HOMEWORK#7

-Shaan Chopra (2015090)

Question

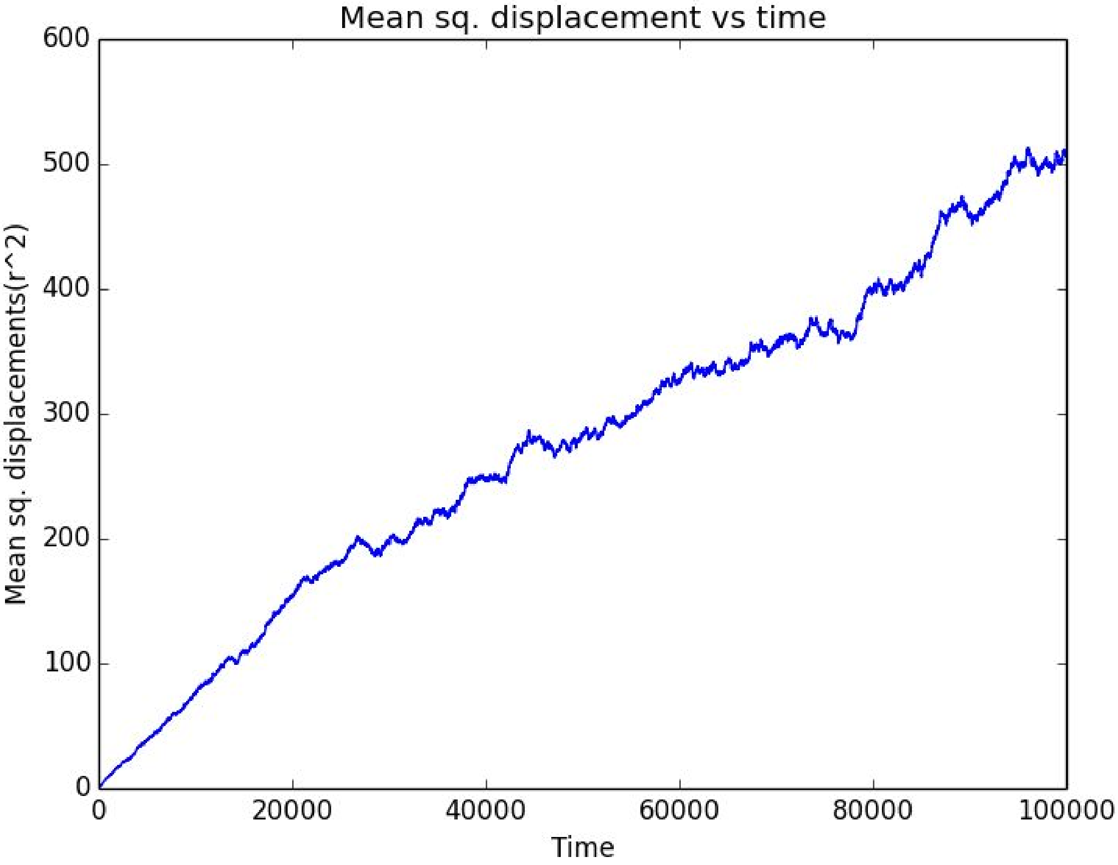
Use kinetic Monte Carlo method to simulate (use MATLAB) the random walk diffusion of molecules (random walkers) on a two-dimensional integer lattice. Random walkers are allowed only four moves: to any of the four nearest neighbor sites. The variable r denotes the displacement from the initial position of random walkers.
Statistical data analysis: compute < r > (mean displacement) and < r2 > (mean square displacement) from this simulation (average is computed over molecular trajectories).
You may simulate 100 molecules on a 50 × 50 lattice.

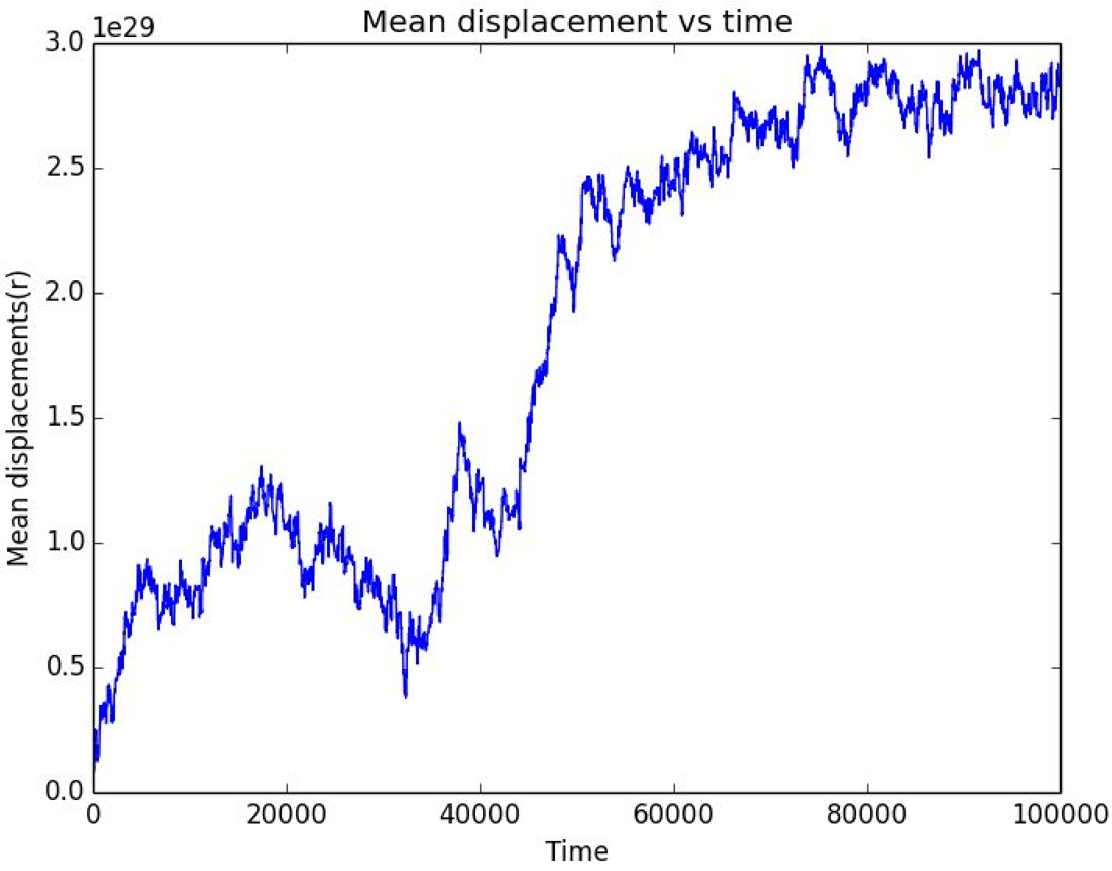
Answer

```
#MONTE CARLO SIMULATION
#Place 100 molecules in the possible positions of a 50x50 matrix
#find mean displacement and mean square displacement and plot the results
import matplotlib.pyplot as plt
import random, math

def mean_disp():
    total = 0.0
    for i in range(100):
        x_pos = pow(mat[i][0]-old[i][0],2)
        y_pos = pow(mat[i][1]-old[i][1],2)
        ans = float(x_pos)+float(y_pos)
        total += total + math.sqrt(ans)
    return total/100

def mean_sq_disp():
    total = 0.0
    for i in range(100):
        x_pos = pow(mat[i][0]-old[i][0],2)
        y_pos = pow(mat[i][1]-old[i][1],2)
        total += float(x_pos)+float(y_pos)
    return total/100

mat = [[0,0] for i in range(100)]
old = [[0,0] for i in range(100)]
pos = [[0 for i in range(50)] for j in range(50)]
points = [[i,j] for i in range(50) for j in range(50)]
arr=[]
arr1=[]

for i in range(100):
    rand = random.choice(points)
    mat[i]=rand
    points.remove(rand)
    pos[rand[0]][rand[1]]=i;
    old[i]=rand[:]

for i in range(100000):
    n=random.uniform(0,1)
    p=random.randint(0,99)
    if(n<=0.25):                #decrease x position (move left)
        a=mat[p]
        if(a[0]>0 and pos[a[0]-1][a[1]]==0):
            pos[a[0]][a[1]]=0
            pos[a[0]-1][a[1]]=p
            mat[p][0]-=1
    elif(n<=0.5):               #increase x position (move right)
        a=mat[p]
        if(a[0]<49 and pos[a[0]+1][a[1]]==0):
            pos[a[0]][a[1]]=0
            pos[a[0]+1][a[1]]=p
            mat[p][0]+=1
    elif(n<=0.75):              #decrease y position (move down)
        a=mat[p]
        if(a[1]>0 and pos[a[0]][a[1]-1]==0):
            pos[a[0]][a[1]]=0
            pos[a[0]][a[1]-1]=p
            mat[p][1]-=1
    else:                           #increase y position (move up)
        a=mat[p]
        if(a[1]<49 and pos[a[0]][a[1]+1]==0):
            pos[a[0]][a[1]]=0
            pos[a[0]][a[1]+1]=p
            mat[p][1]+=1

    if(i%10==0):                #after interval of 10, find mean displacement and mean sq displacement
        mean=mean_disp()
        arr.append(mean)
        mean_sq=mean_sq_disp()
        arr1.append(mean_sq)

avg_r=sum(arr)/float(len(arr))
avg_r2=sum(arr1)/float(len(arr1))
print(avg_r)
print(avg_r2)

plt.figure(0)
plt.plot([i for i in range(0,100000,10)],arr)
plt.xlabel("Time")
plt.ylabel("Mean displacements(r)")
plt.title("Mean displacement vs time")

plt.figure(1)
plt.plot([i for i in range(0,100000,10)],arr1)
plt.xlabel("Time")
plt.ylabel("Mean sq. displacements(r^2)")
plt.title("Mean sq. displacement vs time")

plt.show()
```

Output:

```
1.82911160123e+29       //Avg of mean displacement
274.745062              //Avg of mean square displacement
```


Mean sq. displacement vs time
Mean sq. displacements(r^2)
0
100
200
300
400
500
600
0
20000
40000
60000
80000
100000
Time


Mean displacement vs time
1e29
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Mean displacements(r)
0
20000
40000
60000
80000
100000
Time