**Maths Assigment Q-4**

Question: (Computer Experiment in python: Simulating the Stock Market.) Let Y1,Y2,... be independent random variables such that P(Yi = 1) = P(Yi = −1) = 1/2. Let Xn = ni=1 Yi. Think of Yi = 1 as “the stock price increased by one dollar”, Yi = −1 as “the stock price decreased by one dollar”, and Xn as the value of the stock on day n. (a) Find E(Xn) and V(Xn). (b) Simulate Xn and plot Xn versus n for n = 1,2,...,10,000. Repeat the whole simulation several times. Notice two things. First, it’s easy to “see” patterns in the sequence even though it is random. Second,you will find that the four runs look very different even though they were generated the same way. How do the calculations in (a) explain the second observation?

A paper with writing on it

Description automatically generated

We are going to code something that generates four simulations of stock prices over 10,000 days and plots the results. Even though the simulations are generated using the same method, they can look different due to the inherent randomness in the process. Lets try to recreate this with python.

1. **Import Libraries**

* numpy is a library for numerical operations, and matplotlib.pyplot is used for plotting graphs.

import numpy as np

import matplotlib.pyplot as plt

1. **Let’s define the function simulate\_stock:**

The function simulate\_stock takes two parameters: n\_simulations (number of simulations) and n\_days (number of days for each simulation).

It initializes an empty list called results to store the simulated stock prices for each simulation.

It then runs a loop n\_simulations times, where for each simulation, it generates random values of (-1 or 1) for each day using np.random.choice.

The cumulative sum of these random values is calculated using np.cumsum, representing the cumulative stock prices for each day.

The resulting list of cumulative prices is appended to the results list.

The function returns the list of simulated stock prices for all simulations.

def simulate\_stock(n\_simulations, n\_days):

results = []

for \_ in range(n\_simulations):

stock\_prices = np.random.choice([-1, 1], size=n\_days)

cumulative\_prices = np.cumsum(stock\_prices)

results.append(cumulative\_prices)

return results

1. **Set Simulation Parameters:**

These variables specify the number of simulations (n\_simulations) and the number of days in each simulation (n\_days).

n\_simulations = 5

n\_days = 10000

1. The simulate\_stock function is called with the specified parameters, and the results are stored in the simulation\_results variable.

simulation\_results = simulate\_stock(n\_simulations, n\_days)

1. The code then iterates over the simulation results using a for loop. For each simulation, it prints the simulation number and the corresponding simulated stock prices.

It also plots the cumulative stock prices for each simulation using plt.plot. The label for each plot is set to indicate the simulation number.

The title, xlabel, and ylabel for the plot are set for better understanding, and the legend is displayed to differentiate between different simulations.

Finally, plt.show() is called to display the plot.

# Plot the results

for i, result in enumerate(simulation\_results):

print(f"Simulation {i}: {result} ")

plt.plot(result, label=f'Simulation {i+1}')

plt.title('Simulation of Stock Prices')

plt.xlabel('Days')

plt.ylabel('Stock Price')

plt.legend()

plt.show()

Output:

A screenshot of a computer program

Description automatically generated

Graph:

A graph of different colored lines

Description automatically generated