Ferromagnetic N-state Potts model in 2D studied using Metropolis algorithm

Aristo Kevin Ardyaneira P

(Dated: July 13, 2023)

This project aims to study the critical behavior of 2D Potts model in square lattice numerically by means of Metropolis algorithm. The critical temperature T_{crit} and critical exponents are extracted through finite size scaling analysis and they are compared with the ones in Ising model.

I. INTRODUCTION

As simple as it may seem, only in a few very special cases are many equations in statistical mechanics exactly soluble. The solutions of various equations in that field are of great interest for physicists, as sole understandings of microscopic interactions between particles do not suffice to describe macroscopic properties we often observe in daily lives. Hence, more often than not physicist are compelled to use numerical approach to gain insight in the physics of many-body system.

An example of simple equations of the above kind is a classical Ising model which describes the magnetization of interacting particles in fixed lattice sites. In this model interacting particles are described by the Hamiltonian

$$H = -J \sum_{<ij>} s_i s_j - h \sum_i s_i \tag{1}$$

with interaction term J, external field strength h, and spin value $s_i \in \{-1, 1\}$. The interaction term J can take either non-zero positive value or non-zero negative value. The former corresponds to ferromagnetic Ising model, while the latter depicts antiferromagnetic Ising model. It turns out that for dimensions larger or equal two and $h = 0$, this model predicts the existence of phase transition, characterized with critical temperature T_{crit}, below which (anti-)ferromagnetism emerges and above which paramagnetism exists[1].

Despite of its success, this model is still far from reality in describing magnetism. This is because the spin value s_i may only take a value of either 1 or -1, which implies only two possible directions of magnetization, namely along either positive $z-$axis or negative $z-$axis. In reality, however, a much higher degree of freedom in magnetization is observed at macroscopic level. This motivates a generalization of Ising model to Potts model, which permits more than two discrete spin values yet does not complicate the former model significantly. This brief project deals with the numerical study of this model and discusses its critical behavior.

II. THEORY

A. Potts model

Developed for the first time in 1951 by Renfrey Potts in his Ph.D. thesis[2], his model branches into two distinct yet similar models, namely vector Potts model and standard Potts model. This brief project deals with the latter with vanishing external field ($h = 0$), whose Hamiltonian reads

$$H = -J \sum_{<ij>} \delta\big(s_i, s_j\big). \tag{2}$$

Just like the Ising model, the term J represents the interaction term between spin values of neighboring lattice sites. A positive value of J induces ferromagnetism below certain critical temperature T_{crit}, while a negative value of J yields antiferromagnetism. The term $\delta(s_i, s_j)$ is Dirac delta function and takes a value 1 iff $s_i = s_j$ and a value 0 iff $s_i \neq s_j$. Unlike Ising model, the spin value $s_i \in \{0, 1, 2, ..., \mathrm{N}-1\}$ may take more than two different discrete values, resulting in a higher degree of freedom in magnetization direction.

The total magnetization of a system in Potts model can be calculated using

$$m = \frac{1}{N_{\text{site}}} \sum_k \mathrm{e}^{i\frac{2\pi}{N}\cdot s_k}, \tag{3}$$

where N_{site} denotes the total number of sites, while N denotes the number of possible spin values each site may take. In general the magnetization m takes a complex value $m = |m|\mathrm{e}^{i\,\theta_{\mathrm{m}}}$ with the cutting angle θ_{m} between m and real axis of complex plane. This cutting angle θ_{m} represents the orientation of magnetization, while $|m|$ provides the magnetization strength.

In contrast to Ising model, ferromagnetic Potts model exhibits a phase transition already for the dimension $N \geq 1$[3], for which reason this model is useful for the study of phase transitions. The critical point is located at $J/T_{\text{crit}} = \log\big(1 + \sqrt{N}\big)$, whilst the phase transition is of second order for $1 \leq N \leq 4$ and of first order for $N > 4$[4,5]. Phase transition of first order is characterized with a discontinuity in the first derivative of a thermodynamic potential at the critical point, while the discontinuity of a thermodynamic potential appears in second derivative for second order phase transition. In addition, a phase transition is recognizable by the presence of singularity in specific heat and magnetic susceptibility vs temperature. It is also to be noted that phase transition appears in the thermodynamic limit, that is for $N_{\text{site}} \to \infty$. In a machine it is, however, impossible to simulate Potts model in a thermodynamic limit, for which reason no discontinuity is observed at the critical point. Hence, one needs to do finite size scaling analysis to extrapolate critical temperature and critical exponents.

B. Metropolis algorithm

Given a classical Hamiltonian in statistical mechanics, it is of interests for physicists to know the partition function

$$Z = \sum_i \mathrm{e}^{-\beta E_i} \tag{4}$$

of the system, as many statistical properties in thermodynamic equilibrium can be derived from it. Thus, this function connects the microscopic description of interacting particles with their macroscopic behaviors. However, it is unfortunate that in many cases of Hamiltonian, i.e. Ising and Potts model in high dimension, the partition function cannot be calculated analytically. Therefore, direct sampling from probability distribution corresponding to the Hamiltonian is in many cases challenging. To overcome this difficulty, a Markov chain Monte Carlo method called Metropolis algorithm was developed in 1953[6]. The aim of this algorithm is to generate a Markov chain, a sequence of random variables, from a probability distribution, whose direct calculation is difficult. Applied on Potts model, this algorithm works as follows:

Algorithm 1 Metropolis algorithm on Potts model

```
Require: Initial spin configuration on $N_{\text{site}}$ lattice sites, $N$ possible spin values, physical quantity $Q$ to be measured, and fixed number of sweeps $N_{\text{sweeps}}$
  $i \leftarrow 0$
  while $i < N_{\text{sweeps}}$ do
      $n \leftarrow \text{random}(\{0, 1, 2, ..., N_{\text{site}}\})$
      $\tilde{s}_n \leftarrow \text{random}(\{0, 1, 2, ..., N-1\} \setminus \{s_n\})$
      $\Delta E \leftarrow E(\tilde{s}_n) - E(s_n)$        ▷ Measure energy difference after flipping $s_n$
      $s_n \leftarrow \tilde{s}_n$ with probability $\min(1, \mathrm{e}^{-\beta \Delta E})$
      $Q \leftarrow$ measure $Q$
      $i \leftarrow i + 1$
  end while
```

It is to be noted that the number of sweeps N_{sweeps} needs to be sufficiently large in comparison to the number of lattice sites N_{site}. Otherwise, sampled Qs are not close to the actual Q in thermodynamic equilibrium. In practice, this is done by letting algorithm 1 run for a certain number of thermalization N_{eq} without making any measurement, before measurement begins. To generate uncertainties of measured Q, algorithm 1 typically runs with N_{bins} repetitions.

III. RESULT AND DISCUSSION

A. Spin configuration

Before the critical behavior in the magnetization and in the specific heat of the system is studied, it is interesting to know how spin configuration of the system at various temperatures looks like under algorithm 1. For that, the parameters $N_{\text{sweeps}} = 500 \cdot N_{\text{site}} = 500 \cdot L^2$, $N_{\text{eq}} = N_{\text{sweeps}}/5$, $N = 4$ are set with $L = 16$. Under this parameters algorithm 1 yields Fig. 1.

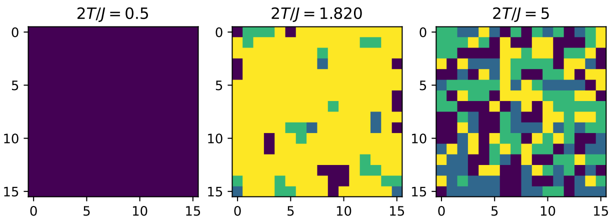

FIG. 1. Spin configuration of 4-state standard Potts model at $T < T_{\text{crit}}$ (left), $T = T_{\text{crit}}$ (middle), $T > T_{\text{crit}}$ (right). $T_{\text{crit}} = 1/\log\left(1 + \sqrt{4}\right) \approx 0.910$

It can be seen on Fig. 1 that each spin at each site takes only one spin value for $T < T_{\text{crit}}$ and begins to undergo phase transition at $T = T_{\text{crit}}$ by flipping some of the spins, thus starts to have $\mathbb{Z}^2$ symmetry. At $T > T_{\text{crit}}$ the system takes random spin configuration that no preferred direction of magnetization is observed. Therefore, it is inferred that algorithm 1 together with parameters $N_{\text{sweeps}} = 500 \cdot L^2$ and $N_{\text{eq}} = N_{\text{sweeps}}/5$ works well enough to determine the proper spin configuration at various temperatures. These parameters are used subsequently to study the critical behavior of the system.

B. Critical behavior

One noticeable sign of phase transition is a singularity in specific heat c_V vs temperature T for $N_{\text{site}} \to \infty$. In a finite system, however, this singularity is recognized by the increase of maximum c_V with lattice sizes, which for instance can be seen in Fig. 2 Moreover phase transition is marked by inflection point of absolute magnetization $|m|$ vs temperature T in a finite system. This is demonstrated in Fig. 3.

An inflection point has a property of having a highest absolute gradient. As magnetic susceptibility χ grows proportional to the variance of absolute magnetization, a peak of χ is also observed at this inflection point. Therefore a temperature T_{max}, for which c_V and χ take their maximum values can be extracted and plotted against

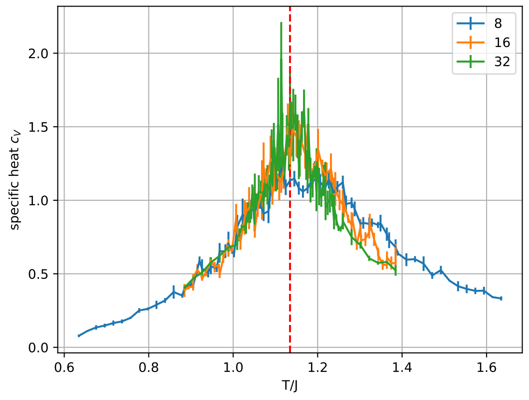

FIG. 2. Specific heat c_V vs temperature for 2-state Potts model with various number of lattices $N_{\text{site}} \in \{8^2, 16^2, 32^2\}$. Vertical red dashed line marks theoretical critical temperature $T_{\text{crit}}/J = 1/\log\left(1+\sqrt{2}\right) \approx 1.13$.

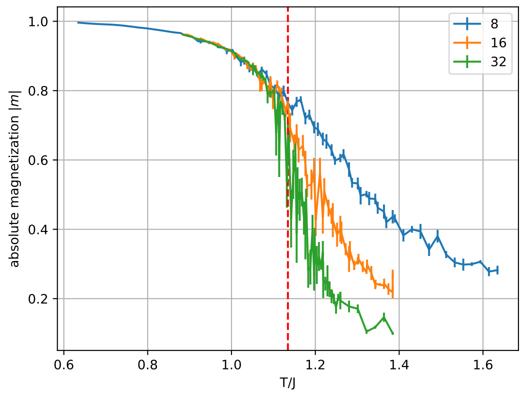

FIG. 3. Absolute magnetization $|m|$ vs T for 2-state Potts model with various number of lattices $N_{\text{site}} \in \{8^2, 16^2, 32^2\}$. Vertical red dashed line marks theoretical critical temperature $T_{\text{crit}}/J = 1/\log\left(1+\sqrt{2}\right) \approx 1.13$.

L^{-1}. The data points are then fitted with a linear function

$$T_{\max}(L^{-1}) = m \cdot L^{-1} + T_{\text{crit}}$$

to obtain the intersection between $T_{\max}(L^{-1})$ and y-axis, which yields T_{crit} (see Fig. 4 as an example). This approach is done on 2-, 4-, and 6-state Potts model and results in Tab. I.

Another way to obtain critical temperature T_{crit} is by

N	T_{crit} from $c_V^{\max}$	T_{crit} from $\chi^{\max}$	$T_{\text{crit}}^{\text{theo}}$	deviation (%)
2	1.1216	1.1447	1.1346	0.89 − 1.15
4	0.8945	0.9536	0.9102	1.72 − 4.77
6	0.8023	0.8048	0.8076	0.35 − 0.67

TABLE I. Critical points extracted from peak of c_V and peak of χ using linear fit. In spite of high autocorrelation time around critical point, this approach produces T_{crit} with pretty low deviation from the theoretical one.

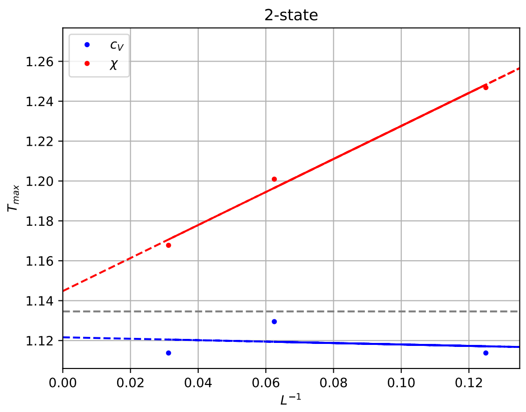

FIG. 4. $T_{\max}$ vs. L^{-1} ($J = 1$). Gray dashed horizontal line marks theoretical critical temperature $T_{\text{crit}}/J = 1/\log\left(1+\sqrt{2}\right) \approx 1.13$. Deviation of intersection between $T_{\max}(L^{-1})$ and y-axis from theoretical T_{crit} are attributed to the inaccuracies of determining $c_V^{\max}$ and $\chi^{\max}$, which come from high autocorrelation time of algorithm 1 around critical point.

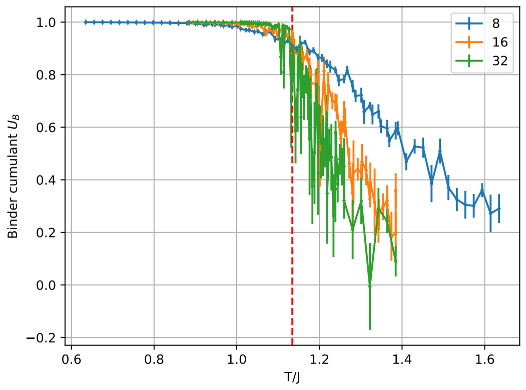

FIG. 5. Binder cumulant U_B vs temperature T of 2-state Potts model for various number N_{site} of lattice sites. Vertical red dashed line marks theoretical critical temperature $T_{\text{crit}}/J = 1/\log\left(1+\sqrt{2}\right) \approx 1.13$. Large fluctuations and error bars for $N_{\text{site}} = 32^2$ are attributed to high autocorrelation time of algorithm 1.

plotting a so-called Binder cumulant

$$U_B = \frac{3}{2}\left(1 - \frac{\langle s^4\rangle}{3\langle s^2\rangle^2}\right) \tag{5}$$

with order parameter $s = |m|$ vs T for various system sizes N_{site} (see for instance Fig. 5). Afterwards, one searches for a crossing point between U_B of various system sizes. At critical temperature T_{crit} all U_B take the same value, since around $T \approx T_{\text{crit}}$ Binder cumulant U_B behaves as $U(T,L) = b\left(\frac{T-T_{\text{crit}}}{T_{\text{crit}}} \cdot L^{1/\nu}\right) \approx b(0)$ with critical exponent ν[7]. Thus, it is independent of the number of lattice sites.

In order to find a crossing point between U_B of various system sizes, a linear function is fitted to data

N	$N_{\text{site},1} \times N_{\text{site},2}$	T_{crit}	deviation (%)
2	8, 16	1.1282	0.56 − 11.68
	8, 32	1.0906	
	16, 32	1.0021	
4	8, 16	0.9132	0.33 − 4.98
	8, 32	0.8963	
	16, 32	0.8649	
6	8, 16	0.8018	0.72 − 5.77
	8, 32	0.7804	
	16, 32	0.7610	

TABLE II. Critical points obtained from crossing points between U_B of various number N_{site} of lattice sites.

points of U_B vs T around $T \approx T_{\text{crit}}$. This approach yields Tab. II. One can observe from Tab. II that Binder cumulant approach is also able of extracting critical temperature T_{crit}, although for large numbers of lattice sites, i.e. $N_{\text{site}} = 32^2$, a considerable deviation from $T_{\text{crit}}^{\text{theo}}$ occurs. This happens due to statistically dependent measurement of U_B during the course of algorithm 1, which causes high fluctuation in U_B. A large number of lattice sites is especially susceptible to this issue, for which reason a greater number N_{sweeps} of sweeps should have been set instead. This comes unfortunately with a price of a more computationally expensive run, due to which a slightly smaller N_{sweeps} and N_{eq} were chosen.

In summary, T_{crit} is able to be successfully extrapolated using both c_V, χ approach and U_B approach, yielding a value, which deviates 11.68 % from $T_{\text{crit}}^{\text{theo}}$ at most.

Beside T_{crit}, one is also interested in critical exponents of N-state Potts model, as these exponents govern the relationship between several statistical properties and absolute relative temperatures $|t| = |T - T_{\text{crit}}|/T_{\text{crit}}$. Since χ and c_V are measured, the corresponding critical exponents γ and α are of interest. They can be gained by plotting $\chi^{\max}$ and $c_V^{\max}$ against L in logarithmic scale and by fitting exponential function

$$\chi^{\max} = m_\chi \cdot L^{\frac{\gamma}{\nu}}$$
$$c_V^{\max} = m_{c_V} \cdot L^{\frac{\alpha}{\nu}}$$

to data points (See Fig. 6 as an example). This results in γ/ν and α/ν presented in Tab. III. Therein, the critical exponent ν is extracted by plotting U_B against $(T - T_{\text{crit}})/T_{\text{crit}} \cdot L^{\frac{1}{\nu}}$ and varying ν until all U_{B} appear on a single line (see Fig. 7).

Overall, critical exponent α, γ, and ν can be extracted by fitting exponential function to c_V-L diagram and to χ-L diagram and by varying ν until all U_B appear on a single line. This approach, however, fails to deliver physically reasonable α, as insufficient number N_{sweeps} of sweeps causes inaccuracy of $c_V^{\max}$. A higher N_{sweeps} for algorithm 1 is therefore suggested to measure c_V.

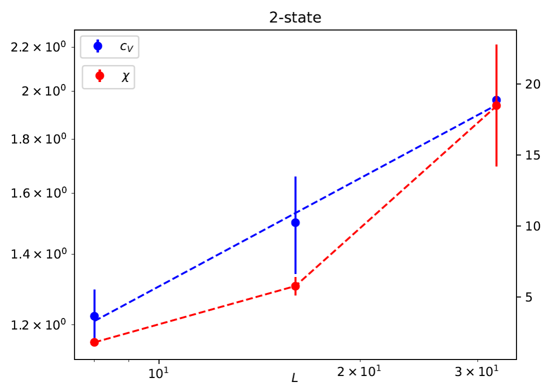

FIG. 6. $c_V^{\max}$ (left) and $\chi^{\max}$ (right) vs L as well as their corresponding fit functions (dashed line).

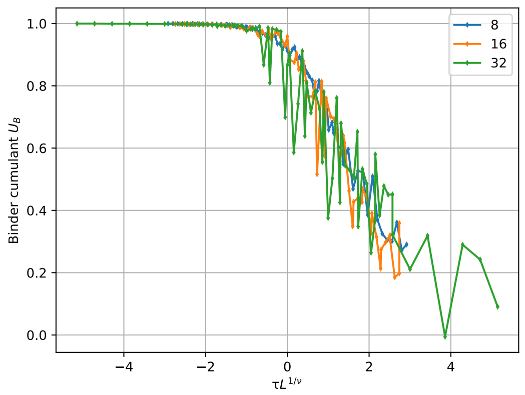

FIG. 7. U_B vs $\frac{T - T_{\text{crit}}}{T_{\text{crit}}} \cdot L^{1/1.1}$ of 2-state Potts model with various number N_{site} of lattice sites. Despite fluctuations in U_B for $N_{\text{site}} = 32^2$ all U_Bs seem to appear on a single line for $\nu = 1.1$.

IV. CONCLUSION

By means of finite size scaling analysis this brief project successfully confirms theoretical value of critical temperature $J/T_{\text{crit}} = \log(1 + \sqrt{N})$ in N-state Potts model with deviation of not larger than 12%. A particularly large uncertainty is observed around T_{crit} due to high autocorrelation time around $T \approx T_{\text{crit}}$, for which reason higher number N_{sweeps} of sweeps should have been set. Moreover, critical exponents α, γ, and ν of 2- and 4-state Potts model are extracted by fitting exponential

N	α/ν	γ/ν	ν
2	0.3404	1.6754	1.1
4	0.2054	1.1261	1.2
6	−0.3453	0.5863	2.2

TABLE III. Critical exponents of 2-, 4-, and 6-state Potts model. A doubt is cast on the critical exponent for $N = 6$, as it exhibits a negative value. This value is caused possibly by a large fluctuation of c_V around $T \approx T_{\text{crit}}$ due to insufficient number N_{sweeps} of sweeps.

function to c_V-L diagram and to χ-L diagram and by varying ν until all U_B appear on a single line. The result is presented in Tab. III. Due to large uncertainty in c_V, χ, and in U_B, α presented in Tab. III is physically not meaningful.

In comparison to 2D Ising model in square lattice, 2-state Potts model exhibits a half of Ising model critical temperature $\left(T_{\text{crit}}^{\text{Potts}} = 0.5 \cdot T_{\text{crit}}^{\text{Ising}}\right)$. This difference can be explained by considering two different extreme cases of configuration of spins neighboring an arbitrary site i. If all neighboring spins are aligned parallel to s_i, the energy coming from the local interaction between s_i and each of its neighboring spin is $-4J$ for both Ising and Potts model. However, if the neighboring spins are aligned anti-parallel to s_i the interaction energy would be $4J$ for Ising and 0 for Potts model. Thus, the difference in interaction energy between those two extreme cases are $8J$ ($4J$) in Ising (Potts) model, leading to a difference in T_{crit} by a factor of 2. The critical exponents of Ising model are $\alpha \approx 0.2608$, $\gamma \approx 1.7627$, $\nu \approx 1$, which are close to the critical exponents of 2-state Potts model (see Tab. III).

[1] G. Gallavotti, "Statistical mechanics," (Berlin: Springer-Verlag, 1999).
[2] R. B. Potts, Mathematical Proceedings of the Cambridge Philosophical Society **48**, 106 (1952).
[3] V. Beffara and H. Duminil-Copin, Probability Theory and Related Fields **153**, 511 (2012).
[4] H. Duminil-Copin, V. Sidoravicius, and V. Tassion, Communications in Mathematical Physics **349**, 47 (2015).
[5] H. Duminil-Copin, M. Gagnebin, M. Harel, I. Manolescu, and V. Tassion, "Discontinuity of the phase transition for the planar random-cluster and potts models with $q > 4$," (2017), arXiv:1611.09877 [math.PR].
[6] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, The Journal of Chemical Physics **21**, 1087 (2004), https://pubs.aip.org/aip/jcp/article-pdf/21/6/1087/8115285/1087_1_online.pdf.
[7] K. Binder, Z. Physik B - Condensed Matter **43**, 119 (1981).

Appendix A: Metropolis algorithm applied on Potts model

Below are several functions - written in python - used to run algorithm 1.

```python
import numpy as np
import matplotlib.pyplot as plt

def dirac_delta(x,y):
    """Dirac delta function delta(x,y)=1 iff x=y else 0"""
    if x == y: return 1
    else:return 0

def energy(system, i, j):
    """Energy function of spins connected to site (i, j)."""
    L = system.shape[0]
    assert L == system.shape[1], "Need to be square lattice!"
    e_ij = dirac_delta(system[i,j],system[np.mod(i-1,L),j])+\
    dirac_delta(system[i,j],system[np.mod(i+1,L),j])+\
    dirac_delta(system[i,j],system[i,np.mod(j-1,L)])+\
    dirac_delta(system[i,j],system[i,np.mod(j+1,L)])
    return -2*e_ij

def calc_theta(m):
    """Calculate the cutting angle between m and real axis"""
    if m.imag >= 0: theta=np.arccos(m.real/abs(m))
    elif m.real >=0: theta=np.arcsin(m.imag/abs(m))
    else: theta=np.arctan(m.imag/m.real)-np.pi
    return theta

def measure_magnetization(system,N):
    """Measure total magnetization of the system.
    Returns m (complex)"""
    L = system.shape[0]
    quant=2*np.pi/N
    m = 0
    for i in range(L):
        for j in range(L):
            m += np.exp(1j*quant*system[i,j])
    return m

def measure_energy(system):
    """Measure total energy of the system"""
    L = system.shape[0]
    E = 0
    for i in range(L):
        for j in range(L):
            E += energy(system, i, j) / 2.
    return E

def prepare_system(N,L):
    """Initialize the system."""
    system = np.random.randint(0, N, size=(L, L))
    return system

def metropolis_loop(system, N, T, N_sweeps, N_eq, N_flips):
    """ Main loop doing the Metropolis algorithm."""
    E = measure_energy(system)
    M = measure_magnetization(system,N)
    quant=2*np.pi/N
    L = system.shape[0]
    E_list, M_list = [E], [M]
    for step in range(N_sweeps + N_eq):
        #Choose site (i,j) randomly'
        i,j = np.random.randint(0, L), np.random.randint(0,L)
        #Take the value of spin at site (i,j)'
        old_val=system[i,j]
        #Flip the spin at site (i,j)'
        new_val=np.random.choice(list(range(old_val))+list(range(old_val+1,N)))

        #Calculate the change of energy
        dE = -2*(dirac_delta(new_val,system[np.mod(i-1,L),j])+\
             dirac_delta(new_val,system[np.mod(i+1,L),j])+\
             dirac_delta(new_val,system[i,np.mod(j-1,L)])+\
             dirac_delta(new_val,system[i,np.mod(j+1,L)]))-energy(system, i, j)
        #Calculate the change of magnetization
        dM = np.exp(1j*quant*new_val)-np.exp(1j*quant*old_val)
        if dE <= 0.:
            system[i,j] = new_val
            E += dE
            M += dM
        elif np.exp(-1. / T * dE) > np.random.rand():
            system[i,j] = new_val
            E += dE
            M += dM
        if step >= N_eq and np.mod(step, N_flips) == 0:
            # measurement
            E_list.append(E)
            M_list.append(M)
    return system, np.array(E_list), np.array(M_list)

def spec_heat(system,N,T,N_sweeps,N_eq,N_flips,N_bins):
    """Calculate specific heat for a given N-state potts model with temperature T"""
    L=system.shape[0]
    c_bin = []
    for k in range(N_bins):
        Es = metropolis_loop(system, N, T, N_sweeps, N_eq, N_flips)[1]

        c_bin.append(1. / T**2. / L**2. * np.var(Es))

    return np.mean(c_bin), np.std(c_bin)/np.sqrt(N_bins)

def abs_magnetization(system,N,T,N_sweeps,N_eq,N_flips,N_bins):
    """Calculate |m|=|M/(Lx*Ly)| for a given N-state potts model with temperature T"""
    L=system.shape[0]
    abs_m_bin = []
    for k in range(N_bins):
        Ms = metropolis_loop(system, N, T, N_sweeps, N_eq, N_flips)[2]

        mean_abs_m = np.mean(np.abs(Ms))/L**(2)

        abs_m_bin.append(mean_abs_m)

    return np.mean(abs_m_bin), np.std(abs_m_bin)/np.sqrt(N_bins)

def energy_density(system,N,T,N_sweeps,N_eq,N_flips,N_bins):
    """Calculate e=E/(Lx*Ly) for a given N-state potts model with temperature T"""
    L=system.shape[0]
    e_bin = []
    for k in range(N_bins):
        Es = metropolis_loop(system, N, T, N_sweeps, N_eq, N_flips)[1]

        mean_e = np.mean(Es/L**(2))

        e_bin.append(mean_e)

    return np.mean(e_bin), np.std(e_bin)/np.sqrt(N_bins)

def magn_susceptibility(system,N,T,N_sweeps,N_eq,N_flips,N_bins):
    """Calculate magnetic susceptibility for a given N-state potts model with temperature T"""
    L=system.shape[0]
    chi_bin = []
    for k in range(N_bins):
        Ms = metropolis_loop(system, N, T, N_sweeps, N_eq, N_flips)[2]

        chi_bin.append(1. / T / L**2. * np.var(np.abs(Ms)))

    return np.mean(chi_bin), np.std(chi_bin)/np.sqrt(N_bins)

def Binder_e_cumulant(system,N,T,N_sweeps,N_eq,N_flips,N_bins):
    """Calculate Binder cumulant with order parameter e"""
    L=system.shape[0]
    UBe_bin=[]
    for k in range(N_bins):
        Es = metropolis_loop(system, N, T, N_sweeps, N_eq, N_flips)[1]

        mean_e4 = np.mean((Es/L**(2))**4)
        mean_e2 = np.mean((Es/L**(2))**2)

        UBe_bin.append(3/2*(1 - 1/3 * mean_e4 / mean_e2**2))

    return np.mean(UBe_bin), np.std(UBe_bin)/np.sqrt(N_bins)

def Binder_m_cumulant(system,N,T,N_sweeps,N_eq,N_flips,N_bins):
    """Calculate Binder cumulant with order parameter |m|"""
    L=system.shape[0]
    UBm_bin=[]
    for k in range(N_bins):
        Ms = metropolis_loop(system, N, T, N_sweeps, N_eq, N_flips)[2]

        mean_m4 = np.mean(np.abs(Ms/L**(2))**4)
        mean_m2 = np.mean(np.abs(Ms/L**(2))**4)

        UBm_bin.append(3/2*(1 - 1/3 * mean_m4 / mean_m2**2))

    return np.mean(UBm_bin), np.std(UBm_bin)/np.sqrt(N_bins)

def produce_result(system,N,T,N_sweeps,N_eq,N_flips,N_bins):
    """Run metropolis algorithm to measure e, cv, m, |m|, chi, UBe, and UBm"""
    #Take system size
    L=system.shape[0]

    #Prepare list of measurement results
    e_bin = [] #for energy density
    c_bin = [] #for specific heat
    m_bin = [] #for (complex) magnetization
    abs_m_bin = [] #for |m|
    chi_bin = [] #for magnetic susceptibility
```

```python
    UBe_bin = [] #for Binder cumulant calculated using e
    UBm_bin = [] #for Binder cumulant calculated using |m|

    #Fill list of measurement results
    for k in range(N_bins):
        system, Es, Ms = metropolis_loop(system, N, T, N_sweeps, N_eq, N_flips)

        mean_e4 = np.mean((Es/L**(2))**4) # <(E/N)^4>=<e^4>
        mean_e2 = np.mean((Es/L**(2))**2) # <(E/N)^2>=<e^2>
        mean_abs_m4 = np.mean((np.abs(Ms)/L**(2))**4) # <|m|^4>
        mean_abs_m2 = np.mean((np.abs(Ms)/L**(2))**2) # <|m|^2>

        e_bin.append(np.mean(Es/L**(2))) #fill energy density list
        c_bin.append(np.var(Es) / (L**(2) * T**(2))) #fill specific hear list
        m_bin.append(np.mean(Ms)/L**(2)) #fill (complex) magnetization
        abs_m_bin.append(np.mean(np.abs(Ms))/L**(2)) #fill <|m|>
        chi_bin.append(np.var(np.abs(Ms))/(L**(2) * T)) #fill magnetic susceptibility
        UBe_bin.append(3/2*(1 - 1/3*mean_e4/mean_e2**2)) #fill Binder cumulant UBe
        UBm_bin.append(3/2*(1 - 1/3*mean_abs_m4/mean_abs_m2**2)) #fill Binder cumulant UBm

    return np.mean(e_bin), np.std(e_bin)/np.sqrt(N_bins),\
           np.mean(c_bin), np.std(c_bin)/np.sqrt(N_bins),\
           np.mean(m_bin),\
           np.mean(abs_m_bin), np.std(abs_m_bin)/np.sqrt(N_bins),\
           np.mean(chi_bin), np.std(chi_bin)/np.sqrt(N_bins),\
           np.mean(UBe_bin), np.std(UBe_bin)/np.sqrt(N_bins),\
           np.mean(UBm_bin), np.std(UBm_bin)/np.sqrt(N_bins)
```

To produce data used for finite size scaling analysis, the following codes are used.

```python
print("Produce data...")
Ns = [2,4,6]
Ls = [8,16,32]

data_ferro=dict((N, dict((L, []) for L in Ls)) for N in Ns)

for N in Ns:
    Tc_guess = 2/np.log(1+np.sqrt(N))
    for L in Ls:
        print("Start N={N}, L={L}".format(N=N,L=L))
        Ts = np.linspace(Tc_guess - 0.5, Tc_guess + 0.5, 25)
        Ts = np.append(Ts, np.linspace(Tc_guess - 8./L, Tc_guess + 8./L, 50))
        Ts = np.sort(Ts)[::-1]
        data_ferro[N][L].append(Ts) # fill data_ferro {N:{L:[Ts]},...}
        init_system = prepare_system(N,L) # initiate system

        # prepare list of measurement results
        e_list = np.array([])
        c_list = np.array([])
        m_list = np.array([])
        abs_m_list = np.array([])
        chi_list = np.array([])
        UBe_list = np.array([])
        UBm_list = np.array([])

        #set parameters for metropolis loop
        N_sweeps = int(500 * L**2)
        N_eq = int(N_sweeps // 5)
        N_flips = 10
        N_bins = 5

        # measure above units for each T using for-loop
        for T in Ts:
            e, e_err, \
            c, c_err, \
            m, \
            abs_m, abs_m_err, \
            chi, chi_err, \
            UBe, UBe_err, \
            UBm, UBm_err = produce_result(init_system,N,T,N_sweeps,N_eq,N_flips,N_bins)

            e_list = np.append(e_list, [e, e_err])
            c_list = np.append(c_list, [c, c_err])
            m_list = np.append(m_list, m)
            abs_m_list = np.append(abs_m_list, [abs_m, abs_m_err])
            chi_list = np.append(chi_list, [chi, chi_err])
            UBe_list = np.append(UBe_list, [UBe, UBe_err])
            UBm_list = np.append(UBm_list, [UBm, UBm_err])

        data_ferro[N][L].append(e_list) # fill data_ferro {N:{L:[Ts, e_list]},...}
        data_ferro[N][L].append(c_list) # fill data_ferro {N:{L:[Ts, e_list, c_list]}}
        data_ferro[N][L].append(m_list) # fill data_ferro {N:{L:[Ts, e_list, c_list, m_list]}}
        data_ferro[N][L].append(abs_m_list) # fill data_ferro {N:{L:[Ts, e_list, c_list, m_list, abs_m_list]}}
        data_ferro[N][L].append(chi_list) # fill data_ferro {N:{L:[Ts, e_list, c_list, m_list, abs_m_list, chi_list]}}
        data_ferro[N][L].append(UBe_list) # fill data_ferro {N:{L:[Ts, e_list, c_list, m_list, abs_m_list, chi_list, UBe_list]}}
        data_ferro[N][L].append(UBm_list) # fill data_ferro {N:{L:[Ts, e_list, c_list, m_list, abs_m_list, chi_list, UBe_list, UBm_list]}}

        print("Finish N={N}, L={L}".format(N=N,L=L))

    data_ferro[N]['2Tc/J']=Tc_guess # fill data_ferro {N:{'2Tc/J':Tc_guess}}

print("...Production of data finished")
```