In [19]:
```
import numpy as np
import random
import matplotlib.pyplot as plt
import pickle
from scipy.io import loadmat
```

In [20]:
```
def MetroSweep(lattice,J,B):#Sweeps through each element of the lattice
    for j in range(n):
        for i in range(n):
            #Energy of site i,j
            Ei = -J*lattice[(i)%n][(j)%n]*(lattice[(i+1)%n][(j)%n]+lattice[(i-
1)%n][(j)%n]+lattice[(i)%n][(j+1)%n]+lattice[(i)%n][(j-1)%n])
            #Energy of site i,j if flipped
            Ef = -Ei
            #Change in energy
            delE = Ef - Ei
            #if energy is favourable flip
            if delE <= 0:
                lattice[i%n][j%n] = -lattice[i%n][j%n]
            #if energy not favourable flip with probability
            elif random.random() < np.exp(-B*delE):
                    lattice[i%n][j%n] = -lattice[i%n][j%n]
    return lattice
```

In [24]:
```
#calculates the magnetization for different temperatures
def MagCalc(lattice):
    mag = np.sum(lattice)
    return mag
```

In [25]:
```
#calculates the energy
def Energy(lattice):
    energy = 0
    for i in range(n):
        for j in range(n):
            S = lattice[i,j]
            Ei = lattice[(i+1)%n][(j)%n]+lattice[(i-1)%n][(j)%n]+lattice[(i)%
n][(j+1)%n]+lattice[(i)%n][(j-1)%n]
            energy += -Ei*S
    return energy/4.
```

In [32]:
```
#Define variables
J = 1
k = 1
T = np.arange(0,4,0.1) #temperature values
n=50 #size of lattice
t = 100000 #MC time steps
E,M,C,X = np.zeros(T.size), np.zeros(T.size), np.zeros(T.size), np.zeros(T.siz
e)
n1, n2  = 1.0/(t*n*n), 1.0/(t*t*n*n)
MagTot, EngTot = [],[]
```

In [33]:
```
for i in range(T.size):
    E1 = M1 = E2 = M2 = 0
    #Generate random initial configuration
    lattice = 2*np.random.randint(0,2,(n,n))-1;
    B=1.0/T[i]; B2=B*B;
    MagAr, EngAr, tPlot =[],[],[]

    for j in range(t):
        MetroSweep(lattice,J,B)

    #sweep and calculate Energy and Mag at each site
    for j in range(t):
        MetroSweep(lattice,J,B)
        Ene = Energy(lattice)
        Mag = MagCalc(lattice)
        eng = Energy(lattice)
        tPlot.append(j)
        MagAr.append(Mag)
        EngAr.append(eng)

        E1 = E1 + Ene
        M1 = M1 + Mag
        M2 = M2 + Mag*Mag
        E2 = E2 + Ene*Ene

    MagTot.append(MagAr)
    EngTot.append(EngAr)
    E[i] = n1*E1
    M[i] = n1*M1
    C[i] = (n1*E2 - n2*E1*E1)*B2 #given by B^2(<Ei^2>-<Ei>^2)
    X[i] = (n1*M2 - n2*M1*M1)*B #given by B(<Si^2>-<Si>^2)
```

```
C:\Users\aorfi\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: RuntimeWa
rning: divide by zero encountered in double_scalars
  """
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: