

Carlo_Doc

March 31, 2022

0.1 This Document will describe and demonstrate the way that the Python code works.

0.1.1 Dependencies:

- [Python] (<https://www.python.org/>)
- [Jupyter Notebook] (<https://jupyter.org/>)
- [Numpy] (<https://www.numpy.org/>)
- [Matplotlib] (<https://matplotlib.org/>)
- [Pandas] (<https://pandas.pydata.org/>)
- [Math] (<https://docs.python.org/3/library/math.html>)

0.1.2 First the random stock prices are generated.

First the Class that will be used to generate the random stock prices is defined. This Code is shown below:

```
[9]: # Import all the necessary packages

import matplotlib.pyplot as plt
import numpy as np
import math
import pandas as pd

class GBM:

    def simulate(self):
        while(self.total_time > 0):
            # ***
            dS = self.current_price*self.drift*self.time_period + self.
            ↪current_price*self.volatility*np.random.normal(0, math.sqrt(self.
            ↪time_period))
            self.prices.append(self.current_price + dS)
            self.current_price += dS
            self.total_time -= self.time_period

    def __init__(self, initial_price, drift, volatility, time_period,
            ↪total_time):
```

```

# Initialize fields
self.initial_price = initial_price
self.current_price = initial_price
self.drift = drift
self.volatility = volatility
self.time_period = time_period
self.total_time = total_time
self.prices = []
# Simulate the diffusion process
self.simulate() # Simulate the diffusion proces

```

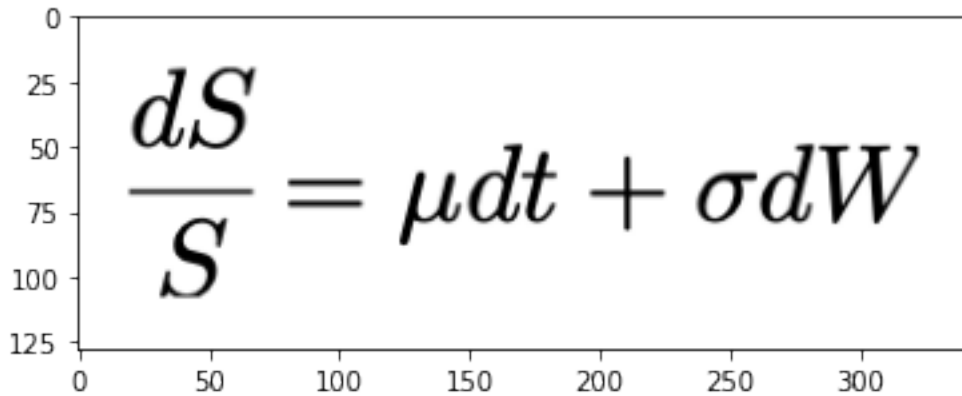
[10]: # Displays the Equation used for the Generating Brownian Motion

```

import matplotlib.image as mpimg

img = mpimg.imread('GBM_Equation.png')
imgplot = plt.imshow(img)
plt.show()

```



$$\frac{dS}{S} = \mu dt + \sigma dW$$

0.1.3 This Class is defined so that it can be called later in the code to generate the random stock prices.

Next we must define the parameters that will be used to generate the random stock prices.

Variables:

- Simulations: [], This creates the list that will contain the random stock prices. It is empty.
- Number of Simulations: "n", This is the number of random stock prices that will be generated.
- Total Time: "total_time", This is the total time that the random stock prices will be generated.
- Initial Stock Price: "initial_price", This is the initial stock price that will be used to generate the random stock prices.

- Drift: "drift", This is the drift that will be used to generate the random stock prices.
- Time Step: "time_period", This is the time between each random stock price. 1 = 1 year. 1/12
- Volatility: "volatility", This is the volatility that will be used to generate the random stock prices.

These are all of the parameters that will be used to generate the random stock prices. Later in the code there will be more parameters will be defined to generate a fair price for the stock.

The Class is called and the parameters are passed to the Class. The Class will then generate the random stock prices and fill the empty list with the random stock prices.

0.1.4 Code:

```
[11]: simulations = []
      n = 500
      initial_price = 100
      drift = .02
      volatility = .25
      time_period = 1/365 # Daily
      total_time = 1

      for i in range(0, n):
          simulations.append(GBM(initial_price, drift, volatility, time_period,
                                  total_time))
```

Finally To complete the Geometric Brownian Motion process, the graph of the random stock prices is generated.

0.1.5 Code:

```
[12]: for sim in simulations:
      plt.plot(np.arange(0, len(sim.prices)), sim.prices)
      plt.title('Simulation of GBM')
      plt.xlabel('Time (Days)')
      plt.ylabel('Price ($)')
```

