Computational Physics - Exercise 3

November 12, 2021

Exercise 3
Keito Watanabe (s6kewata), Haveesh Singirikonda (s6gusing)
1). From the sheet, we have the expression for the average magnetization per site as follows:

$$< m >= \frac{1}{N\beta}\frac{\partial}{\partial h}(\log Z) = \frac{1}{N\beta}\frac{1}{Z}\frac{\partial Z}{\partial h}$$

So using the partition function (with $J > 0$) given in the exercise sheet, we have that:

$$\begin{aligned}
\frac{\partial Z}{\partial h} &= \frac{\partial}{\partial h}\left(\int_{-\infty}^{+\infty}\frac{d\phi}{\sqrt{2\pi\beta\hat{J}}}\exp\left[-\frac{\phi^2}{2\beta\hat{J}} + N\log\left(2\cosh(\beta h \pm \phi)\right)\right]\right) \\
&= \int_{-\infty}^{+\infty}\frac{d\phi}{\sqrt{2\pi\beta\hat{J}}}\frac{\partial}{\partial h}\left(\exp\left[-\frac{\phi^2}{2\beta\hat{J}} + N\log\left(2\cosh(\beta h \pm \phi)\right)\right]\right) \\
&:= \int_{-\infty}^{+\infty}\frac{d\phi}{\sqrt{2\pi\beta\hat{J}}}\frac{\partial}{\partial h}\left(\exp\left[\kappa(\beta, h)\right]\right) \\
&= \int_{-\infty}^{+\infty}\frac{d\phi}{\sqrt{2\pi\beta\hat{J}}}e^{\kappa(\beta,\phi)}\frac{\partial\kappa}{\partial h}
\end{aligned}$$

Now:

$$\frac{\partial\kappa}{\partial h} = N\frac{2\sinh(\beta h \pm \phi)}{2\cosh(\beta h \pm \phi)}\beta = N\beta\tanh(\beta h \pm \phi)$$

Substituting Eq. (3) into Eq. (2), and thus substituting this into Eq. (1), we get our average magnetization per site as follows:

$$\begin{aligned}
< m > &= \frac{1}{N\beta}\frac{1}{Z}\int_{-\infty}^{+\infty}\frac{d\phi}{\sqrt{2\pi\beta\hat{J}}}e^{\kappa(\beta,\phi)}N\beta\tanh(\beta h \pm \phi) \\
&= \frac{1}{Z}\int_{-\infty}^{+\infty}\frac{d\phi}{\sqrt{2\pi\beta\hat{J}}}e^{\kappa(\beta,\phi)}\tanh(\beta h \pm \phi)
\end{aligned}$$

Comparing this to the expression for the expectation value for some operator O, we see that:

$$\boxed{m[\phi] = \tanh(\beta h \pm \phi)}$$

In a similar fashion, we can evaluate the average energy per site. We have the following expression for $< \epsilon >$ as such:

$$< \epsilon > = -\frac{1}{N}\frac{\partial}{\partial \beta}(\log Z) = -\frac{1}{NZ}\frac{\partial Z}{\partial \beta}$$

Now using the expression for the partition function (with $J > 0$) given in the exercise sheet, we have:

$$\begin{aligned}
\frac{\partial Z}{\partial \beta} &= \frac{\partial}{\partial \beta}\left(\int_{-\infty}^{+\infty} \frac{d\phi}{\sqrt{2\pi\beta\hat{J}}} \exp\left[-\frac{\phi^2}{2\beta\hat{J}} + N\log\left(2\cosh(\beta h \pm \phi)\right)\right]\right) \\
&= \int_{-\infty}^{+\infty} d\phi \frac{\partial}{\partial \beta}\left(\frac{\exp\left[-\frac{\phi^2}{2\beta\hat{J}} + N\log\left(2\cosh(\beta h \pm \phi)\right)\right]}{\sqrt{2\pi\beta\hat{J}}}\right) \\
&= \int_{-\infty}^{+\infty} d\phi \left(\frac{e^{\kappa(\beta,h)}}{\sqrt{2\pi\beta\hat{J}}}\frac{\partial \kappa}{\partial \beta} + e^{\kappa(\beta,h)}\left(-\frac{1}{2\beta\sqrt{2\pi\beta\hat{J}}}\right)\right) \\
&= \int_{-\infty}^{+\infty} \frac{d\phi e^{\kappa(\beta,h)}}{\sqrt{2\pi\beta\hat{J}}}\left(\frac{\partial \kappa}{\partial \beta} - \frac{1}{2\beta}\right)
\end{aligned}$$

Now we observe that:

$$\begin{aligned}
\frac{\partial \kappa}{\partial \beta} &= \frac{\partial}{\partial \beta}\left(-\frac{\phi^2}{2\beta\hat{J}} + N\log\left(2\cosh(\beta h \pm \phi)\right)\right) \\
&= \frac{\phi^2}{2\beta^2\hat{J}} + N\frac{2\sinh(\beta h \pm \phi)}{2\cosh(\beta h \pm \phi)}h \\
&= \frac{\phi^2}{2\beta^2\hat{J}} + Nh\tanh(\beta h \pm \phi)
\end{aligned}$$

Thus combining our expressions, we have the following:

$$\begin{aligned}
< \epsilon > &= -\frac{1}{NZ}\int_{-\infty}^{+\infty} \frac{d\phi e^{\kappa(\beta,h)}}{\sqrt{2\pi\beta\hat{J}}}\left(\frac{\phi^2}{2\beta^2\hat{J}} + Nh\tanh(\beta h \pm \phi) - \frac{1}{2\beta}\right) \\
&= \frac{1}{Z}\int_{-\infty}^{+\infty} \frac{d\phi e^{\kappa(\beta,h)}}{\sqrt{2\pi\beta\hat{J}}}\left(\frac{\phi^2}{2\beta^2 N\hat{J}} + h\tanh(\beta h \pm \phi) - \frac{1}{2\beta N}\right)
\end{aligned}$$

Thus we see that the average energy per site is given as:

$$\boxed{\epsilon[\phi] = \frac{\phi^2}{2\beta^2 N\hat{J}} + h\tanh(\beta h \pm \phi) - \frac{1}{2\beta N}}$$

2). We are given that the Hamiltonian is,

$$\mathcal{H}(p,\phi) = \frac{p^2}{2} + \frac{\phi^2}{2\beta\hat{J}} - N\log(2\cosh(\beta h + \phi)) \tag{1}$$

The equations of motion are,

$$\dot{\phi} = \frac{\partial\mathcal{H}}{\partial p}, \dot{p} = -\frac{\partial\mathcal{H}}{\partial\phi} \tag{2}$$

After evaluating the derivatives, the equations would look like,

$$\dot{\phi} = p \tag{3}$$

$$\dot{p} = -\frac{\phi}{\beta\hat{J}} + N\tanh(\beta h + \phi) \tag{4}$$

[ ]:
```
import numpy as np
```

Defining the *artificial* Hamiltonian.

And using this with the equations of motion, we get $\dot{p}$ and $\dot{q}$. We will use these expressions to evaluate the leapfrog algorithm.

[ ]:
```
b = 1

def H(p,q,J,h,N):

    ham = p**2/2. + q**2/(2*b*J) - N*np.log(2*np.cosh(b*h+q))

    return ham

def p_dot(q,p,J,h,N):

    pd =  q/(b*J) - N*np.tanh(b*h+q)

    return -pd

def q_dot(q,p,J,h,N):

    return p

def P_acc(p,q,J,h,N):

    return np.exp(-H(p,q,J,h,N))
```

3). The leapfrog algorithm follows in the next block.

[ ]:
```
def leapfrog(N_md,p_0,q_0,J,h,N):

    dt = 1/N_md

    p = p_0
```

```python
    q = q_0

    q = q + 0.5*q_dot(q,p,J,h,N)*dt

    for i in range(N_md):

        p = p + p_dot(q,p,J,h,N)*dt

        if i!=N_md-1:

            q = q + q_dot(q,p,J,h,N)*dt

    q = q + 0.5*q_dot(q,p,J,h,N)*dt

    return p,q
```

[ ]:
```python
N_md = 100

p_0 = 0.1
q_0 = 1

h = 1

N = 20

J = 1/N

p_lf,q_lf = leapfrog(N_md,p_0,q_0,J,h,N)
```

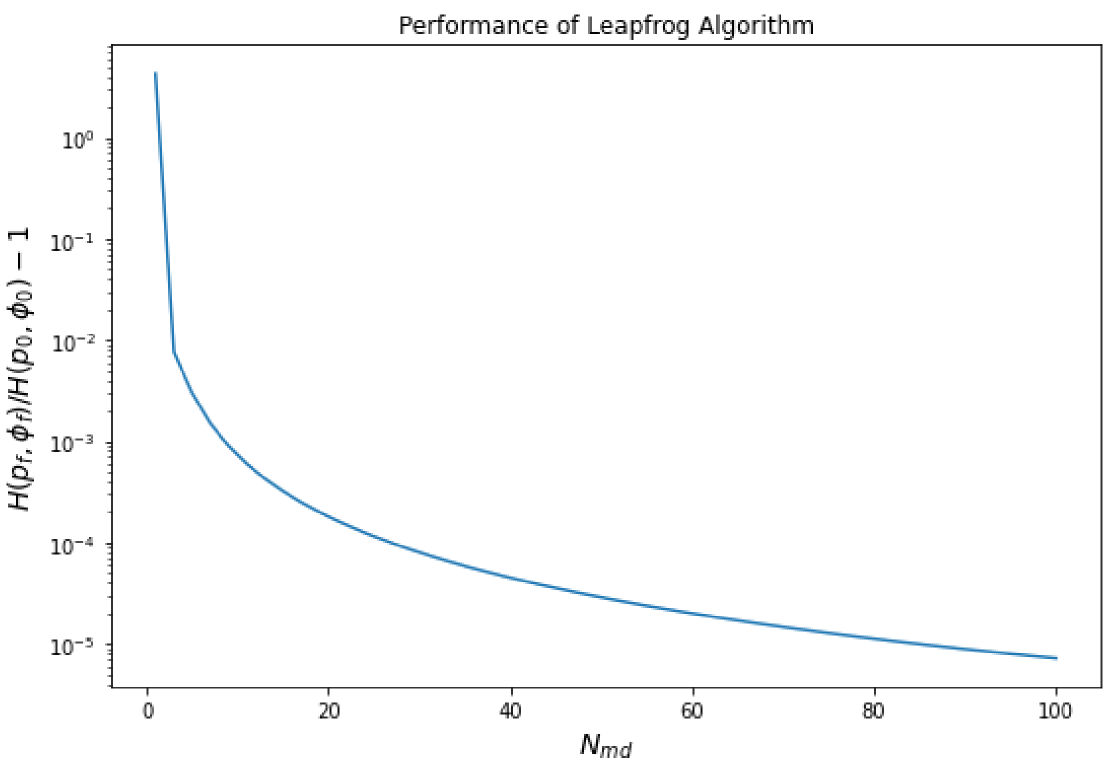
[ ]:
```python
# check if the difference in Hamiltonian would be a small value
dH = H(p_lf,q_lf,J,h,N) - H(p_0,q_0,J,h,N)

dH = dH/H(p_0,q_0,J,h,N)

dH
```

[ ]: -2.1365720552909926e-08

[ ]:
```python
N_md = np.linspace(1,100,dtype='int')

p_0 = 1
q_0 = 1

h = 1
N = 20
J = 1/N

h0 = H(p_0,q_0,J,h,N)
```

```
H_md = np.ones(len(N_md))

for i in range(len(N_md)):

    p,q = leapfrog(N_md[i],p_0,q_0,J,h,N)

    H_md[i] = (H(p,q,J,h,N) - h0)/h0

H_md = np.abs(H_md)
```

[ ]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(9,6))
plt.semilogy(N_md,H_md)
plt.xlabel('$N_{md}$', fontsize=14)
plt.ylabel('$H(p_f,\phi_f)/H(p_0,\phi_0)-1$', fontsize=14)
plt.title("Performance of Leapfrog Algorithm");
```



The plot looks like the one given in Figure 1.
4). The code using the HMC algorithm is written in the next block.

[ ]:
```
def HMC(N_s,N_md,J,h,N):

    q_mc = np.ones(N_s)
```

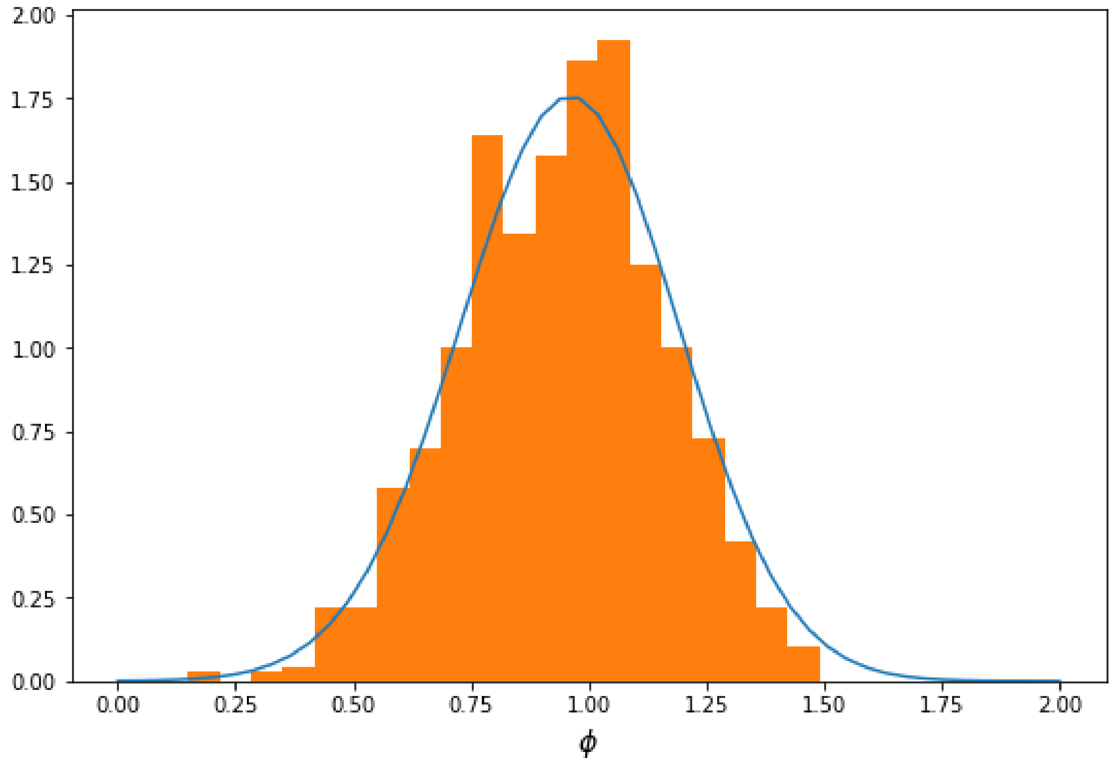
```
    p_mc = np.ones(N_s)

    acc = 0

    for i in range(N_s):

        p_0 = np.random.normal()
        q_0 = 1.0

        p_l,q_l = leapfrog(N_md,p_0,q_0,J,h,N)

        P_0 = P_acc(p_0,q_0,J,h,N)
        P_l = P_acc(p_l,q_l,J,h,N)

        r = np.random.normal()

        if P_l>P_0:
            q_mc[i] = q_l
            p_mc[i] = p_l
            acc += 1

        elif P_l/P_0>r:
            q_mc[i] = q_l
            p_mc[i] = p_l
            acc += 1

        else:
            q_mc[i] = q_0
            p_mc[i] = p_0


    return q_mc,p_mc,acc/N_s
```

[ ]:
```
N_s = 1000

h = 1
N = 20
J = 1/N

N_md = 100

q_mc, p_mc , acc = HMC(N_s,N_md,J,h,N)

acc  # the acceptance rate
```

[ ]: 0.976

The acceptance rate for N_{md} is $\sim 98\%$
Plotting the histogram to check if our sampling is correct.

[ ]:
```
q_th = np.linspace(0,2)

h = 1
N = 20
J = 1/N
b = 1

P_th = np.exp(-q_th**2/(2*b*J))*np.exp(N*np.log(2*np.cosh(b*h+q_th)))
P_th = P_th/(np.max(P_th))*1.75

# q_tmp = q_mc/(np.max(q_mc))
plt.figure(figsize=(9,6))
plt.plot(q_th,P_th)
plt.hist(q_mc,density=True,bins=20)
plt.xlabel('$\phi$', fontsize=14);
```



The plot follows the theoretical distribution, so the HMC code works.
Calculating $< m >$ and $< \epsilon >$ for $J = 1$

[ ]:
```
m_q = np.sum(np.tanh(b*h+q_mc))

m_q = m_q/N_s

m_q
```

[ ]: 0.9557907701665327

```
e_q = np.sum(q_mc**2/(2*J*b**2) + N*h*np.tanh(b*h+q_mc) - 1/(2*b))

e_q = e_q/N_s

e_q
```

[ ]: 27.94442696099703

5). We plot the values of $< m >$ and $< \epsilon >$ vs $\hat{J}$.

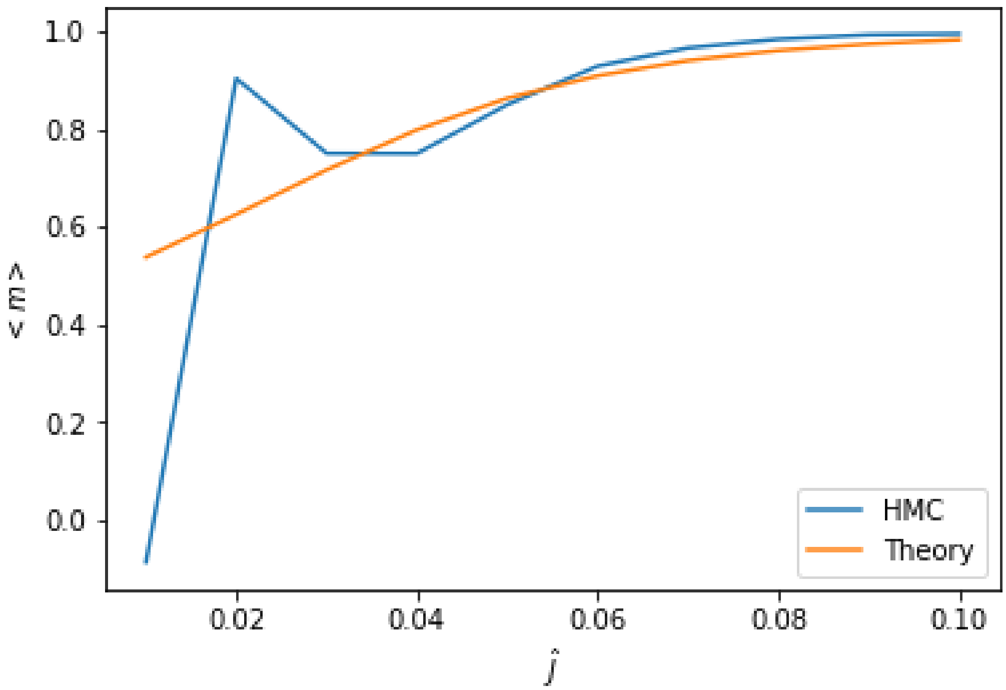
```
h = 0.5
N = 20

J_arr = np.linspace(0.2,2,10)/N

N_md = 100

N_s = 1000

m_arr = np.ones(len(J_arr))
e_arr = np.ones(len(J_arr))

for i in range(len(J_arr)):

    q_arr, p_arr, acc = HMC(N_s,N_md,J_arr[i],h,N)

    m = np.sum(np.tanh(b*h+q_arr))

    m = m/N_s

    e = np.sum(-q_arr**2/(2*N*J_arr[i]*b**2) - h*np.tanh(b*h+q_arr) + 1/(2*b*N))
    e = e/N_s

    m_arr[i] = m
    e_arr[i] = e
```

```
import math

# def nCr(n,r):
#     f = math.factorial
#     return f(n) / f(r) / f(n-r)

def nCr(n,r):
    f = math.factorial
    return f(n) / (f(r) * f(n-r))
```

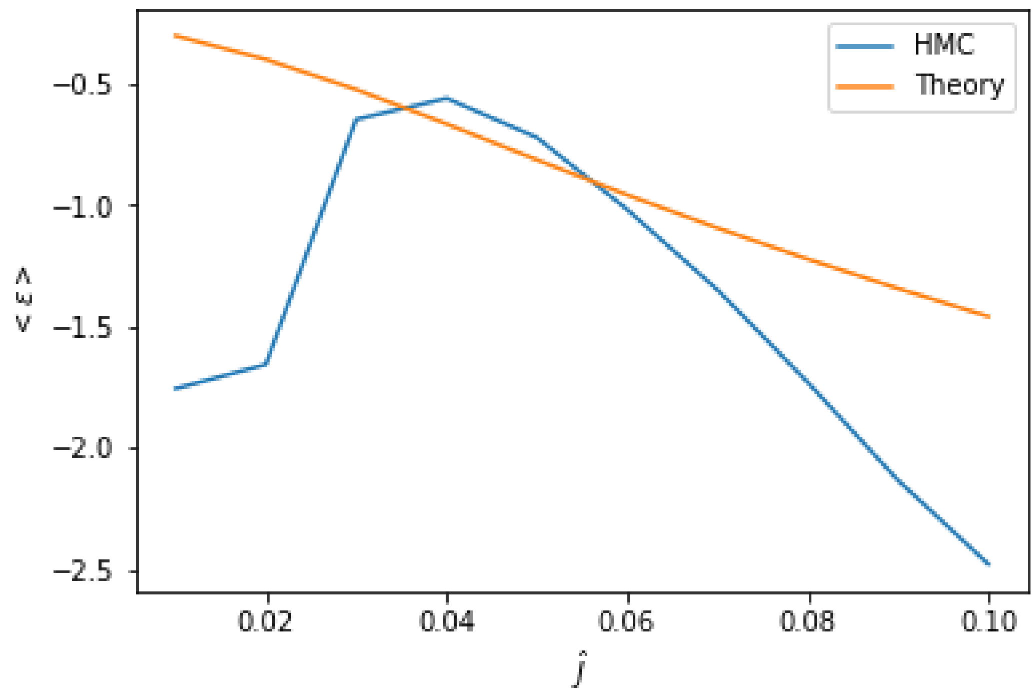
```
Z_th = np.ones(len(J_arr))
e_th = np.ones(len(J_arr))
```

```
m_th = np.ones(len(J_arr))

N = 20
b = 1

for i in range(len(J_arr)):

    Z_th[i] = 0
    e_th[i] = 0
    m_th[i] = 0

    for j in range(N+1):

        x = N - 2*j

        Z_tmp = nCr(N,j)*np.exp(0.5*b*J_arr[i]*x**2 + b*h*x)
        Z_th[i] = Z_th[i] + Z_tmp

        e_tmp = nCr(N,j)*np.exp(0.5*b*J_arr[i]*x**2 + b*h*x)*(0.
 ↪5*b*J_arr[i]*x**2 + b*h*x)
        e_th[i] = e_th[i] + e_tmp

        m_tmp = nCr(N,j)*np.exp(0.5*b*J_arr[i]*x**2 + b*h*x)*x
        m_th[i] = m_th[i] + m_tmp

m_th = m_th/(N*Z_th)

e_th = -e_th/(N*Z_th)
```

[ ]:
```
plt.plot(J_arr,m_arr,label='HMC')
plt.plot(J_arr,m_th,label='Theory')
plt.xlabel('$\hat{J}$')
plt.ylabel('$<m>$')
plt.legend();
```



[ ]:
```
plt.plot(J_arr,e_arr,label='HMC')
plt.plot(J_arr,e_th,label='Theory')
plt.xlabel('$\hat{J}$')
plt.ylabel('$<\epsilon>$')
plt.legend();
```



We see that while there are deviations for smaller values of $\hat{J}$, for larger values of $\hat{J}$ the HMC reaches the theoretical solution for $< m >$. For energy however, there are still sizable deviations between the theoretical and numerical results, observed by the different slope behaviour between the two.