Caitlyn Jones

Prof. Yangyang Wang

MATH 40A: Intro to Applied Math

25 November 2024

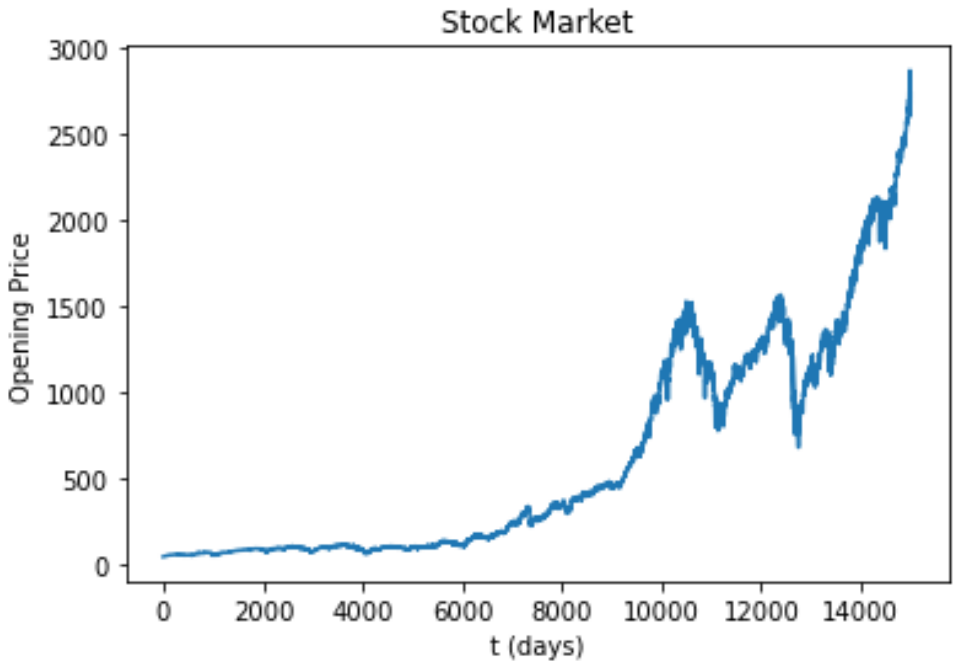
Homework 5

1.

a. To plot the data of the opening price, I input that the `column_index = 1`in the given code.

```
#convert stock data into floats
column_index = 1 #put column index to plot here
stock_data = [float(x) for x in s[column_index][-N:] ]
```

Here is the resulting graph.

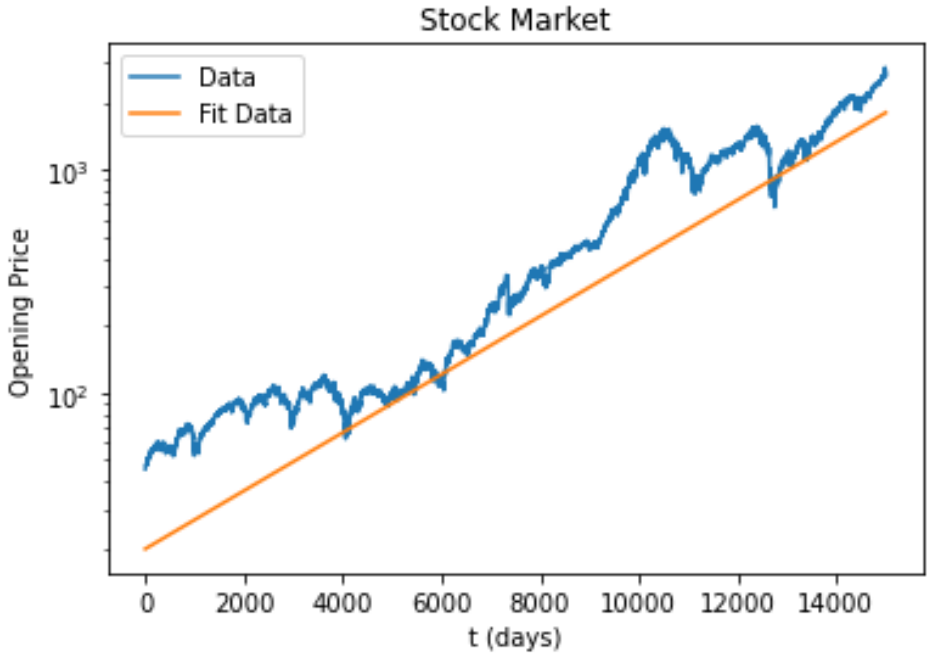


b. Then, I made a semi-log plot. To do so, I used `plt.semilogy` and the function S(t) in order to prove that the data fits to a straight line.

```
time = np.arange(N)
S_t = 20*np.exp(3e-4*time)

plt.figure()
plt.plot(time,stock_data, label = "Data")
plt.semilogy(time, S_t, label = "Fit Data")
plt.xlabel('t (days)')
plt.ylabel('Opening Price')
plt.title("Stock Market")
plt.legend()
plt.show()
```

And here at the results, which show that the data fits to the straight line.



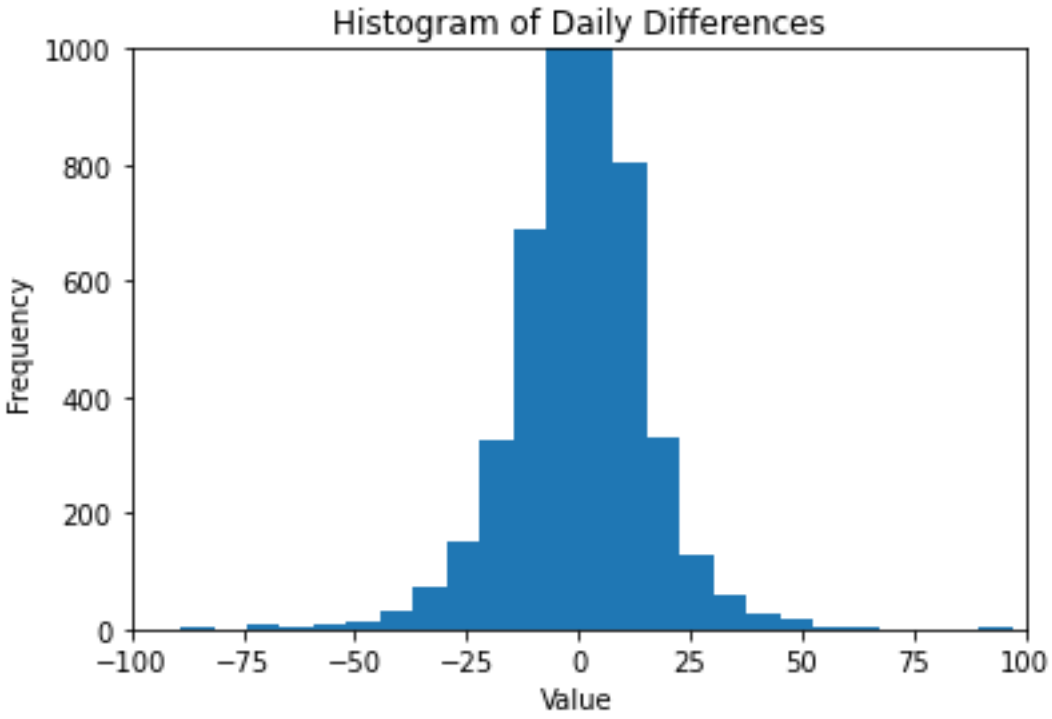
2.

a. First, I created a list to store the opening prices of the data.

```
opening_prices = np.array(s[1], dtype=float)
```

Then, I calculated the daily differences and made a histogram to show the distribution.

```
differences = np.diff(opening_prices)
plt.hist(differences, bins = 30)
plt.xlim(-100,100)
plt.ylim(0,1000)
plt.title("Histogram of Daily Differences")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```



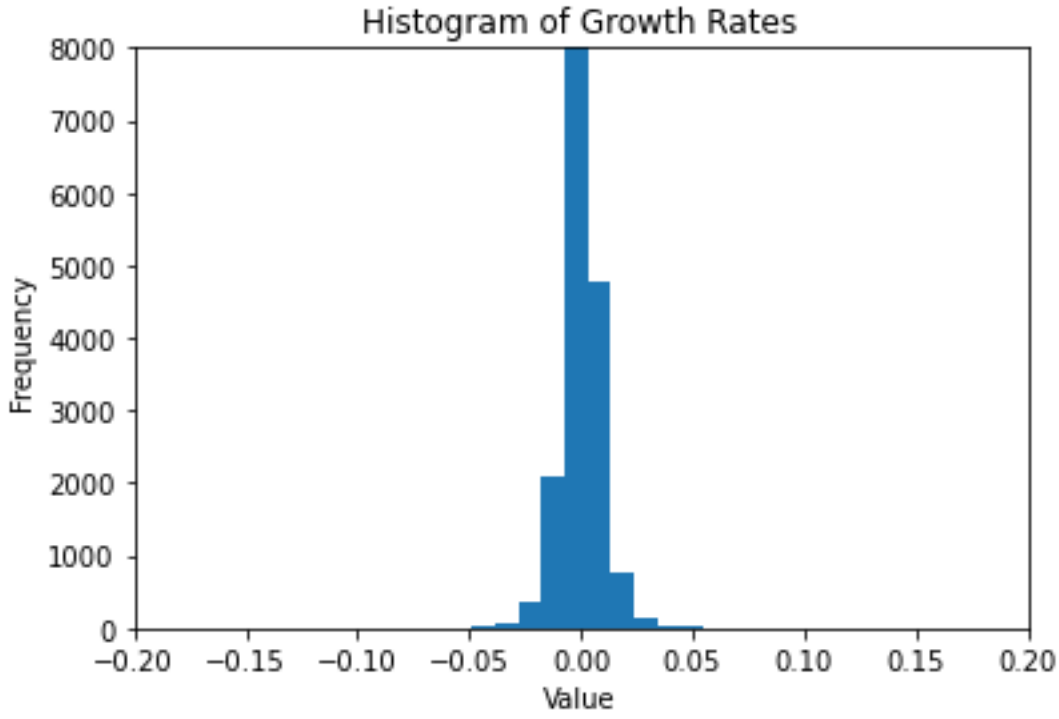
As you can see, the highest frequency of the difference in days was around zero. Since the frequency at zero keeps increasing, I changed the limits of the x and y axes in order to better see the outliers in the data. As you can see, there are a few differences that are rather large, which we can predict from the steep changes in opening prices from the graph in question 1.

b. Next, I calculated the rate at which the stock prices grow.

```
opening_prices = np.array(s[1], dtype=float)
growth_rates = np.diff(opening_prices)/opening_prices[:-1]
```

Then I made a histogram to display this difference.

```
plt.hist(growth_rates, bins = 30)
plt.xlim(-.2,.2)
plt.ylim(0,8000)
plt.title("Histogram of Growth Rates")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```



Lastly, I calculated the mean.

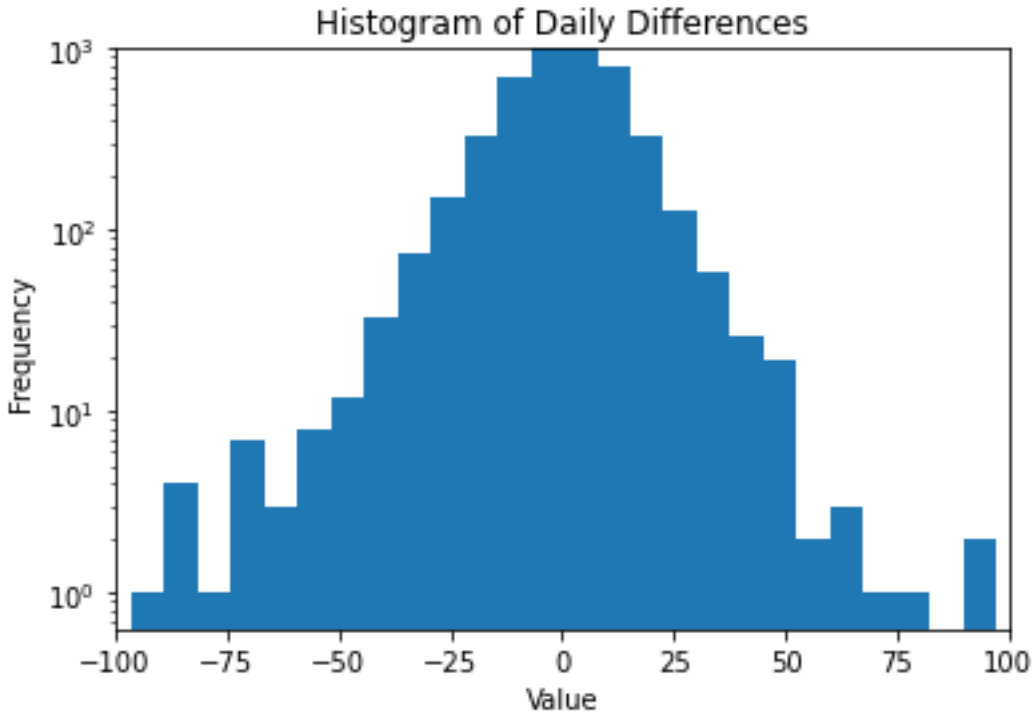
```
mean_growth_rate = np.mean(growth_rates)
print("The mean is", mean_growth_rate)
```

```
The mean is 0.0003421084295267601
```

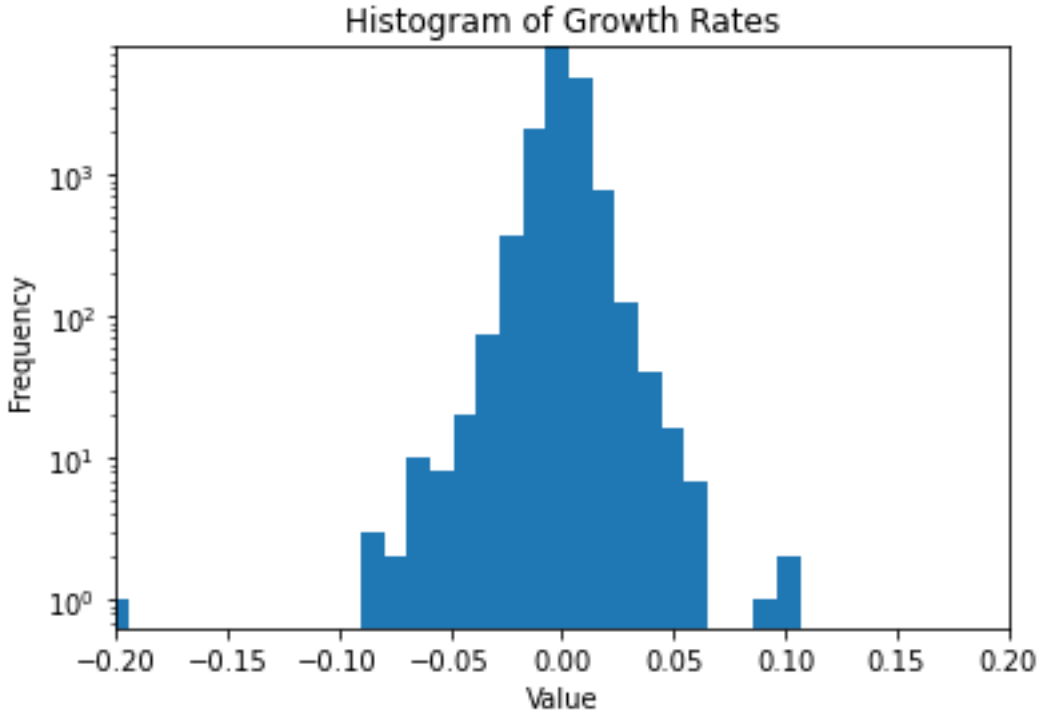
This means that on average, the stocks grow about .03% per day.

c. First, I changed the scale of the vertical y axis to be log to get the following graphs derived from part (a) and (b).

```
opening_prices = np.array(s[1], dtype=float)
differences = np.diff(opening_prices)
plt.hist(differences, bins = 30, log = True)
plt.xlim(-100,100)
plt.ylim(0,1000)
plt.title("Histogram of Daily Differences")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```



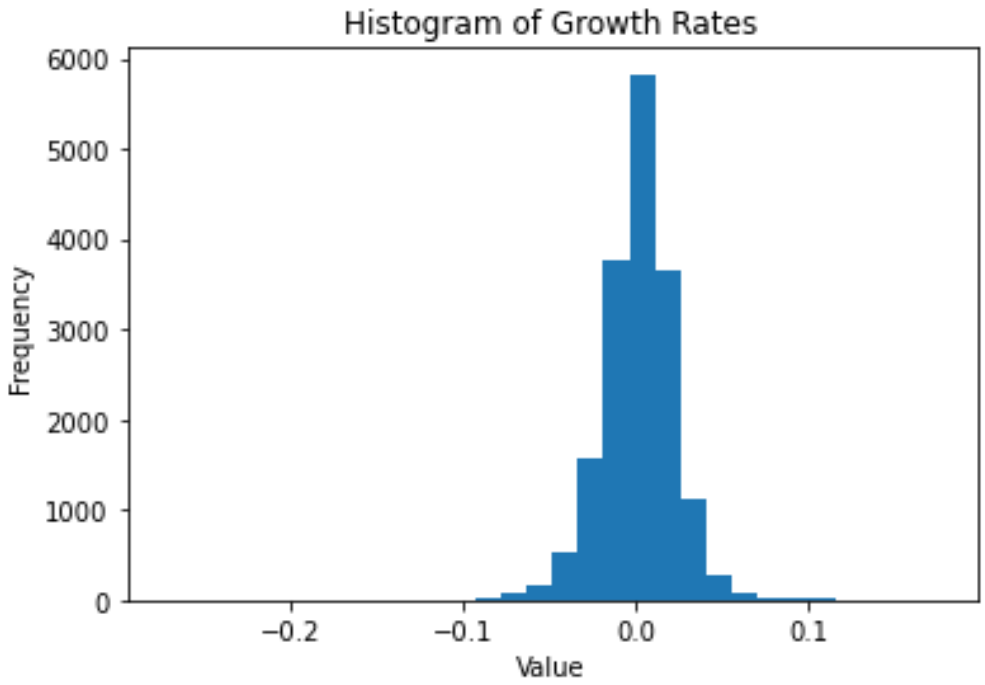
```
opening_prices = np.array(s[1], dtype=float)
growth_rates = np.diff(opening_prices)/opening_prices[:-1]
plt.hist(growth_rates, bins = 30, log = True)
plt.xlim(-.2,.2)
plt.ylim(0,8000)
plt.title("Histogram of Growth Rates")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```



The graphs are a good fit because the order of magnitude present in them is incredibly similar to that of the graphs in part (a) and (b).

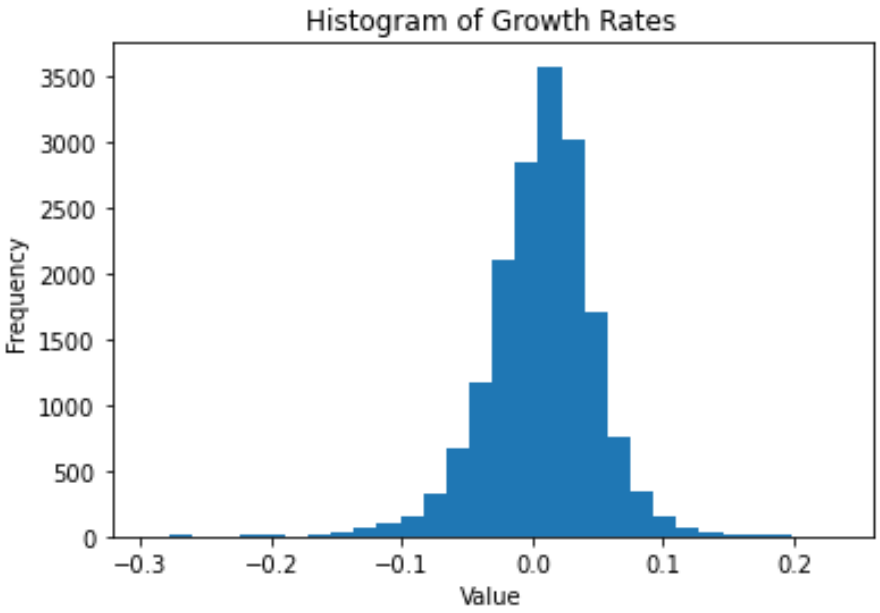
d. For k = 5, we get the following graph.

```
opening_prices = np.array(s[1], dtype=float)
growth_rates5 = [(opening_prices[i+5]- opening_prices[i])/opening_prices[i] for i in range(len(opening_prices) - 5)]
plt.hist(growth_rates5, bins = 30)
plt.title("Histogram of Growth Rates")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```




```
opening_prices = np.array(s[1], dtype=float)
growth_rates20 = [(opening_prices[i+20]- opening_prices[i])/opening_prices[i] for i in range(len(opening_prices) - 20)]
plt.hist(growth_rates20, bins = 30)
plt.title("Histogram of Growth Rates")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```

And for k = 20, we get the following graph.



The standard deviation for step size 5 and step size 20 are the following.

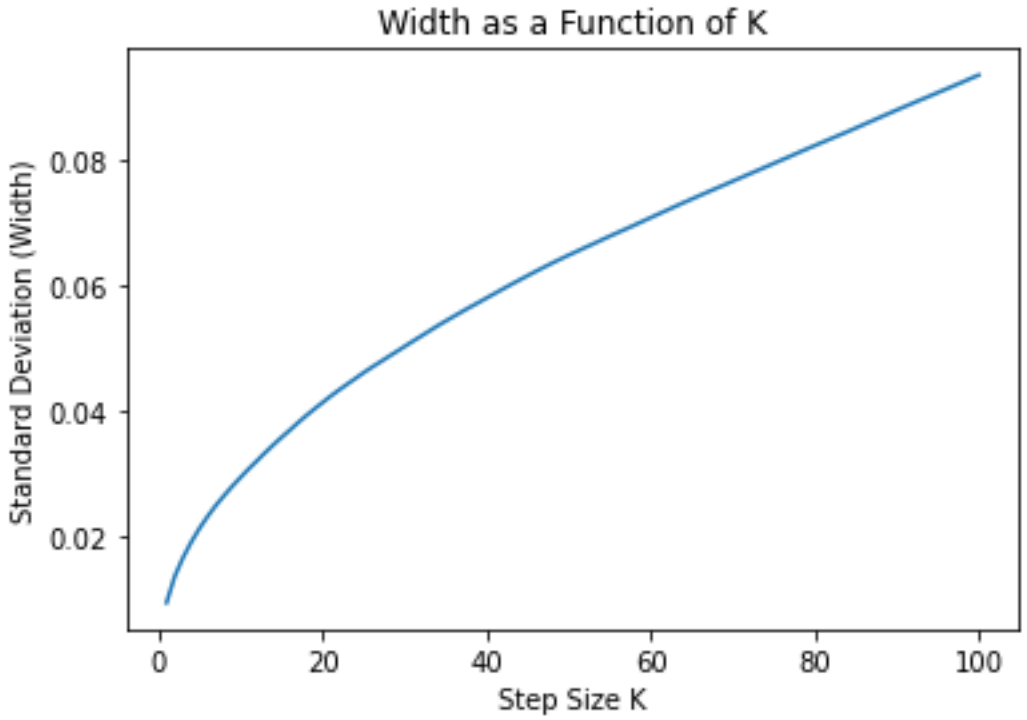
```
print("Standard deviation for step size 5:",np.std(growth_rates5))
print("Standard deviation for step size 20:", np.std(growth_rates20))
```

```
Standard deviation for step size 5: 0.02126017561502398
Standard deviation for step size 20: 0.0413627923700834
```

Then, I calculated the width of each value from 1 to 100 and plotted the results.

```
k_vals = np.arange(1, 101)
sigma_t = []
for k in range(1,101):
    growth_rates_k = [(opening_prices[i+k]-opening_prices[i])/opening_prices[i] for i in range(len(opening_prices) - k)]
    sigma_t.append(np.std(growth_rates_k))


plt.plot(k_vals, sigma_t)
plt.xlabel("Step Size K")
plt.ylabel("Standard Deviation (Width)")
plt.title("Width as a Function of K")
plt.show()
```



As you can see, the width does increase very consistently with our random walks because the width is proportional to k and the above graph reflects that.

Lastly, I calculated the diffusion constant D.

```
sigma2 = np.array(sigma_t)**2
diffusion_constant = (sigma2)/ (2*k_vals)
print("D =",np.mean(diffusion_constant))
```

```
D = 4.266601328199795e-05
```