Experiment 9

Bhuvi Ghosh
60009210191

Aim: Build a Monte Carlo simulation of coupon-bearing bonds using CIR.
Objective:

- To understand the CIR model and how it can be used to price coupon-bearing bonds.
- To implement a Monte Carlo simulation of coupon-bearing bonds using the CIR model.
- To analyze the results of the Monte Carlo simulation and compare them to the theoretical prices.

Theory:
The Cox-Ingersoll-Ross (CIR) model is a widely used stochastic process in finance to describe interest rate movements over time. It is an extension of the Vasicek model, designed to address some of its limitations. The CIR model is particularly relevant for pricing coupon-bearing bonds as it accounts for the mean reversion observed in interest rates. This theory aims to explain the CIR model, its application in pricing coupon-bearing bonds, the implementation of a Monte Carlo simulation using the CIR model, and the subsequent analysis and comparison of the simulation results with theoretical prices.

The CIR Model and Pricing Coupon-Bearing Bonds: The CIR model is a one-factor stochastic model that describes the evolution of interest rates over time. It is expressed as follows:

$$d(r(t)) = a(b - r(t))dt + \sigma\sqrt{r(t)}dW(t)$$

where:
- $r(t)$ is the instantaneous interest rate at time t.
- a is the speed of mean reversion, which determines how quickly the interest rate converges to the mean b.
- b is the long-term mean or equilibrium interest rate.
- σ is the volatility of the interest rate process.
- $W(t)$ is a Wiener process or Brownian motion.

To price coupon-bearing bonds using the CIR model, we need to simulate future paths of interest rates and calculate the present value of expected cash flows. By employing numerical methods like Monte Carlo simulation, we can generate multiple interest rate paths and average their corresponding bond prices to obtain a more accurate valuation.

Implementing a Monte Carlo Simulation: To implement a Monte Carlo simulation for pricing coupon-bearing bonds with the CIR model, follow these steps:

Step 1: Discretization: Discretize the CIR model using numerical methods like Euler's method or the Milstein method to generate a sequence of interest rates over time.

Step 2: Simulate Interest Rate Paths: Simulate multiple paths of interest rates by employing the discretized CIR model. Each path represents a possible evolution of interest rates over time.

Step 3: Calculate Bond Prices: For each simulated interest rate path, calculate the present value of expected cash flows for the coupon-bearing bond. Discount the cash flows back to the present using the simulated interest rates.

Step 4: Average and Calculate Theoretical Prices: Average the bond prices obtained from Step 3 to arrive at the theoretical price of the coupon-bearing bond. This average represents the Monte Carlo estimate of the bond's fair value based on the CIR model.

The Cox-Ingersoll-Ross (CIR) model offers a valuable framework for modeling interest rate movements and pricing coupon-bearing bonds. By employing Monte Carlo simulation, we can effectively estimate the fair value of these bonds under uncertain interest rate scenarios. Comparing the simulation results with theoretical prices and market data allows us to assess the model's effectiveness and make informed decisions in financial markets.

Lab Experiment to be done by students:

1. Implement a Monte Carlo simulation of coupon-bearing bonds using the CIR model.
2. Run the Monte Carlo simulation for a range of interest rates, volatilities, and coupon rates.
3. Analyze the results of the Monte Carlo simulation and compare them to the theoretical prices.
4. Discuss the implications of the results for the pricing of coupon-bearing bonds.

```
import numpy as np
import matplotlib.pyplot as plt
import yfinance as yf
```

```
kappa = 0.1
theta = 0.05
sigma = 0.02
r0 = 0.03
T = 5
N = 1000
M = 10000
dt = T / N
```

```
[ ] def simulate_cir_paths(kappa, theta, sigma, r0, T, N, M):
        rates = np.zeros((N + 1, M))
        rates[0] = r0
        dt_sqrt = np.sqrt(dt)

        for t in range(1, N + 1):
            dW = np.random.normal(0, 1, M)
            rates[t] = rates[t - 1] + kappa * (theta - rates[t - 1]) * dt + sigma * np.sqrt(np.maximum(rates[t - 1], 0)) * dW * dt_sqrt

        return rates
```

```
[ ] coupon_rate = 0.04
    face_value = 1000
    coupons_per_year = 2
```

```
[ ] def price_coupon_bond(rates, T, coupon_rate, face_value, coupons_per_year):
        coupon = coupon_rate * face_value / coupons_per_year
        bond_prices = np.zeros(rates.shape[1])

        for i in range(1, int(T * coupons_per_year) + 1):
            discount_factors = np.exp(-np.sum(rates[:i * (N // (T * coupons_per_year))] * dt, axis=0))
            bond_prices += coupon * discount_factors
        discount_factors = np.exp(-np.sum(rates * dt, axis=0))
```

```
        bond_prices += face_value * discount_factors[-1]

        return np.mean(bond_prices)
```

```
rates = simulate_cir_paths(kappa, theta, sigma, r0, T, N, M)
```

```
bond_price = price_coupon_bond(rates, T, coupon_rate, face_value, coupons_per_year)
print(f"Simulated price of the coupon-bearing bond: {bond_price:.2f}")
```

```
Simulated price of the coupon-bearing bond: 1043.60
```

```
interest_rates = [0.03, 0.04, 0.05]
volatilities = [0.01, 0.02, 0.03]
coupon_rates = [0.03, 0.04, 0.05]
for r0 in interest_rates:
    for sigma in volatilities:
        for coupon_rate in coupon_rates:
            rates = simulate_cir_paths(kappa, theta, sigma, r0, T, N, M)
            bond_price = price_coupon_bond(rates, T, coupon_rate, face_value, coupons_per_year)
            print(f"r0: {r0}, sigma: {sigma}, coupon_rate: {coupon_rate}, Bond Price: {bond_price:.2f}")
```

```
r0: 0.03, sigma: 0.01, coupon_rate: 0.03, Bond Price: 978.89
r0: 0.03, sigma: 0.01, coupon_rate: 0.04, Bond Price: 1028.57
r0: 0.03, sigma: 0.01, coupon_rate: 0.05, Bond Price: 1067.34
r0: 0.03, sigma: 0.02, coupon_rate: 0.03, Bond Price: 963.68
r0: 0.03, sigma: 0.02, coupon_rate: 0.04, Bond Price: 1026.13
r0: 0.03, sigma: 0.02, coupon_rate: 0.05, Bond Price: 1064.41
r0: 0.03, sigma: 0.03, coupon_rate: 0.03, Bond Price: 956.67
r0: 0.03, sigma: 0.03, coupon_rate: 0.04, Bond Price: 990.77
r0: 0.03, sigma: 0.03, coupon_rate: 0.05, Bond Price: 1036.66
r0: 0.04, sigma: 0.01, coupon_rate: 0.03, Bond Price: 942.14
r0: 0.04, sigma: 0.01, coupon_rate: 0.04, Bond Price: 988.16
r0: 0.04, sigma: 0.01, coupon_rate: 0.05, Bond Price: 1029.90
r0: 0.04, sigma: 0.02, coupon_rate: 0.03, Bond Price: 918.09
r0: 0.04, sigma: 0.02, coupon_rate: 0.04, Bond Price: 1000.15
r0: 0.04, sigma: 0.02, coupon_rate: 0.05, Bond Price: 1042.59
r0: 0.04, sigma: 0.03, coupon_rate: 0.03, Bond Price: 949.34
r0: 0.04, sigma: 0.03, coupon_rate: 0.04, Bond Price: 996.25
r0: 0.04, sigma: 0.03, coupon_rate: 0.05, Bond Price: 1059.15
r0: 0.05, sigma: 0.01, coupon_rate: 0.03, Bond Price: 899.29
```


r0: 0.04, sigma: 0.02, coupon_rate: 0.04, Bond Price: 1000.15
r0: 0.04, sigma: 0.02, coupon_rate: 0.05, Bond Price: 1042.59
r0: 0.04, sigma: 0.03, coupon_rate: 0.03, Bond Price: 949.34
r0: 0.04, sigma: 0.03, coupon_rate: 0.04, Bond Price: 996.25
r0: 0.04, sigma: 0.03, coupon_rate: 0.05, Bond Price: 1059.15
r0: 0.05, sigma: 0.01, coupon_rate: 0.03, Bond Price: 899.29

r0: 0.05, sigma: 0.01, coupon_rate: 0.04, Bond Price: 948.76
r0: 0.05, sigma: 0.01, coupon_rate: 0.05, Bond Price: 1001.02
r0: 0.05, sigma: 0.02, coupon_rate: 0.03, Bond Price: 882.26
r0: 0.05, sigma: 0.02, coupon_rate: 0.04, Bond Price: 967.17
r0: 0.05, sigma: 0.02, coupon_rate: 0.05, Bond Price: 974.51
r0: 0.05, sigma: 0.03, coupon_rate: 0.03, Bond Price: 874.31
r0: 0.05, sigma: 0.03, coupon_rate: 0.04, Bond Price: 957.75
r0: 0.05, sigma: 0.03, coupon_rate: 0.05, Bond Price: 966.49