Financial Securities and Markets

Yuqian Zhang
yz6572@nyu.edu

May 13, 2023

Contents

1 Problem Overview

For the condition that buying a contract paying at maturity T the amount in USD:

$$\max\left[0, \left(\frac{S(T)}{S(0)} - k\right) \cdot \left(k' - \frac{L(T-\Delta, T-\Delta, T)}{L(0, T-\Delta, T)}\right)\right]$$

with:

- $S(t)$ the Nikkei-225 spot price quantoed from JPY into USD
- $L(t, T-\Delta, T)$ is the 3-month USD LIBOR rate between $T-\Delta$ and T
- Δ is a period of 3-month (convert to 0.25 years)
- T the expiration date (here we set as 3 years)
- k, k' given relative strike prices

In this project, I'll use Hull-White method to calculate the price of this contract with input of deal term of the contract. The model will first generate the Nikkei-225 spot price's path by geometric Brownian motion, and then calculate the interest path by Hull-White model. Finally, we use the zero-coupon bond price to get LIBOR rate and calculate the payoff by the contract function. More details will be illustrated in the next part.

2 Assumptions and Data

2.1 Hull White Model's General Assumptions

The Hull-White model aims at describing the evolution of interest rates (3.1). As the extension of Vasicek model, the Hill-White model allows the volatility of the interest rate to be time-independent, and it is a one-factor model so that only one source of market risk is considered. Here are some basic assumptions for applying this model.

- **No Arbitrage**
  The model assumes that there are no arbitrage possibilities, which means that it's impossible to make risk-free profits.
- **Market Completeness**
  The market shall be complete – every derivative security can be perfectly hedged using the underlying asset.Thus, combine with the arbitrage free assumption, the model requires the condition that $N = R$ where R means the number of random sources, and N means the number of risky assets.
- **Risk-Neutral Measure**
  The Hull-White assumes a risk-neutral measure so the expected return for all assets is the risk-free rate.

- **Normal Distribution**

  To make the model tractable, I simply let the changes in the short rate are normally distributed. However, one potential problem is the negative interest rate might appear.

- **Mean Reversion**

  The model assumes that interest rates are mean-reverting, which means that they tend to return to a long-term average level over time. This property also fits the economic patterns that the interest rate would converge to a stable value after a period of one-time changes of some variables.

- **Time-Dependent Volatility**

  The Hull-White model allows the volatility of the interest rate to be a function of time, which is a more realistic assumption than the constant volatility assumed by the Vasicek model.

- **Instantaneous Short Rate**

  The model assumes that the short rate (the interest rate for an infinitesimally short period of time) is the only factor driving the evolution of the term structure of interest rates.

2.2 Assumptions for Data Generating and Real Data Source

- **Brownian Motion**

  In this project, I assume that *the dynamics of the US interest rate*, *the short rate dynamics*, *the price of the stock*, and *the exchange rate between USD and JPY* all follow stochastic random walks driven by some Wiener processes. Though the Brownian motion cannot be proved in the real world, it's still reasonable to make such assumption because of the central limit theory, the observation of independent stock price change, and the continuous market trading.

- **Correlation**

  Assume that there's correlation between random process of the stock (US) and the random process of the interest rate (US) for applying the Cholesky decomposition (3.1).

- **Real Data Source**

  - σ_x is the volatility of the exchange rate USD/JPY, 0.18 from Yahoo Finance and Macro Trend.
  - σ_s is the volatility of the Nikkei-225 equity, 0.2 which calculated from Nikkei-225 one year price curve converted in USD by Yahoo Finance.
  - r_f risk-free interest rate in Japan, -0.0015, gained from Yahoo Finance, which will be used in calculating LIBOR rate.
  - S_0, r_0 are the initial stock price and the interest rate for the model collected on May 2nd, in USD. At the same time, assign May 2nd as time $t = 0$. I get the interest rate r_0 from Trading Economics as 0.0525 and the stock price from Google Finance: NI225 as 291.57.

3 Math Reasoning: Hull-White Method

3.1 Hull-White Model

Hull-white model (extended Vasiček) [1]

$$dr = (\Theta(t) - a(t)r)dt + \sigma(t)dW^{Q^d}, \quad (a(t) > 0) \tag{1}$$

where

- α is a constant
- σ is a constant
- Θ is a deterministic function of time

  $$\Theta(t) = \frac{\partial f(0,t)}{\partial T} + af(0,t) + \frac{\sigma_r^2}{2a}\left(1 - e^{-2at}\right)$$

  where $\{f(0,T); T > 0\}$ means the theoretical instantaneous forward rate which I'll take the observed $\{f^*(0,T); T > 0\}$ from the real data in this model.
- W^{Q^d} is a Brownian motion under martingale measure, Q^d
- To simplify the model, I directly give a proper Θ value, 0.03, to avoid calculating $f(t,T)$ by the formula above.

We choose a and σ to get a volatility function and choose Θ to fit the theoretical bond prices $\{p(0,T); T > 0\}$ to the observed curve $\{p^*(0,T); T > 0\}$. As we assume there's correlation in the second part, we can use Cholesky decomposition , and rewrite dW^{Q^d} as

$$dW^{Q^d} = \rho_{sr}dW + \sqrt{1 - \rho_{sr}^2}dW^{\perp} \tag{2}$$

where $dW^{\perp}$ stands for a Wiener process independent of dW^{Q^d} and ρ_{sr} is the correlation between the short rate and the Nikkei-225 spot price.

3.2 Short Rate Process

3.3 Spot price

Assume the spot price of Nikkei-225 have the following dynamics form under the Japanese risk-neutral measure Q^f where *yen* is the numeraire:

$$\frac{dS}{S} = (r_f - q)\,dt + \sigma_s dW^{Q^J} \tag{3}$$

where r_f is the risk-free rate, q is the dividend yield, and σ_s is the volatility of the Nikkei-225 spot price. Then, change the measure dW^{Q^J} into the measure where USD is the new numeraire, $dW^{Q_{\text{USD}}}$. We have

$$dS_t = (r_{\text{JPY}} - q - \sigma_s\sigma_x\rho_{sx})S_t dt + \sigma_s S_t dW^{Q_{\text{USD}}} \tag{4}$$

where σ_x is the volatility of exchange rate of USD and Nikkei-225 prices, and ρ_{sx} is the correlation between the Nikkei-225 spot price and the exchange rate above. Then, we solve the SDE by the following steps:

3.3.1 Solving SDE

Proof. Firstly, isolating the process terms to get

$$\frac{dS_t}{S_t} = (r_{\text{JPY}} - q - \sigma_s\sigma_x\rho_{sx})dt + \sigma_s dW^{Q_{\text{USD}}} \tag{5}$$

then, we introduce the function that

$$f(t, S_t) = \ln(S_t) \tag{6}$$

After taking partials, we have

$$f_{S_t} = \frac{1}{S_t}dS_t, \quad f_{S_tS_t} = -\frac{1}{S_t^2}(dS_t)^2$$

We'd like to compute $(dS_t)^2$ to get the solution S_t. Notice that

$$(dS_t)^2 = \sigma_s^2 S_t^2 dt$$

we can have

$$\begin{aligned} df(t, S_t) &= \frac{1}{S_t}dS_t + \frac{1}{2}(-\frac{1}{S_t^2}(dS_t)^2) \\ &= \frac{1}{S_t}[(r_{\text{JPY}} - q - \sigma_s\sigma_x\rho_{sx})S_t dt + \sigma_s S_t dW^{Q_{\text{USD}}}] + \frac{1}{2}(-\frac{1}{S_t^2}(dS_t)^2) \\ &= (r_{\text{JPY}} - q - \sigma_s\sigma_x\rho_{sx})dt + \sigma_s dW^{Q_{\text{USD}}} - \frac{1}{2}\sigma_s^2 dt \\ &= (r_{\text{JPY}} - q - \sigma_s\sigma_x\rho_{sx} - \frac{1}{2}\sigma_s^2)dt + \sigma_s dW^{Q_{\text{USD}}} \end{aligned}$$

Then, integrating both sides to get:

$$\ln(S_t) = \ln(S_0) + (r_{\text{JPY}} - q - \sigma_s\sigma_x\rho_{sx} - \frac{1}{2}\sigma_s^2)t + \sigma_s W^{Q_{\text{USD}}}$$

taking exponent of both sides to get:

$$S_t = S_0 \cdot exp\left[(r_{\text{JPY}} - q - \sigma_s\sigma_x\rho_{sx} - \frac{1}{2}\sigma_s^2)t + \sigma_s W^{Q_{\text{USD}}}\right]$$

□

3.4 LIBOR Rate

3.4.1 Hull-White Term Structure

By Björk textbook Chapter 24 [1], we have hull-white term structure:

$$p(t,T) = \frac{p^*(0,T)}{p^*(0,t)}\exp\left\{B(t,T)f^*(0,t) - \frac{\sigma^2}{4a}B^2(t,T)\left(1 - e^{-2at}\right) - B(t,T)r(t)\right\} \tag{7}$$

and the bond prices:

$$p(t,T) = e^{(A(t,T) - B(t,T)r(t))} \tag{8}$$

where A and B are solve:

$$\begin{cases} A_t(t,T) = \Theta(t)B(t,T) - \frac{1}{2}\sigma^2 B^2(t,T) \\ A(T,T) = 0 \end{cases}$$

$$\begin{cases} B_t(t,T) = -1 \\ B(T,T) = 0 \end{cases}$$

$$A(t,T) = \int_t^T \Theta(s)(s-T)ds + \frac{\sigma^2}{2} \cdot \frac{(T-t)^3}{3}$$

$$B(t,T) = \frac{1}{a}\left\{1 - e^{-a(T-t)}\right\}$$

Thus, we can get the bond price $p(T-\Delta, T)$ as follows:

$$p(t,T) = E_{t,r}^Q\left[\exp\left\{-\int_t^T r_s ds\right\}\right] \tag{9}$$

$$p(T-\Delta,T) = E_{t,r}^Q\left[\exp\left\{-\int_{T-\Delta}^T r_s ds\right\}\right] \tag{10}$$

3.4.2 Calculating LIBOR Rate

With the bond price formula above, we can calculate the LIBOR rate for the cutoff

$$L(0,T-\Delta,T) = -\frac{p(0,T) - p(0,T-\Delta)}{\Delta \cdot p(0,T)} \tag{11}$$

$$L(T-\Delta,T-\Delta,T) = -\frac{p(T-\Delta,T) - 1}{\Delta \cdot p(T-\Delta,T)} \tag{12}$$

3.5 Payoff

Finally, using the $S(t)$, LIBOR rate, and strikes, we can price our contract with the payoff function:

$$\Pi(T) = \max\left[0, \left(\frac{S(T)}{S(0)} - k\right) \cdot \left(k' - \frac{L(T-\Delta,T-\Delta,T)}{L(0,T-\Delta,T)}\right)\right] \tag{13}$$

We take the average of payoff to get the expected payoff, then convert it into the present payoff by timing $\exp\{-r_f \cdot T\}$. The discounting process will have the following fomula;

$$D(T) = exp\left\{-\sum_{i=0}^{n-1} r_i \Delta t\right\} \tag{14}$$

where r_i means the short rate, and the price of contract is

$$V(0) = \mathbb{E}[D(T)\Pi(T)] \tag{15}$$

4 Illustrations of Code

4.1 Structure and Key Functions

My code can mainly divided into three parts: the key functions, the implement with given values and the implement with real variables, and in this writeup I'll only exhibit the results with the real values (5).

There are six main functions with descriptions in the first part:

```
# generating ranbdom functions
def random_functions(rho, num_steps, num_paths, T)
    return dt, dW, dW_perp, dWq

# generate the path for the stock
def stock_path(S0, rf, q, sigma_s, num_paths, num_steps, dt, rho_sx, sigma_x)
    return S

# generate the path for the interest by hull-white model
def interest_path(r0, theta, a, sigma, dW, dt, num_paths, num_steps, dWq, rho_sr):
    return r

# calculate the zero-coupon bond price by hull-white
def Hull_White_bp(price1, price2, forward_rate, a, sigma, t, T, r):
    return bp

# calculate LIBOR rate
def libor(price1, price2, delta)
    return lib

#the contract function
def calpayoff(k,k_prime,S ,L_delta,L_T)
    return payoff
```

Listing 1: Def for the first part

4.2 Calibration with Real Data

As discussed in the second part of the data source (2.2), I finally set up the real value as follows:

```
# for the actual price
    k = 1
    k_prime = 1
    S0 = 291.57
    rf = -0.0015
    r0 = 0.0525
    theta = 0.03
    a = 0.5
    sigma_r = 0.02
    q = 0.0152
```

```
    rho_sr = 0.6
    rho_sx = 0.5
    sigma_s = 0.2
    sigma_x = 0.18
    T = 5                    #5 years
    num_steps = 252 * 5      #time step : every day
    num_paths = 10000

    delta = 1 / 4 #3 month (1/4 year)
    P_0_T = np.exp(-0.0341 * 5)
    P_0_t = np.exp(-0.0341 * (5 - 0.25))
    forward_rate = 0.030215
    delta = 1 / 4            #3 month (1/4 year)

    P_0_T = np.exp(-0.0341 * 5)
    P_0_t = np.exp(-0.0341 * (5 - 0.25))
    forward_rate = 0.030215
```

Listing 2: Real values

5 Result and Validation

5.1 Stock Price's Path and Short Rate's Path

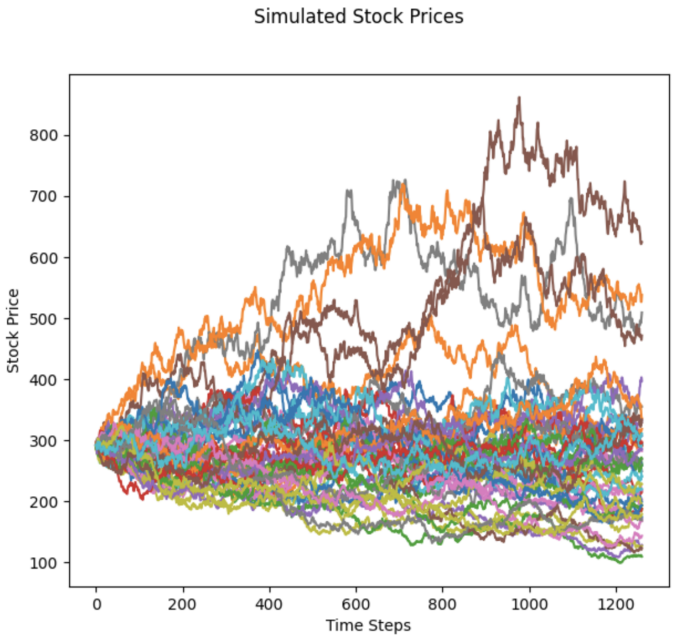

Figure 1: Stock Price's Path

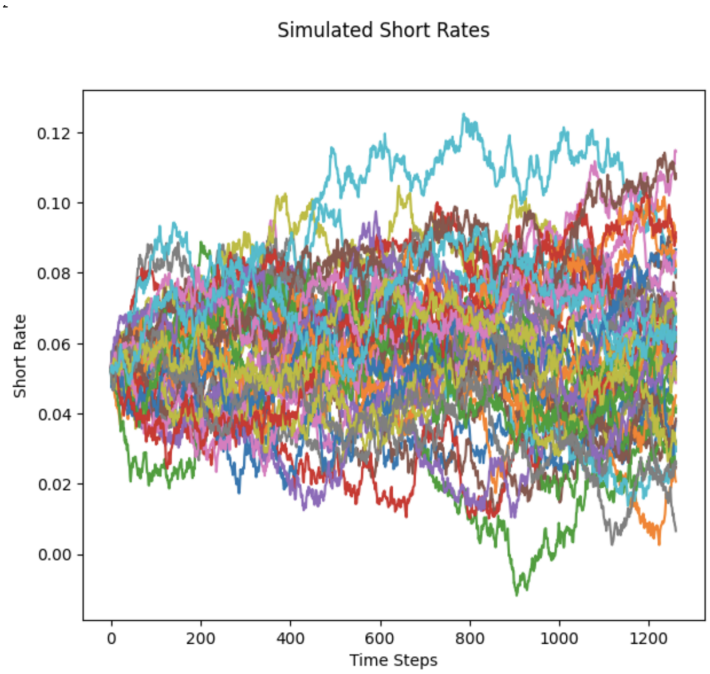

Figure 2: Short Rate's Path

By these two pictures, we can see that most simulated prices and short rates are concentrated in a certain range.

5.2 Calculated Results

```
L_delta [0.03568583 0.09078221 0.04477003 ... 0.03220727 0.0461357  0.07173338]
s_ratio [1.05369996 1.09231626 0.89651681 ... 0.78124901 1.09397725 1.19825085]
L_ratio [1.04205092 2.65090326 1.30731578 ... 0.9404744  1.34719426 2.09466439]
expected payoff 0.20752419158777644
present value 0.2090864742615181
```

Figure 3: Calculated Results

With such calculated results, we can find that each ratio fluctuates around a certain values and the expected payoff is different present value after applying the change.

5.3 Frequency

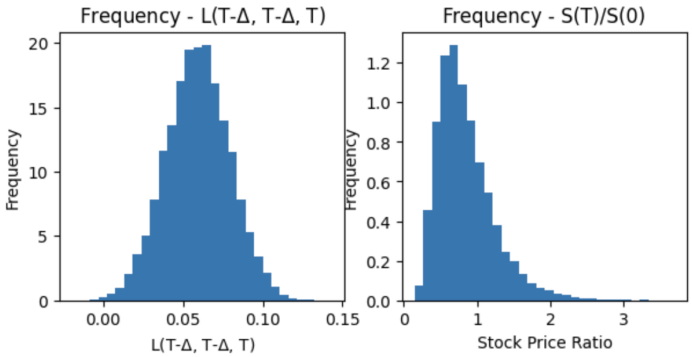

Figure 4: Frequency

Finally the distribution of L-ratio's frequency is normally distribution, with the center around 0.065. Both frequency pictures proves the statement given by the picture of stock price and the short rate, showing the concentration of a certain value.

6 Appendix

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# !pip install -Uqq ipdb
# import ipdb

# generating ranbdom functions
def random_functions(rho, num_steps, num_paths, T):
  dt = T / (num_steps) # time steps
  dW = np.random.normal(0, np.sqrt(dt), (num_paths, num_steps))
  dW_perp = np.random.normal(0, np.sqrt(dt), (num_paths))
  dWq = rho * dW[:, num_steps - 1] + np.sqrt(1 - rho ** 2) * dW_perp
  return dt, dW, dW_perp, dWq

# Brownian motion

# generate the path for the stock
# S0: the initial price
# rf: the risk-free interest rate
# q: the dividend yield of the stock
# sigma_s: the volatility of the stock
# num_paths: the number of paths to simulate
# num_steps: the number of steps in each path
# time_steps: the time step for each step
# rho_sx: the correlation between the stock and another asset
# sigma_x: the volatility of the other asset

def stock_path(S0, rf, q, sigma_s, num_paths, num_steps, dt, rho_sx, sigma_x):
  S = np.zeros((num_paths, num_steps + 1))
  S[:, 0] = S0
  drift_stock = rf - q - rho_sx * sigma_s * sigma_x
  dW = np.random.normal(0, np.sqrt(dt), (num_paths, num_steps))
  for step in range(num_steps):
      S[:, step + 1] =  S[:, step] * np.exp(drift_stock * dt - 0.5 * sigma_s**2 * dt + sigma_s
    * dW[:, step])
return S

# generate the path for the interest by hull-white model
def interest_path(r0, theta, a, sigma, dW, dt, num_paths, num_steps, dWq, rho_sr):
  r = np.zeros((num_paths, num_steps + 1))
  r[:, 0] = r0
  for t in range(num_steps):
    drift_interest = theta - a * r[:, t]
    # r[:, t + 1] = r[:, t] + drift_interest * dt + sigma * dWq
    r[:, t+1] = r[:, t] + (theta - a * r[:, t]) * dt + sigma * (rho_sr * dW[:, t] + np.sqrt(1
    - rho_sr**2) * np.random.normal(0,
```

```
  return r

#%pdb on

# calculate the zero-coupon bond price by hull-white
def Hull_White_bp(price1, price2, forward_rate, a, sigma, t, T, r):
  p = price1 / price2
  B = (1 - np.exp(-a * (T - t))) / a
  e = np.exp(B * forward_rate - (sigma**2 / 4*a) * (B**2) * (1 - np.exp(-2*a*t)) - B * r[:,-1]
     )
  bp = p * e
return bp

# calculate LIBOR rate
def libor(price1, price2, delta):
  lib = (price1 - price2) / (delta * price2)
return lib

#the contract function
def calpayoff(k,k_prime,S ,L_delta,L_T):
   s_ratio = S[:, -1] / S[:, 0]
  print("s_ratio", s_ratio)
  L_ratio = L_delta / L_T
  print("L_ratio", L_ratio)
  payoff = np.maximum(0,(s_ratio - k) * (k_prime - L_ratio))
  return payoff

# for the actual price
k = 1
k_prime = 1
S0 = 291.57
rf = -0.0015
r0 = 0.0525
theta = 0.03
a = 0.5
sigma_r = 0.02
q = 0.0152
rho_sr = 0.6
rho_sx = 0.5
sigma_s = 0.2
sigma_x = 0.18
T = 5                        #5 years
num_steps = 252 * 5          #time step : every day
num_paths = 10000

delta = 1 / 4                #3 month (1/4 year)
P_0_T = np.exp(-0.0341 * 5)
P_0_t = np.exp(-0.0341 * (5 - 0.25))
forward_rate = 0.030215

```

```
95
96 dt, dW, dW_perp, dWq = random_functions(rho_sr, num_steps, num_paths, T)
97
98 s = stock_path(S0, rf, q, sigma_s, num_paths, num_steps,  dt, rho_sx, sigma_x)
99 r = interest_path(r0, theta, a, sigma_r, dW, dt, num_paths, num_steps , dWq, rho_sr)
100
101 p_delta_T = Hull_White_bp(P_0_T,P_0_t,forward_rate,a,sigma_r,T-delta,T,r)
102
103
104 L_delta = libor(1,p_delta_T,delta)
105
106 L_T = libor(P_0_t,P_0_T,delta)
107
108 # print('LIBOR', L_T)
109 print('L_delta', L_delta)
110
111 # print('stock', s)
112 # print('interest rate', r)
113
114 payoff = calpayoff(k,k_prime,s,L_delta,L_T)
115 expected_payoff = np.mean(payoff)
116 present_value = expected_payoff * np.exp(-rf * T)
117
118 # ipdb.set_trace(context = 28)
119 print('expected payoff', expected_payoff)
120 print('present value', present_value)
121
122 plt.fig = plt.subplots(figsize=(7, 6))
123 plt.plot(r[:50].T)
124 plt.suptitle("Simulated Short Rates")
125 plt.xlabel("Time Steps")
126 plt.ylabel("Short Rate")
127 plt.show()
128
129 plt.fig = plt.subplots(figsize=(7, 6))
130 plt.plot(s[:50].T)
131 plt.suptitle("Simulated Stock Prices")
132 plt.xlabel("Time Steps")
133 plt.ylabel("Stock Price")
134 plt.show()
135
136 # fig, ax = plt.subplots(figsize=(7, 6))
137 # ax.plot(r[:, -2])
138 # ax.set_title("Frequency - L(T-Delta, T-Delta, T)")
139 # ax.set_xlabel(r"$L(T-Delta, T-Delta, T)$")
140 # ax.set_ylabel("Frequency")
141 # plt.show()
142
143
144 # Histograms of L(T-Delta, T-Delta, T) and S(T)/S(0)
```

```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(7, 3))

# Histogram for L(T-Delta, T-Delta, T)
L_T_Delta = r[:, -2]
axes[0].hist(L_T_Delta, bins=30, density=True)
axes[0].set_title("Frequency - L(T-$\Delta$, T-$\Delta$, T)")
axes[0].set_xlabel("L(T-$\Delta$, T-$\Delta$, T)")
axes[0].set_ylabel("Frequency")

# Histogram for S(T)/S(0)
S_ratio = s[:, -1] / s[:, 0]
axes[1].hist(S_ratio, bins=30, density=True)
axes[1].set_title("Frequency - S(T)/S(0)")
axes[1].set_xlabel("Stock Price Ratio")
axes[1].set_ylabel("Frequency")

plt.show()
```

Listing 3: Code

References

[1] Tomas Björk *Arbitrage Theory in Continuous Time*, Oxford university press, 3nd ed, 2009.