Study of ferromagnetism using the Ising Model

Apoorva Bhatia[1], Roshni Sahoo[2], Nomya Mahana[3]

[1]*Email: apoorva.bhatia@xaviers.edu.in, UID:* ***192216***
[2]*Email: roshni.sahoo@xaviers.edu.in, UID:* ***192131***
[3]*Email: nomya.mahana@xaviers.edu.in, UID:* ***192138***

August 2021

Department of Physics

St. Xavier's College (Autonomous), Mumbai, India – 400001

Table of Contents

Abstract

Ferromagnetic substances are those substances in which the electrons have the tendency to align their spins in a common direction. The Ising model is a mathematical model used for the modelling of ferromagnetic substances. This can be done both analytically and computationally. In this project we solve this model computationally using the Metropolis algorithm which is a type of Monte Carlo algorithm. Furthermore, we have studied the various thermodynamic properties of the system such as energy, entropy, specific heat and magnetization as a function of temperature in case of both absence and presence of an external magnetic field. Analytical solution has also been discussed in case of no external magnetic field by defining the partition function which further helps us to find the probability for all states to be UP over temperature which in turn gives us an idea of why no phase transition is observed in case of 1-D Ising model in absence of external magnetic field.

Keywords: Ferromagnetism, Ising Model, Partition Function, Magnetization, Energy, Metropolis Algorithm

Study of ferromagnetism using the Ising Model

The Ising model is named after Ernst Ising who himself worked on this in his dissertation in 1925 to show phase transition in ferromagnetic substances. However, since then the model has found wide applications in various topics including protein folding, neural networks and social behavior apart from statistical physics. The Ising model is a very useful model for modelling a ferromagnetic substance. Ising himself solved the model exactly in one dimension but to his disappointment, was unable to see any phase transition. Two decades later, Lars Onsager solved the model in two dimensions and observed phase transition in the absence of an external magnetic field. In this project, we follow the footsteps of Ernst Ising and solve the 1-D Ising model for a ferromagnetic substance both analytically and computationally. To solve the Ising model computationally, the Metropolis- Hastings algorithm which is a Monte Carlo algorithm is implemented in Python. Furthermore, various thermodynamic properties of the ferromagnetic substance including energy, entropy, specific heat and magnetization as a function of temperature is computed. These properties are computed under 2 specific conditions – 1) in the absence of an external magnetic field and 2) in the presence of an external magnetic field. Furthermore, the magnetization as a function of the external magnetic field is also computed. The results obtained are then compared with the well-known behaviors of a ferromagnetic substances both in the presence and absence of magnetic fields. Lastly, the justification for the absence of phase transition in the 1-D Ising model in the absence of external magnetic field is explored.

Theory

Ferromagnetism is the strongest type of magnetism that is encountered in magnets in everyday life. A ferromagnet when kept in the presence of an external magnetic field gets magnetized and then remains magnetized even after the external field is removed. Some common ferromagnetic substances include iron, cobalt and nickel. Ferromagnetism forms the basis of many electrical and electrochemical devices such as electromagnets, electric motors, generators, etc.

Magnetic Properties

The origin of magnetism is the unpaired electrons in any substance that has a magnetic dipole moment ($\vec{\mu}$) producing its own magnetic field. This magnetic dipole moment of an electron comes from its *spin*. An electron is a spin-half particle, i.e., it can have only two spin states: +1 or -1. Thus, the magnetic field corresponding to the magnetic dipole can point either UP or DOWN. When placed in an external magnetic field $\vec{B}$, the field exerts a torque on the dipoles ($\vec{B} \times \vec{\mu}$) which lines the dipoles parallel to the magnetic field. Due to Pauli's exclusion principle, any pair of electrons will have opposite spin thus nullifying the effect of the torque. However, some unpaired electrons may be present which will align themselves to $\vec{B}$. When the external field is removed, the magnetic moments will randomize again. Such a material is a paramagnetic substance.

In ferromagnets, due to quantum mechanical effects the magnetic dipoles try to align in the same direction as their neighbors. These materials then form magnetic domains in which all the spins are aligned in the same direction. Thus, a ferromagnetic substance will have a large number of such domains oriented randomly in different directions. The average magnetic moment of these dipoles thus cancels out and the net magnetization is 0. These magnetic domains can then be aligned in the same direction by an external magnetic field. In ferromagnets, even when the magnetic field is removed, a net magnetization remains.

Ferromagnetic substances have a critical temperature or Curie point, after which the ferromagnets lose their magnetization and become paramagnetic, i.e., the magnetic moments orient themselves randomly. Below this temperature, this is a spontaneous magnetization where the ferromagnet behaves as a permanent magnet.

Modelling a ferromagnetic substance

To model a ferromagnetic substance using the 1-D Ising model, we have to first start by preparing a lattice of randomly oriented +1 and -1 spins. The line segment between any two spins is called the “bond”. Any spin only interacts with its nearest neighbor (this is done for simplicity), i.e., it will interact with another spin *only* to its immediate right or left.

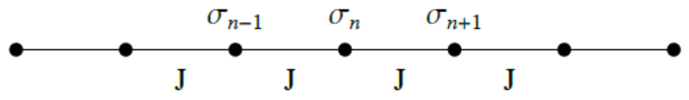


***Figure 1**. 1-D lattice of spins*

Each lattice site is denoted by $\sigma_j \in \{-1, 1\}$, representing the spin at the site (either up or down). Any distribution of spins (σ_1 , σ_2 , σ_3 …. σ_N) is known as the *microstate* of the system. For a system with N spins, there are going to be 2^N microstates.

The energy of each state of the system is given by the Hamiltonian which is defined as:

$$H = -\sum_{\langle i_j j\rangle} J_{ij}\sigma_i\sigma_j - \mathrm{h}\sum_j \sigma_j$$

J_{ij} is the interaction constant between the spins. For a ferromagnet, J is a positive value as each dipole tries to align itself to its nearest neighbors.

h is the external magnetic field. When h is positive, the domains try to align themselves in the direction of the field and when h is negative the domains align themselves opposite to the direction of the applied field.

The 1-D Ising model in the absence of an external magnetic field can be solved exactly as follows.

In the absence of magnetic field, h = 0 and the Hamiltonian comes out to be:

$$H = -\sum_{\langle i_j j \rangle} J_{ij}\sigma_i\sigma_j$$

Here, we assume that J > 0 i.e., the spins are interacting with each other. The equation is then solved using boundary conditions.

For this case, we consider an open boundary, so, the energy level comes out to be:

$$E_i = -J\sum_{i,i+1} \sigma_i\sigma_{i+1}$$

In total there are N spins but only (N – 1) bonds. The ground-state (i.e., the lowest energy state) is obtained when we set all spins up:

$$E_{gs} = -(N-1)J$$

The ground-state is therefore ferromagnetic (all spins aligned). But to know if this is true at a finite temperature, we calculate the partition function:

$$Z = \sum_{\sigma_1,\cdots,\sigma_N} e^{-\beta E}$$

The partition function is a sum over all possible configurations of the spin variables $\sigma_1, \cdots, \sigma_N$. Since each σ_N takes on the values ±1, there is a total of 2^N distinct terms in this sum. We consider the whole Z as the spins are interacting.

To do this, we first compute the sum of σ_N, as

$$Z = \sum_{\sigma_1,\cdots,\sigma_{N-1}} e^{\beta J(\sigma_1\sigma_2+\cdots+\sigma_{N-2}\sigma_{N-1})} \left(\sum_{\sigma_N} e^{\beta J\sigma_{N-1}\sigma_N}\right)$$

This sum gives us:

$$\sum_{\sigma_N} e^{\beta J\sigma_{N-1}\sigma_N} = e^{\beta J\sigma_{N-1}} + e^{-\beta J\sigma_{N-1}} = 2\cosh(\beta J\sigma_{N-1})$$

Now, cosh is an even function and $\sigma_{N-1} = \pm 1$. Hence, $\cosh(\beta J \sigma_{N-1}) = \cosh(J)$. This part of the sum therefore factors out of the partition function and we are left with

$$Z = 2\cosh(\beta J) \sum_{\sigma_1,\cdots,\sigma_{N-1}} e^{\beta J(\sigma_1\sigma_2+\cdots+\sigma_{N-2}\sigma_{N-1})}$$

What remains is exactly the partition function for a system with (N - 1) spins. If we continue to repeat this procedure, we get a factor of 2coshβJ each time. The only different term will be the last one (the sum over σ_1). It will have no argument so we will get simply $\sum_{\sigma_1} 1 = 2$. Hence, the partition function is:

$$Z = 2^N (\cosh \beta J)^{N-1}$$

Now, we compute the probability that all spins are up. The probability for any state $\sigma = (\sigma_1, \cdots, \sigma_N)$ is,

$$P_\sigma = \frac{e^{-\beta E_\sigma}}{Z}$$

The probability for the state in which all spins are up, $\sigma_1\sigma_2 + \cdots + \sigma_{N-1}\sigma_N = (N-1)$, will be:

$$P\,(all\ up) = \frac{e^{\beta J(N-1)}}{2^N(\cosh \beta J)^{N-1}} = \frac{1}{2^N}\left(\frac{e^{\beta J}}{\cosh \beta J}\right)^{N-1}$$

By symmetry, this is the same as the probability that all spins are down. When N is moderately small, P (all up) changes gradually from a small value at high T to 1/2 at T = 0. However, when value of N is increased the probability of finding all spins up tends to a step function, being zero above a certain temperature and exactly 1/2 below.

Moreover, this temperature where the jump occurs is also pushed to smaller and smaller values as N increases. Hence, as ($N \to \infty$), P (all up) becomes discontinuous, being exactly zero for all finite temperatures but then jumping to 1/2 exactly at T = 0. This is a strong indication that finite temperatures destroy the ferromagnetic order (i.e., that the model is ferromagnetic only at T = 0).

Assuming that N is significantly large, we can say that, $N \approx N-1$, and the thermodynamic quantities come out as:

$$\text{Free energy, } F = -T\,ln\,Z = -T\,ln\,2 - NT\,ln\left(2\cosh(\tfrac{J}{T})\right)$$

$$\text{Internal Energy, } U = -\frac{\partial}{\partial\beta} ln\,Z = -NJtanh(\tfrac{J}{T})$$

$$\text{Specific Heat, } C = \frac{\partial U}{\partial\beta} = N\left(\frac{J}{T}\right)^2 \text{sech}^2(\tfrac{J}{T})$$

$$\text{Entropy, } S = \frac{U-F}{T} = ln2 + N\left\{-\frac{J}{T}\tanh\left(\frac{J}{T}\right) + \ln\left[2\cosh\left(\frac{J}{T}\right)\right]\right\}$$

$$\text{Magnetization, } M = -\frac{\partial F}{\partial h}$$

For any microstate, the formula for magnetization is given by,

$$M = \sum_{n=1}^{N} \sigma_n$$

Metropolis Algorithm

In statistics and statistical physics, the Metropolis–Hastings algorithm is a Markov chain Monte Carlo (MCMC) method used for obtaining a sequence of random samples from a probability distribution from which direct sampling is difficult. This sequence can be used to approximate the distribution or to compute an integral.

To use it, we take a random variable x which occurs with probability π(x) and we calculate the average value of function f(x). It is given by:

$$\bar{f} = \sum_i f(x_i)\pi\,(x_i)$$

Since, we consider a real-life system, the value of i is very large, so we divide this into subsets:

$$\bar{f} = \frac{1}{M}\sum_i f(x_i)x\,M_i$$

Which can also be written as,

$$\bar{f} = \frac{1}{M}\sum_r f_r$$

here r refers to a ‘trial’ in which the variable x is generated with probability π(x), f_r is the value of f for that trial and M is the total number of trials. In this expression, a given value f_r will usually occur multiple times, in proportion to the probability $\pi(x_r)$ of x_r occurring.

But when the system is subjected to a “Heat Bath” that probability $\pi(x) \propto e^{-\beta E(x)}$ (*Boltzmann Distribution*)

One possible solution is to follow the condition of the Detailed Balance, which is

$$\pi(\mu)P(\mu \to \nu) = \pi(\nu)P(\nu \to \mu)$$

$$\frac{P(\mu \to \nu)}{P(\nu \to \mu)} = \frac{\pi(\nu)}{\pi(\mu)} = \frac{e^{-\beta E_\nu}}{e^{-\beta E_\mu}}$$

We decompose this transfer probability as:

$$P(\mu \to \nu) = g(\mu \to \nu)A(\mu \to \nu)$$

Where $g(\mu \to \nu)$ is the selection probability and $A(\mu \to \nu)$ is the acceptance ration.

This is the main gist of Monte-Carlo Algorithms that is to decide what this transition probability should be. It is the particular choice of the ratio given in that makes it the Metropolis Algorithm. If we are dealing with physical system such that the energy of two states (μ and ν) is E_μ and E_ν respectively then by Canonical Probability Distribution we can find the acceptance ratio as

$$\frac{A(\mu \to \nu)}{A(\nu \to \mu)} = e^{-\beta(E_\nu - E_\mu)}$$

At the end what we get is,

$$A(\mu \to \nu) = 1 \; if \; E_\nu < E_\mu$$

$$= e^{-\beta(E_\nu - E_\mu)} \; if \; E_\nu > E_\mu$$

Results & Discussion

Absence of external field

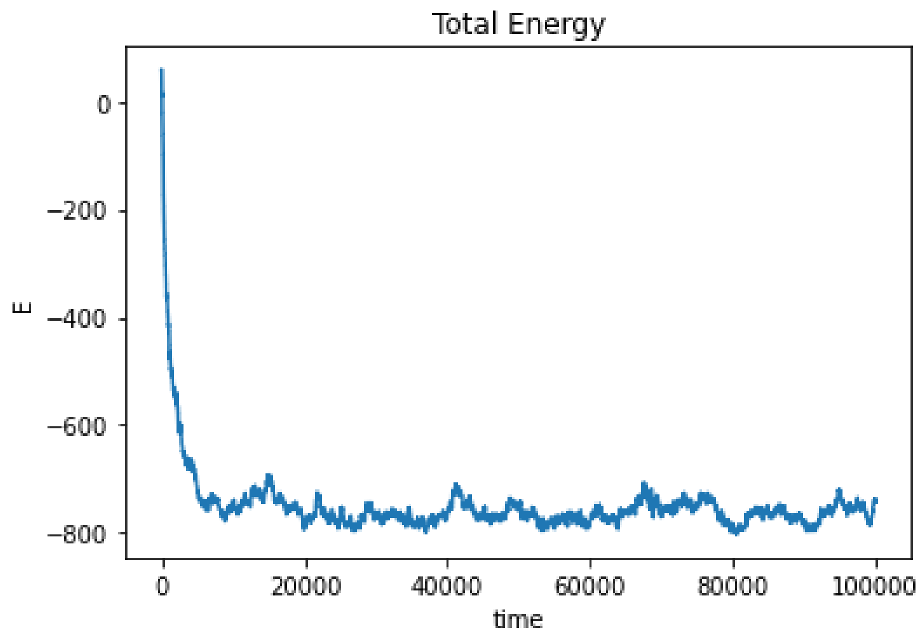


Figure 2. A system of 1000 spins with a randomized initial microstate. Energy reaches equilibrium value after finite time.

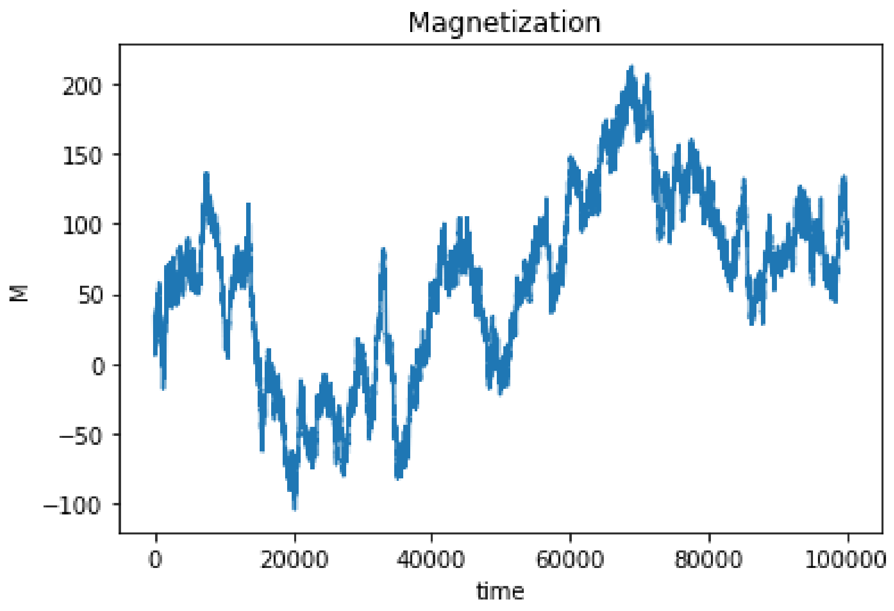
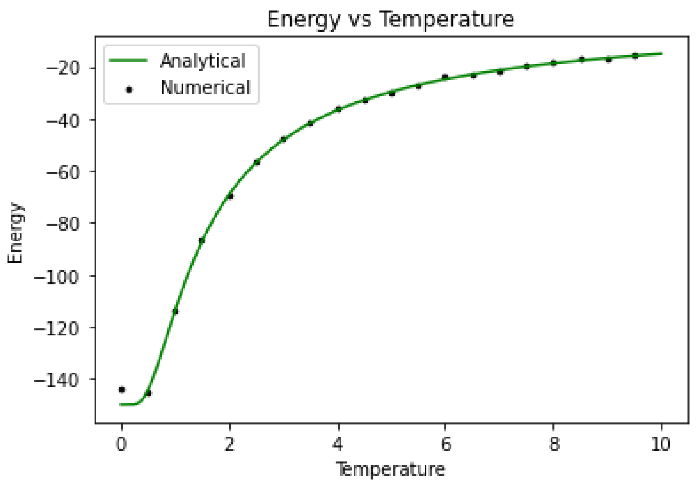
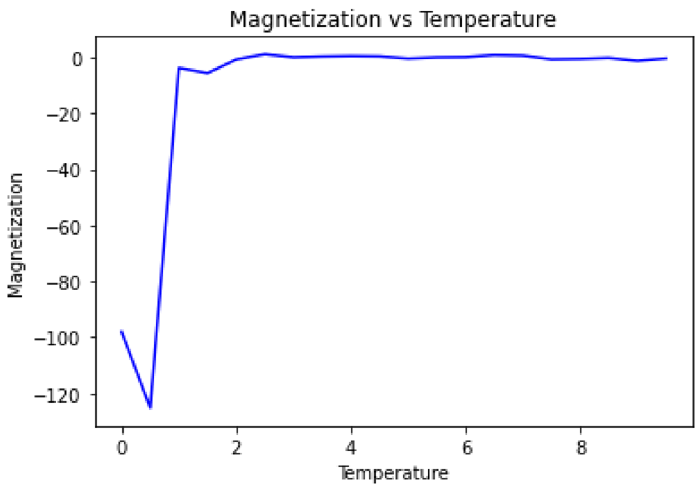


Figure 3. System of 1000 spins with a randomized initial microstate. Magnetization of the system changes randomly with time in the absence of external magnetic field.

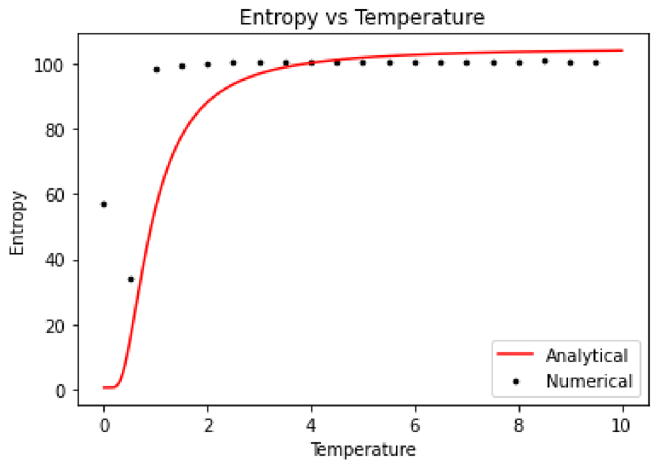
The state of the system is made to evolve with a timestep of 1 unit using the Metropolis algorithm for a long time until the energy of the system reaches remains constant with some minor fluctuations. Thus, once this system attains equilibrium, its energy fluctuates about its average value.



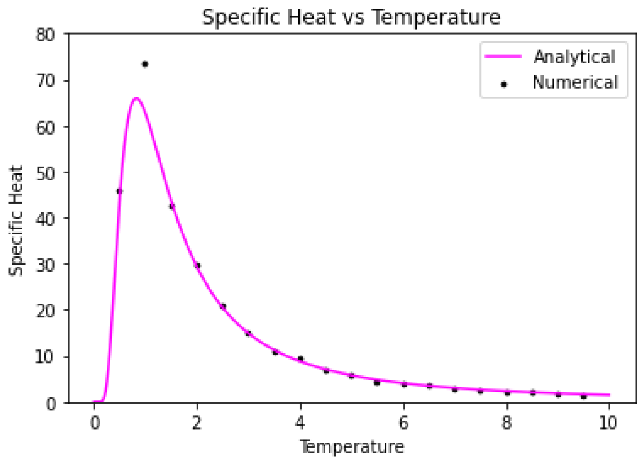
Energy as a function of temperature for a system of 150 spins.



Magnetization as a function of Temperature for a system of 150 spins. Magnetization is 0 for finite temperatures.



Entropy as a function of temperature for a system of 150 spins.



Specific heat as a function of temperature for a system of 150 spins

Figure 4. Thermodynamic properties of a ferromagnetic substance with 150 spins starting with a randomized initial state. The initial state of the system at any temperature was kept the same.

Each of the plots in Figure 4 represent a thermodynamic property of the system. As one can clearly observe, the plot for the energy, specific heat and entropy of the system as a function of temperature calculated numerically using the Metropolis algorithm are quite accurate when compared with the exact solution. In the graph for the magnetization as a function of temperature, one can see that for finite temperature the net magnetization is 0. Thus, there is no phase transition of the ferromagnetic substance in the 1-D Ising model in the absence of external magnetic field. This is exactly the result that Ernst Ising himself obtained when he analytically solved the 1-D Ising model. Thus, this result was *expected*. This can be further demonstrated by finding the probability for all the spins to be in the up or down state as a function of temperature.

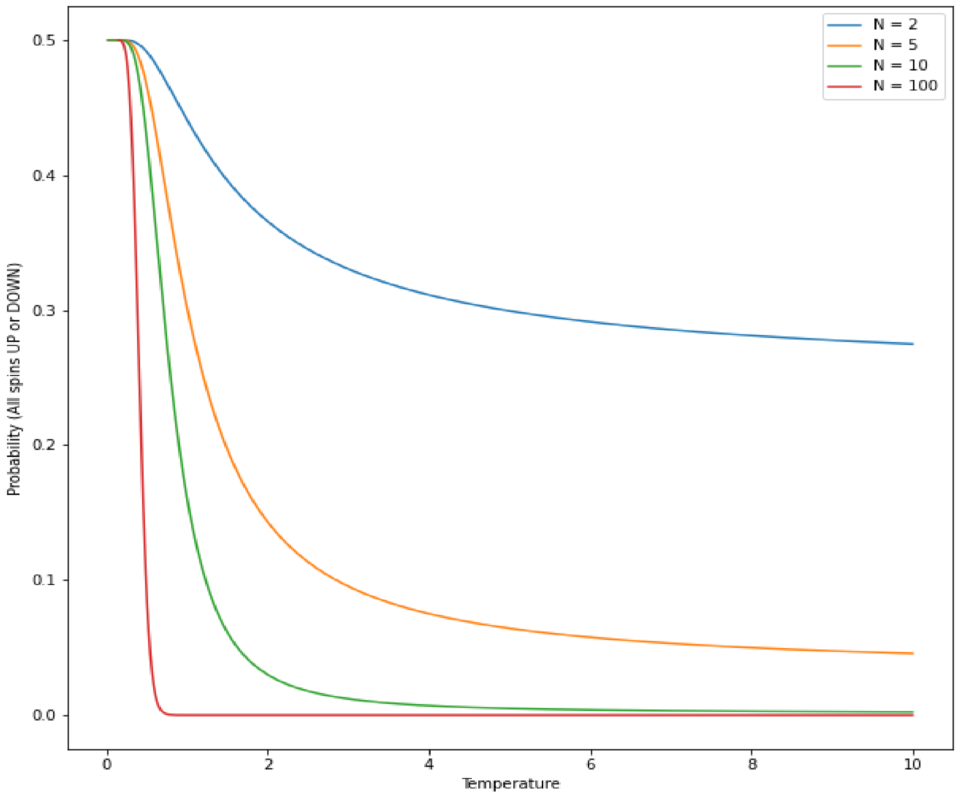


Figure 5. Probability for all the spins to be up as a function of temperature with number of spins as a parameter.

In figure 5 we can see that for large values of N, the probability for all the spins to be up (i.e., to show ferromagnetic behavior) is 0 for finite values of temperatures. Thus as N → ∞, P → 0 at finite temperatures.

Presence of magnetic field

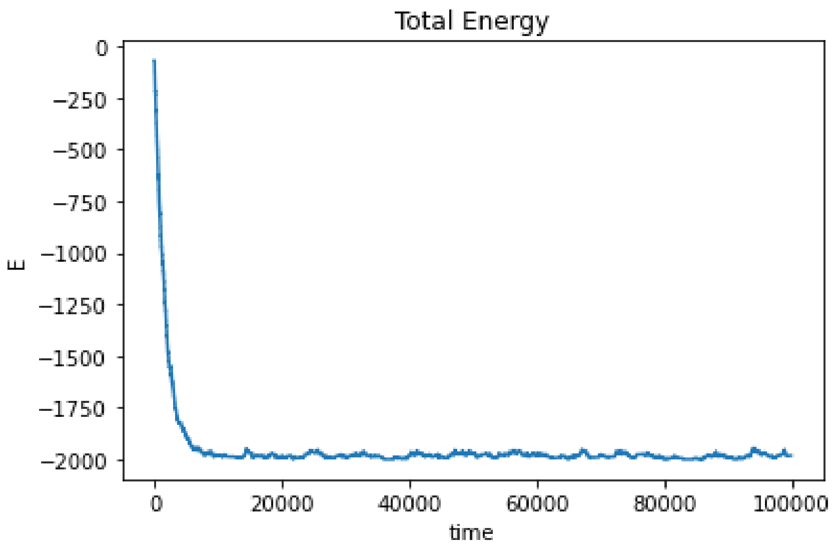


Figure 6. System of 1000 spins in the presence of an external magnetic field. System attains equilibrium after finite time and energy of the system thus attains equilibrium value with small fluctuations

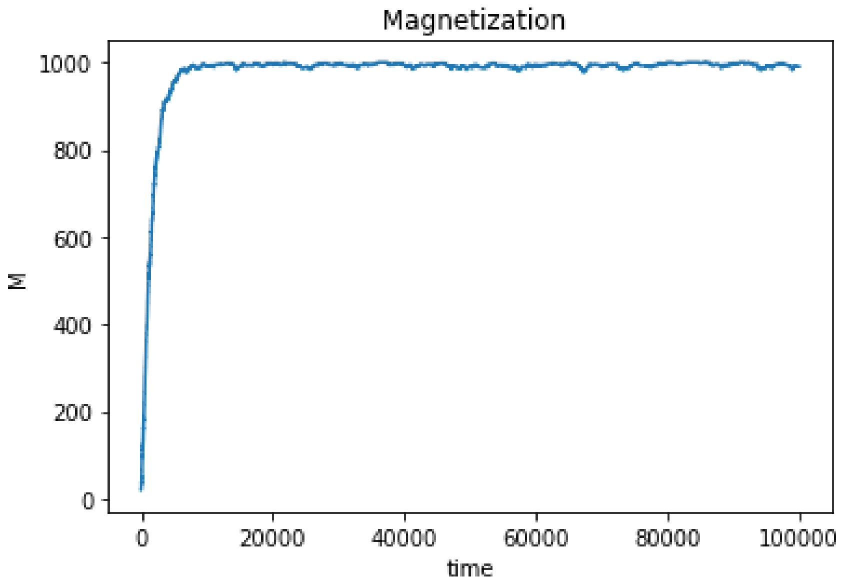


Figure 7. System of 1000 spins in the presence of an external magnetic field. System attains equilibrium after finite time and magnetization of the system thus attains equilibrium value with small fluctuations

Figure 6 and Figure 7 demonstrates how the system attains equilibrium after finite time when under the influence of an external magnetic field. The initial state of the system is randomized and then made to evolve with timestep equal to 1 unit using the Metropolis Algorithm. When the system attains equilibrium, the energy and magnetization of the system fluctuates about its average value. Since an external magnetic field is applied to the system, it is seen that the net magnetization of the system at equilibrium is around 1000, which implies the system is fully magnetized in the direction of the applied magnetic field (i.e., all the spins are up hence net magnetization is 1000). This is exactly opposed Figure 3 where the magnetization at each timestep is some random value (as there is no external magnetic field).

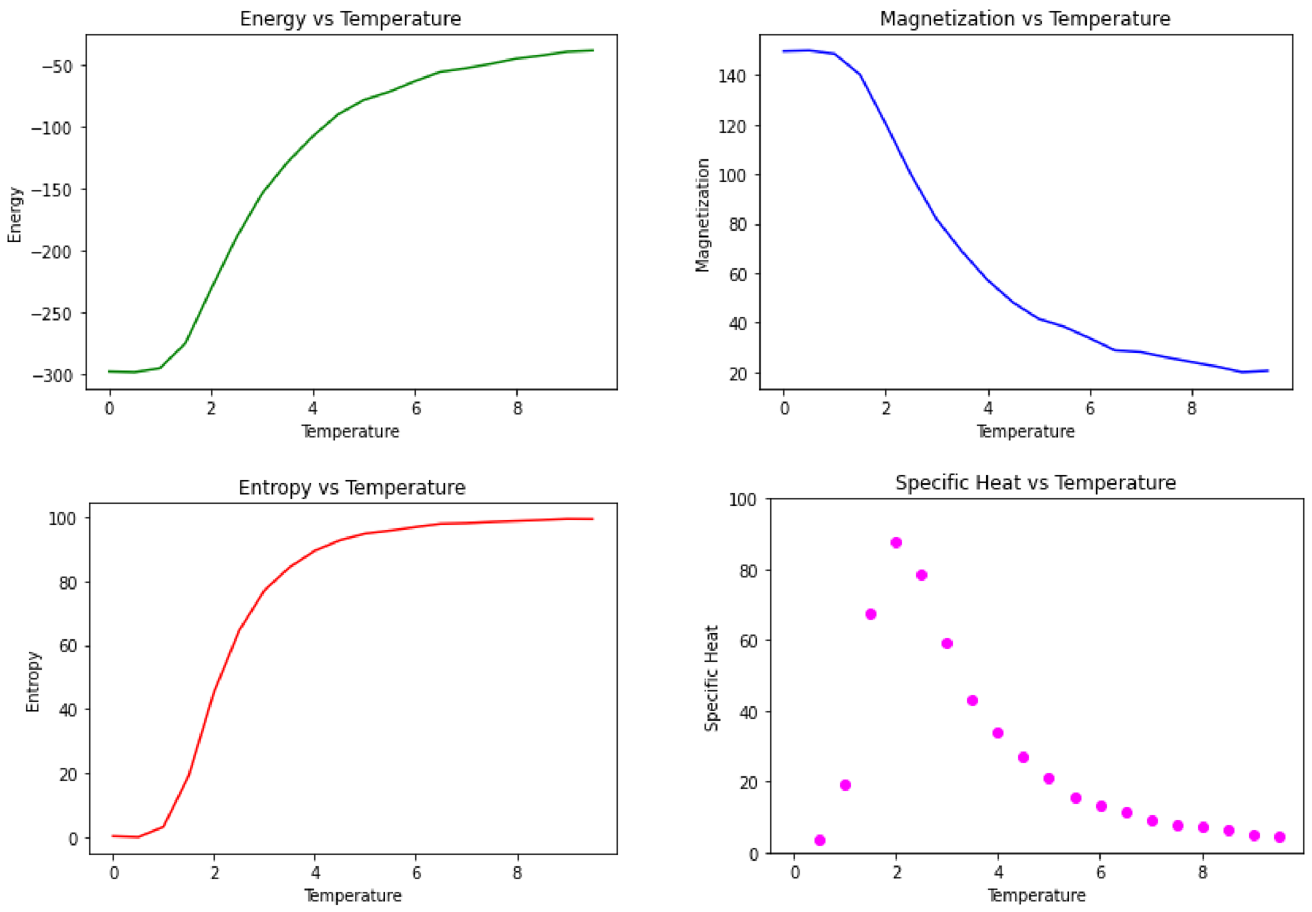


Figure 8. Thermodynamic properties of ferromagnetic substance in the presence of an external magnetic field. The system has 150 and the initial state of the system is selected at random which remains the same for all the temperatures.

In Figure 8, one can observe the phase transition of the ferromagnetic substance in the magnetization vs temperature plot. The magnetization of the system decreases after a certain value of temperature (called the Curie temperature). Beyond the Curie temperature, the material can no longer undergo spontaneous magnetization and thus behaves like a paramagnet.

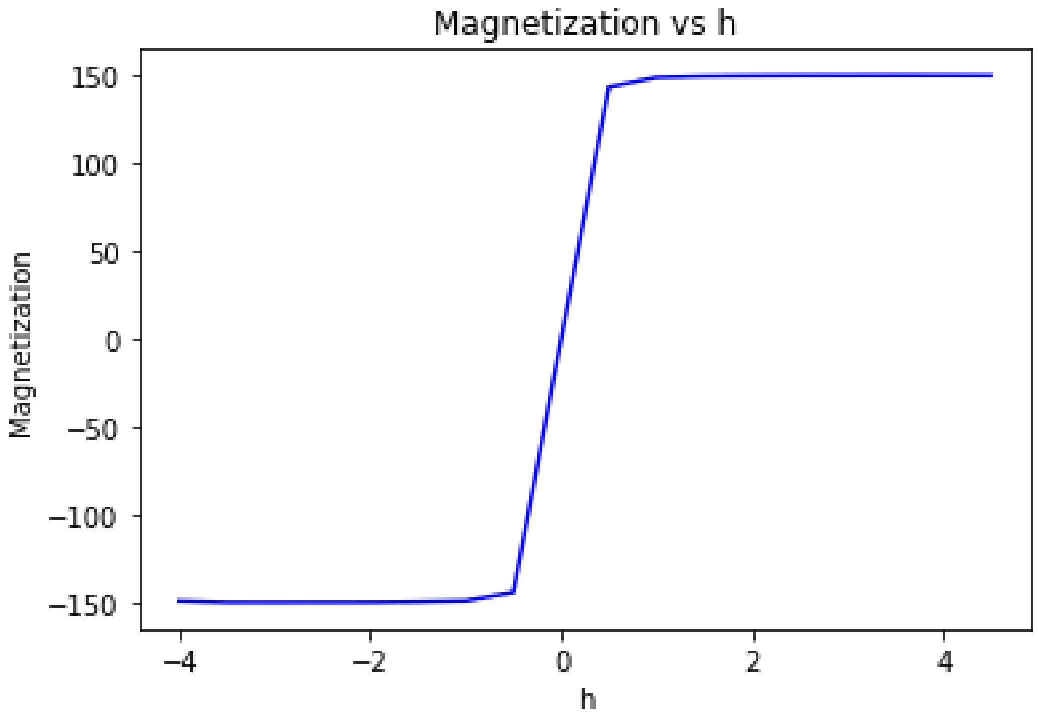


Figure 9. Magnetization as a function of external magnetic field 'h' at T = 1

Figure 9 represents the magnetization of the system as a function of the external magnetic field. It can be seen that at h = 0, there is a sudden jump in the magnetization.

Conclusion

The thermodynamics properties like energy, specific heat, entropy and magnetization of a ferromagnetic material has been modelled using the 1-D Ising model and computed using the Metropolis Algorithm and compared with the analytical solution.

Limitations

1. The number of spins taken are in the range of 1000 which is a very small compared to real life situations. Taking extremely high values of N will be computationally expensive and almost impossible to execute.
2. In the 1-D Ising model with 0 external magnetic field, no phase transition is observed. This is a limitation of the 1-D Ising model. To be able to observe phase transition under 0 external magnetic field, one has to use a 2-D Ising model.

Applications

1. Ising model came into the picture to investigate ferromagnetic properties for a system with a large fraction of electrons.
2. With the Ising model the spin glasses can also be described by the usual Hamiltonian where the S-variables describe the Ising spins, while the $J_{i,k}$ are taken from a random distribution.
3. The Ising model also has applications in neuroscience. The activity of neurons in the brain can be modelled statistically. Each neuron at any time is either active + or inactive −. Because the neural activity at any one time is modelled by independent bits, a dynamical Ising model would provide a first approximation to a neural network which is capable of learning.
4. Used to predict the beta regions for secondary structure of globular proteins.
5. Ising model can be used to interpret Lattice gas, a statistical model for the motion of atoms. This is used to understand a range of binding behaviors, including the binding of ligands to receptors in the cell surface, the binding of chemotaxis proteins to the flagellar motor, and the condensation of DNA.

Acknowledgments

We would like to give a special vote of thanks to our Physics Department for their continued guidance and support and helping us in successful completion of this project.

References

- Peierls, R., “On Isings model of ferromagnetism”.
- Loknathan and Gambhir. “Statistical and Thermal Physics”
- Ising, E. (1925), "Beitrag zur Theorie des Ferromagnetismus", *Z. Phys.*, **31** (1)

Appendix: Source Code

Link to Colab: https://colab.research.google.com/drive/1C6KeK9Jjk-FGSJ5GE-iDRDHMgzE3zpvU?usp=sharing

```
#1-D ISING MODEL (PRESENCE OF EXTERNAL MAGNETIC FIELD)

#import packages
import numpy as np
import random
import matplotlib.pyplot as plt

#Input for total spins in 1-D array
N = 1000

#Array of the initial microstate of the system

state0_list = []                                                    #empty
array for initial microstate
tot = 0
while (tot < N):
    p = random.choice([-1, 1])
    state0_list.append(p)                                           #append
ing state0 with random choice of -1 or 1
    tot += 1                                                        #append
ing state0 N times only

state0 = np.array(state0_list)                                      #list t
o array

#print('Initial state of the system is:',state0)

#Initial Parameters
h = 1
J = 1
T = 1     #Kelvin
Beta = 1/T

#Energy
E = []                                                               #list
for energy
energy = 0
for spin in range(len(state0) - 1):
  energy += -
J*state0[spin]*state0[spin+1] - h*state0[spin]         #hamiltonian

E.append(energy)                                                    #appen
ding initial energy into the energy list
print('Initial Energy =', E)

#Magnetization
M = []
m = np.sum(state0)                                                  #initi
al magnetization
M.append(m)
print('Initial Magnetization =', M)

iter = 100000                                                      #time

for i in range(iter):
    n = random.randint(0, N-1)
    state0[n] = -1*state0[n]
    energy_new = 0
    for spin in range(len(state0) - 1):
        energy_new += -J*state0[spin]*state0[spin+1] - h*state0[spin]
        del_energy = energy_new - E[i]
    p = np.random.uniform()
    if p <= np.exp(-Beta*del_energy):
        m_new = np.sum(state0)
        M.append(m_new)
        E.append(energy_new)
    else:
        state0[n] = -1*state0[n]
        E.append(E[i])
        M.append(M[i])



E = np.array(E)
M = np.array(M)

#Final energy and magnetization
print('Final Magnetization:', M[iter])
print('Final Energy:', E[iter])
print('Average Energy:',np.mean(E))
print('Average Manetization:',np.mean(M))
```

```
t = np.linspace(0, iter, iter+1)                                    #times
tep is 1 unit
plt.figure(1)
plt.title("Total Energy")
plt.xlabel("time")
plt.ylabel("E")
plt.plot(t, E)

plt.figure(2)
plt.title(" Magnetization")
plt.xlabel("time")
plt.ylabel("M")
plt.plot(t, M)
plt.show()


#Input for number of spins
N = 150

#Array of the initial microstate of the system

state0_list = []                                                   #empty
array for initial microstate
tot = 0
while (tot < N):
    p = random.choice([-1, 1])
    state0_list.append(p)                                          #append
ing state0 with random choice of -1 or 1
    tot += 1                                                       #append
ing state0 N times only

#Initial microstate of the system
state0 = np.array(state0_list)

#print('Initial state of the system is:',state0)

#Initial Parameters
h = 0              #External Magnetic field constant (0 for no magnetic fi
eld, non-zero for presence of magnetic field)
J = 1               #Interaction constant of the spins (+1 for ferro, -
1 for anti-ferro)

#Energy of the initial state
```

```
energy = 0                                                     #initi
alising
for spin in range(len(state0) - 1):
  energy += -
J*state0[spin]*state0[spin+1] - h*state0[spin]         #hamiltonian of the
 system


#Magnetization of the initial state
m = np.sum(state0)

#Entropy of the initial state
sigma_pos = 0
for sigma in state0:
  if sigma==+1:
    sigma_pos += 1
sigma_neg = N - sigma_pos
entropy = np.log(scipy.special.factorial(N)) - (np.log(scipy.special.facto
rial(sigma_pos)) + np.log(scipy.special.factorial(sigma_neg)))      #entro
py calculation

def energy_avg(T,h):
  Beta = 1/T
  E =[]
       #list for energy of each state
  E.append(energy)
  E_2 = []
       #list for energy^2 of each state
  E_2.append(energy**2)
  M = []
       #list for magnetization of each state
  M.append(m)
  S = []
       #list for entropy of each state
  S.append(entropy)

  #Metropolis starts

  iter = 100000
       #time
  for i in range(iter):
      n = random.randint(0, N-1)
      state0[n] = -1*state0[n]
      energy_new = 0
      for spin in range(len(state0)-1):
          energy_new += -J*state0[spin]*state0[spin+1] - h*state0[spin]
      del_energy = energy_new - E[i]
      p = np.random.uniform()
      if p <= np.exp(-Beta*del_energy):
            m_new = np.sum(state0)
            M.append(m_new)
            E.append(energy_new)
            E_2.append(energy_new**2)
      else:
            state0[n] = -1*state0[n]
            E.append(E[i])
            E_2.append(E[i]**2)
            M.append(M[i])

      #Entropy Calculation
      sigma_pos = 0
      for sigma in state0:
        if sigma==+1:
          sigma_pos += 1
      sigma_neg = N - sigma_pos
      Entropy = np.log(scipy.special.factorial(N)) - (np.log(scipy.special
.factorial(sigma_pos)) + np.log(scipy.special.factorial(sigma_neg)))
#entropy calculation
      S.append(Entropy)
  #Average Energy
  E_avg = sum(E)/len(E)

  #Specific Heat Calculation
  C = []
  for energy_2 in E_2:
    specific_heat = (Beta/T)*(energy_2 - (E_avg)**2)
    C.append(specific_heat)

  #Average Specific Heat
  C_avg = sum(C)/len(C)

  #Average Entropy
  S_avg = sum(S)/len(S)

  #Average Magnetization
  M_avg = sum(M)/len(M)

  return {'E_avg':E_avg, 'M_avg':M_avg,'S_avg':S_avg,'C_avg':C_avg}

h = []
```

```
E_avg =[]                                                                  #list
for collecting average energy at each temperature
M_avg = []                                                                 #list
for collecting average magnetization at each temperature
S_avg = []
C_avg =[]
M_h = []                                                              #list for c
ollecting average entropy at each temperature
Temp_list =[]
for Temp in np.arange(0.001,10,0.5):
  result = energy_avg(Temp,0)
  E_avg.append(result['E_avg'])
  M_avg.append(result['M_avg'])
  S_avg.append(result['S_avg'])
  C_avg.append(result['C_avg'])
  Temp_list.append(Temp)

#for hi in np.arange(-4,5,0.5):
 # result = energy_avg(1,hi)
  #M_h.append(result['M_avg'])
  #h.append(hi)


#Converting all lists to array for plotting purposes
E_avg = np.array(E_avg)
M_avg = np.array(M_avg)
S_avg = np.array(S_avg)
C_avg = np.array(C_avg)
Temp_list = np.array(Temp_list)
M_h = np.array(M_h)
h = np.array(h)


#Analytical solution for h = 0

T = np.linspace(0.01,10,1000)       #Temperature
N = 150
J = 1

#Average Energy Formula
def energy(T,N,J):
  return -N*J*np.tanh(J/T)

#Average Entropy Formula
def entropy(T,N,J):
  Beta = 1/T
  return np.log(2) + N*(-(J/T)*np.tanh(J/T) + np.log(2*np.cosh(J/T)))

#Average specific heat
def specific(T,N,J):
  return N*((J/T)**2)*(1/(np.cosh(J/T)**2))


#Energy vs Temperature
plt.figure(1)
plt.scatter(Temp_list, E_avg,color='black',label='Numerical')
         #Numerical
plt.plot(T, energy(T,N,J),color='green',label='Analytical')
      #Analytical
plt.title('Energy vs Temperature')
plt.xlabel('Temperature')
plt.ylabel('Energy')
plt.legend()

#Magnetization vs Temperature
plt.figure(2)
plt.plot(Temp_list,M_avg,color='blue')
plt.title('Magnetization vs Temperature')
plt.xlabel('Temperature')
plt.ylabel('Magnetization')

#Entropy vs Temperature
plt.figure(3)
plt.scatter(Temp_list,S_avg,color='black',label = 'Numerical')        #Num
erical
plt.plot(T, entropy(T,N,J), color = 'red', label='Analytical')
      #Analytical
plt.title('Entropy vs Temperature')
plt.xlabel('Temperature')
plt.ylabel('Entropy')
plt.legend()

#Specific heat vs Temperature
plt.figure(4)
plt.scatter(Temp_list,C_avg,color='black',alpha=1,label = 'Numerical')
    #Numerical
plt.plot(T, specific(T,N,J), color = 'magenta', label='Analytical')
           #Analytical
plt.title('Specific Heat vs Temperature')
plt.xlabel('Temperature')
plt.ylabel('Specific Heat')
plt.ylim(0,90)
plt.legend()


plt.figure(4)
plt.plot(h,M_h,color='blue')
plt.title('Magnetization vs h')
plt.xlabel('h')
plt.ylabel('Magnetization')
plt.show()


#formula for probability that all spins are UP (when h = 0)
def P_allUP(N,T,J):
  Beta = 1/T
  return np.exp(Beta*J*(N-1))/((2**N)*(np.cosh(Beta*J))**(N-1))

#Plotting
T = np.linspace(0.01,10,1000)
N = [2,5,10,100]
plt.figure(figsize=(10,10))
plt.ylabel('Probability (All spins UP or DOWN)')
plt.xlabel('Temperature')
for i in N:
  plt.plot(T,P_allUP(i,T,1),label='N = {}'.format(i))
  plt.legend()
```