < T_c$, it can take on either of the latter two non-zero values from equation 6, which when plugged into our expression above makes it

$$\chi = \frac{1}{2a(T - T_c) + 12b\frac{a(T_c - T)}{2b}} = \frac{1}{4a(T_c - T)}$$

Clearly, at $T = T_c$, the susceptibility will diverge, creating a discontinuity. This discontinuity in the second derivative of the free energy is a hallmark of second-order phase transitions, and characterises the Ising Model as having such a one.

As mentioned when discussing the Metropolis-Hastings algorithm above, analysing the behaviour of these second derivatives of the free energy as above is an impractical and not very conclusive method to characterise phase transitions, even if it greatly benefits our mathematical model. If we want to obtain a more conclusive result computationally, we can rely on the analysis of the above free energy performed by Lee and Kosterlitz.

In their original paper, they claimed that when ΔF is expanded as a function of the order parameter as well as the lattice size, we see that it is a universally increasing function of the lattice size for a first-order phase transition and a constant function of lattice size for a second-order phase transition. So, analyzing the Landau free energy near the point of transition against the order parameter will give us the double minima that indicates the possible existence of phase transition and varying it with lattice size and then analysing it will give us the order of the phase transition.

At the point of phase transition, the plot of the free energy against the order parameter will yield a pronounced double minima and we know that as $L \to \infty$ we have $T_c(L) \to T_c$ for a finite system, therefore our value of critical temperature will be close to the theoretical value but, never exact. Still, we can work with an approximate a value of critical temperature that is $T_c(L)$, for that if we equalize the depths of the two minima by changing the temperature (we will look it in a little more detail in the next section). That particular temperature will be the critical temperature for the corresponding lattice size. Once we have the ΔF at that lattice size, we can repeat it for multiple lattices and use it to make draw more conclusive observations about the system.

3.3 Lee-Kosterlitz Algorithm

Once we have identified the critical temperature using the Metropolis-Hastings algorithm, we can turn the Lee-Kosterlitz Analysis into a computer algorithm to characterize phase transitions. We run the Metropolis-Hastings algorithm again, but this time just for a single temperature value, the critical temperature. We run it for a large number of Monte Carlo sweeps (around 100,000) and store the data for the order parameter. The reason for this is that we can utilize the behavior of the system near the critical temperature to obtain data about its behaviour at nearby temperatures as well.

Using the Single Histogram Method detailed by Ferrenberg and Swendsen, we create a histogram of the stored order parameter data, thus getting an array of values (say the energies E), which in our code are the midpoints of the bins, and the corresponding counts $H(E)$. Once we have this, we can get the value of any function of the energy $A(E)$ at a temperature close to the critical temperature, using the expression

$$A(E,\beta) = \frac{\sum_i A(E_i) \cdot H(E_i) \cdot e^{-(\beta-\beta_c)E_i}}{\sum_i \cdot H(E_i) \cdot e^{-(\beta-\beta_0)E_i}} \tag{7}$$

where $\beta = 1/T$ is the inverse temperature at which we want to extrapolate data, and $\beta_c = 1/T_c$ is the inverse of the critical temperature at which we have performed the simulation that gave us the histogram. As can be seen in the graph below, where we first simulated a 2D Ising lattice over multiple temperatures individually to obtain the energy, then compared it with the results from this single histogram method, the two methods give very close results. Thus, we can conclusively use the single histogram method to obtain data of our system near the critical temperature, which we will then use to perform the Lee-Kosterlitz algorithm.

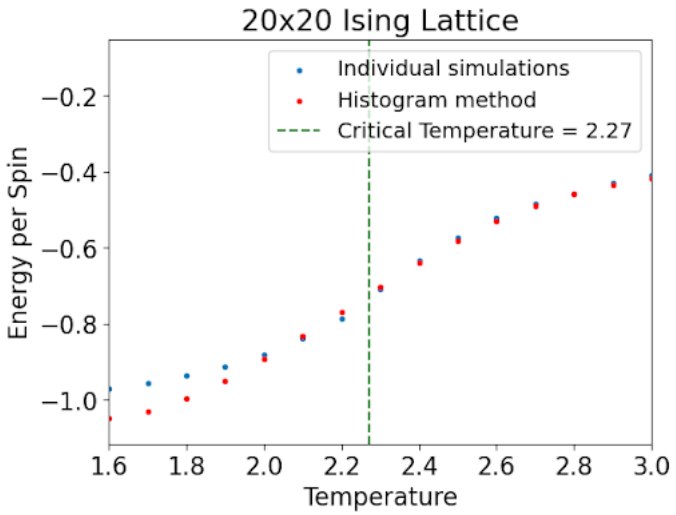

Figure 7: Comparison between energy data for a 2D Ising lattice near the critical temperature, as obtained from individual simulations (in blue) and from the Single Histogram method (in green)

Given that we can extrapolate the behavior of the system near the critical temperature using the technique developed by Ferrenberg and Swendsen, this saves us a lot of computational memory and time. We take the data we extracted at the critical temperature and calculate the free energy using equation 8 and plot it against the order parameter. It is important to note that the binning needed to do for the Histogram method requires multiple attempts to get the right number. It should be done carefully and all the empty bins should be removed when calculating the free energy. The free energy itself is calculated as

$$F(E) \propto -\ln H(E) + (\beta - \beta_0)E \tag{8}$$

Here, β is the inverse temperature that is very close to the critical temperature β_0.

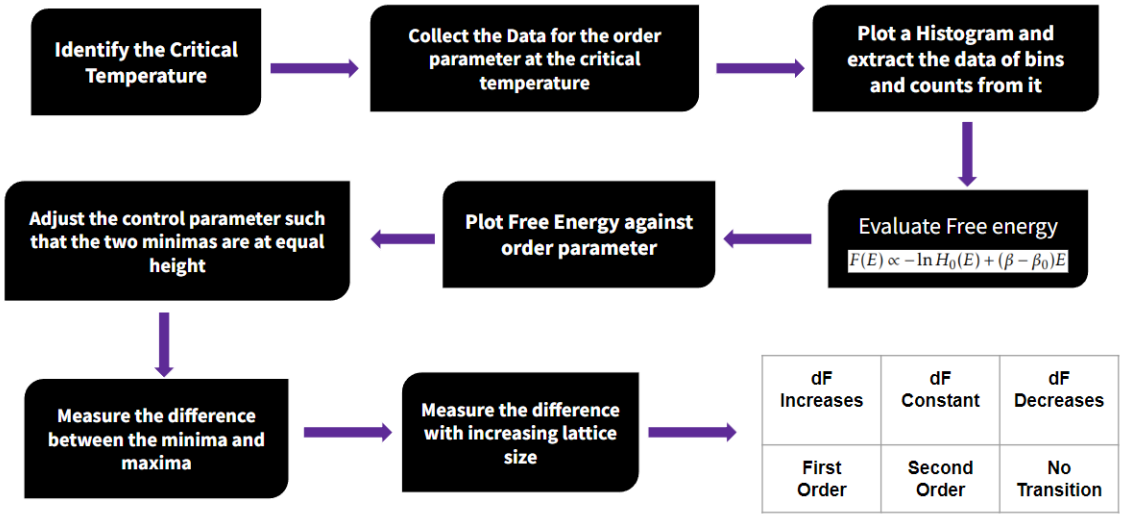

Figure 8: Flowchart depicting the steps for implementing the Lee-Kosterlitz algorithm

The plot of free energy against the order parameter would give us a double minima, and our next step would be to get an approximate value of critical temperature by varying β in the above equation till the two minima are at equal depth. This β is the representation of the critical temperature for that particular lattice size. This is an important step as we are working with finite lattice size, and as mentioned before we will have different critical temperature ($T_c(L)$) value for different lattice sizes. We can now measure the difference between the minima and the maxima $\Delta F(L)$ by curve fitting a polynomial function to the graph to smoothly observe these extrema. Finally, we repeat these steps for multiple lattice sizes and plot $\Delta F(L)$ as a function of lattice size.

If we observe that the value of $\Delta F(L)$ is increasing with increasing lattice size, then the system will exhibit first-order phase transition. If $\Delta F(L)$ is constant with increasing lattice size we get a second-order phase transition. Finally if we see a decreasing trend then the system shows no phase transition.

Though this quantity should ideally be calculated when the two minima are at the same depth, which we can get by varying the temperature T in the Lee-Kosterlitz algorithm, this is a task that must be performed manually for each lattice (even for different runs of the same lattice size), and would therefore be impractical to do each time we run the Lee-Kosterlitz algorithm. Thus, what we instead did was to implement the Lee-Kosterlitz algorithm at a temperature $T = T_c$, where we would get double minima that while not at the same depth, were at depths close to each other. Then, to calculate ΔF, we took the distance between the y-coordinate of the central maximum and the mean of the y-coordinates of the two minima.

3.4 Methodology

We will be working with three different models and for each of them, we will follow a procedure that would help us conclusively characterize the phase transition for them. The procedure involves several steps.

1. First, we need to identify the order parameter of our system as it differentiates the ordered and disordered phases.

2. After that, we need to look for indications of phase transition and at what temperature it might be. This can be done using the plots of the second derivatives of free energy. The discontinuity in the second derivative will give us an approximate value of our critical temperature.

3. We will then use the Histogram method to extract the data for the order parameter near the critical temperature to extrapolate the behavior of our system in the vicinity of the critical temperature.

4. After that, we will plot a histogram of the order parameter and collect the data for the count and midpoint of each bin. Using the counts and bins, we will calculate the free energy using the equation 8.

5. If we observe the double minima, our next step would be to equalize the depths of the minima to get an approximate critical temperature for that particular lattice size. We will then calculate the difference between the minima and the maxima for multiple lattice sizes.

4 Results and Discussion

4.1 q-Potts Model

In our analysis of phase transitions, we first used the q-Potts model. We initialize a lattice of spins that can have any of the q-values for a particular q-state and simulated it over a large range of temperature using the Metropolis-Hastings algorithm (running around 100,000 sweeps) for lattice size of 32x32. Now, to determine the order parameter of the system, we plotted the energy per spin against the temperature and observed its behavior (energy is used as the order parameter for the system in the original paper by Lee and Kosterlitz). The plot of energy per spin showed a transition from a non-zero to a zero value as shown in Fig 9 for multiple q values, which is a direct behavior of the order parameter. Thus, we conclude that energy per spin is the order parameter for our system.

Once we had identified the order parameter, we looked for indications of phase transition and the value of critical temperature by plotting the second derivative of the free energy against the temperature for q=8 state Potts Model. We observed the peak for q=8 to be very close to the theoretical value of 0.744, as shown in Fig 10. We can use the peak of specific heat to evaluate the critical temperature for various q values because we now know it is reliable, these critical temperature values will be used in our analysis of phase transition.

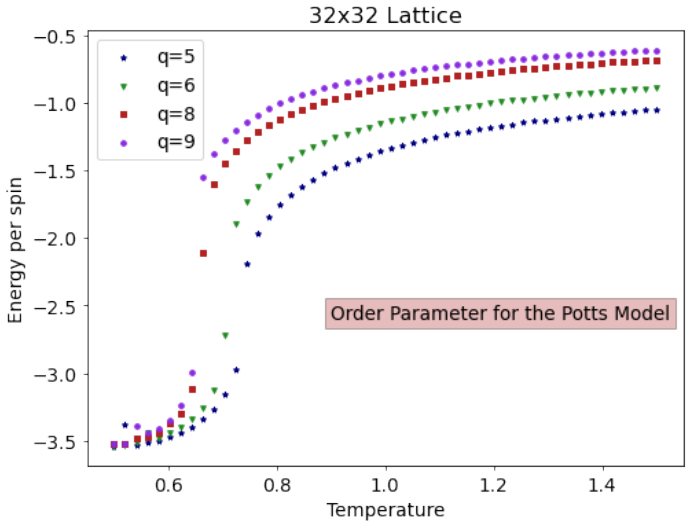

Figure 9: The energy per spin as a function of temperature reveals its nature of being an order parameter. It takes a non-zero value around 0.6-0.7 and then starts to asymptote towards a zero value for higher temperature. We performed it for multiple q values to ensure that the behavior remains the same throughout. The simulation was performed for a 32x32 lattice and 100,000 Monte Carlo sweeps.

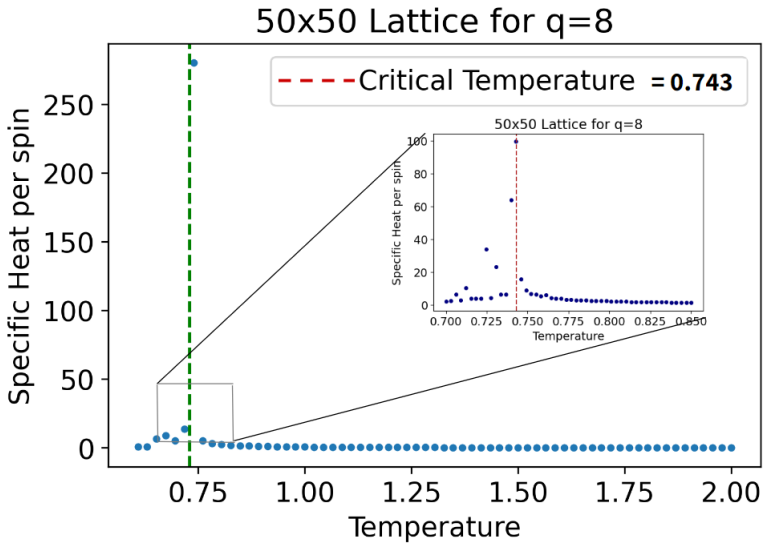

Figure 10: The specific heat shows a diverging value at a temperature around 0.75 for q=8 state Potts Model. When the area near 0.75 is zoomed in we see a much more pronounced behavior of the specific heat at a temperature of 0.743, this is the critical temperature for our system and this value is very close to the theoretical value of 0.744. Therefore, the peak of specific heat can be used to identify the critical temperature for various other q values.

It is important to consider that the discontinuity observed in the first or the second derivatives would only be a weak indication of the order of phase transition and could not be used as a conclusive result for characterizing phase transition. Our main goal from the plots of derivatives of free energy was to confirm the order parameter and identify the critical temperature.

Next, we extract the data for the order parameter and the temperature near the critical point and plot a histogram to get the counts and bins. Using equation 8, we calculated the free energy for a particular lattice size and tuned the value of β to equalize the depths of the two minima, this is our critical temperature for the particular lattice $\beta_c(L)$.

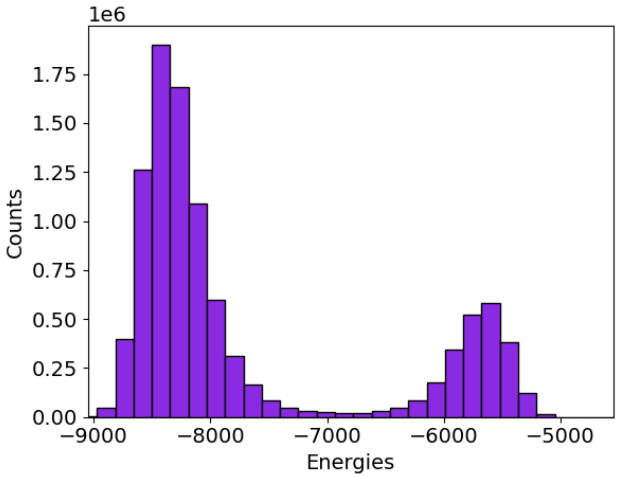

Figure 11: Histogram of Energies of a 72x72 8-state-Potts Lattice at the critical temperature $T_c = 0.74$

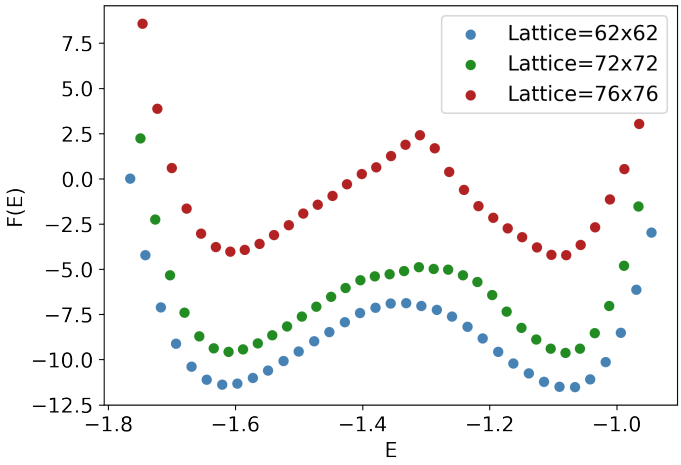

Figure 12: Free Energy against Energy for different lattice sizes of an 8-state-Potts Lattice; as the lattice size increases, the depth of the minima visibly increases

We plotted (Fig 13) the obtained free energy against the order parameter and did a curve fit to identify the minima and maxima of the curve. We then calculated the difference between them and called it $\Delta F(L)$. We repeated the same free energy analysis for multiple lattice sizes and then plotted free energy as a function of lattice size. From the resulting plot, we observed that the value of ΔF decreases with the inverse of lattice size (meaning it would increase with increasing lattice size), indicating a first-order phase transition.

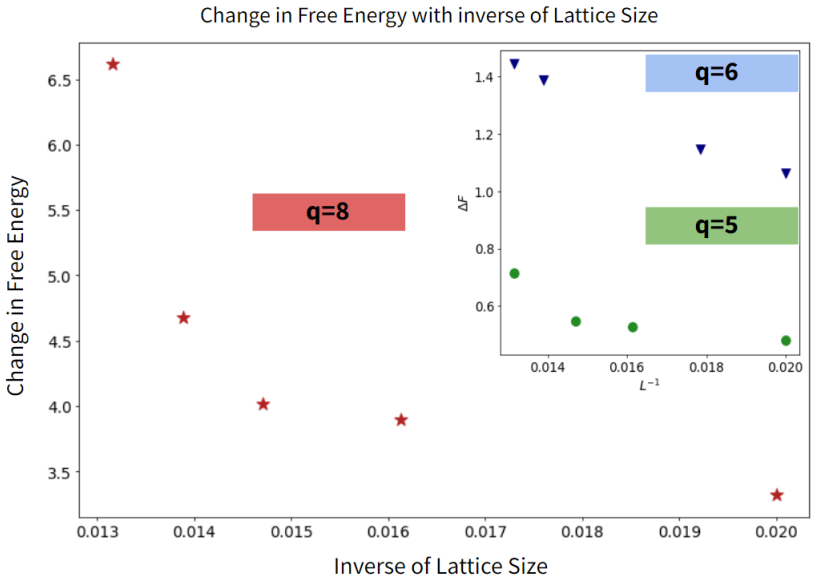

Figure 13: There is a clear decrease in the value of ΔF for q=8 when plotted against inverse of lattice size. The plot for q=6 and q=5 have been added as a subset to the plot of q=8 because the change they showed with respect to q=8 made it very difficult to see the trend line.

We can also cross check our result with the original work presented by Lee and Kosterlitz where they had also plotted the change in Free energy for q=8,6 and 5 state Potts Model against the inverse of lattice size. We have scaled up the values for q=6 and q=5 to highlight the trend they follow, and we find our result to be in agreement with their original research work as shown in Fig 14.

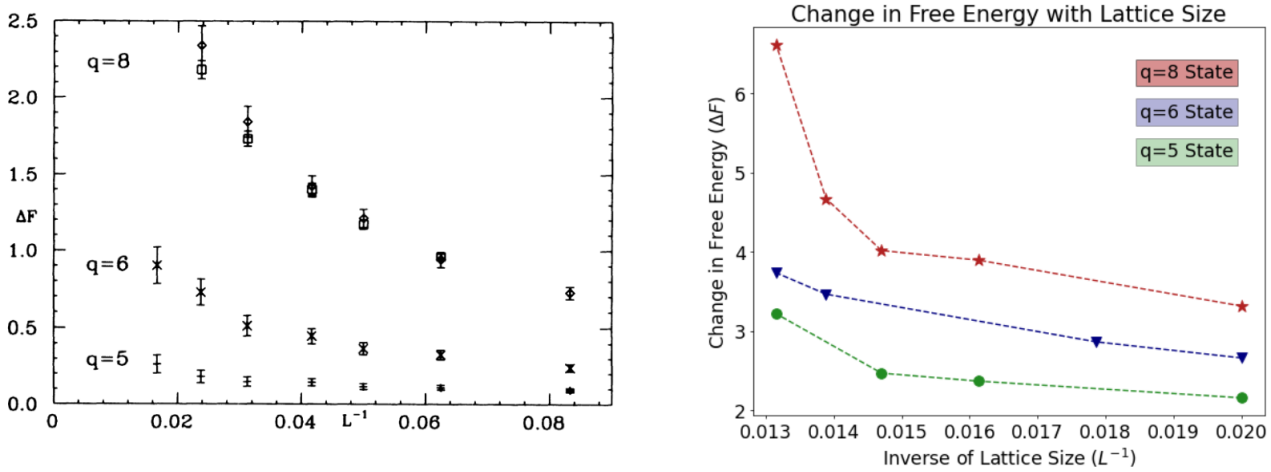

Figure 14: On the left we have the original work performed by Lee and Kosterlitz and on the right we have our analysis done for the same state Potts Model. Our results are in agreement with the original work.

4.2 2D Ising Model

After successfully identifying the phase transition of the q-Potts model, we moved on to the 2D Ising Model. As with the q-Potts model, we created a system of spins all starting with an orientation +1, and simulated it over a range of temperatures using the Metropolis-Hastings algorithm (running 10,000 Monte Carlo sweeps), plotted the resulting energies and magnetisations against their corresponding temperatures. Here, we observed that it was the magnetisation that went from a non-zero value to a zero value. Thus, we identified the magnetisation to be the order parameter, and plotted its susceptibility against temperature as well to observe anomalous behaviour and thus identify a critical temperature.

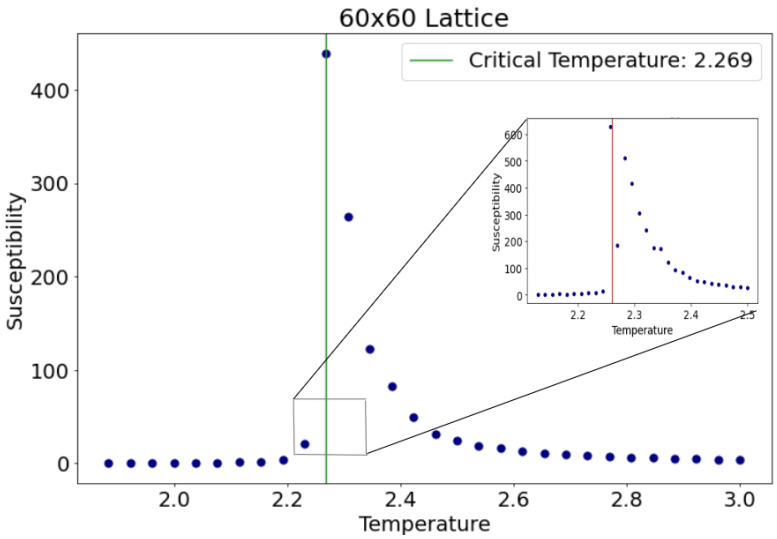

Figure 15: The Susceptibility shows a diverging value at a temperature around 2.25 for 2D Ising Model. When the area near 2.25 is zoomed in we see a much more pronounced behavior of the susceptibility at a temperature of 2.269 (or 2.27), this is the critical temperature for our system and this value is very close to the theoretical value. Therefore, the peak of susceptibility can be used to analyze the behavior of the system near the critical point.

The graph of the susceptibility against temperature shows a peak at a temperature somewhere around 2.5 (thus being a preliminary, albeit weak indicator of a second-order phase transition), and to further narrow this down we performed the same steps again but for a smaller range of temperatures. This gave us a critical temperature of about 2.269.

Now with the critical temperature identified, we used the single histogram method, simulating the system at this critical temperature using the Metropolis-Hastings algorithm for a large number (100,000) of Monte Carlo sweeps, and collecting the magnetisation values after each sweep after equilibrium had been achieved, what we took to be after 3000 sweeps. We plotted a histogram of this data, and used it to calculate the free energy at different magnetisations.

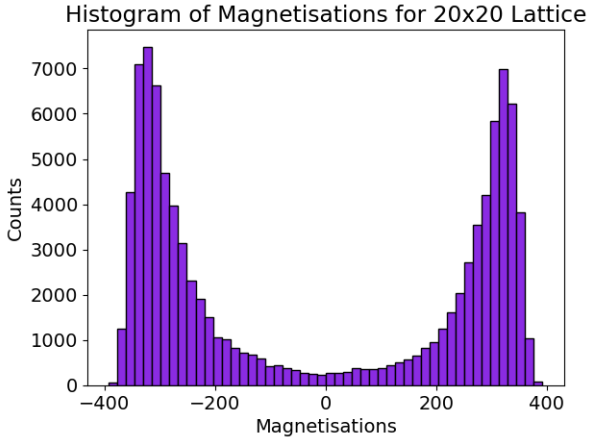

Figure 16: Histogram of the values of magnetisation of the system at the critical temperature $T_c = 2.27$

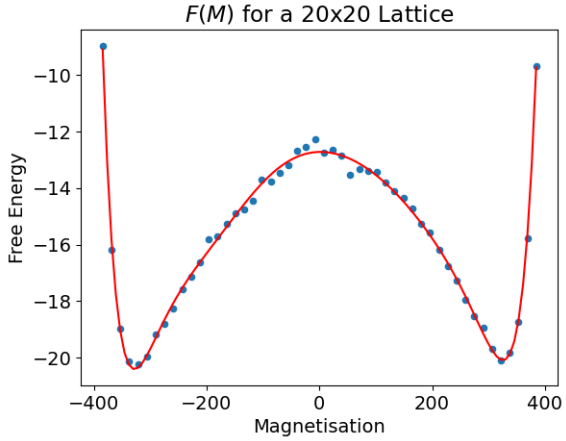

Figure 17: Free energy as a function of magnetisation, showing a double minima, with a polynomial fit (in red) to determine more accurately the extrema

Having successfully seen the double minima in the graph of $F(M)$ against M, we can calculate the distance between the central maximum and minima ΔF. To accurately determine the extrema of the graph, we fitted a 10th order polynomial[‡] to the points, and used the `scipy find_peaks` function to find the minima and

[‡]This polynomial does not represent a theoretical curve that the free energy follows, it is merely a curve fitted to more smoothly identify the extrema.

maxima of that. We determine this for a range of lattice sizes L, and plot the resulting ΔF against the corresponding L.

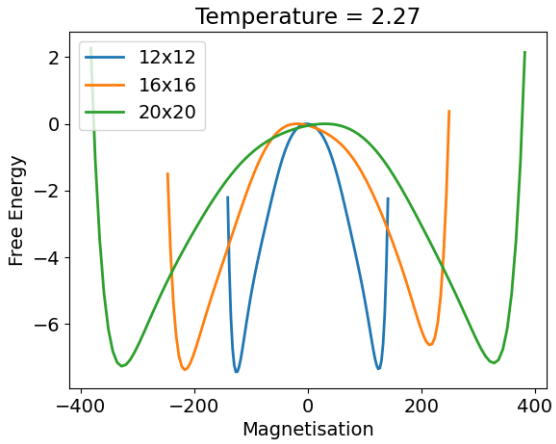

Figure 18: Free energy against magnetisation for some lattice sizes, all normalised so their maxima lie on the line $y = 1$

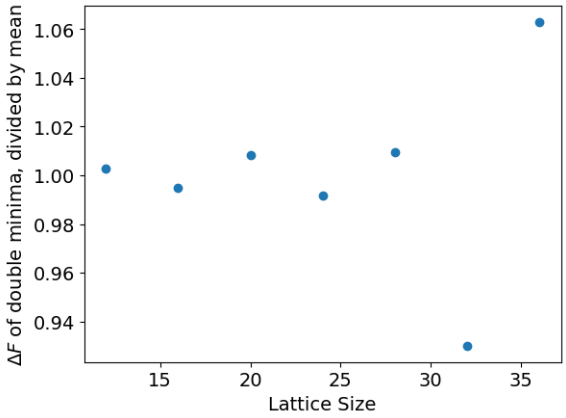

Figure 19: The distance between the central maximum and minima for the F against M graphs of different lattice sizes, all divided by their mean value

The graph shows points that remain more or less constant, with changes only by a small amount. This can also be seen by the image above on the left, where the distance between the maximum and minima remains more or less the same even as we increase the size of the lattice. From this, we can conclude that the 2D Ising Model shows a second-order, or continuous phase transition.

4.3 2D Heisenberg Model

Finally, after having succesfully used the Lee-Kosterlitz algorithm on the q-Potts and Ising models, we turned to the Classical Heisenberg model. Though this model does not exhibit any phase transitions, we still wanted to see what the Lee-Kosterlitz algorithm could do with it. As with our previous two models, we first attempted to identify an order parameter, by observing the behavior of both energy and magnetisation[§] against the temperature.

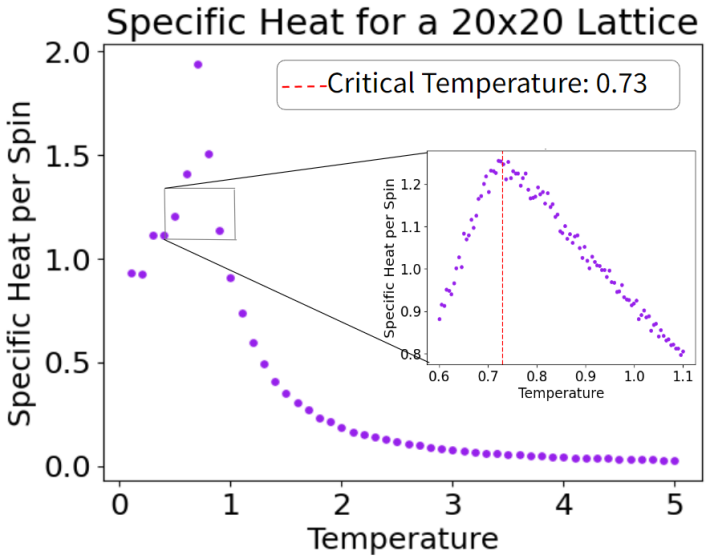

Figure 20: The specific heat shows a diverging value at a temperature around 0.75. When the area near 0.75 is zoomed in we see a much more pronounced behavior of the specific heat at a temperature of 0.73, this is the critical temperature for our system. Therefore, the peak of specific heat can be used to analyze the behavior of the system near the critical point.

[§]The magnetisation was calculated by taking a vector sum of all spins in the lattice

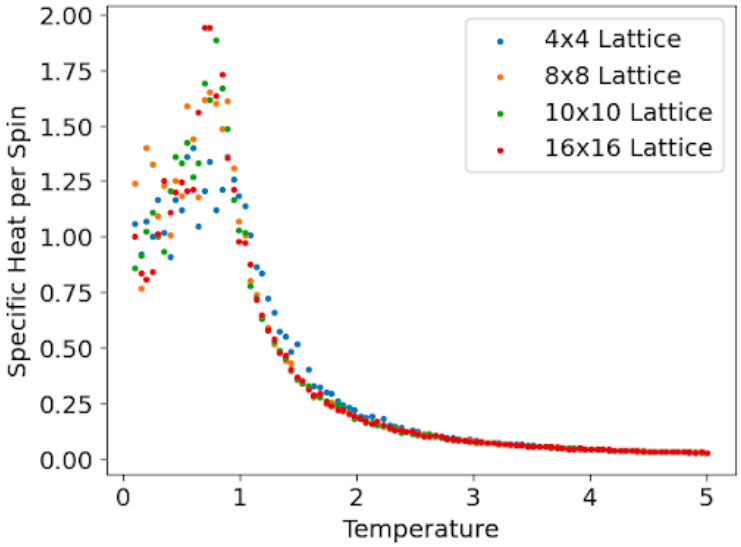

Figure 21: Specific heat against Temperature for multiple Lattice sizes, clearly showing an increase in the peak as the lattice size increases

Curiously, both the energy of the system seems to behave much like an order parameter, being zero in one 'phase' and non-zero in another. Moreover, the specific heat of the system also peaks at some non-zero temperature. By obtaining the graphs of specific heat against temperature for larger and larger lattice sizes, we were also able to see that this was a peak that was not disappearing, and is thus not an anomaly in the smaller lattices. Treating this temperature (0.73) then as some sort of critical temperature for the system, we used the single histogram method to get data for the energy at this 'critical' temperature, then used it to calculate the free energy.

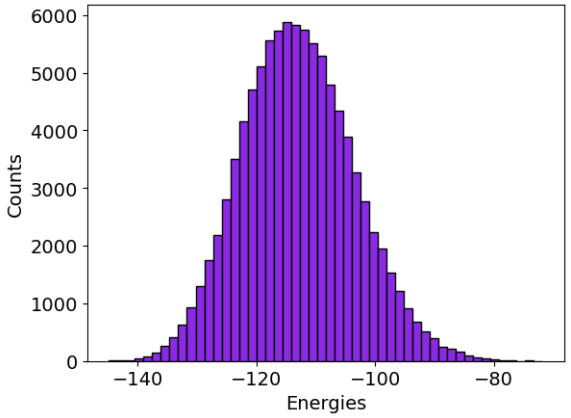

Figure 22: Histogram of the values of energy of the system at the 'critical temperature' $T_c = 0.73$

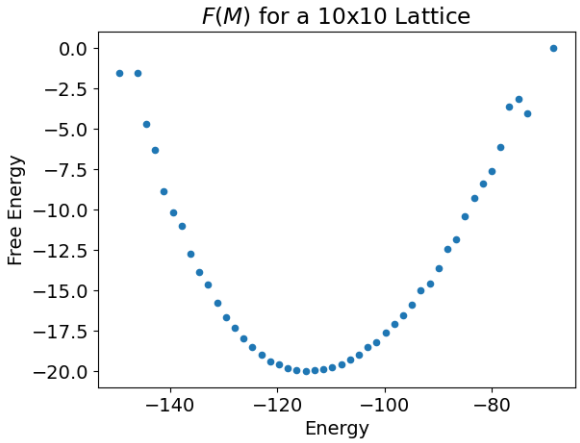

Figure 23: Free energy as a function of energy, showing only a single minimum

The presence of only a single minimum suggests that there may not be any sort of phase transition occurring in this system, which would mean the peak we saw in the specific heat was not an indicator of a phase transition at all. Just to be certain of this result though, we also plotted the free energy against energy for larger lattices and for greater number of Monte Carlo sweeps as well, and here too we observed only a single minimum. Thus, we can conclude that the 2D Heisenberg Model shows no phase transition.

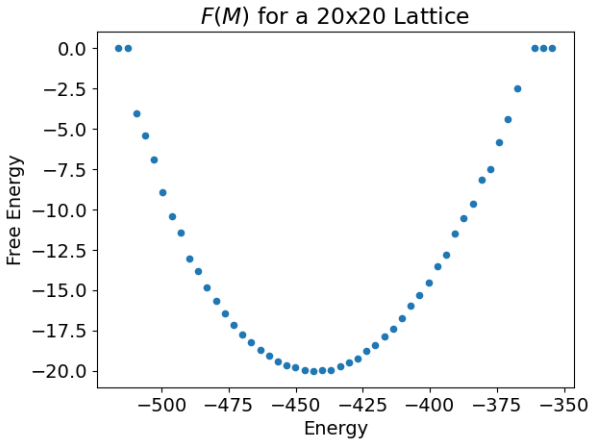

Figure 24: Free energy as a function of energy for a 20x20 Lattice, showing only a single minimum

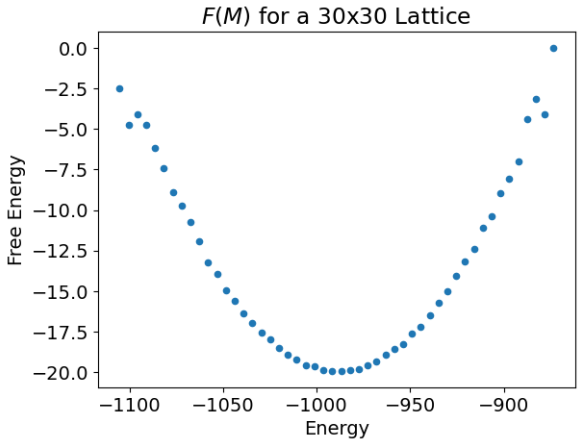

Figure 25: Free energy as a function of energy for a 30x30 Lattice, showing only a single minimum

5 Conclusion

In our project, we tested the Lee-Kosterlitz analysis for three different models and obtained conclusive results that helped us characterize the phase transition for these models. We started by testing the method with the q-State Potts Model for q values greater than or equal to the critical value of 4. We found that this system showed a first-order phase transition, and our results were in agreement with the existing analytical and computational results. We then moved on to test the method on the 2D Ising Model, which exhibits a continuous or second-order phase transition. Once again, we were able to determine the order of the phase transition successfully. Both these systems have discrete symmetry and show phase transition (these models were also analyzed by Lee and Kosterlitz in their original paper). However, in our final model, the 2D Classical Heisenberg Model, which possesses continuous symmetry, we found no evidence of any phase transition. We couldn't use the analysis of free energy because we didn't see any double minima, even when the system was run closer to the critical temperature. This lack of a double minimum can serve as standing proof of the absence of any phase transition in the system.

Although, this is not an irrefutable evidence to support our claim, it does agree with the current theoretical explanation given by Mermim-Wagner that states that a 2D Classical Heisenberg model will show no phase transition. This method of Lee-Kosterlitz analysis is a powerful tool for characterizing the phase transition of various systems. It can also be used for systems about which we might not have any knowledge initially, and still, it can yield conclusive and reliable results about the system's behavior near the critical point.

6 Challenges

- Firstly, we were unable to determine the appropriate method to calculate the magnetization for the classical Heisenberg model. Although we calculated it individually for the x, y, and z directions, we couldn't be certain about the correctness of our method.
- For the q-Potts model, it was computationally difficult to obtain good results for lattices smaller than 50×50 for the free energy plot. This made it challenging to perform the simulation for multiple q values.
- We also tried to implement the multiple histogram method with Lee-Kosterlitz Analysis, but it was slightly difficult for us to execute. We expected to observe a double minimum with different coordinate magnetizations (x, y, and z), but we weren't able to obtain it. We are unsure if this is something we would observe for our 2D Classical Heisenberg Model.
- Furthermore, the Multiple Histogram method also failed to give accurate results for the 2D Ising Model, although it worked well for the Lee-Kosterlitz Analysis. The behavior observed was not that accurate from the multiple histogram method.

7 Acknowledgements

We would like to thank Professor Bikram Phookun and the Graduate Assistant Philip for giving us the opportunity to work on this wonderful project. We would also like to thank Professor Somendra Bhattacharjee and Philip for the invaluable comments and help they gave us when we were working on this project. Finally, we would like to thank our batchmates for lending their support and help to us.

8 References

1. Lee, J., & Kosterlitz, J. (1990). New numerical method to study phase transitions. Physical review letters, 65(2), 137.
2. Soane, S., & Galloway, C. Model.Finite-Size Lattice Effects in Potts
3. Naya, S., & Nambu, S. (1977). Order-Parameter Expansion of Free Energy and Its Application to Some Models with First and Second Order Phase Transitions. Progress of Theoretical Physics, 58(6), 1679-1691.
4. Ferrenberg, A. M., & Landau, D. P. (1991). Critical behavior of the three-dimensional Ising model: A high-resolution Monte Carlo study. Physical Review B, 44(10), 5081.
5. Statistical and Thermal Physics: With Computer Applications, Second Edition by Tobochnik & Gould
6. Machine Learning and Phase Transitions-John Martyn
7. Heffern, E. F., Huelskamp, H., Bahar, S., & Inglis, R. F. (2021). Phase transitions in biology: from bird flocks to population dynamics. Proceedings of the Royal Society B, 288(1961), 20211111.
8. Lectures on Landau Theory of Phase Transitions, Department of Physics, Georgetown University Peter D Olmsted

9 Appendix : Python Code

9.1 Python Libraries Imported

Listing 1: Imported Libraries.

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.signal import find_peaks
from numba import njit

plt.rcParams['font.size'] = 14
```

9.2 Determining Energy for the Three Models

Listing 2: Obtaining the Energy for a 2D Ising Lattice.

```
@njit
def energyI(box, h=0, J=1):
    '''
    Returns energy of a 2D Ising lattice of spins, where the energy per
        bond is the product of the spins making the bond

    PARAMETERS:
    -----------
     - box: 2D (shape (x, y)) lattice of spins
```

```
     - h: External magnetic field; set by default to 0
     - J: Interaction energy; set by default to 1

    RETURNS:
    --------
     - e: Energy of the lattice, a number
    '''

    x,y = box.shape
    eJ = 0 # Initialising interaction energy
    eH = 0 # Initialising external magnetic field energy
    for i in range(x):
        for j in range(y):
            sides = box[i-1,j] + box[i,j-1] # By the loop over every
                particle, this loops over every bond
            eJ -= box[i,j]*sides
            eH -= box[i,j]
    eJ *= J
    eH *= h
    e = eJ + eH

    return e
```

Listing 3: Obtaining the Energy for a 2D q-Potts Lattice.

```
@njit
def kronecker(spin1, spin2):
    '''
    Returns energy of a pair of spins on a q-Potts lattice

    PARAMETERS:
    -----------
     - spin1: Spin 1 of the pair, an integer between 0 and q-1
     - spin2: Spin 2 of the pair, an integer between 0 and q-1

    RETURNS:
    --------
     - Energy of the pair: 1 if the spins are the same, and 0 if they are
         not
    '''

    if int(spin1)==int(spin2):
        return 1
    else:
        return 0

@njit
def eng(spins,i,j,L,J=1):
    '''
    Returns energy of a spin on a q-Potts lattice

    PARAMETERS:
    -----------
     - spins: 2D (shape (x, y)) lattice of spins
     - i: x-coordinate of spin whose energy is to be determined
     - j: y-coordinate of spin whose energy is to be determined
     - L: Length of the lattice
     - J: Interaction energy; set by default to 1

    RETURNS:
    --------
```

```
     - Energy of the spin: a number
    '''

    energy=(kronecker(spins[i][j],spins[(i+1)%L][j])+kronecker(spins[i][j
       ],spins[i][(j+1)%L])+kronecker(spins[i][j],spins[i][j-1])+kronecker
       (spins[i][j],spins[i-1][j]))

    return -J*energy
```

Listing 4: Obtaining the Energy for a 2D Classical Heisenberg Lattice.

```
@njit
def energyH(box, hz=0, J=1):
    '''
    Returns energy of a 2D Classical Heisenberg lattice of spins, where
       the energy per bond is the dot product of the spins making the bond

    PARAMETERS:
    -----------
     - box: 3D (shape (x, y, 3)) lattice of spins [2D Lattice, third
        dimension for components of vector spins]
     - h: External magnetic field in the z-direction; set by default to 0
     - J: Interaction energy; set by default to 1

    RETURNS:
    --------
     - e: Energy of the lattice, a number
    '''

    x,y = box.shape[:2]
    eJ = 0 # Initialising interaction energy
    eH = 0 # Initialising external magnetic field energy
    for i in range(x):
        for j in range(y):
            sides = box[i-1,j] + box[i,j-1] # By the loop over every
               particle, this loops over every bond
            eJ -= np.sum(box[i,j]*sides)
            eH -= box[i,j,2]
    eJ *= J
    eH *= hz
    e = eJ + eH

    return e
```

9.3 Determining Magnetisation for the Three Models

Listing 5: Obtaining the Magnetisation for a 2D Ising Lattice.

```
@njit
def magnetI(box):
    '''
    Returns magnetisation of a 2D Ising lattice of spins.

    PARAMETERS:
    -----------
     - box: 2D (shape (x, y)) lattice of spins

    RETURNS:
    --------
     - m: The sum of all spins of the lattice, a number
```

```
    '''

    x,y = box.shape[:2]
    m = 0 # Initialising sum of all spins
    for i in range(x):
        for j in range(y):
            m += box[i,j]

    return m
```

Listing 6: Obtaining the Magnetisation for a 2D q-Potts Lattice.

```
@njit
def magnetQ(box):
    '''
    Returns magnetisation of a 2D q-Potts lattice of spins.

    PARAMETERS:
    -----------
     - box: 3D (shape (x, y, 2)) lattice of spins

    RETURNS:
    --------
     - m: The sum of all spins of the lattice, a 1D array with the 2
        elements
    '''

    x,y = box.shape[:2]
    m = np.array([0.,0.]) # Initialising sum of all spins
    for i in range(x):
        for j in range(y):
            m += box[i,j]

    return m
```

Listing 7: Obtaining the Magnetisation for a 2D Classical Heisenberg Lattice.

```
@njit
def magnetH(box):
    '''
    Returns magnetisation of a 2D Classical Heisenberg lattice of spins.

    PARAMETERS:
    -----------
     - box: 3D (shape (x, y, 3)) lattice of spins

    RETURNS:
    --------
     - m: The sum of all spins of the lattice, a 1D array with 3 elements
    '''

    x,y = box.shape[:2]
    m = np.array([0.,0.,0.]) # Initialising sum of all spins
    for i in range(x):
        for j in range(y):
            m += box[i,j]

    return m
```

9.4 Monte Carlo Sweeps for the Three Models

Listing 8: One Monte Carlo sweep for a 2D Ising Lattice.

```
@njit
def montecarloI(box, T, h=0, J=1):
    '''
    Performs one Monte Carlo sweep at a given temperature over a 2D Ising
        lattice of spins.

    PARAMETERS:
    -----------
     - box: 2D (shape (x, y)) lattice of spins
     - T: Temperature at which Monte Carlo sweep is performed
     - h: External magnetic field; set by default to 0
     - J: Interaction energy; set by default to 1

    RETURNS:
    --------
     - box: The lattice after one Monte Carlo sweep has been performed, a
         3D array of the same shape as the input box
     - en: The energy of the new lattice, a number
     - mag: The sum of all spins of the new lattice, a number
    '''

    x,y = box.shape
    en = energyI(box, h=h, J=J)
    mag = magnetI(box)

    for p in range(x*y):
        i = int(x*np.random.uniform(0,1)) # Choosing a random x-coordinate
            in the lattice
        j = int(y*np.random.uniform(0,1)) # Choosing a random y-coordinate
            in the lattice

        nlatij = -box[i,j] # Trial is flip of random spin

        sides = box[(i+1)%x,j] + box[(i-1)%x,j] + box[i,(j+1)%y] + box[i,(
            j-1)%y] # Sum
        dE = -J*(nlatij - box[i,j])*sides - h*(nlatij - box[i,j]) # Change
            in energy from trial flip/rotation

        if dE <= 0: # Trials lowering energies are always accepted
            en += dE
            mag += -box[i,j] + nlatij
            box[i,j] = nlatij
        else:
            r = np.random.uniform(0,1)
            if r <= np.exp(-dE/T): # Trials increasing energy are accepted
                with a probability from the Boltzmann distribution
                en += dE
                mag += -box[i,j] + nlatij
                box[i,j] = nlatij

    return box, en, mag
```

Listing 9: One Monte Carlo sweep for a 2D q-Potts Lattice.

```
@njit
def oneMCS(energy, beta, spins,q, J=1):
    '''
    Performs one Monte Carlo sweep at a given temperature over a 2D q-
        Potts lattice of spins.

```

```
    PARAMETERS:
    -----------
         energy: Energy of the lattice of spins
         beta: Inverse temperature at which Monte Carlo sweep is performed
         spins: 2D (shape (x, y)) lattice of spins
         q: Number of states for q-Potts model
         J: Interaction energy; set by default to 1

    RETURNS:
    --------
         energy: The energy of the new lattice, a number
         spins: The lattice after one Monte Carlo sweep has been performed
        , a 2D array of the same shape as the input spins
    '''

    L = np.shape(spins)[0]                # Linear dimension of our square
        lattice
    N = L**2  # Number of spins
    #energy=0
    for nothing in range(N): #running a loop over all the spins


        i=int(L*np.random.uniform(0,1))# choosing a coordinate in x
        j=int(L*np.random.uniform(0,1)) #choosing a coordinate in y
        ns=int(q*np.random.uniform(0,1))
        #print(ns)
        temp=spins[i,j]
        E_i=eng(spins,i,j,L)
        #dE=2*J*spins[i,j]*(spins[(i+1)%L,j]+spins[(i-1)%L,j]+spins[i,(j
           +1)%L]+spins[i,(j-1)%L])+2*h*spin[i,j]
        # using boundary condition we can calculate the change in energy
           if the spin is flipped, the factor of 2 comes from subtracting
           final energy from initial
        spins[i,j]=ns
        E_f=eng(spins,i,j,L)

        dE=E_f-E_i

        e=np.exp(-dE*beta) #defining the probability of acceptance
        r=np.random.uniform(0,1)# calling a random uniform number
        #check if dE is less than equal to 0 or not, if yes then increment
            the change and flip the spins
        if dE<=0 or r<e:
            energy += dE
            spins[i,j]=ns

        else:
            spins[i,j]=temp

    return energy, spins #return energy and spin configuration after
       oneMCS
```

Listing 10: One Monte Carlo sweep for a 2D Classical Heisenberg Lattice.

```
@njit
def montecarloH(box, T, q=8, hz=0, J=1):
    '''
    Performs one Monte Carlo sweep at a given temperature over a 2D
       Classical Heisenberg lattice of spins.

    PARAMETERS:
```

```
    -----------
     - box: 3D (shape (x, y, 3)) lattice of spins; p is 1 for an Ising
        lattice, 2 for q-Potts, and 3 for Heisenberg
     - T: Temperature at which Monte Carlo sweep is performed
     - hz: External magnetic field in the z-direction; set by default to 0
     - J: Interaction energy; set by default to 1

    RETURNS:
    --------
     - box: The lattice after one Monte Carlo sweep has been performed, a
        3D array of the same shape as the input box
     - en: The energy of the new lattice, a number
     - mag: The sum of all spins of the new lattice, a 1D array with 3
        elements
    '''

    x,y = box.shape[:2]
    en = energyH(box, hz=hz, J=J)
    mag = magnetH(box)

    for p in range(x*y):
        i = int(x*np.random.uniform(0,1)) # Choosing a random x-coordinate
            in the lattice
        j = int(y*np.random.uniform(0,1)) # Choosing a random y-coordinate
            in the lattice

        ntheta = np.arccos(2*np.random.uniform(0,1) - 1) # Choose random
           altitude angle
        nphi = 2*np.pi*np.random.uniform(0,1) # Choose random azimuthal
           angle
        nlatij = np.array([np.sin(ntheta)*np.cos(nphi), np.sin(ntheta)*np.
           sin(nphi), np.cos(ntheta)]) # Trial is 3D unit vector with
           angular coordinate chosen above

        sides = box[(i+1)%x,j] + box[(i-1)%x,j] + box[i,(j+1)%y] + box[i,(
           j-1)%y] # Sum
        dE = -J*np.sum((nlatij - box[i,j])*sides)  - hz*(nlatij[-1] - box[
           i,j,-1]) # Change in energy from trial flip/rotation

        if dE <= 0: # Trials lowering energies are always accepted
            en += dE
            mag += -box[i,j] + nlatij
            box[i,j] = nlatij
        else:
            r = np.random.uniform(0,1)
            if r <= np.exp(-dE/T): # Trials increasing energy are accepted
                with a probability from the Boltzmann distribution
                en += dE
                mag += -box[i,j] + nlatij
                box[i,j] = nlatij

    return box, en, mag
```

9.5 Obtaining Data for the Three Models at the Critical Temperature

Listing 11: Obtaining Critical Temperature Data for a 2D Ising Lattice.

```
@njit
def crit_temp_dataI(p, Tc, J=1, h=0, mcs=100_000):
    '''
```

```
    Returns energies and magnetisations of a 2D lattice of spins at
        equilibrium at its critical temperature, over many Monte Carlo
        sweeps.

    PARAMETERS:
    -----------
     - p: Number of spins along one dimension of the 2D lattice
     - Tc: Critical temperature at which Monte Carlo sweep is performed
     - h: External magnetic field (in the z-direction for Heisenberg Model
         ); set by default to 0
     - J: Interaction energy; set by default to 1
     - mcs: Number of Monte Carlo sweeps to be run; set by default to 100
         _000

    RETURNS:
    --------
     - e: The equilibrium energies of the lattice, a 1D array with mcs
         elements
     - m: The equilibrium sums of all spins of the lattice, a 1D array
         with mcs elements
    '''

    box = np.ones((p, p)) # Create lattice for Monte Carlo sweeps to be
        performed on
    e = np.zeros(mcs) # Initialise array of energies
    m = np.zeros(mcs) # Initialise array of sums of spins
    for i in range(mcs):
        box, en, mag = montecarloI(box, T=Tc, h=h, J=J)
        e[i] = en
        m[i] = mag

    e = e[3000:] # Equilibrium taken to be
    m = m[3000:] # achieved after 3000 MC sweeps

    return e, m
```

Listing 12: Obtaining Critical Temperature Data for a 2D q-Potts Lattice.

```
@njit
def simulate(q,L, beta,J=1, n_mcs=10_000):
    '''
    Returns energies and magnetisations of a 2D q-Potts lattice of spins
        at equilibrium at a temperature, over many Monte Carlo sweeps.

    PARAMETERS:
    -----------
     - q: Number of states for q-Potts model
     - L: Number of spins along one dimension of the 2D lattice
     - beta: Temperature at which Monte Carlo sweep is performed
     - J: Interaction energy; set by default to 1
     - mcs: Number of Monte Carlo sweeps to be run; set by default to 10
         _000

    RETURNS:
    --------
     - systemEnergy: The equilibrium energies of the lattice, a 1D array
         with mcs elements
     - m: The equilibrium sums of all spins of the lattice, a 2D array of
         shape (mcs, 2)
     - spins: The spins of the lattice after the Monte Carlo sweeps have
         been run, a 2D array of shape (L, L)
```

```
    '''

    N=L*L
    val=np.arange(1,q+1,1)
    systemEnergy = np.zeros(n_mcs)
    m=np.zeros_like(systemEnergy)
    spins = np.random.choice(val,(L,L))
    E=0.0
    for i in range(L):
        for j in range(L):
            E+=eng(spins,i,j,L,J=1)
    E=E/2


    #m = np.zeros_like(systemEnergy)
    systemEnergy[0]= E#,np.sum(spins) #From the hamiltonian
    m[0]=np.abs(np.sum(spins))

    for i in (range(1,n_mcs)): #run over n_mcs-1
        energy,spins=oneMCS(systemEnergy[i-1],beta,spins,q)# input
            systemEnergy[0]
        systemEnergy[i]=energy #m[mc] = np.sum(spins)
        m[i]=np.abs(np.sum(spins))

    return systemEnergy,m,spins
```

Listing 13: Obtaining Critical Temperature Data for a 2D Classical Heisenberg Lattice.

```
@njit
def crit_temp_dataH(p, Tc, J=1, hz=0, mcs=100_000):
    '''
    Returns energies and magnetisations of a 2D lattice of spins at
        equilibrium at its critical temperature, over many Monte Carlo
        sweeps.

    PARAMETERS:
    -----------
     - p: Number of spins along one dimension of the 2D lattice
     - Tc: Critical temperature at which Monte Carlo sweep is performed
     - hz: External magnetic field in the z-direction; set by default to 0
     - J: Interaction energy; set by default to 1
     - mcs: Number of Monte Carlo sweeps to be run; set by default to 100
         _000

    RETURNS:
    --------
     - e: The equilibrium energies of the lattice, a 1D array with mcs
         elements
     - m: The equilibrium sums of all spins of the lattice, a 2D array of
         shape (mcs, 3)
    '''

    box = np.zeros((p, p, 3)) # Create lattice for Monte Carlo sweeps to
        be performed on
    box[:,:,0] = 1
    e = np.zeros(mcs) # Initialise array of energies
    m = np.zeros((mcs, 3)) # Initialise array of sums of spins
    for i in range(mcs):
        box, en, mag = montecarloH(box, T=Tc, hz=hz, J=J)
        e[i] = en
        m[i] = mag
```

```
28
29      e = e[3000:] # Equilibrium taken to be
30      m = m[3000:] # achieved after 3000 MC sweeps
31
32      return e, m
```

9.6 Lee-Kosterlitz Algorithm

Listing 14: Performing Lee-Kosterlotz Analysis.

```
1  @njit
2  def lee_kost(o, t, Tc, bins=50):
3      '''
4      Creates a histogram of the values of the order parameter of a 2D
           lattice at equilibrium, then determines the free energy as a
           function of this order parameter.
5
6      PARAMETERS:
7      -----------
8       - o: 1D array of values of the order parameter of a lattice
9       - t: Temperature at which to perform Lee-Kosterlitz analysis to
            calculate free energy
10      - Tc: Critical temperature of the system the lattice represents
11      - bins: Number of bins to make for the histogram; set by default to
            50
12
13     RETURNS:
14     --------
15      - os: The values of the order parameter at which the free energy has
            been determined, a 1D array with a number of elements equal to the
             number of bins of the order parameter with a non-zero count
16      - fs: The values of the free energy corresponding to the above values
             of the order parameter, a 1D array with the same number of
            elements as os
17     '''
18
19     c, b = np.histogram(o, bins=bins)
20     os = (b[1:] + b[:-1])*0.5 # The midpoint of each bin is the value of
           the order parameter for that bin
21
22     delt = np.array([0]) # Initialise array of c and os elements to be
           deleted
23     for i in range(len(os)):
24         if c[i] == 0:
25             delt = np.append(delt, i) # Note which bins in the histogram
                   have counts of zero
26     if len(delt > 1):
27         c = np.delete(c, delt[1:]) # Delete zero-valued elements of c
28         os = np.delete(os, delt[1:]) # Delete corresponding elements of os
29
30     fs = np.zeros_like(os) # Initialise array of free energies
31     for i in range(len(os)):
32         fs[i] = -t*np.log(c[i]) + t*(os[i]*(1/t - 1/Tc))
33         pass
34
35     return os, fs
```

Listing 15: Performing Lee-Kosterlotz Analysis for a q-Potts Model.

```
1  def lee_kostQ(e,b,bc,bins=50):
```

```
    '''
    Creates a histogram of the values of the order parameter of a 2D q-
       Potts lattice at equilibrium, then determines the free energy as a
       function of this order parameter.

    PARAMETERS:
    -----------
     - e: 1D array of values of the energies of the lattice
     - b: Inverse temperature at which to perform Lee-Kosterlitz analysis
        to calculate free energy
     - bc: Inverse of critical temperature of the system the lattice
        represents
     - bins: Number of bins to make for the histogram; set by default to
        50

    RETURNS:
    --------
     - es: The values of the energy at which the free energy has been
        determined, a 1D array with a number of elements equal to the
        number of bins of the order parameter with a non-zero count
     - fs: The values of the free energy corresponding to the above values
         of the order parameter, a 1D array with the same number of
        elements as os
    '''

    cutoff=len(e)//2

    count,bin_E=np.histogram(e[cutoff:],bins=bins)
    energy=np.zeros(len(bin_E))

    for i in range(len(bin_E)-1):
        energy[i]=(bin_E[i]+bin_E[i+1])/2

    energy=energy[:-1]

    Hist_count=[]
    Ef=[]

    for i in range(len(count)):
        if count[i]!=0:
            Hist_count.append(count[i])
            Ef.append(energy[i])

    Hist_count=np.array(Hist_count)
    Ef=np.array(Ef)

    F=-B0*np.log(Hist_count)+(b-bc)*Ef

    return es, fs
```

9.7 Finding the distance between the Central Maximum and Double Minima

Listing 16: Determining ΔF between the Maximum and Minima.

```
def find_dF(os, fs):
    '''
    Determines the distance between the minima and maximum in the free
       energy v/s order parameter graph.

    PARAMETERS:
```

```
    -----------
     - os: The values of the order parameter at which the free energy has
         been determined
     - fs: The values of the free energy corresponding to the above values
          of the order parameter

    RETURNS:
    --------
     - dF: the distance between the central maximum and the double minima
         in the fs v/s os graph
     - ymax: the Y-coordinate of the central maximum
     - ymin: the mean of the Y-coordinates of the two minima
     - x: the list of x values covered by os
     - y: the y values from the best fit corresponding to x

    '''

    coeffs = np.array(np.polyfit(os,fs,10))
    x = np.linspace(min(os), max(os), 100)
    y = np.zeros_like(x)
    for i in range(len(coeffs)):
        y += coeffs[i]*(x**(10-i))

    ymax_arg = find_peaks(y)[0][0]
    ymin_arg1 = find_peaks(-y)[0][0]
    ymin_arg2 = find_peaks(-y)[0][1]

    ymax = y[ymax_arg]
    ymin = (y[ymin_arg1] + y[ymin_arg2])*0.5

    dF = ymax - ymin

    return dF, ymax, ymin, x, y
```