



## Experiment 8

Name: Umang Kirit Lodaya

SAP ID: 60009200032

Batch: D11

**Aim: Analyze various Interactive plots of Monte Carlo Simulations of CPPI and GBM.**

### Objective:

- Learn about CPPI and drawdown constraints.
- Implement CPPI and drawdown constraints in Python.
- Test the CPPI and drawdown constraints on a specified data source.

### Theory:

**Monte Carlo Simulation:** A Monte Carlo simulation is a mathematical technique that uses repeated random sampling to approximate the probability distribution of a random variable. This can be used to estimate the value of a financial asset, the risk of a financial portfolio, or the performance of a trading strategy.

The basic idea behind a Monte Carlo simulation is to generate a large number of random samples from the probability distribution of the random variable. These samples are then used to estimate the expected value, variance, and other statistical properties of the random variable.

For example, to estimate the value of a financial asset, we could generate a large number of random samples from the asset's price distribution. These samples would then be used to estimate the asset's expected price, its volatility, and its risk.

**CPPI:** CPPI stands for Constant Proportion Portfolio Insurance. It is a dynamic asset allocation strategy that aims to protect investors from downside risk while still allowing them to participate in market upside.

The basic idea behind CPPI is to maintain a constant proportion of the portfolio in risky assets. This proportion is called the floor value. The remaining part of the portfolio is invested in risk-free assets.

As the value of the risky assets rises, the floor value is increased. This means that more of the portfolio is invested in risky assets. As the value of the risky assets falls, the floor value is decreased. This means that less of the portfolio is invested in risky assets.

The CPPI strategy can be implemented using a Monte Carlo simulation. The simulation would start with a certain initial portfolio value and a certain floor value. The simulation would then generate a large number of random samples from the asset's price distribution. These samples would then be used to track the value of the portfolio over time.



**GBM:** GBM stands for geometric Brownian motion. It is a stochastic process that is used to model the evolution of asset prices over time.

The GBM process is characterized by two parameters: the drift rate and the volatility. The drift rate is the expected rate of return of the asset. The volatility is a measure of the uncertainty of the asset's returns.

The GBM process can be used to model the price of a financial asset over time. The price of the asset is modeled as a random walk with drift. The drift rate represents the expected rate of return of the asset, and the volatility represents the uncertainty of the asset's returns.

The GBM process can be implemented using a Monte Carlo simulation. The simulation would start with a certain initial asset price and a certain drift rate and volatility. The simulation would then generate a large number of random samples from the GBM process. These samples would then be used to track the price of the asset over time.

#### Formulae

Here are some of the formulae that are used in Monte Carlo simulations of CPPI and GBM:

- Expected value: The expected value of a random variable is the average of its possible values. For example, if a random variable can take on the values 1, 2, and 3, then the expected value is  $(1 + 2 + 3) / 3 = 2$ .
- Variance: The variance of a random variable is a measure of its uncertainty. The variance is calculated by taking the average of the squared deviations from the mean. For example, if a random variable can take on the values 1, 2, and 3, then the variance is  $(1 - 2)^2 + (2 - 2)^2 + (3 - 2)^2 / 3 = 1/3$ .
- Standard deviation: The standard deviation of a random variable is the square root of its variance. The standard deviation is a measure of the typical distance from the mean. For example, if a random variable has a variance of  $1/3$ , then the standard deviation is  $\sqrt{1/3} = \sim 0.577$ .
- Drift rate: The drift rate of a GBM process is the expected rate of return of the asset. The drift rate is typically expressed as a percentage. For example, if the drift rate is 5%, then the asset is expected to increase in value by 5% on average each year.
- Volatility: The volatility of a GBM process is a measure of the uncertainty of the asset's returns. The volatility is typically expressed as a percentage. For example, if the volatility is 10%, then the asset's price is expected to fluctuate by 10% on average each year.

#### **Lab Experiment to be done by students:**

1. Implement a Monte Carlo simulation of a CPPI strategy.
2. Implement a Monte Carlo simulation of a GBM strategy.
3. Create interactive plots of the Monte Carlo simulations

**NAME: UMANG KIRIT LODAYA**

**SAP ID: 60009200032**

**BATCH: D11**

```
In [5]: import numpy as np
import matplotlib.pyplot as plt
import yfinance as yf
```

```
In [6]: # Function to implement the CPPI strategy
def cppi_simulation(stock_prices, floor_value, m, n, num_simulations):
    results = []
    for _ in range(num_simulations):
        wealth = m
        cushion = 1.0

        for i in range(1, len(stock_prices)):
            cushion = (wealth - floor_value) / wealth
            risky_asset = cushion * wealth
            risk_free_asset = wealth - risky_asset
            wealth = risky_asset * stock_prices[i] / stock_prices[i - 1] + risk_free_asset

        results.append(wealth)
    return results
```

```
In [7]: # Function to implement the GBM strategy
def gbm_simulation(initial_price, drift, volatility, n_steps, n_simulations, dt=1.0):
    results = []
    for _ in range(n_simulations):
        price = initial_price
        for _ in range(n_steps):
            price *= np.exp((drift - 0.5 * volatility ** 2) * dt + volatility * np.sqrt(dt) * np.random.randn())
        results.append(price)
    return results
```

```
In [8]: # Fetching stock data from yfinance
def fetch_stock_data(ticker, start_date, end_date):
    stock_data = yf.download(ticker, start=start_date, end=end_date)['Adj Close']
    return stock_data
```

```
In [9]: # Parameters for CPPI simulation
m = 1000000 # Initial investment
floor_value = 800000 # Floor value, CPPI strategy will not let the wealth go below this value
```

```
In [10]: # Parameters for GBM simulation
initial_price = 100 # Initial stock price
drift = 0.05 # Average return (drift) of the stock
volatility = 0.2 # Volatility of the stock
n_steps = 252 # Number of trading days in a year
n_simulations = 1000 # Number of simulations
```

```
In [11]: # Fetching stock data
ticker = 'AAPL' # You can change the ticker symbol to any other valid symbol
start_date = '2020-01-01'
end_date = '2021-01-01'
stock_data = fetch_stock_data(ticker, start_date, end_date)
```

[\*\*\*\*\*100%\*\*\*\*\*] 1 of 1 completed

```
In [12]: # Running simulations
cpqi_results = cpqi_simulation(stock_data, floor_value, m, n_steps, n_simulations)
gbm_results = gbm_simulation(initial_price, drift, volatility, n_steps, n_simulations)
```

```
In [14]: # Interactive plots
plt.hist(cpqi_results, bins=50, alpha=0.5, label='CPPI Strategy')
plt.hist(gbm_results, bins=50, alpha=0.5, label='GBM Strategy')
plt.xlabel('Final Wealth')
plt.ylabel('Frequency')
plt.title('Monte Carlo Simulations')
plt.legend()
plt.grid(True)
plt.show()
```

