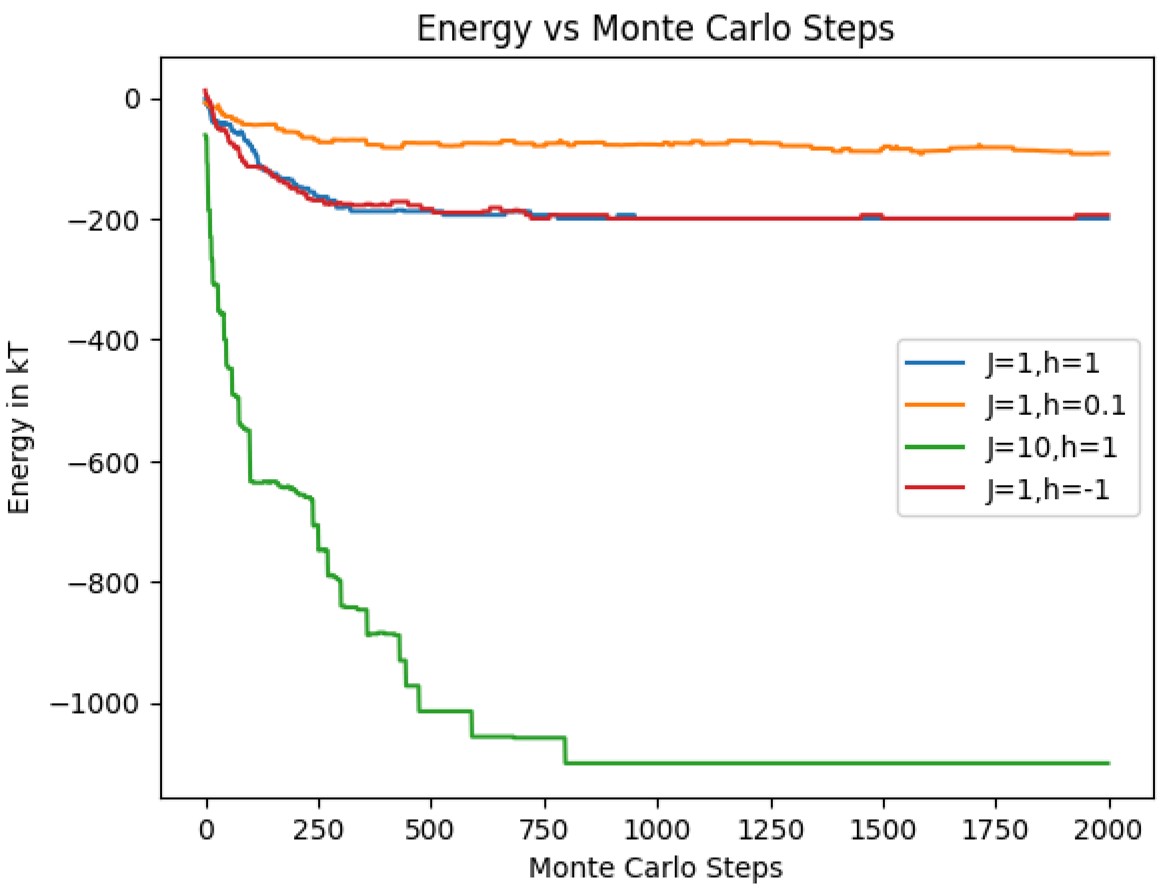
```
import numpy as np
import matplotlib.pyplot as plt
```

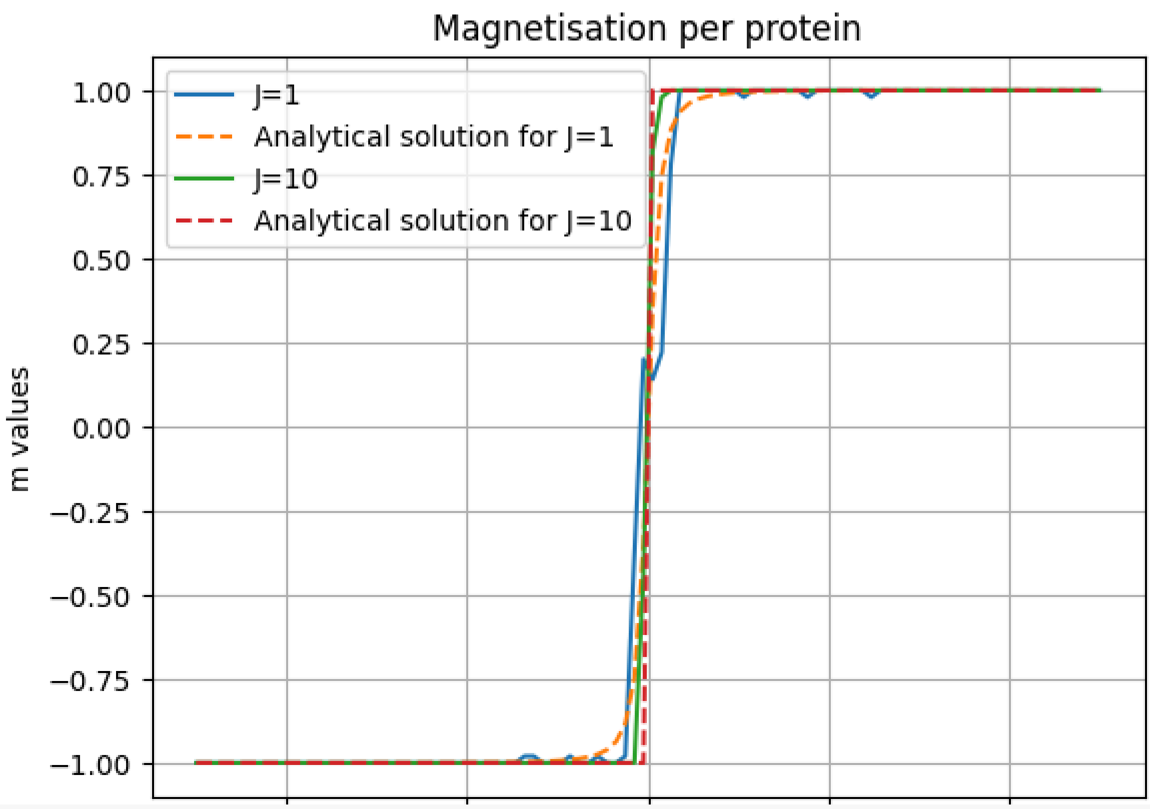
```
class Model:
  def __init__(self, J: float, h: float, m: int):
    self.J = J
    self.h = h
    self.m = m
    self.energies = []

  def get_energy(self):
    return (np.dot(self.S,np.roll(self.S,-1)))*self.J*(-1) - np.sum(self.S)*self.h # Usir

  def simulate(self, N: int):
    self.S = np.random.choice([1,-1],self.m)

    while (N!=0):
      N-=1
      old_E = self.get_energy()

      state = np.random.choice(np.arange(0,self.m))
      self.S[state] *= -1

      new_E = self.get_energy()
      delta_E = new_E - old_E
      if delta_E > 0:
        p = np.exp(-delta_E)
        if np.random.random() > p:
          self.S[state] *= -1
          self.energies.append(old_E)
          continue

      self.energies.append(new_E)

  def plot_graph(self):
    plt.title("Energy vs Monte Carlo Steps")
    plt.plot(self.energies,label=f"J={self.J},h={self.h}")
    plt.xlabel("Monte Carlo Steps")
    plt.ylabel("Energy in kT")
```

```
for J,h in ((1,1),(1,0.1),(10,1),(1,-1)):
  my_model = Model(J,h,100)
  my_model.simulate(2000)
  my_model.plot_graph()
plt.legend()
plt.show()
```



After simulation with 2000 steps, models have reached a steady state

```
def anal(J,h):
  return np.sinh(h)/np.sqrt(np.cosh(h)*np.cosh(h) - 2*np.exp(-2*J)*np.sinh(2*J))

h_values = np.linspace(-5,5,100)
for J in (1,10):
  m_values = []
  for h in h_values:
    mag_model = Model(J,h,100)
    mag_model.simulate(4000)
    M = np.sum(mag_model.S)
    m = M/100
    m_values.append(m)

  plt.title("Magnetisation per protein")
  plt.plot(h_values,m_values,'-',label=f"J={J}")
  plt.plot(h_values,anal(J,h_values),'--',label=f"Analytical solution for J={J}")

plt.xlabel("h values")
plt.ylabel("m values")
plt.legend()
plt.grid()
plt.show()
```




```
for J,h in ((1,1),(1,0.1),(10,1),(1,-1)):
  my_model = Model(J,h,100)
  my_model.simulate(6000)
  eq_energies = my_model.energies[3000:] # Energies after equilibrium attained
  avg_energy = np.average(eq_energies)
  avg_sq_energy = np.average(np.array(eq_energies)**2)
  # print(avg_energy)
  # print(avg_sq_energy)
  print(f"Specific Heat for J={J} and h={h} is: {avg_sq_energy - avg_energy**2:.4f}")
```

```
Specific Heat for J=1 and h=1 is: 4.3967
Specific Heat for J=1 and h=0.1 is: 42.2328
Specific Heat for J=10 and h=1 is: 0.0000
Specific Heat for J=1 and h=-1 is: 2.9287
```