P1: Brownian Motion

Edgar

February 2022

1 Introduction

For this homework, the goal was to make a script that could simulate Brownian motion by using a random number generator, observing the movement in different dimensions and printing a graph to showcase the data produced by the script, with 30 repetitions for runs of duration of 100, 1000 and 10000 steps.

2 Code

```
from random import random, randint, getrandbits
from math import fabs, sqrt
import matplotlib.pyplot as plt
import numpy as np
from time import time
```

Here we have the different libraries needed to produce the random numbers, the math equations, the plotted graph and the time to measure the duration of the process.

```
runs = 30
steps = [100, 1000, 10000]
results = []

for i in range(3):
    dur = steps[i]
    for dim in range(1, 6):
        mayores = []
        for rep in range(runs):
            before = time()*1000
            pos = [0] * dim
            mayor = 0
            for paso in range(dur):
                eje = randint(0, dim - 1)
                if pos[eje] > -100 and pos[eje] < 100:
```

```
                    if random() < 0.5:
                        pos[eje] += 1
                    else:
                        pos[eje] -= 1
                else:
                    if pos[eje] == -100:
                        pos[eje] += 1
                    if pos[eje] == 100:
                        pos[eje] -= 1
                mayor = max(mayor, sqrt(sum([p**2 for p in pos])))
            mayores.append(mayor)
            after = time()*1000
        results.append(mayores)
tiempo = after - before
print(tiempo)

walks_1 = results[0:5]
walks_2 = results[5:10]
walks_3 = results[10:15]
```

The code here is running the different graphs, of 100, 1000 and 10000 steps, 30 times, with a 5 dimensional movement.It was primarily referenced from the code by our teacher Dr. E. Shaeffer [2]. it also runs a calculation from the beginning of the entire process until the end, counting the time in seconds it took for the simulation to run.

```
ticks = ['1', '2', '3', '4', '5']

def set_box_color(bp, color):
    plt.setp(bp['boxes'], color=color)
    plt.setp(bp['whiskers'], color=color)
    plt.setp(bp['caps'], color=color)
    plt.setp(bp['medians'], color='orange')

plt.figure()

bpl = plt.boxplot(walks_1, positions=np.array(range(len(walks_1)))*5.0-1.0, sym='', widths=0
bpc = plt.boxplot(walks_2, positions=np.array(range(len(walks_2)))*5.0, sym='', widths=0.8)
bpr = plt.boxplot(walks_3, positions=np.array(range(len(walks_3)))*5.0+1.0, sym='', widths=0
set_box_color(bpl, '#D7191C')
set_box_color(bpc, '#2CB62C')
set_box_color(bpr, '#2C7BB6')

plt.plot([], c='#D7191C', label='100 steps')
plt.plot([], c='#2CB62C', label='1000 steps')
plt.plot([], c='#2C7BB6', label='10000 steps')
```

```
plt.legend()

plt.xticks(range(0, len(ticks)*5, 5), ticks)
plt.xlim(-2, len(ticks)*5)
plt.title('Euclidean distance')
plt.xlabel('Dimensions')
plt.ylabel('Max distance')
plt.tight_layout()
plt.savefig('EuclideanDistance.png')
plt.show()
```

This last section is used to draw the graph, name each axis, create the color info graphic, and select the respective color for each graph. It was taken from the code worked on by a classmate with the alias FeroxDeitas [1].

3 Results

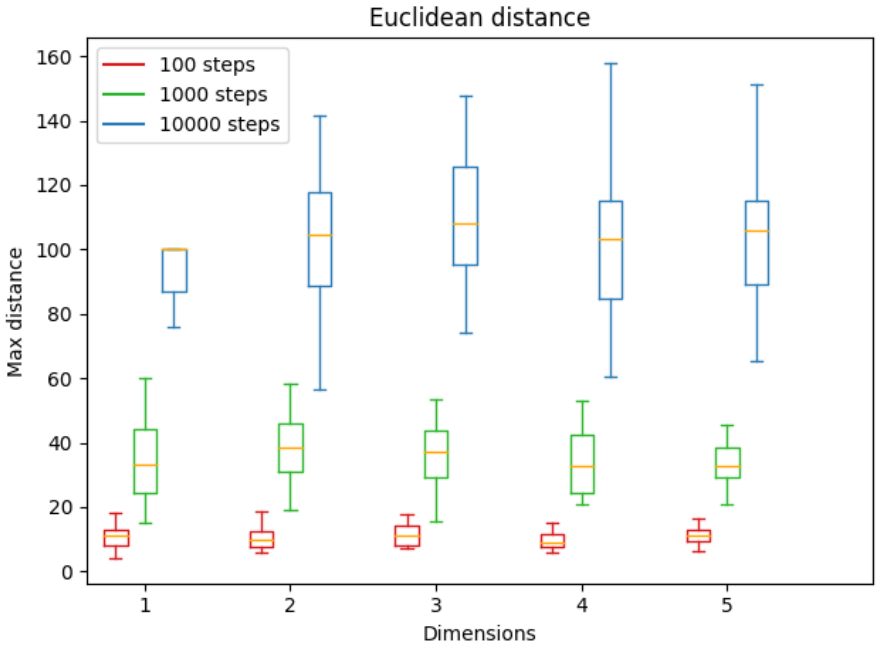

Figure 1: Plotted data

4 Interpretation

Here in figure 1 we can see how for the most part all the movement seems to be evenly spread out in the runs of 100 steps, however as the runs get longer the variation between dimension becomes larger. We can observe this gradual change in the 1000 steps runs, and so much more pronounced in the 10000 steps runs.

An interesting detail is how the first dimension in the 10000 step run differs so drastically statistically to all the others, this difference means that the range of movement in this dimension of space was a lot more limited than all the others, meaning that the particle was moving along this dimensional axis predominantly.

5 Conclusion

This tool is very helpful to represent the statistics of a moving particle and it can be applied on different scenarios with multiple dimensions. The data also showed how with higher duration of runs one is able to obtain better and more accurate data that better represents the movement of a particle, rather than the very limited sample size of the smaller duration runs.

References

[1] J.FeroxDeitas. *GitHub,pasos$_a$leatorios.py*. URL: `https://github.com/FeroxDeitas/Simulacion-Nano/blob/main/Tareas/P1/pasos_aleatorios.py`.

[2] E. Schaeffer. *GitHub,sinpar.py*. URL: `https://github.com/satuelisa/Simulation/blob/master/BrownianMotion/sinpar.py`.