Vasicek Model: Calibration and Monte Carlo Pricing

Alessandro Ciniltani

February 2023

1 Introduction

The Vasicek model [1] is a popular tool used to model the evolution of interest rates. It is described by the following Stochastic Differential Equation:

$$dr_t = \kappa(\theta - r_t)dt + \sigma dW_t \tag{1}$$

It is a mean reverting process, with rate of mean reversion k, θ represents the mean, dt the time step, σ the Standard Deviation and dW_t are the increment of a Brownian Motion.
The equation has two main components:

- **The drift term** $\kappa(\theta - r_t)dt$: This term represents the expected change in the short-term interest rate over time. It depends on two parameters: κ and θ. The parameter κ is the mean reversion rate, which represents the speed at which the interest rate reverts to its long-term mean. The parameter θ is the long-term mean interest rate, which represents the level to which the interest rate is expected to converge in the long run.
- **The diffusion term** σdW_t: This term represents the random fluctuations in the short-term interest rate over time. It is a stochastic differential, where dW_t is the differential of a standard Brownian motion process and σ is the volatility of interest rate changes. The term σdW_t.

The SDE above can be solved, obtaining:

$$r_t = r_{t-1}e^{-\kappa(t-s)} + \theta(1 - e^{-\kappa(t-s)}) + \sigma \int_t^s e^{-k(t-u)}dW(u) \tag{2}$$

Assuming the Vasicek model, we have that:

$$P(t,T) = E_t[e^{-\int_t^T r_s ds}] = A(t,T)e^{-B(t,T)r_t} \tag{3}$$

where,

$$A(t,T) = exp((\frac{\theta - \sigma^2}{2\kappa})[B(t,T) - T + t] - \frac{\sigma^2}{2\kappa}B(t,T)^2) \tag{4}$$

$$B(t,T) = \frac{1}{\kappa}[1 - e^{\kappa(T-t)}] \tag{5}$$

1.1 Model Calibration

Calibrating the Vasicek model to the interest rate curve on a specific date provides the optimal parameters that best fit the observed market data. In turn, this information can be used to simulate the risk-neutral dynamics of interest rates and price financial instruments.

This section will provide a calibration of the Vasicek model to the interest rate curve as of 23 January 2023, using the fmin function from the scipy.optimize library.

The steps involved in calibrating the Vasicek model are as follows:

1. **Obtain interest rate data**: Obtain historical data for the interest rate curve on 23 January 2023, which can be sourced from Refinitiv.

   Interpolation of the yield curve on the time horizon of interest is also needed, it has been implement for a year and a half with the following code:

   ```
   sdate = date(2023,1,23)   # start date
   edate = date(2024,7,23)   # end date
   calendar = pd.date_range(sdate, edate - timedelta(days=1),freq='d')
   f = CubicSpline(data['MaturityDate'], data['Yield'])
   ```

   Obtaining the orange curve displayed in Figure 1.

2. **Define the objective function**: Define the objective function that will measure the goodness of fit between the Vasicek model and the observed interest rate data. The objective function should be a measure of the sum of squared differences between the observed and generated interest rates. In this paper we used the following functions:

   ```
   def model_prices(kappa,theta,sigma,r0):

       B = (1 - np.exp(-kappa * t)) / kappa
       A = np.exp((theta - sigma**2 / (2 * kappa**2)) * (B - t) - sigma**2 / (4 *
                                                   kappa) * B**2)
       p = A * np.exp(-B * r0)
       R = - np.log(p)/ t

       return R, A, B

   def vasicek_error_function(params,r_observed,t):
       kappa, theta, sigma, r0 = params
       r_predicted = np.zeros_like(r_observed)
       r_predicted[0] = r0
       err = 0
       R,A,B=model_prices(kappa,theta,sigma,r0)
       for i in range(1, len(t)-1):
           err = err + (r_observed[i] - R[i]) ** 2

       return err
   ```

   Which can be written as:

$$\text{err} = \sum_{i=1}^{N-1} (R_{\text{observed}}[i] - R[i])^2 \tag{6}$$

which is the sum of squared error between observed and fitted R, which will be minimized.

3. **Minimize the objective function**: Utilize the fmin function from scipy.optimize to minimize the objective function and obtain the optimal parameters of the Vasicek model.

```
def libor_calibration(libor_rates,maturities):
  # Supply initial guess. Use theta array from assignment as baseline.
  init_theta = np.array([0.5, 0.05, 0.06, 0.03])
  init_kappa, init_mu, init_sigma, init_r0 = init_theta
  return so.fmin(vasicek_error_function, init_theta, args = (libor_rates,
                                             maturities), disp = 0)
```

In order to calibrate the model, given the assumptions about mean reversion, we need to fix appropriate bounds for the coefficients, especially for $\kappa > 0$.
The following parameters has been obtained from the calibration:

- $\kappa = 1.1667$
- $\theta = 0.0753$
- $\sigma = 0.3751$
- $r_0 = 0.0190$

Once the Vasicek model has been calibrated to the interest rate curve as of 23 January 2023, it can be used to simulate the risk-neutral dynamics of interest rates and price financial instruments.
It is possible to plot the observed and the fitted values after calibration in order to evaluate the model's performance (Fig. 1).

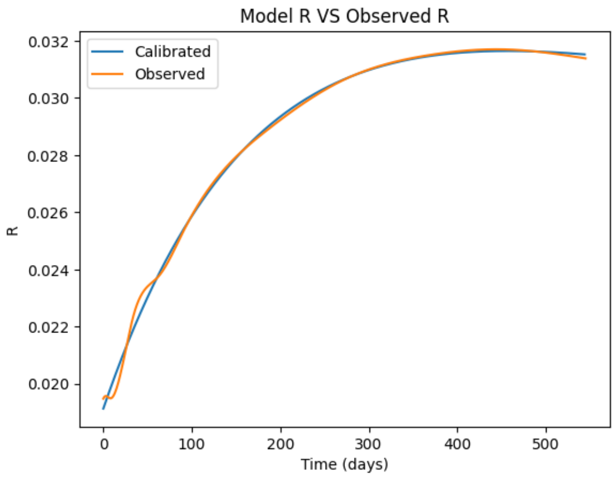

Figure 1

Now, that the calibrated parameters are known, equations 3, 4 and 5 can be easily computed with $R(0,T) = \frac{-\log P(0,T)}{T}$.

1.2 Risk-Neutral Dynamics of Interest Rates

The calibrated Vasicek model can be used to simulate the risk-neutral dynamics of interest rates on a given time horizon. This information can then be used to price financial instruments and assess their value.

In this section, we will explain how to simulate the risk-neutral dynamics of interest rates on a 1 year time horizon in terms of the Money Market Account (MMA) and 6 months Euribor rates.

The steps involved in simulating the risk-neutral dynamics of interest rates are as follows:

1. **Define the simulation parameters**: Define the length of the simulation period, the number of simulations to be performed, and the time step size.
2. **Generate random interest rate paths** : Generate random interest rate paths based on the calibrated Vasicek model parameters and the defined simulation parameters.
3. **Calculate the MMA and 6 months Euribor rates**: Calculate the MMA and 6 months Euribor rates for each of the generated interest rate paths.
4. **Analyze the results**: Analyze the results of the simulation to determine the expected value and the distribution of the MMA and 6 months Euribor rates over the simulation period.

1.2.1 Money Market Accouunt

The Money Market Account (MMA) evolves as follows:

$$dBt = B_t \cdot r_t \cdot dt, B_0 = 1 \tag{7}$$

Here, r_t evolves under the Vasicek model, so we can simulate it:

```python
def sim_vasicek(params, nsim):
    kappa,theta,sigma,r0 = params

    dt = 1 / nsim
    sqrt_dt = np.sqrt(dt)

    dW = sqrt_dt * np.random.normal(size=nsim)
    dr = kappa * (theta - r0) * dt + sigma * dW
    r = np.zeros(nsim + 1)
    r[0] = r0
    r[1:] = r0 + np.cumsum(dr)

    return r
```

From this function, a simulated path for r is obtained [Figure 2].
Now, a way for approximating the integral is needed, trapezoidal rule have been implemented.
The trapezoidal rule is a numerical method for approximating the value of a definite integral. It works by approximating the area under the curve of the integrand with a series of trapezoids. The basic idea is to partition the interval of integration into a set of subintervals, and then approximate the area under the curve on each subinterval with a trapezoid.

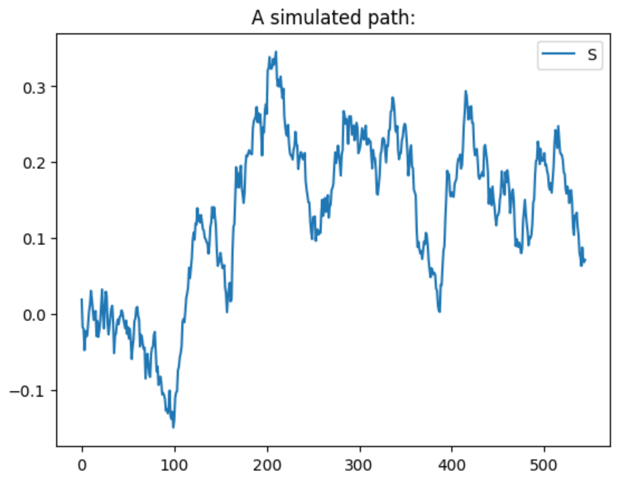

Figure 2

$$A_i = \frac{(f(x_i) + f(x_{i+1}))h}{2}, \tag{8}$$

This formula has been implement for computing:

$$\int\limits_{t-1}^{t} r_s ds = \frac{r_{t-1} + r_t}{2}(t_1 - t_{i-1}) \tag{9}$$

The following code illustrates the implementation of the trapezoidal rule for approximating the integral:

```
def trapezoidal_rule(curve, tenor):

    curve_shift = np.roll(curve,-1)
    curve_shift[-1] = 0

    integral = 0.5 * (curve + curve_shift) * (tenor)

    return integral

def mma_approx(curve, tenor):
    mma = np.zeros_like(curve)
    mma[0] = 1

    integral = trapezoidal_rule(curve, tenor)

    for i in range(len(mma)-1):
        mma[i+1] = mma[i] * np.exp(integral[i])

    return mma
```

The obtained curve is displayed in Figure 3.

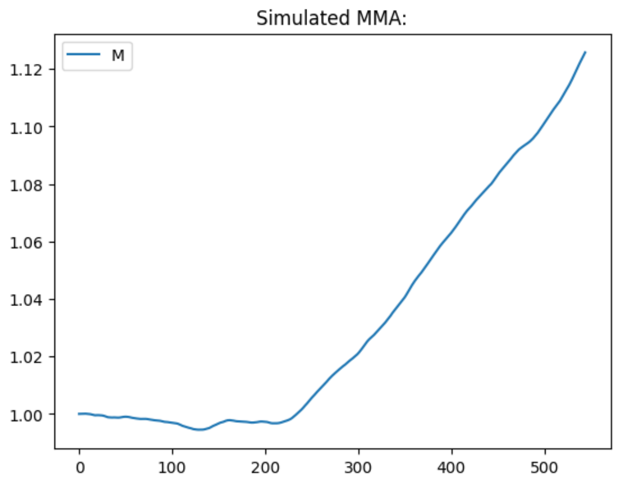

Figure 3

1.2.2 6 months Euribor Rates

The 6 months Euribor rates have been simulated starting from:

$$L(t_{i-1,t_1}) = \left(\frac{1}{P(t_{i-1}, t_i)} - 1\right) \frac{1}{\alpha(t_{i-1}, t_i)} \tag{10}$$

Given that the tenor is six months, $\alpha(t_{i-1}, t_i) = 0.5$. As a first step, daily forward prices are computed with the following code:

```
x, A, B = model_prices(kappa, theta, sigma, r0)

P = A * np.exp(-B * sim_vasicek(calibrated_theta, 544))

dt=np.zeros(len(t)) #vector of t(i)-t(i-1) for 1.5 y
for i in range(len(t)-1):
    dt[i]=t[i+1]-t[i]

t = dt[:-1]
x, A, B = model_prices(kappa, theta, sigma, r0)
P = A * np.exp(-B * sim_vasicek(calibrated_theta, 543))
```

Obtaining, the rates illustrated in Figure 4.
Finally, spot prices and Euribor rates have been computed as follows:

```
Six_forw=pd.DataFrame(P).rolling(window=180).apply(np.prod, raw = True)
Six_forw = Six_forw.dropna().reset_index().drop("index", axis=1)

euri=((1/Six_forw)-1)*(1/0.5)
```

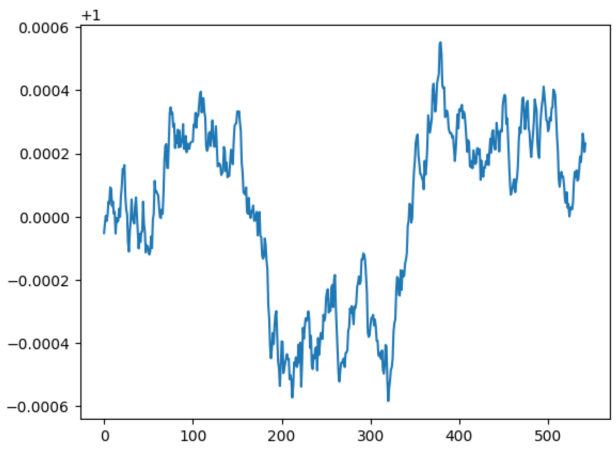

Figure 4: Daily Forward for 6m Euribor

Obtaining rates as in Figures 5 and 6

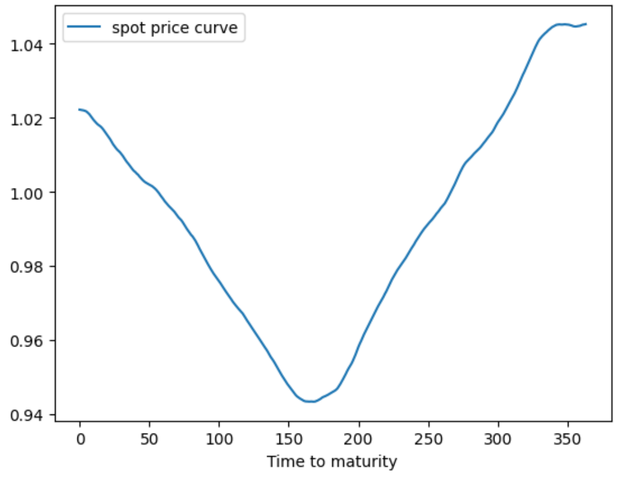

Figure 5: Spot prices

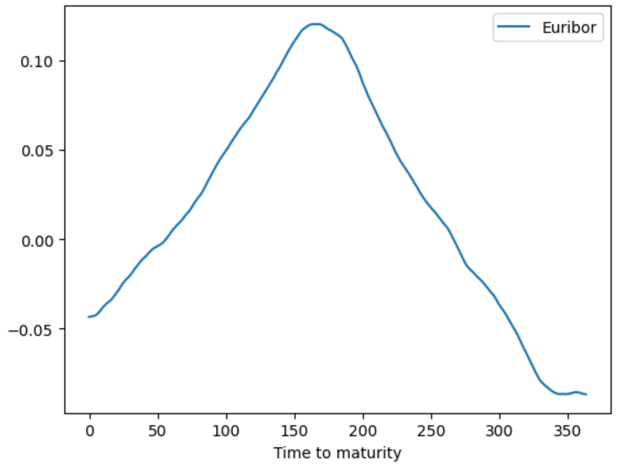

Figure 6: Simulation of Euribor

1.3 Pricing of a ZCB via Monte Carlo

Zero-coupon bonds (ZCBs) are financial instruments that pay no periodic coupon payments and instead pay the entire return at maturity. The price of a ZCB can be determined using the Vasicek model and Monte Carlo simulation.

The steps involved in pricing a ZCB using Monte Carlo simulation are as follows:

1. **Define the simulation parameters**: Define the length of the simulation period, the number of simulations to be performed, and the time step size.

2. **Generate random interest rate paths**: Generate random interest rate paths based on the calibrated Vasicek model parameters and the defined simulation parameters.

3. **Calculate the ZCB price**: For each of the generated interest rate paths, calculate the ZCB price at maturity by discounting the face value with the simulated interest rates.

4. **Analyze the results**: Analyze the results of the simulation to determine the expected value and the distribution of the ZCB price at maturity.

This is done with the following code:

```
nsim = 365
MCsim = 10000

results = np.zeros((MCsim, nsim + 1))
for i in range(MCsim):
    results[i, :] = sim_vasicek(calibrated_theta, nsim)
```

The obtained results are displayed in Figure 7. Then, prices for each simulation are computed within the Vasicek framework, using the following formula:

```
def Price(T, nsim, r):
    dt = 1 / T
    inte = np.zeros((nsim, T))
    for i in range(1, T):
        inte[:, i] = (r[:, i] + r[:, i - 1]) * 0.5 * dt
    summi = np.sum(inte, axis=1)
    P = np.exp(-summi)
    return P
```

The mean of all prices will proxy the estimated price, moreover confidence intervals have been computed, obtaining as mean 0.9692. The upper bound of the Confidence Interval is 0.97343 while the lower bound is 0.9651.

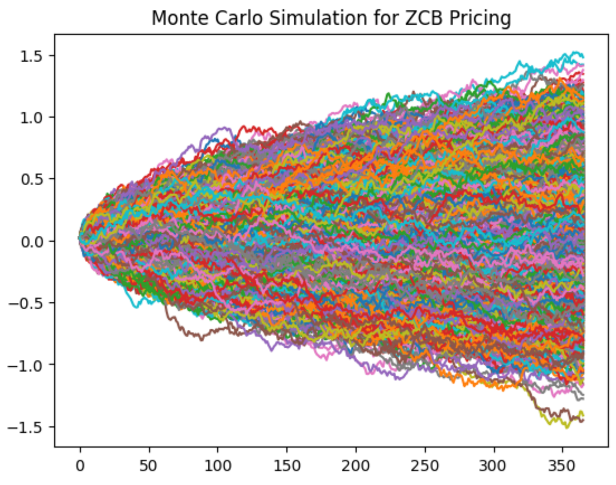

Figure 7

References

[1] Vasicek, O.: An equilibrium characterization of the term structure. J. Financ. Econ. 5, 177–188 (1977)