IB9JH0 Group Assignment
Pricing Interest Rate Derivatives with the Vasicek and CIR Models in C++

Kai Cheema, Chance Keown, Sebastian Murphy,
James Poole, Chunlong Wang

June 5, 2024

Contents

1 Introduction

In this project, we aim to investigate and compare two classical interest rate models: the Vasicek model and the Cox-Ingersoll-Ross (CIR) model. To begin, both models will be implemented to simulate future interest rate paths. These simulated interest rates will then be used to price bonds and interest rate swaptions. Finally, by modifying the parameters of both models, we will investigate how different interest rate models impact the pricing mechanisms of financial derivatives and assess the sensitivity of derivative prices to changes in model parameters.

2 Methodology

2.1 Theory and Modelling Approach

The primary goal of this project is to accurately simulate interest rates and price financial derivatives such as bonds and swaptions. The chosen interest rate models are well-established stochastic models in quantitative finance and both are used to simulate the evolution of interest rates over time, capturing mean-reversion behaviour observed in the real world.

2.1.1 The Vasicek and Cox-Ingersoll-Ross (CIR) Models

The Vasicek model is a type of one-factor short-rate model, characterised by a tendency for interest rates to revert to a long-term mean value. The model is described by the following stochastic differential equation (SDE) [Vasicek, 1977]:

$$dr_t = \kappa(\theta - r_t)dt + \sigma dW_t$$

where r_t is the short-term interest rate at time t, κ is the speed of mean reversion, θ is the long-term mean rate, σ is the volatility of interest rate changes and dW_t is a Wiener process representing the random market risk factor.

The CIR model is another mean-reverting interest rate model, but it ensures interest rates remain non-negative, a plausible real-world assumption. The model is described by the following SDE [Cox et al., 1985]:

$$dr_t = \kappa(\theta - r_t)dt + \sigma\sqrt{r_t}dW_t$$

where the parameters are the same as in the Vasicek model above, with the key difference being the $\sqrt{r_t}$ which ensures the non-negativity.

2.1.2 Rate simulator

To generate interest rate paths using the Vasicek and CIR models, we implement a rate simulator (further explained in section 2.2) that implements the Euler-Maruyama (EM) method. The EM method is a numerical technique used to approximate solutions to SDEs. Given an SDE of the form:

$$dr(t) = a(r(t), t)dt + b(r(t), t)dW(t)$$

where $a(r(t), t)$ is the drift term, $b(r(t), t)$ is the diffusion term and $dW(t)$ is a Wiener process, the Euler-Maruyama discretization is given by [Bayram et al., 2018]:

$$r_{n+1} = r_n + a(r_n, t_n)\Delta t + b(r_n, t_n)\sqrt{\Delta t} \cdot \epsilon_n$$

where Δt is the time step and ϵ_t is a random variable sampled from a standard normal distribution. A sample of simulated interest rate paths for both models is displayed below in Figure 1:

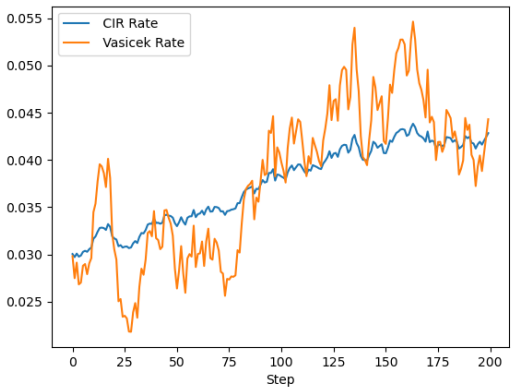

Figure 1: Simulated interest rates for Vasicek and CIR models

2.1.3 Pricing Bonds and Swaptions

A bond is a financial instrument that represents a loan made by an investor to a borrower. It involves periodic coupon payments until the return of the face value at maturity. A bond can be priced by calculating the present value of its future cash flows as follows:

$$P = \sum_{i=1}^{n} \frac{F \cdot C \cdot \frac{f}{2}}{(1 + r_i)^{i \cdot \frac{f}{2}}} + \frac{F}{(1 + r_n)^T}$$

where F is the face value of the bond, C is the coupon rate, T is the maturity of the bond, f is the frequency of the coupon payments and r_i is the interest rate at time i.

A swaption is an option to enter into an interest rate swap agreement at a future date, in which the holders can exchange a fixed and floating rate at a predefined rate. There are two types of swaptions: payer swaptions (the right to pay fixed, receive floating) and receiver swaptions (the right to receive fixed, pay floating). The price of a swaption can be determined using Black's formula [Black, 1976]. For a payer swaption, the price is:

$$P_{\text{payer}} = N \cdot PVA \left(F\Phi(d_1) - K\Phi(d_2) \right)$$

and for a receiver swaption it is:

$$P_{\text{receiver}} = N \cdot PVA \left(K\Phi(-d_2) - F\Phi(-d_1) \right)$$

where N is the notional amount, PVA is the present value of an annuity, F is the forward swap rate, K is the strike rate, $\Phi(x)$ is the c.d.f of the standard normal distribution and d_1 and d_2 are calculated as:

$$d_1 = \frac{\ln(F/K) + \frac{1}{2}\sigma^2 T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

with σ the volatility and T the time-to-maturity.

2.2 Code Structure

2.2.1 Overview

The code is structured into several classes, each responsible for different aspects of the simulation and pricing of derivatives. The main components are a class for interest rate models, classes for representing bonds and swaptions, a class to perform the rate simulation and main.cpp which handles the actual simulation and computation. This structure allowed each member to work on a different class, as well as testing, before combining them and performing integration testing. The following section describes the class structure in more detail.

2.2.2 Class Descriptions

The overall structure of the classes is represented in the class diagram below (figure 2). The *InterestRateModel* class is an abstract base class that includes common protected variables and a pure virtual function implemented by derived classes. The *VasicekModel* and *CIRModel* classes both derive from *InterestRateModel* and implement their respective models. The *simulateNextRate* method then uses the Euler-Maruyama method to update the interest rate in each. The *Bond* class represents a bond, with private variables for each attribute, the *price* method calculates the present value of the bond's cash flows using simulated interest rates. The *Swaption* class represents an interest rate swaption, with private variables for each attribute. The *price* method implements Black's formula to price the swaption based on the simulated rates. Finally, the *RateSimulator* class is responsible for simulating the paths using a specified model. It also provides a high-level interface for users to directly price a bond without manually simulating the rates first. It is dependent on both *InterestRateModel* and *Bond*.

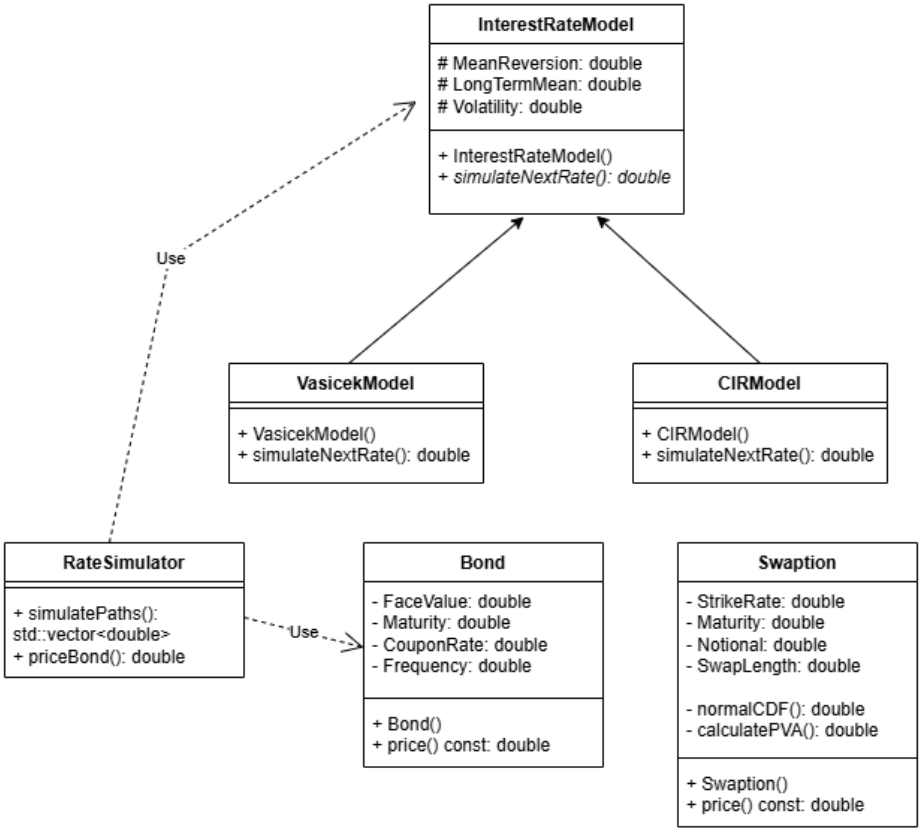

Figure 2: Class Diagram

2.2.3 Simulation and Pricing Workflow

The workflow for simulation and pricing of financial instruments is as follows:

- Initialise the interest rate models (*VasicekModel* and *CIRModel*)
- Create instances of the *Bond* and *Swaption* classes
- Use the *RateSimulator* class to generate interest rate paths for both models
- Pass the simulated rates to *Bond* and *Swaption* to calculate prices
- Output the results to the console and save the simulated paths to a CSV file

2.3 Code Testing

Using the Catch2 testing framework and CMake, we were able to automate and manage our testing process. In the tests folder of our code there are three unit test files, and one integration test file.

2.3.1 Unit Testing

The unit testing performed focused on ensuring the interest rate models were being implemented correctly, and the bond/swaption pricing was also correct.

To test the CIR and Vasicek models, we plotted their paths and examined the output CSV to ensure they looked reasonable. We then implemented tests verifying the long-term mean-reversion behaviour and volatility effect of each model, to ensure the parameters were being handled correctly.

For bonds, these tests included: a constant rates test, a face value scaling test and a test where the coupon rate equals the interest rate. These enabled us to use theoretical bond pricing formulas for constant rates to ensure our method was working correctly. In addition to this, a range of different interest rates, face values, coupon rates, maturities, and coupon frequencies were tested for the bond pricing function, to ensure it could handle inputs that made sense but were perhaps not customary. This included short coupon frequency times, extra-large coupons, and a variety of maturities.

Finally, to test the swaption pricing, we implemented a constant rates test, a payer/receiver swaption pricing test and a volatility effect test. Having passed all of these tests, and ensuring functions were handling parameters correctly, we moved on to integration testing.

2.3.2 Integration Testing

To verify that all of the individual classes functioned cohesively together we next performed integration testing. The testing included generating interest rate paths using the models, pricing bonds and swaptions with these paths, and verifying the results were consistent with financial theory. These tests included ensuring bond prices were positive and consistent, and that the swaption prices increased with higher volatility. This testing approach ensured that individually all of our classes functioned as expected, and when combined they were able to accurately achieve the aims of our project.

3 Discussion

In this section, we present some plots and tables generated by our code (alongside the Python code in the appendix) and discuss the differences between the two models

3.1 Results and Analysis

3.1.1 Interest Rate Models

Using the output CSV file, we extract the simulated rates over a set time period and plot them using Python. An example plot is shown below for an initial rate of 0.03 and a time step of 0.05:

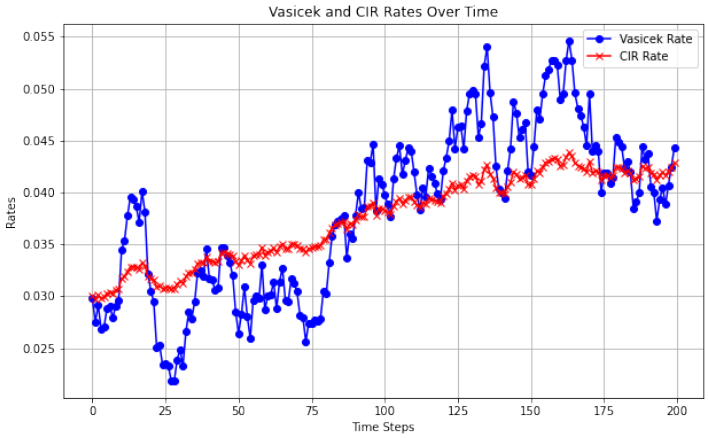

Figure 3: Simulated paths for the Vasicek and CIR models

We can further study the characteristics via the summary statistics in table 1:

Statistic	Vasicek Rate	CIR Rate
Mean	0.038	0.0375
Median	0.0393	0.0385
Standard Deviation	0.008	0.0042
Variance	6.367e-05	1.75e-05
Min	0.0218	0.0297
Max	0.0546	0.0438

Table 1: Summary Statistics of Vasicek and CIR Rates

Both models exhibit mean-reverting behaviour with the Vasicek mean rate slightly higher. The Vasicek model rates show more variability and higher volatility than the CIR rates, which is also evidenced in the plot. Furthermore, the Vasicek model has more frequent and larger fluctuations, evidenced by its max and min spread. Overall the CIR model appears more stable, potentially due to the requirement of no negative rates.

3.1.2 Bond Pricing

To analyse our bond pricing, main.cpp was run multiple times, varying the maturity variable to see how the price changes with increasing time to maturity. The prices of a bond with $1000 face value, 5% coupon rate and semi-annual coupon payments are shown below:

Maturity (Years)	Vasicek Price	CIR Price
5	1048.68	1053.17
10	1047.96	1060.27
15	1062.75	1079.71
20	1074.61	1095.40
25	1084.11	1108.07

Table 2: Bond Prices for Different Maturities and a 5% Coupon Rate Using Vasicek and CIR Models

Both models show an increase in bond prices as maturity increases, which is expected as longer-term bonds often have higher sensitivity to interest rate changes, especially with fixed coupon rates. Interestingly, the CIR prices are consistently slightly higher than the Vasicek prices, potentially due to the enforced non-negativity of rates leading to slightly different discounting effects. Finally, with a 5% coupon rate, the bonds are trading at a premium, which makes sense as our initial rate for the interest rate models is only 3%.

3.1.3 Swaption Pricing

As done in the previous section, we now reran main.cpp multiple times, varying the maturity and strike rate of the swaption to see how the prices of both models are affected, with a volatility of 20% and a notional value of 1000.

Maturity (Years)	Strike Rate (%)	Vasicek Payer Price	Vasicek Receiver Price	CIR Payer Price	CIR Receiver Price
5	2.5	79.99	4.42	70.88	5.44
10	2.5	148.48	23.92	170.90	20.28
15	2.5	248.78	46.63	286.75	40.51
5	5.0	19.21	58.26	15.48	64.04
10	5.0	50.18	135.41	61.62	123.72
15	5.0	102.49	194.94	123.87	179.19

Table 3: Swaption Prices for Different Maturities and Strike Rates Using Vasicek and CIR Models

Both models show that swaption prices increase with maturity, which is consistent with the idea that longer maturities increase the likelihood of a favourable rate movement, therefore increasing the value of the swaption. Additionally, the lower strike rate (2.5%) increases the payer swaption price and decreases the receiver swaption price, compared to the higher strike rate (5%). Intuitively this also makes sense as a lower strike rate makes it more likely for the payer swaption to end up in the money, and less likely for the receiver swaption.

Similar to the bond pricing, the CIR model once again produces slightly higher prices compared to the Vasicek model, particularly for longer maturities. This difference suggests that the CIR model, which accounts for a more realistic non-negative interest rate distribution, might predict higher potential payoffs under volatile conditions.

3.2 Conclusion

In this project, we have successfully implemented the Vasicek and CIR models to simulate interest rates and used these rates to price bonds and swaptions. The Euler-Maruyama method was used to discretize the SDEs and the accuracy of the models was ensured through rigorous unit and integration testing using the Catch2 framework and CMake.

Our results showed that both models can effectively simulate interest rate paths and price financial instruments quickly in C++, with a fairly simple class structure. Results from pricing both bonds and swaptions were consistent with theory, including increasing prices with longer maturity. The successful implementation and validation of these models is significant as in future we could implement new derived classes for other interest rate models that are potentially more complicated, like the Hull-White [Hull and White, 2001] or Ho-Lee model [Hull and White, 1993]. A limitation of this work is the Euler-Maruyama method, though simple and widely used, could introduce discretization errors. Additionally, the assumption of constant volatility may not capture the complexities of real-world interest rate dynamics.

This project offers a good foundation for future work extending the complexity of the models and numerical methods used, and demonstrates the use of interest rate models in risk management and financial planning.

References

[Bayram et al., 2018] Bayram, M., Partal, T., and Orucova Buyukoz, G. (2018). Numerical methods for simulation of stochastic differential equations. *Advances in Difference Equations*, 2018(1).

[Black, 1976] Black, F. (1976). The pricing of commodity contracts. *Journal of Financial Economics*, 3(1).

[Cox et al., 1985] Cox, J. C., Ingersoll, J. E., and Ross, S. A. (1985). A theory of the term structure of interest rates. *Econometrica*, 53(2).

[Hull and White, 1993] Hull, J. and White, A. (1993). One-factor interest-rate models and the valuation of interest-rate derivative securities. *Journal of Financial and Quantitative Analysis*, 28(2).

[Hull and White, 2001] Hull, J. and White, A. (2001). The general hull–white model and supercalibration. *Financial Analysts Journal*, 57(6).

[Vasicek, 1977] Vasicek, O. (1977). An equilibrium characterization of the term structure. *Journal of Financial Economics*, 5(2).

Appendix

Figure 1 Plot Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mlt
ir = pd.read_csv("output (11).csv")
ir.head()
col_names = list(ir.columns)
print('column names : ', col_names)
ax = ir.plot(x = 'Step', y = ' CIR Rate')
ir.plot(ax = ax, x = 'Step', y = ' Vasicek Rate')
plt.savefig('Rates.png')
plt.show()
```

Figure 3 Plot Code

```
# Import libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Read the CSV file
data = pd.read_csv('output.csv')

# Assuming the columns are named 'Time', 'VasicekRate', and 'CIRRate'
time_steps = data['Step']
vasicek_rates = data[' Vasicek Rate']
cir_rates = data[' CIR Rate']

# Create the plot
plt.figure(figsize=(10, 6))

# Plot Vasicek rates
plt.plot(time_steps, vasicek_rates, label='Vasicek Rate', color='blue',
        linestyle='-', marker='o')

# Plot CIR rates
plt.plot(time_steps, cir_rates, label='CIR Rate', color='red',
        linestyle='-', marker='x')

# Adding title and labels
plt.title('Vasicek and CIR Rates Over Time')
plt.xlabel('Time Steps')
plt.ylabel('Rates')
plt.legend()

# Display the plot
plt.grid(True)
plt.savefig('Rates2.png')
plt.show()
```

Table 1 Code

```
def round_to_sf(value, sig_figs=4):
    return round(value, sig_figs - int(f"{value:e}".split('e')[1]) - 1)

# Summary statistics
summary_statistics = {
    'Vasicek Rate': {
        'Mean': round(vasicek_rates.mean(),4),
        'Median': round(vasicek_rates.median(),4),
        'Standard Deviation': round(vasicek_rates.std(),4),
        'Variance': round_to_sf(vasicek_rates.var()),
        'Min': round(vasicek_rates.min(),4),
        'Max': round(vasicek_rates.max(),4)
    },
    'CIR Rate': {
        'Mean': round(cir_rates.mean(),4),
        'Median': round(cir_rates.median(),4),
        'Standard Deviation': round(cir_rates.std(),4),
        'Variance':  round_to_sf(cir_rates.var()),
        'Min': round(cir_rates.min(),4),
        'Max': round(cir_rates.max(),4)
    }
}

# Create LaTeX table format
latex_table = "\\begin{table}[H]\n\\centering\n\\begin{tabular}{|l|c|c|}\n\\hline\n"
latex_table += "Statistic & Vasicek Rate & CIR Rate \\\\\n\\hline\n"

for stat in summary_statistics['Vasicek Rate'].keys():
    vasicek_value = summary_statistics['Vasicek Rate'][stat]
    cir_value = summary_statistics['CIR Rate'][stat]
    latex_table += f"{stat} & {vasicek_value} & {cir_value} \\\\\n"

latex_table += "\\hline\n\\end{tabular}\n\\
                caption{Summary Statistics of Vasicek and CIR Rates}\n\\
                label{tab:summary_statistics}\n\\end{table}"

# Print LaTeX table
print(latex_table)
```