Entropic Forces In Brownian Motion

James Saslow

March 2021

1 Introduction

This report investigates the fundamental characteristics of Brownian motion as well as the introduction of the entropic force highlighted in ”Entropic Forces in Brownian Motion”, by Nico Roos.

The phenomenon of Brownian motion occurs when a particle experiences collisions from nearby particles in its environment. When a particle undergoes a collision, it experiences a change in its momentum, which by definition, is a force. So, when several collisions occur over a time interval on a particle (such as the big yellow particle in Figure 1), it undergoes random, sporadic motion, known as Brownian motion.

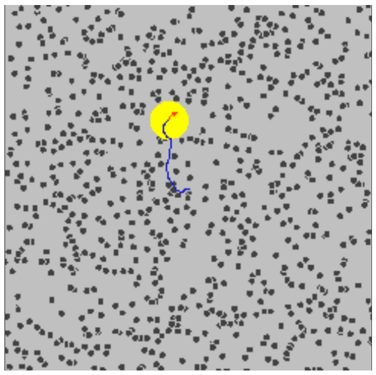

Figure 1: Brownian Motion of a particle in a gaseous medium [2]

2 Random Walks

2.1 Introduction

It turns out, a limiting case of Brownian motion is a random walk. This limiting case is made possible when there is zero drift. A random walk, also known as a drunken walk, is a stochastic process in which you can imagine a drunk person, initially starting at a bar, making equally sized steps in random directions along the basis vectors of a given coordinate system.

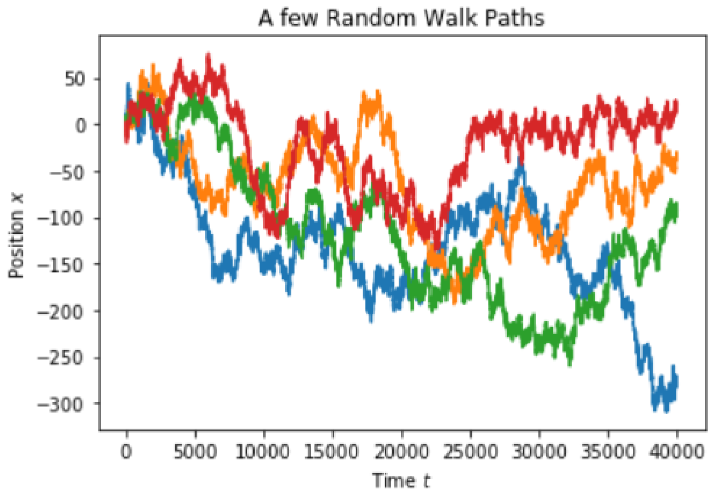

Figure 2: A few Generated 1D Random Walks

Examining one random walk by itself does not yield much useful information. However, if we consider a large ensemble of random walks where each walk starts at the origin, we can formulate a probability distribution that describes the most probable location where the walk will end up. This probability distribution will evolve with time. We know this because as time increases, there will be more macrostates available for the drunk people to walk to. Since all the drunk people start at the bar, We will expect our distribution to start out as a delta function centered at the origin, and continuously flatten out into a bell curve as time progresses and drunk people scatter from the origin. This process of dispersing away from the origin in this manner is known as diffusion. Einstein's Diffusion formula explains that the RMS displacement of diffused particles in 1D is proportional to the square root of elapsed time since particles were released from the origin.

$$< x >^2 = 2Dt \tag{1}$$

where D is the diffusion constant. A large diffusion constant means particles will diffuse faster from some location compared to a family of particles governed by a smaller diffusion constant.

2.2 Numerical Analysis

We can model a large number of 1D random walks by using a Markov Chain simulation in Python. We can implement this numerically by randomly sampling a -1 or $+1$ every time step to tell the drunk person which direction to walk. When we do this, we find that the variance of the walk positions and RMS displacement both increase linearly with time.

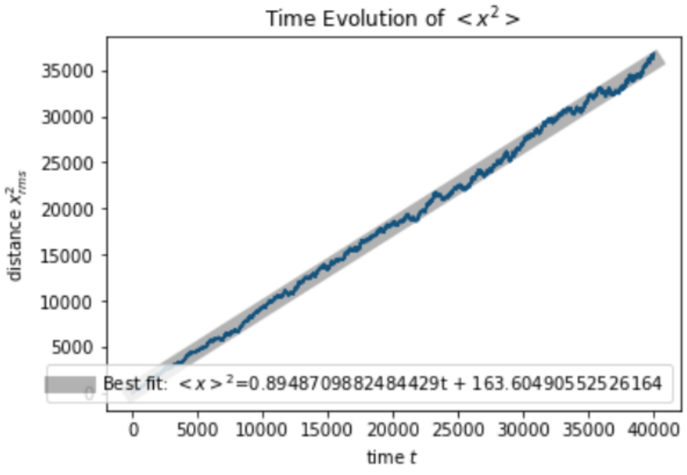

Figure 3: RMS Displacement Squared vs time

This experimental result is just predicted as predicted by Einstein's diffusion model (See Figure 3).

2.3 Probability Distribution

Since standard deviation is a state variable, it would be a reasonable assumption to model our distribution of walk positions with a normal curve.

Assuming a normal distribution model becomes an even more sound idea when we examine histograms of random walk positions at different time steps from the Markov Chain code.

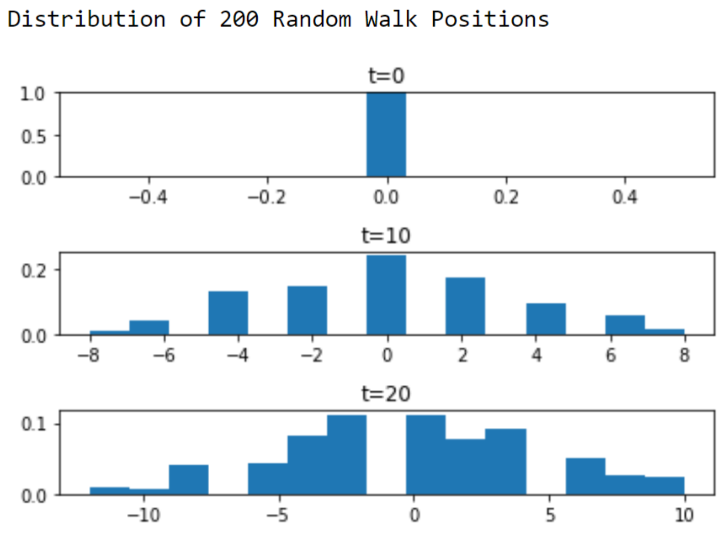

Figure 4: Time Evolution of Probability distribution describing Random Walk Positions

The probability distribution of random walk positions is given by the normal curve with a time dependent variance.

$$f(x,t) = \frac{1}{\sqrt{2\pi[\sigma(t)]^2}} e^{-\frac{x^2}{2[\sigma(t)]^2}} \tag{2}$$

Where $[\sigma(t)]^2 \approx < x^2 >= 2Dt$ by the Einstein diffusion relation. And $\mu = 0$ on average.

Interestingly enough, if we take position and time derivatives of $f(x,t)$, we find that $\frac{\partial f(x,t)}{\partial t} = D\frac{\partial^2 f(x,t)}{\partial t^2}$, which is the Diffusion Equation in one dimension. Roos uses the 3D diffusion equation in radial coordinates as a method to acquire the probability distribution of Brownian diffusion starting from the center of a sphere.

The diffusion equation:

$$\frac{\partial f(\mathbf{r},t)}{\partial t} = D\nabla^2 f(\mathbf{r},t) \tag{3}$$

Of course, the above derivation of the probability distribution isn't as thorough since it is extrapolated from a central limit theorem process by experimental means. Roos covers a more natural, analytical approach to this random walk problem in 3D in his paper by maximizing entropy. However, I believe it is important to shed light on this Markov chain perspective since it provides an easier visualization of why these random walks are governed by a normal distribution.

2.4 Entropic Forces

Let's once again consider the RMS Displacement of these random walks. As shown both by Einstein's diffusion relation, and from the Markov chain data (Figure 5), $< x > \propto t^{1/2}$.

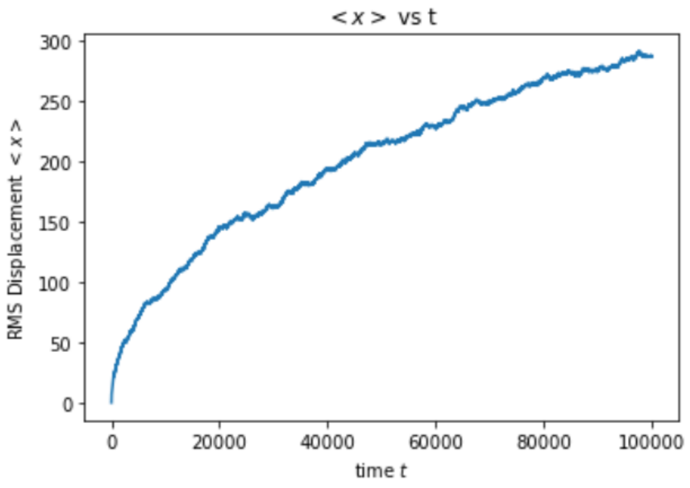

Figure 5: RMS Displacement of several walks vs time

Since this position vs time graph has concavity, we can conclude that the RMS position of these random walks have nonzero acceleration. Therefore, by Newton's Second Law, there must be a force responsible for changing the RMS velocity of these diffusive particles. This force is called the Entropic Force. The Entropic Force is a macroscopic force responsible for diffusing these particles in Brownian motion in order to maximize the system's entropy after a long time has passed.

We can actually obtain an analytical expression for the entropic force in the case of a 1D random walk by taking time derivatives of $< x^2 >= 2Dt$. We find that

$$< F >= -\frac{1}{4} m\sqrt{2D} t^{-3/2} = -mD^2 / < x^3 > \tag{4}$$

Our Markov chain data shows the Entropic force driving the entropy of the system to a maximum (Figure 6). Roos uses this "increasing entropy" characteristic of the Entropic force to his advantage to derive the probability distribution that describes Brownian motion in a spherically symmetric environment by maximizing the entropy of the system.

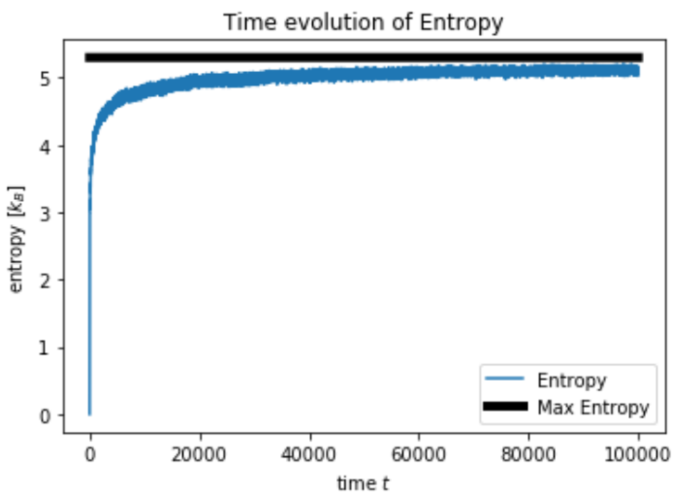

Figure 6: Entropy as a function of time for 200 Random Walks

3 Random Walk From the Center of A Sphere

Roos' paper is interested in diffusion in 3D since it gives a more real life description of Brownian motion.

3.1 Probability

Roos divides a sphere into concentric shells and calculates the probability of a random walk ending up inside a volume of some shell. For a large number of steps or for a long time since the particle's release, he finds the probability distribution in radial coordinates to be

$$p(r,t) = \frac{4}{\sqrt{\pi}}\sigma^{-3}r^2 e^{-r^2/\sigma^2}dr \tag{5}$$

where $\frac{1}{2}[\sigma(t)]^2 = \frac{1}{3} < r^2 >=< x^2 >$. He notes that the same solution can also be obtained by solving the diffusion partial differential equation in radial coordinates.

3.2 Entropic Forces

Roos states Neuman's Entropic Force in section III to be

$$< F_r >= \frac{2k_BT}{< r >} \tag{6}$$

This means that Brownian particles released from the center of a sphere appear to be pulled away by some force inversely proportional to the distance r.

References

[1] Roos, Nico. "Entropic Forces in Brownian Motion." American Journal of Physics, vol. 82, no. 12, 2014, pp. 1161–1166., doi:10.1119/1.4894381.
`https://arxiv.org/ftp/arxiv/papers/1310/1310.4139.pdf`

[2] "Brownian Motion." Wikipedia, Wikimedia Foundation, 11 Mar. 2021,
`https://en.wikipedia.org/wiki/Brownian_motion`

4 Appendix

Markov Chain Python Code for 1D Random Walks

```
import numpy as np
import pylab as plt

# Parameters
t = 100000 # Elapsed Time
num = 200 # Number of walks
step_size = 1 # How large steps are in our walk

steps = np.random.randint(0,2,t*num)
for i in range(t*num):
    if steps[i] == 0:
        steps[i] = -1

steps = step_size*steps.reshape(num,t)

X = np.zeros((num,t))
for i in range(num):
    for j in range(1,t):
        X[i,j] = X[i,j-1] + steps[i,j-1]

# Plotting some of these random walks
i=0
while i<num:
    plt.plot(X[i])
    i+=1
    if i==4: # Max Number of random walks displayed
        break

plt.title('A few Random Walk Paths')
plt.xlabel('Time $t$')
plt.ylabel('Position $x$')
plt.show()
```

```
# Plotting time evolution of histograms
print('Distribution of '+str(num)+ ' Random Walk Positions')
fig, axs = plt.subplots(3, sharex=False, sharey=False)
axs[0].hist(X[:,0], density=True, bins = 15)
axs[0].set_title('t=0')
axs[0].set_ylim([0, 1])
axs[1].hist(X[:,10],density = True, bins = 15)
axs[1].set_title('t=10')
axs[2].hist(X[:,20], density = True, bins = 15)
axs[2].set_title('t=20')
plt.tight_layout()
plt.show()

# Time evolution of mean and standard deviation
mu = np.zeros(t)
var = np.zeros(t)
xrms = np.zeros(t)

# Calculating mean and standard deviation at each time step
for j in range(t):
    mu[j] = np.mean(X[:,j])
    var[j] = np.var(X[:,j])
    xrms[j] = np.sqrt(np.mean(X[:,j]**2))

stdev = np.sqrt(var)

# Linear regression on the RMS Disp.
time = np.arange(0,t)

m_xsq,b_xsq = plt.polyfit(time,xrms**2,1)

plt.plot(time, xrms**2)
plt.plot(time, m_xsq*time+b_xsq,color = 'black',alpha=0.3,lw=10,\
         label = 'Best fit: $<x>^{2}$='+str(m_xsq)+'t + '+str(b_xsq))
plt.title('Time Evolution of $<x^{2}>$')
plt.xlabel('time $t$')
plt.ylabel('distance $x_{rms}^{2}$')
plt.legend(loc = 'lower right')
plt.show()

plt.plot(time, xrms)
plt.title('$<x>$ vs t')
plt.xlabel('time $t$')
plt.ylabel('RMS Displacement $<x>$')
plt.show()
plt.plot(time,mu)
plt.title('Time evolution of Mean')
plt.xlabel('time $t$')
plt.ylabel('mean')
plt.show()

# Diffusion Coefficient
D = m_xsq/2
print('Diffusion Coefficient = D = ', D)

# Counting Microstates
microstates = np.zeros(t)
for j in range(t):
    X_s = np.sort(X[:,j])
    count = [X_s[0]]
    for i in range(1,len(X_s)):
        if X_s[i] != count[-1]:
            count.append(X_s[i])
    microstates[j] = len(count)

#Multiplicity and entropy
plt.plot(time, microstates)
plt.title('Time Evolution of Multiplicity of Several Random Walks')
plt.xlabel('time $t$')
plt.ylabel('Multiplicity')
plt.show()

plt.plot(time, np.log(microstates), label = 'Entropy')
plt.plot(time, np.ones(t) * np.log(num), \
color = 'black', lw = 5, label = 'Max Entropy' )
plt.title('Time evolution of Entropy')
plt.xlabel('time $t$')
plt.ylabel('entropy [$k_{B}$]')
plt.legend(loc = 'lower right')
plt.show()

print('Maximum entropy = ', np.log(num), 'kB')
```