IEOR 4735 – Structured and Hybrid Products
Fall 2022

Sajay Velmurugan (SV2692)

December 15, 2022

1 Question

We would like to price a contract paying on the settlement date T + Δ the payoff amount in USD, defined on the maturity date T as:

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

with:

- S(t) the Nikkei-225 spot price at time t, quantoed from JPY into USD
- L(t, T, T + Δ) is the 3-month USD LIBOR rate between T and T + Δ, observed at time t
- Δ is a period of 3-month (0.25 years). T the maturity date (e.g. 3 years) and T + Δ the settlement date (e.g. 3.25 years)
- k , k' given relative strike prices (e.g. both could be 1.00 or ...)

  Provide a pricing routine (e.g. Python script) calculating the price of this contract, taking as inputs: the deal terms (T , Δ, k,k') and the relevant market data (interest rates, volatilities, spot prices, correlations ....). Explain your precise assumptions and methodology choices clearly in an accompanying write-up.

2 Introduction

2.1 Key Products used

Nikkei - 225 is the mostly common used index for the Tokyo Stock exchange. It is measured in Japanese Yen. The price trajectory of Nikkei - 225 over the past year is shown in Figure 1

Figure 1: Nikkei-225 Price Chart

The London Interbank Offered Rate (LIBOR) is a benchmark interest rate at which major global banks lend to one another in the international interbank market for short-term loans. $L(t, T_1, T_2)$ is referred to as Forward LIBOR Rate. It represents the rate determined at time t to lend to another participant at T_1 for a period till T_2.

2.2 About the contract

- **Philosophy:** The philosophy of this contract is to capture the correlation of the Nikkei - 225 and the Libor rate. The investor stands to gain from the opposite movement of Nikkei - 225 and the forward Libor rate.
- **Quanto Options:** The Nikkei - 225 is traded in Japanese Yen (JPY) but the contract is designed in such a way to get paid the return in USD itself. The exchange rate will be fixed to the prevailing exchange rate at inception of the option transaction.
- **Exotic Derivative:** Since the payoff of the contract is multiplication of two traded products, it can be considered as exotic derivative.

3 Assumptions and Methodology Choices

3.1 Modelling of quantoed Nikkei-225 spot price

3.1.1 Modelling of Nikkei-225 spot price

For the Nikkei - 225, it would be priced according to the risk neutral measure of JPY using Black-Scholes model as follows:

$$\frac{dS(t)}{S(t)} = (r_{JPY} - q)dt + \sigma_S dW^{Q_{JPY}} \tag{1}$$

Since we are using Black-Scholes model to price the spot price of Nikkei - 225, we are inherently assuming all the market assumptions of Black-Scholes model like arbitrage-free, no transaction cost market. The major assumption of the stock price is that we are assuming it to be of lognormal distribution. Ideally the distribution of spot price should be identified from data and the **empirical distribution** should be modelled .

Assumption 1. *Nikkei-225 prices follow a lognormal distribution*

In the above equation 1, r_{JPY} is the risk free interest rate of Japan. r_{JPY} could also be modelled with a stochastic differential equation for spot rate of Japanese Yen. For this question, we assume it to be a constant since modelling the spot rate to model the Nikkei - 225 price would be an added complexity. Also the payout is going to be in USD, so modelling of r_{JPY} was not considered and it was assumed to be deterministic.

Assumption 2. *r_{JPY} is assumed to be deterministic.*

Dividend yield (q) is the relation between a stock's annual dividend payout and its current stock price. Depending on how much a stock price moves during the day, the dividend yield is constantly changing as the price of the stock changes. Here we assume the dividend yield (q) of Nikkei-225 to be constant

Assumption 3. *Dividend yield q is assumed to be constant.*

3.1.2 Calculation of σ_S

The contract payout depending on the Nikkei-225 has only one strike price and not multiple strike price. This removes the need of modelling stochastic volatility or local volatility surface.

Assumption 4. *Modelling of stochastic volatility or local volatility is not needed.*

Due to the presence of one strike price, modelling of Nikkei-225 using geometric Brownian motion model should be good enough. To find the σ_S, there are two methods which can be used.

1. Calculate σ_S from historical price (Historical Volatility)
2. Calculate σ_S from current market price (Implied Volatility)

The future generally doesn't follow the historical prices. Thus, usage of historical volatility may lead to usage of wrong assumptions of volatility. Also a typical feature of implied volatility from stock index options is that it is higher than the historical/realized volatility of the index. Hence we use implied volatility from the current market prices of the options. We use T-maturity option prices (T is a parameter) and input it into the Black Scholes model to identify the implied volatility. To find the volatility, trading volume is critical. It is typically highest for the at-the-money (ATM) option contracts. Hence market prices of the at-the-money options are used to calculate the implied volatility.

Assumption 5. *At the money options market price is used to calculate the implied volatility.*

3.1.3 Quanto Option

Since the spot price of Nikkei - 225 is quantoed from JPY to USD, it would be better to price according to risk neutral measure of USD along with quanto adjustment. The pricing equation along with quanto adjustment is given as below:

$$\frac{dS(t)}{S(t)} = (r_{JPY} - q - quantoadjustment)dt + \sigma_S dW^{Q_{USD}}$$

For the quanto adjustment, we further need the foreign exchange rate JPY/USD and also the correlation between JPY/USD and the Nikkei-225 asset price. Foreign exchange rate JPY/USD would be varying along the duration of simulation. Since it is a quanto option which depends on the exchange rate at inception, modelling of the foreign exchange rate JPY/USD wouldn't be needed.

Assumption 6. *Modelling of the foreign exchange rate JPY/USD wouldn't be needed*

After converting the risk neutral measure from JPY to USD, we get the quanto adjustment as follows:

$$quantoadjustment = \rho_{SFX}\sigma_S\sigma_{FX}$$

ρ_{SFX} is the correlation between Nikkei-225 and the JPY/USD rate. σ_S is the volatility of Nikkei - 225 and σ_{FX} is the volatility of JPY/USD foreign exchange rate. Adding the quanto adjustment, the pricing equation is as follows:

$$\frac{dS(t)}{S(t)} = (r_{JPY} - q - \rho_{SFX}\sigma_S\sigma_{FX})dt + \sigma_S dW^{Q_{USD}} \tag{2}$$

Calculation of σ_S has been explained in subsection 3.1.2. Similarly, σ_{FX} can be calculated. Ideally the correlation between exchange rate and stock price should also be calculated from some option price but such options are hard to find. Also the correlation between these two variables are second order. Hence, the correlation can be calculated from historical prices.

Assumption 7. ρ_{SFX} *calculated using historical prices*

3.1.4 Final pricing equation of Nikkei-225

We have all the necessary elements needed for simulating the quantoed prices of Nikkei-225 in USD. The pricing follows a geometric brownian motion and the formula is given as follows:

$$S(t) = S(0)\exp\left((r_{JPY} - q - \rho_{SFX}\sigma_S\sigma_{FX})t + \sigma_S W_1(t)\right) \tag{3}$$

3.2 Modelling of Libor rate

Libor rate can be modelled in different approaches like:

1. **Libor Rate Model** : Using Libor Market Model, we can directly model for Libor rates and then use for pricing the contract. But the pricing would be done with T-Forward as the nurmeraire. Using Girsanov's theorom, this has to be changed to Q measure of money market to be able to use it along with the quanto option part.

2. **Short Rate Model** : Using short rate models like Hull-White and their affine term structure equations, we can find the prices of Zero coupon Bond. Using the ZCB prices, we find the necessary Libor rate.

   For the pricing routine, short rate model approach has been used. This approach wouldn't need any changes of numeraire to use it along with Nikkei - 225 quanto options part.

3.2.1 Modeling of short rate

To model the short rate, the following short rate models were tried:

- Hull White Model
- Ho-Lee Model

Both the models chosen above possess Affine Term Structure. Hence the term structure $\{p(t,T); 0 \leq t \leq T, T > 0\}$ has the form

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

The functions A(t, T) and B(t, T) are deterministic functions.

Hull White Model

Hull White model is a single factor, no-arbitrage yield curve model in which the short term interest rate is driven by a mean reverting process. The dynamics of short rate is given by:

$$dr(t) = \lambda(\theta(t) - r(t))dt + \eta dW(t) \tag{4}$$

$$r(0) = r_0$$

$\theta(t)$ is a time dependent drift term which is generally used to fit the mathematical bond prices to that of the market. Parameter η and λ has to be calibrated for this model. Parameter η determines the volatility and λ is the reversion rate parameter. The time dependent drift term $\theta(t)$ is given by:

$$\theta(t) = \frac{1}{\lambda}\frac{\delta}{\delta t}f^r(0,t) + f^r(0,t) + \frac{\eta^2}{2\lambda^2}(1 - e^{-2\lambda t}) \tag{5}$$

To find the instantaneous forward rate, we use the price of ZCB from the market. Once the market price of ZCB is obtained, we construct a spline for ZCB to get continous values for differentiation.

Assumption 8. *B-spline was constructed to interpolate the market prices of Zero Coupon bond.*

$$f^r(0,t) = -\frac{\delta}{\delta t}\log P_{mkt}(0,t) \tag{6}$$

For pricing the ZCB, we obtain r(t) using simulations of the above stochastic differential equation.

To find the η, volatility of short rate, similar method described to find the volatility of stock price Nikkei-225 has to be used. Here market prices of at the money floorlet needs to be used to find the implied volatility of short rate. Since the floorlet prices are not easily available, historical volatility can also be used.

Assumption 9. *Volatility parameter η can be calibrated using historical short rates or implied volatility can be calculated using market floorlet prices*

To find the reversion rate parameter λ, historic short rates could be used to calibrate or common values of 0.02 or 0.03 could be used.

Assumption 10. *Reversion rate parameter λ can be calibrated using historical short rates or common values of 0.02 or 0.03 can be used.*

Different paths of r(t) obtained using Hull white model are shown in the Figure 6a
As it can be seen from the figure, short rate can even go to negatives in the hull white model.
Once r(t) is obtained, we use the affine term struture to find the price of ZCB as follows.

$$\begin{aligned} p(t,T) &= e^{A(t,T)-B(t,T)r(t)} \\ \tau &= T-t \\ B(\tau) &= \frac{1}{\lambda}(e^{-\lambda\tau}-1) \\ A(\tau) &= \lambda\int_0^\tau \theta(T-z)B(z)dz + \frac{\eta^2}{4\lambda^3}[e^{-2\lambda\tau}(4e^{\lambda\tau}-1)-3] + \frac{\eta^2\tau}{2\lambda^2} \end{aligned} \tag{7}$$

Using the above equations P(t, T) could be calculated.

Ho Lee Model

Ho Lee short rate model is another model which could be used to model short rates. Negative rates are possible in this model but this model doesn't incorporate mean reversion. The dynamics of short rate are given as:

$$dr(t) = \theta(t)dt + \sigma dW(t) \tag{8}$$

$\theta(t)$ is a time dependent drift term which is generally used to fit the mathematical bond prices to that of the market. Parameter σ determines the volatility and can be derived similar to the η as explained in Assumption 9. The time dependent drift term is given as:

$$\theta(t) = \frac{\delta}{\delta t}f^r(0,t) + \sigma^2 t \tag{9}$$

To find the instantaneous forward rate, we use the price of ZCB from the market

$$f^r(0,t) = -\frac{\delta}{\delta t}\log P_{mkt}(0,t)$$

Similar to Hull white model, affine term structure equations could be used to get prices of ZCb for the Ho Lee model as well. Bjorg book provides a closed form solution for pricing ZCB using Ho Lee model in proposition 24.4. For the Ho Lee model, the ZCB prices are given by:

$$p(t,T) = \frac{p^*(0,T)}{p^*(0,t)}\exp\left\{(T-t)f^*(0,t) - \frac{\sigma^2}{2}t(T-t)^2 - (T-t)r(t)\right\} \tag{10}$$

3.2.2 Calculation of Libor rate

Using the definition of Libor rate, we can calculate the Libor rates from the prices of ZCB which have been calculated. The simple forward rate for [S, T] contracted at t, henceforth referred to as the LIBOR forward rate, is defined as

$$L(t;S,T) = -\frac{p(t,T)-p(t,S)}{(T-S)p(t,T)} \tag{11}$$

For the pricing of ZCB, we could either use the Hull white model or the Ho Lee model

3.3 Putting it all together

Using the above modelling of Nikkei-225 price and Libor rate, we have the necessary models to combine them to find the price of the contract.

3.3.1 Correlation between the Wiener process

The main philosophy for this contract is itself to capture the correlation between Nikkei - 225 and the Libor rate. Hence this paramater ρ_{Sr} is of utmost importance for this simulation. Again as discussed before, it is always better to use the implied correlation rather than the realised correlation from historical data. But this contract is exotic and highly liquid. Hence we wouldn't have high trading volume and finding the implied volatility from the prices (even if they exist) wouldn't be accurate.

Hence to find the correlation parameter, we use the historical data of short rate prices and Nikkei - 225 prices. Once the volatility is calculated, we bump the value little higher to be more aggressive in pricing. Generally the historic correlation underestimates the implied correlation. Hence bumping up the historical correlation would make it closer to real correlation value.

Assumption 11. *Correlation between two Wiener process for simulating short rate and Nikkei - 225 prices, ρ_{Sr} are found using historical values and then bumped up by a small percentage to be aggressive and have realistic values*

The trajectory of short rate and Nikkei - 225 are shown in the Figure 5.

Once the correlation is calculated, we use this correlation paramater to find the Wiener process for simulation of short rate and Nikkei - 225 prices

$$\begin{aligned} dW_1(t) &= \mathcal{N}(0, dt) \\ dZ(t) &= \mathcal{N}(0, dt) \\ dW_2(t) &= \rho_{Sr} dW_1(t) + \sqrt{(1 - \rho_{Sr}^2)} dZ(t) \end{aligned} \tag{12}$$

Now we have two correlated Wiener processes of which $W_1(t)$ could be used for simulation of Nikkei - 225 and $W_2(t)$ can be used for simulation of r(t).

3.3.2 Payout of contract at T

Though the contract expires at time T, the settlement happens at $T + \Delta$. This is similar to the Forward rate agreement, where settlement happens in the future. To find the payout at T, the settlement price has to be discounted from $T + \Delta$ to T.

$$V(T) = \frac{\max\left[0, \left(\frac{S(T)}{S(0)} - k\right).\left(k' - \frac{L(T,T,T+\Delta)}{L(0,T,T+\Delta}\right)\right]}{1 + \Delta L(T, T, T + \Delta)} \tag{13}$$

3.3.3 Pricing of the contract

We have been using the Money market as the numeraire for simulation and pricing the contracts. We know that $\frac{V(t)}{M(t)}$ is a martingale. Hence by definition of martingale, we can price the contract at time t = 0 as

$$\frac{V(0)}{M(0)} = E^Q\left[\frac{V(T)}{M(T)} \mid F(0)\right] \tag{14}$$

We know the dynamics of money market as the following:

$$\begin{aligned} dM(t) &= r(t)M(t)dt \\ M(t) &= e^{\int_0^t r(s)ds} \end{aligned} \tag{15}$$

As shown in the above equation, we haven't assumed deterministic short rate for the money market. We have simulated the short rate using Hull White or Ho lee model which could be leveraged.

Assumption 12. *Money market value determined using stochastic short rates*

Thus the final pricing equation is as follows:

$$V(0) = E^Q\left[e^{-\int_0^T r(s)ds}V(T) \mid F(0)\right] \tag{16}$$

In this final equation, we have simulated runs of V(T) and and r(t), so we will be able to find the price of this contract.

3.4 Assumptions

All the assumptions used throughout the modelling exercise have been called out here from the individual sections discussed above.

1. Nikkei-225 prices follow a lognormal distribution
2. r_{JPY} is assumed to be deterministic.
3. Dividend yield q is assumed to be constant.
4. Modelling of stochastic volatility or local volatility is not needed.
5. At the money options market price is used to calculate the implied volatility.
6. Modelling of the foreign exchange rate JPY/USD wouldn't be needed
7. ρ_{SFX} calculated using historical prices
8. B-spline was constructed to interpolate the market prices of Zero Coupon bond.
9. Volatility parameter η can be calibrated using historical short rates or implied volatility can be calculated using market floorlet prices
10. Reversion rate parameter λ can be calibrated using historical short rates or common values of 0.02 or 0.03 can be used.
11. Correlation between two Wiener process for simulating short rate and Nikkei - 225 prices, ρ_{Sr} are found using historical values and then bumped up by a small percentage to be aggressive and have realistic values
12. Money market value determined using stochastic short rates

4 Results

4.1 Quantoed Nikkei - 225 spot price

4.1.1 Constants of the model

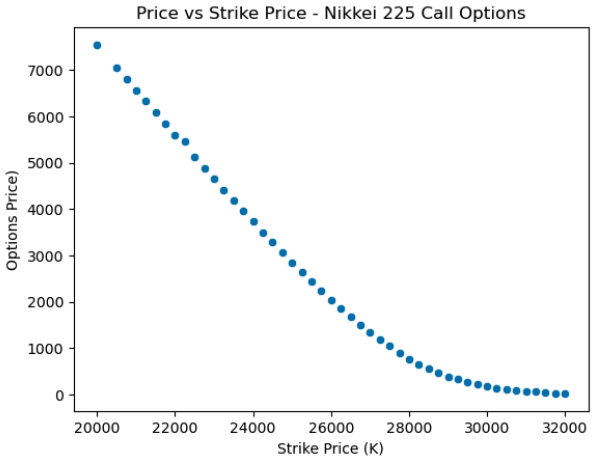
For the model defined in Equation 2, multiple constants needs to be obtained from market data or calibrated from market data.

The constants used in the model are as follows:

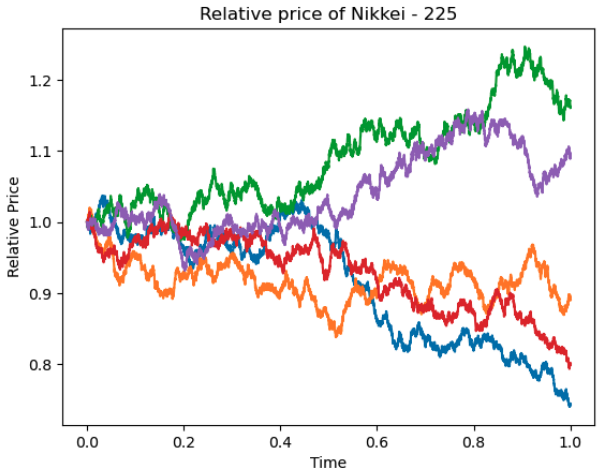
- Risk free rate of Japanese Yen, r_{JPY} : **-0.089%** - Obtained from market data

- Dividend yield of Nikkei - 225, q : **1.8%** - Obtained from market data
- Correlation between exchange rate and index price : **0.172** - Calculated from historical data
- Volatility of exchange rate : **0.131** - Implied volatility data from market
- Volatility of Nikkei - 225 : **0.12** (For k = 1) - Calibrated through Black Scholes model by adjusting the strike price according to parameter K

Market data for Nikkei - 225 call options price for different values of strike price shown in Figure 2a



(a) Nikkei - 225 call options price



(b) Nikkei - 225 relative price

4.1.2 Simulated Paths

The simulated paths of relative quantoed price $\frac{S(t)}{S(0}$ are shown in Figure 2b

4.2 Short rate

For simulation of short rate, the market prices of Zero coupon are needed. The market price of ZCB for different maturities are shown in the Figure 3

A B-spline was constructed on these market prices to get interpolate the values of ZCB for continous time differentiation.

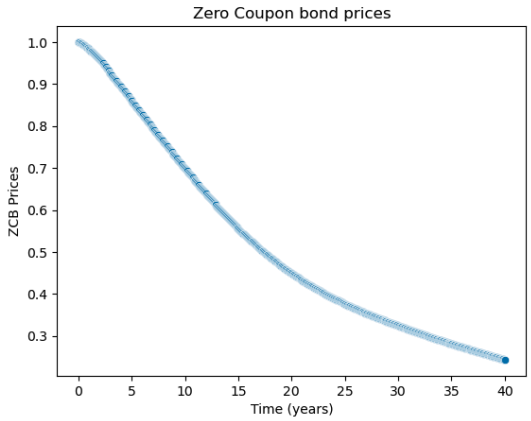
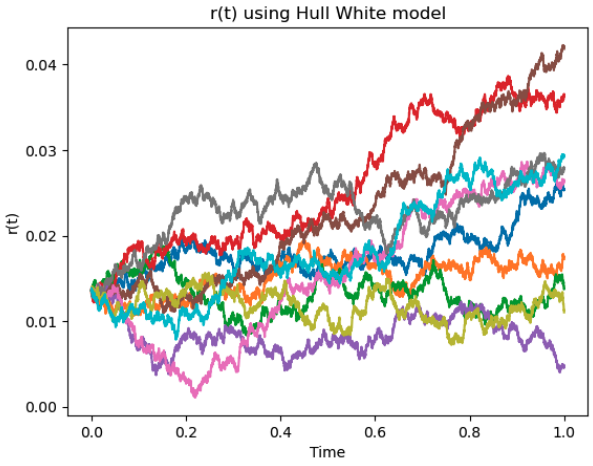


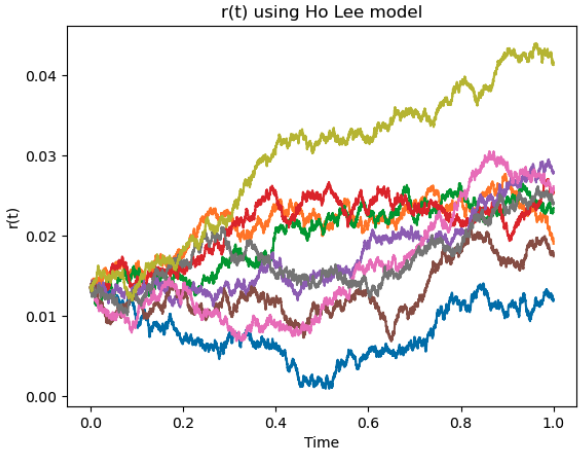
Figure 3: Market price of ZCB

4.2.1 Hull White Model

For Hull White model, λ was taken to be 0.03. The volatility of short rate η was calculated to be 0.008 Different paths of r(t) obtained using Hull white model are shown in the Figure 6a



(a) Short rate obtained using Hull White Model



(b) Short rate obtained using Ho Lee Model

The mean reversion property of short rate in the Hull White model can be seen in the trajectories

4.2.2 Ho Lee Model

For pricing using Ho-Lee model, we need only one parameter, the volatility of short rate σ. It was calculated to be 0.008 Different paths of r(t) obtained using Hull white model are shown in the Figure 6b On comparing the r(T) values from simulations of Hull White and Ho Lee model, we can infer that short rate obtained from Ho - Lee model seems to have higher volatility than Hull - White model. The analysis is shown in the below table 1

Short Rate Model	Mean	Standard Deviation
Hull White	2.15%	0.0083
Ho Lee	2.12%	0.0096

Table 1: Simulated Short rate at time T

4.3 Pricing the contract

4.3.1 Correlation between S(t) and r(t)

The trajectory of short rate and Nikkei - 225 are shown in the Figure 5. The historical correlation calculated between these trajectories were **0.4**. For the simulation, correlation was bumped by 10% to 0.44

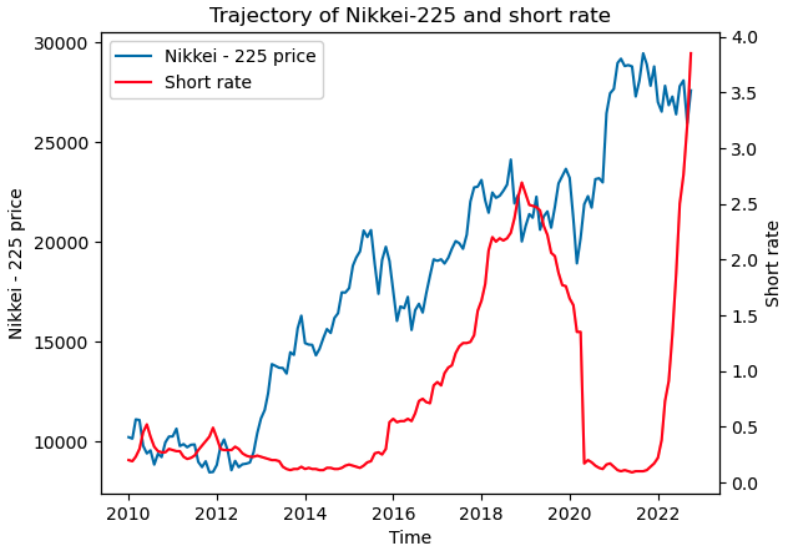
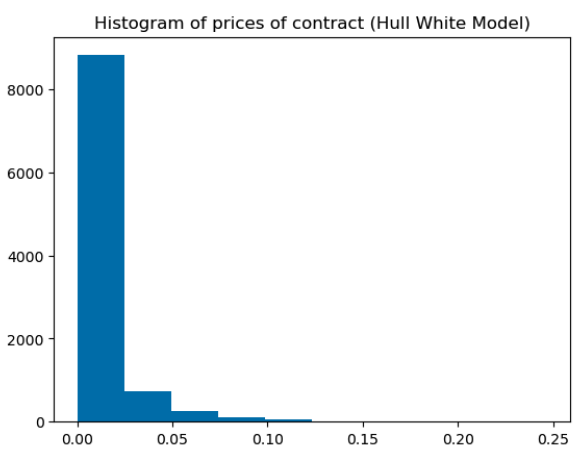


Figure 5: Historical trajectory of Nikkei - 225 and short rate

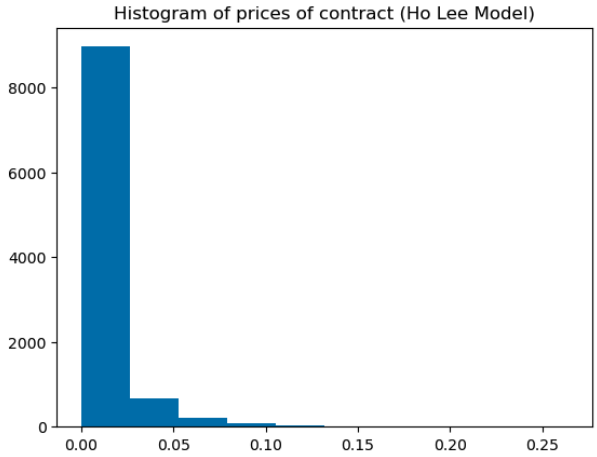
4.3.2 Pricing of contract

The contract was priced using the equation 16 using the default parameters given in the question. The number of time steps taken was 10,000. Also the number of simulations was also 10,000.

Short rates were modelled using two different models and hence we had two different pricing routines. The histogram of prices calculated by both models are shown below in Figure.



(a) Histogram of prices using Hull White Model



(b) Histogram of prices using Ho Lee Model

The contract price was obtained around **0.0070 - 0.0078** depending on the short rate model chosen and also the different assumptions taken throughout. The price of the contract **had higher standard deviation** when Ho-Lee model was used for simulation of short rate.

5 Conclusions

A pricing routine has been created for the given contract. The general approach taken was to model the index price using Geometric brownian motion and libor rate by simulation of short rate. Other methods which could be evaluated are to build an empirical distribution for index prices and libor rate model instead of assuming certain distributions. This could have led to more realistic prices for these contracts. Another method to simulate Libor is to directly use the Libor rate model. Even in short rate models, there are different models with higher degree of freedom which could be tried on. We could have also tried obtaining closed form solution for the given contract

References

[1] Tomas Bjork. *Arbitrage Theory in Continuous Time*. OUP Catalogue 9780199574742. Oxford University Press, Nov. 2009. ISBN: ARRAY(0x5af015f0). URL: https://ideas.repec.org/b/oxp/obooks/9780199574742.html.

[2] *Nikkei 225 Options*. https://www.jpx.co.jp/english/derivatives/products/domestic/225options/index.html. Accessed: 2022-12-15.

[3] Cornelis Oosterlee and Lech Grzelak. *Mathematical Modeling and Computation in Finance: With Exercises and Python and MATLAB Computer Codes*. Dec. 2019. ISBN: 978-1-78634-794-7. DOI: 10.1142/q0236.

[4] *Short rate prices*. https://data.oecd.org/interest/short-term-interest-rates.htm. Accessed: 2022-12-15.

[5] F. de Weert. *Exotic Options Trading*. The Wiley Finance Series. Wiley, 2011. ISBN: 9781119995180. URL: https://books.google.com/books?id=eCvSMeLhvhkC.

IEOR 4735 – Structured and Hybrid Products

Name: Sajay Velmurugan

UNI: SV2692

Imports

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import scipy.integrate as integrate
#from scipy.interpolate import interp1d
from scipy import interpolate
import scipy.stats as st
import yfinance as yahooFinance
import pandas as pd
```

Parameters

All the necessary market data and parameters have been defined in a dictionary which can be used through out the program.

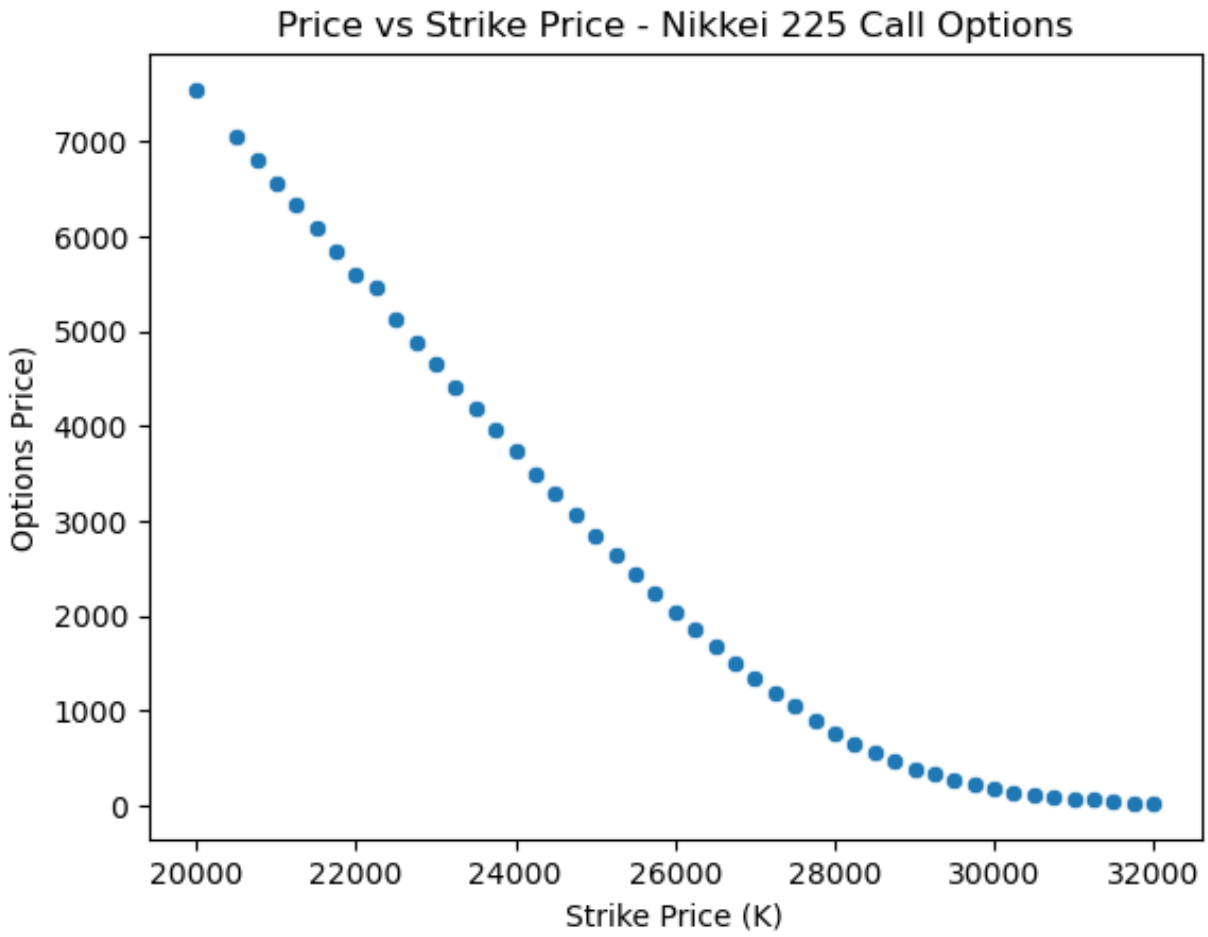
Any parameters which needs to be modifed for the pricing routine can be modifed in the pricing_params key in the dictionar

In [2]:
```
params = {
    'pricing_params' : {
        'K' : 1,
        'K_prime' : 1,
        'delta' : 0.25,
        'T' : 1
    },
    'simulation_params' : {
        'num_paths' : 10000,
        'num_steps' : 10000,
    },
    'nikkei_225' : {
        "options_K" : np.array([20000, *range(20500, 32250, 250)]),
        "options_price" : np.array([7545, 7060, 6815, 6570, 6330, 6090, 5
                4420, 4190, 3960, 3730, 3505, 3285, 3065, 2850, 2640, 24
                1850, 1675, 1500, 1340, 1185, 1045, 890, 770, 655, 560,
                220, 180, 145, 115, 95, 76, 60, 48, 36, 28]),
        "current_price" : 27000,
        "dividend_yield" : 0.018

    },
    'rf_yen' : -0.089/100,
    'JPY_USD' : {
        "sigma_JPY_USD" : 0.131729
    },
    'ZCB' : {
        'market_price' : [1.0,0.999966573000000,0.998930882000000,0.99782
        'market_time' : [0.0,0.00273972600000000,0.0876712330000000,0.172
    },
    'spot_rate': {
        # https://data.oecd.org/interest/short-term-interest-rates.htm Ja
        'historical' : [0.2,0.19,0.23,0.3,0.45,0.52,0.41,0.32,0.28,0.27,0
    },

    'Hull_White': {
        # TO verify
        'lambd' : 0.03
    }
}
```

Market Data of Nikkei - 225 Options price

In [3]:
```
# March 2023 expiry  12/08 prices

K = np.array([20000, *range(20500, 32250, 250)])
price = np.array([7545, 7060, 6815, 6570, 6330, 6090, 5840, 5600, 5465, 5
                4420, 4190, 3960, 3730, 3505, 3285, 3065, 2850, 2640, 24
                1850, 1675, 1500, 1340, 1185, 1045, 890, 770, 655, 560,
                220, 180, 145, 115, 95, 76, 60, 48, 36, 28])

sns.scatterplot(x = K, y = price)
plt.title('Price vs Strike Price - Nikkei 225 Call Options')
plt.xlabel('Strike Price (K)')
plt.ylabel('Options Price)')
```

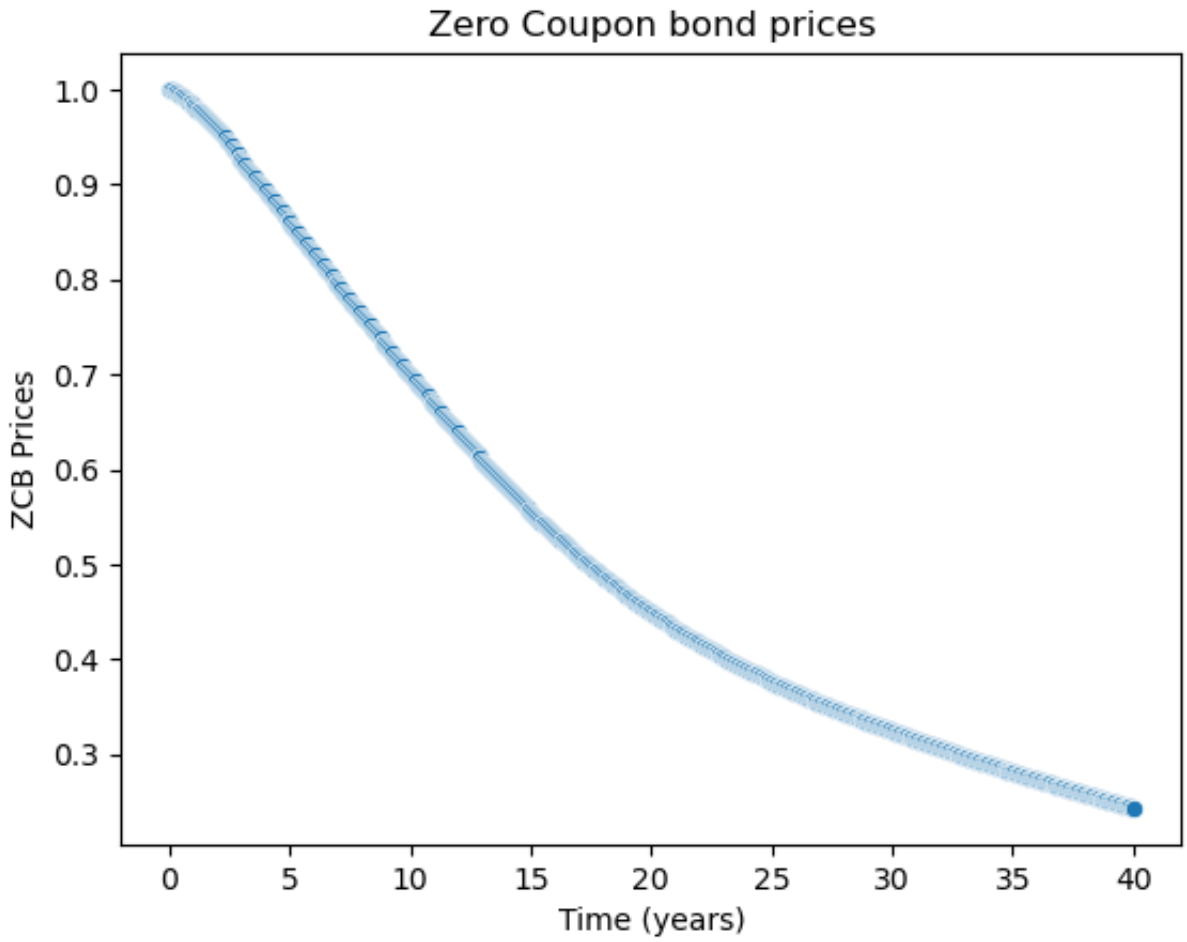
Out[3]: Text(0, 0.5, 'Options Price)')



Market Data of ZCB price

In [4]:
```
sns.scatterplot(x = params['ZCB']['market_time'], y = params['ZCB']['mark
plt.title('Zero Coupon bond prices')
plt.xlabel('Time (years)')
plt.ylabel('ZCB Prices')
```

Out[4]: Text(0, 0.5, 'ZCB Prices')



Calculation of Implied Volatility

Newton's raphson method is used to minimise the difference between the options price calculated using Black scholes model and the options price obtained from the market

In [5]:
```
def ImpliedVolatility(CP,S_0,K,sigma,tau,r, V_market):


    error    = 1e10; # initial error

    optPrice = lambda sigma: BS_Call_Option_Price(CP,S_0,K,sigma,tau,r)
    vega= lambda sigma: dV_dsigma(S_0,K,sigma,tau,r)


    n = 1.0
    while error>10e-10:
        g         = optPrice(sigma) - V_market
        g_prim    = vega(sigma)
        sigma_new = sigma - g / g_prim

        #error=abs(sigma_new-sigma)
        error=abs(g)
        sigma=sigma_new;

#         print('iteration {0} with error = {1}'.format(n,error))

        n= n+1
    return sigma

def dV_dsigma(S_0,K,sigma,tau,r):
    #parameters and value of Vega
    d2    = (np.log(S_0 / float(K)) + (r - 0.5 * np.power(sigma,2.0)) * ta
    value = K * np.exp(-r * tau) * st.norm.pdf(d2) * np.sqrt(tau)
    return value

def BS_Call_Option_Price(CP,S_0,K,sigma,tau,r):

    d1    = (np.log(S_0 / float(K)) + (r + 0.5 * np.power(sigma,2.0)) * t
    d2    = d1 - sigma * np.sqrt(tau)


    if str(CP).lower()=="c" or str(CP).lower()=="1":
        value = st.norm.cdf(d1) * S_0 - st.norm.cdf(d2) * K * np.exp(-r *
    elif str(CP).lower()=="p" or str(CP).lower()=="-1":
        value = st.norm.cdf(-d2) * K * np.exp(-r * tau) - st.norm.cdf(-d1
    return value
```

Given the mean and sigma, Geometric brownian motion simulation would be generated. Vectrosised form has been implemented taking into account any number of paths and steps

In [6]:
```
def geometric_brownian_motion(mu, sigma, T, S_0, num_paths, num_steps, W)
    """
    Fix the initial part
    """


    X = np.zeros(shape = [num_paths, num_steps + 1])
    S = np.zeros(shape = [num_paths, num_steps + 1])

    time = np.linspace(0, T, num_steps + 1)

    S = S_0 * np.exp((mu - 0.5 * sigma **2) * time + sigma*W)
    return S
```

Simulation of Quantoed Nikkei - 225

Function to calculate the historical correlation between Nikkei - 225 and Fx rate. The market prices have been obtained from Yahoo finance

In [7]:
```
def correlation_S_Fx():


    jpy_usd_prices = yahooFinance.Ticker("JPYUSD=X")
    jpy_usd_prices_6month = jpy_usd_prices.history(period="6mo").reset_in
    jpy_usd_prices_6month['Date'] = pd.to_datetime(jpy_usd_prices_6month[
    jpy_usd_prices_6month = jpy_usd_prices_6month.set_index(pd.DatetimeIn
    jpy_usd_prices_6month = jpy_usd_prices_6month['Close']



    nikkei_225_prices = yahooFinance.Ticker("^N225")
    nikkei_225_prices_6month = nikkei_225_prices.history(period="6mo")

    nikkei_225_prices_6month = nikkei_225_prices_6month.reset_index()
    nikkei_225_prices_6month['Date'] = pd.to_datetime(nikkei_225_prices_6
    nikkei_225_prices_6month = nikkei_225_prices_6month.set_index(pd.Date
    nikkei_225_prices_6month = nikkei_225_prices_6month['Close']
    combined = pd.merge(nikkei_225_prices_6month, jpy_usd_prices_6month,
    rho_S_Fx = combined.corr().iloc[0, 1]

    return rho_S_Fx
```

Quanto Option Part

Simulation of the below equation is being done here

$$\left(\frac{S(T)}{S(0)} - k\right)$$

To determine the implied volatility, we need to use the at the money call options price. So according to the parameter K choosen in the pricing routine, the at the money strike price would differ. So at the money options price corresponding to the nearest strike price calculated based on K has been chosen for calculating the implied volatility

In [8]:
```
def quanto_option(params, W):
    """
    K - relative strike price
    r_d - Domestic risk free rate (Japan interest free rate in this case)

    """


    K = params['pricing_params']['K']
    r_d = params['rf_yen']
    rho_S_Fx = correlation_S_Fx()
    sigma_Fx = params['JPY_USD']['sigma_JPY_USD']
    S_0 = params['nikkei_225']['current_price']
    q = params['nikkei_225']['dividend_yield']

    T = params['pricing_params']['T']
    num_paths = params['simulation_params']['num_paths']
    num_steps = params['simulation_params']['num_steps']

    nikkei_options_price = {}
    for i in range(len(params['nikkei_225']['options_K'])):
        nikkei_options_price[params['nikkei_225']['options_K'][i]] = para

    # Finds the implied volatity - exmplain more
    K_S = min(nikkei_options_price.keys(), key=lambda k: abs(k-K*S_0))
    V_mkt = nikkei_options_price[K_S]

    sigma_S = ImpliedVolatility('c', S_0, K_S, 0.1, T, r_d, V_mkt)

    mu = r_d - q - rho_S_Fx*sigma_S*sigma_Fx
    sigma = sigma_S
    asset_price = geometric_brownian_motion(mu, sigma, T, S_0, num_paths,
    relative_price = asset_price[:, -1]/S_0

    quanto_option_part = (relative_price - K)

    return quanto_option_part
```

LIBOR Forwards option part

Simulation of the following part is done here

$$\left(k' - \frac{L(T,T,T+\Delta}{L(0,T,T+\Delta}\right)$$

The entire contract is priced by the following equation:

$$V(0) = E^Q\left[e^{-\int_0^T r(s)ds}\frac{\max\left[0,\left(\frac{S(T)}{S(0)}-k\right).\left(k'-\frac{L(T,T,T+\Delta)}{L(0,T,T+\Delta}\right)\right]}{1+\Delta L(T,T,T+\Delta)} \mid F(0)\right]$$

Of which three parts need the short rate simulation:

1. The floorlet option
2. Discouting the payment at $T+\Delta$ to time T
3. Discounting using money market from T to 0

Hence we combine these three part together and call them as the floorlet_part in the below function. The three parts combined together will look like

$$e^{-\int_0^T r(s)ds}\frac{\left(k'-\frac{L(T,T,T+\Delta)}{L(0,T,T+\Delta}\right)}{1+\Delta L(T,T,T+\Delta)}$$

Simulation of short rate and calculation of Libor forward rate

Short rate simulated using Hull White

The definitions are given in the project report

In [9]:
```
def HW_A(lambd,eta,P0T,t,T, theta):

    tau = T-t
    zGrid = np.linspace(0.0,tau,250)

    B_r = lambda tau: 1.0/lambd * (np.exp(-lambd *tau)-1.0)

    temp1 = lambd * integrate.trapz(theta(T-zGrid)*B_r(zGrid),zGrid)

    temp2 = eta*eta/(4.0*np.power(lambd,3.0)) * (np.exp(-2.0*lambd*tau)*(

    return temp1 + temp2

def HW_B(lambd,eta,t,T):
    return 1.0/lambd *(np.exp(-lambd*(T-t))-1.0)


def HW_P_tT(lambd,eta, P0T, t, T, R, theta):

    num_paths = params['simulation_params']['num_paths']
    num_steps = params['simulation_params']['num_steps']
    delta = params['pricing_params']['delta']
    dt = (T + delta)/num_steps
```

```
    if(t == T):
        return np.ones([params['simulation_params']['num_paths']])



    t_index = int(t/dt)
    T_index = int(T/dt)
    R_t = R[:, t_index]

    B_r = HW_B(lambd,eta,t,T)
    A_r = HW_A(lambd,eta,P0T,t,T, theta)

    P_tT = np.exp(A_r + B_r *R_t)

    return P_tT



def floorlet_part_HW(params, W):

    ti = params['ZCB']['market_time']
    Pi = params['ZCB']['market_price']
    interpolator =  interpolate.splrep(ti, Pi, s=0.00001)

    P0T = lambda T: np.exp(interpolate.splev(T, interpolator, der=0))

    # time step for differentiatation dt = 0.001
    f0T = lambda T: - (np.log(P0T(T + 0.001)) - np.log(P0T(T - 0.001)))/(

    r0 = f0T(0)

    K_prime = params['pricing_params']['K_prime']
    # If options price was avaialble, the implied volatility function cou
    eta = np.std(np.array(params['spot_rate']['historical'])/100)

    lambd = params['Hull_White']['lambd']

    T = params['pricing_params']['T']
    delta = params['pricing_params']['delta']
    num_paths = params['simulation_params']['num_paths']
    num_steps = params['simulation_params']['num_steps']
    dt = (T + delta)/num_steps

    theta = lambda t: 1.0/lambd * (f0T(t+dt)-f0T(t-dt))/(2.0*dt) + f0T(t)


    Z = np.random.normal(0.0,1.0,[num_paths,num_steps])
    R = np.zeros([num_paths, num_steps+1])
    R[:,0] = r0
    time = np.zeros([num_steps+1])

    for i in range(0,num_steps):
        R[:,i+1] = R[:,i] + lambd*(theta(time[i]) - R[:,i]) * dt + eta* (
        time[i+1] = time[i] +dt

    t_index = int(T/dt)
```

```
    M_t = np.zeros([num_paths,num_steps])
    for i in range(0,num_paths):
        M_t[i,:] = np.exp(np.cumsum(R[i,:-1])*dt)

    floorlet_part_ = K_prime - (HW_P_tT(lambd,eta, P0T, T, T, R, theta)/H
    discount_part = HW_P_tT(lambd,eta, P0T, T, T, R, theta)/HW_P_tT(lambd

    t_index = int(T/dt)
    M_T = M_t[:, t_index]

    return floorlet_part_/(discount_part*M_T)
```

Short rate simulated using Ho-Lee model

The definitions are given in the project report

In [10]:
```
def P_tT_HL(params, t, T, P0T, f0T, sigma, R):

    num_paths = params['simulation_params']['num_paths']
    num_steps = params['simulation_params']['num_steps']
    delta = params['pricing_params']['delta']
    dt = (T + delta)/num_steps
    if(t == T):
        return np.ones([params['simulation_params']['num_paths']])

    t_index = int(t/dt)
    T_index = int(T/dt)
    R_t = R[:, t_index]

    P_tT = P0T(T)/P0T(t)*np.exp((T - t) * f0T(t) - sigma*sigma/2*t*((T -

    return P_tT


def floorlet_part_HL(params, W):

    ti = params['ZCB']['market_time']
    Pi = params['ZCB']['market_price']
    interpolator =  interpolate.splrep(ti, Pi, s=0.00001)

    P0T = lambda T: np.exp(interpolate.splev(T, interpolator, der=0))

    # time step for differentiatation dt = 0.001
    f0T = lambda T: - (np.log(P0T(T + 0.001)) - np.log(P0T(T - 0.001)))/(

    r0 = f0T(0)

    K_prime = params['pricing_params']['K_prime']
    sigma = np.std(np.array(params['spot_rate']['historical'])/100)

    lambd = params['Hull_White']['lambd']

    T = params['pricing_params']['T']
```

```
    delta = params['pricing_params']['delta']
    num_paths = params['simulation_params']['num_paths']
    num_steps = params['simulation_params']['num_steps']
    dt = (T + delta)/num_steps

    theta = lambda t: (f0T(t+dt)-f0T(t-dt))/(2.0*dt) + sigma**2.0*t

    R = np.zeros([num_paths, num_steps+1])
    M_t = np.zeros([num_paths,num_steps+1])
    M_t[:,0]= 1.0
    R[:,0]=r0
    time = np.zeros([num_steps+1])

    dt = T / float(num_steps)
    for i in range(0,num_steps):

        R[:,i+1] = R[:,i] + theta(time[i]) * dt + sigma* (W[:,i+1]-W[:,i]
        M_t[:,i+1] = M_t[:,i] * np.exp((R[:,i+1]+R[:,i])*0.5*dt)
        time[i+1] = time[i] +dt


    t_index = int(T/dt)


    P_TT = P_tT_HL(params, T, T, P0T, f0T, sigma, R)
    P_T_TD = P_tT_HL(params, T, T + delta, P0T, f0T, sigma, R)
    P_0_T = P_tT_HL(params, 0, T, P0T, f0T, sigma, R)
    P_0_TD = P_tT_HL(params, 0, T + delta, P0T, f0T, sigma, R)

    floorlet_part_ = K_prime - (P_TT/P_T_TD - 1)/(P_0_T/P_0_TD - 1)
    discount_part = P_TT/P_T_TD

    t_index = int(T/dt)
    M_T = M_t[:, t_index]

    return floorlet_part_/(discount_part*M_T)
```

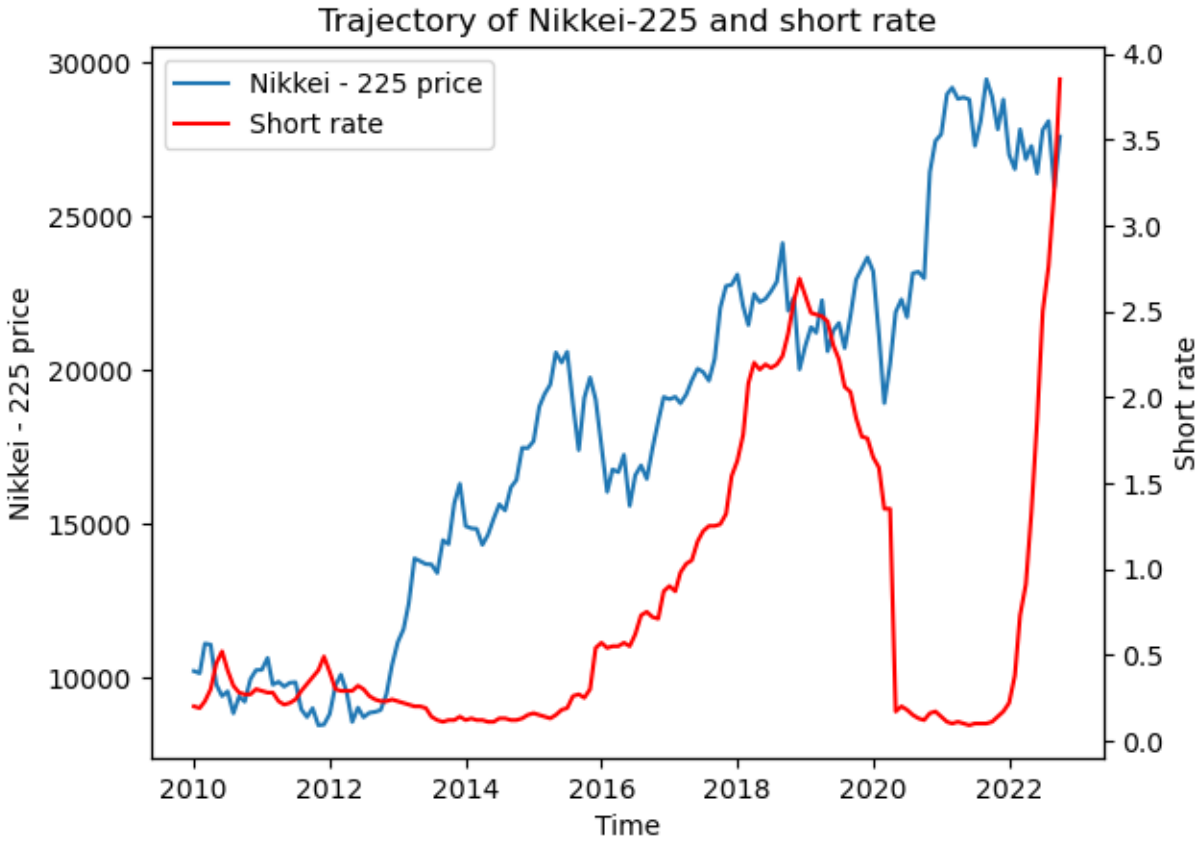
Putting it all together

Correlation between the wiener process for simulations of short rate and Nikkei-225 have been found below from the historical prices and have been bumped up to be aggressive since realised correlation is generally lesser than implied correlation

In [11]:
```
def correlation_S_r():
    historical_r = np.array(params['spot_rate']['historical'])

    nikkei_225_prices = yahooFinance.Ticker("^N225")
    nikkei_225_prices_close = nikkei_225_prices.history(period="6mo", int
    nikkei_225_prices_close = nikkei_225_prices_close['Close'].values

    rho_S_r = np.corrcoef(historical_r, nikkei_225_prices_close)[0, 1]
    return rho_S_r
```

Trajectory of Nikkei - 225 and short rates

In [12]:
```
# def correlation_S_r():
historical_r = np.array(params['spot_rate']['historical'])

nikkei_225_prices = yahooFinance.Ticker("^N225")
nikkei_225_prices_close = nikkei_225_prices.history(period="6mo", interva
x = nikkei_225_prices_close.index
nikkei_225_prices_close = nikkei_225_prices_close['Close'].values

rho_S_r = np.corrcoef(historical_r, nikkei_225_prices_close)[0, 1]
#     return rho_S_r/

fig, ax1 = plt.subplots()

ax2 = ax1.twinx()
ax1.plot(x, nikkei_225_prices_close, label='Nikkei - 225 price')
ax2.plot(x, historical_r, 'r-', label='Short rate')

# lns = ax1 + ax2
# labs = [l.get_label() for l in lns]
# ax2.legend(lns, labs, loc=0)
lines, labels = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax1.legend(lines + lines2, labels + labels2, loc=0)


ax1.set_xlabel('Time')
ax1.set_ylabel('Nikkei - 225 price')
ax2.set_ylabel('Short rate')
# plt.legend(loc="upper left")
plt.title('Trajectory of Nikkei-225 and short rate')
plt.show()
```


Trajectory of Nikkei-225 and short rate
Nikkei - 225 price
Short rate
Nikkei - 225 price
Short rate
Time
30000
25000
20000
15000
10000
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2010
2012
2014
2016
2018
2020
2022

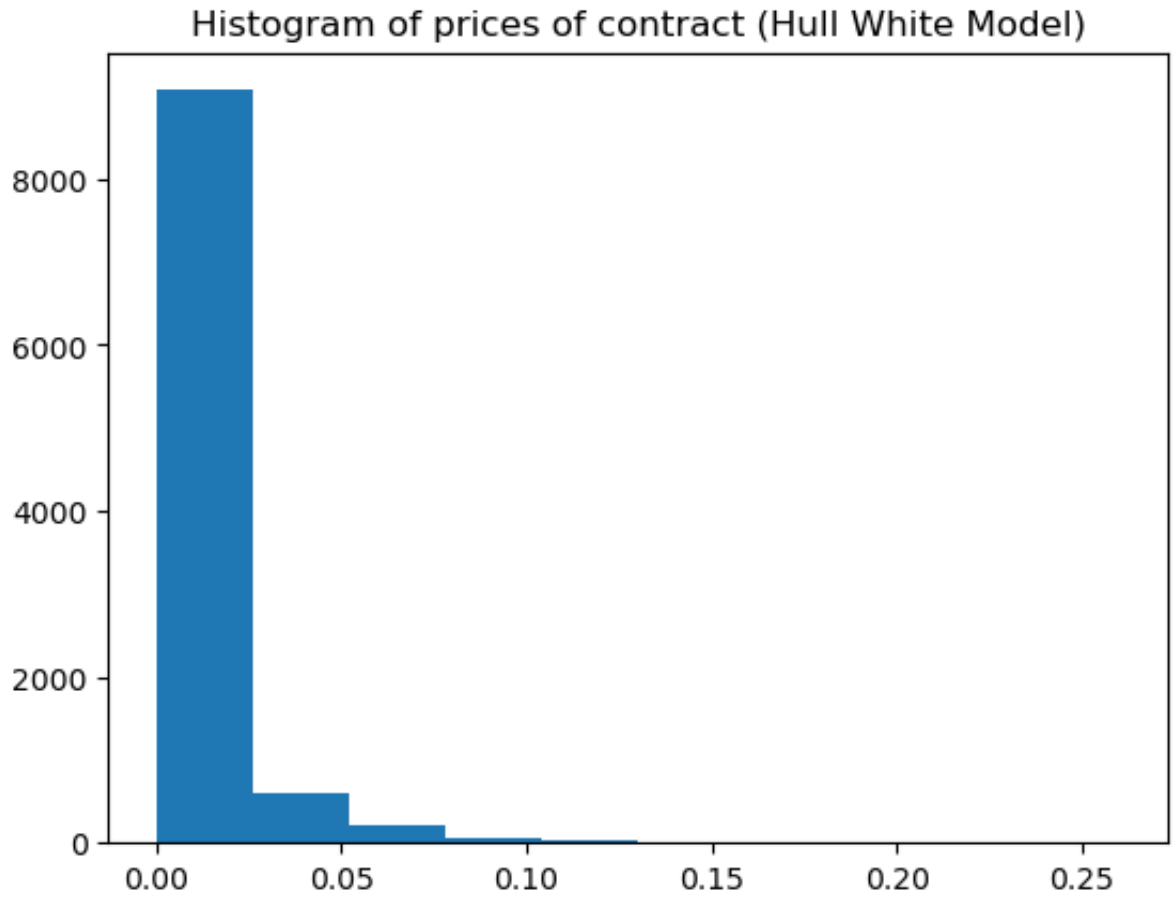
In [13]:
```
def pricing_exotic_option_HW(params):

    T = params['pricing_params']['T']
    delta = params['pricing_params']['delta']
    num_paths = params['simulation_params']['num_paths']
    num_steps = params['simulation_params']['num_steps']
    dt = (T + delta)/num_steps

    rho_S_r = correlation_S_r()
    rho_S_r = rho_S_r*1.1

    # dW for the equity part
    W1 = np.zeros(shape = [num_paths, num_steps + 1])
    dW1 = np.random.normal(0, np.power(dt, 0.5), size = [num_paths, num_s
    W1[:, 1:] = np.cumsum(dW1, axis = 1)

    # dW for the interest rate part
    dZ = np.random.normal(0, np.power(dt, 0.5), size = [num_paths, num_st
    W2 = np.zeros(shape = [num_paths, num_steps + 1])
    dW2 = rho_S_r*dW1 + np.sqrt(1 - (rho_S_r**2))*dZ
    W2[:, 1:] = np.cumsum(dW2, axis = 1)

    # Simulating the quanto option part
    quanto_option_part = quanto_option(params, W1)
    # Simulating the libor rate part
    floorlet_part_ = floorlet_part_HW(params, W2)
#     print(np.mean(floorlet_part_))
    price = np.mean(np.maximum(quanto_option_part*floorlet_part_, 0))
    plt.hist(np.maximum(quanto_option_part*floorlet_part_, 0), density=Fa
    plt.title("Histogram of prices of contract (Hull White Model)");
    print(f'Price of the contract: {price}')
    return price, np.std(np.maximum(quanto_option_part*floorlet_part_, 0)
```

In [14]:
```
def pricing_exotic_option_HL(params):

    T = params['pricing_params']['T']
    delta = params['pricing_params']['delta']
    num_paths = params['simulation_params']['num_paths']
    num_steps = params['simulation_params']['num_steps']
    dt = (T + delta)/num_steps

    rho_S_r = correlation_S_r()
    rho_S_r = rho_S_r*1.1

    # dW for the equity part
    W1 = np.zeros(shape = [num_paths, num_steps + 1])
    dW1 = np.random.normal(0, np.power(dt, 0.5), size = [num_paths, num_s
    W1[:, 1:] = np.cumsum(dW1, axis = 1)

    # dW for the interest rate part
    dZ = np.random.normal(0, np.power(dt, 0.5), size = [num_paths, num_st
    W2 = np.zeros(shape = [num_paths, num_steps + 1])
    dW2 = rho_S_r*dW1 + np.sqrt(1 - (rho_S_r**2))*dZ
    W2[:, 1:] = np.cumsum(dW2, axis = 1)

    # Simulating the quanto option part
    quanto_option_part = quanto_option(params, W1)
    # Simulating the libor rate part
    floorlet_part_ = floorlet_part_HL(params, W2)

    price = np.mean(np.maximum(quanto_option_part*floorlet_part_, 0))
    plt.hist(np.maximum(quanto_option_part*floorlet_part_, 0), density=Fa
    plt.title("Histogram of prices of contract (Ho Lee Model)");
    print(f'Price of the contract: {price}')

    return price, np.std(np.maximum(quanto_option_part*floorlet_part_, 0)
```

In [15]:
```
# Pricing of the contract using Hull white model for simulating short rat
pricing_exotic_option_HW(params)
```

```
Price of the contract: 0.007451218423250279
```

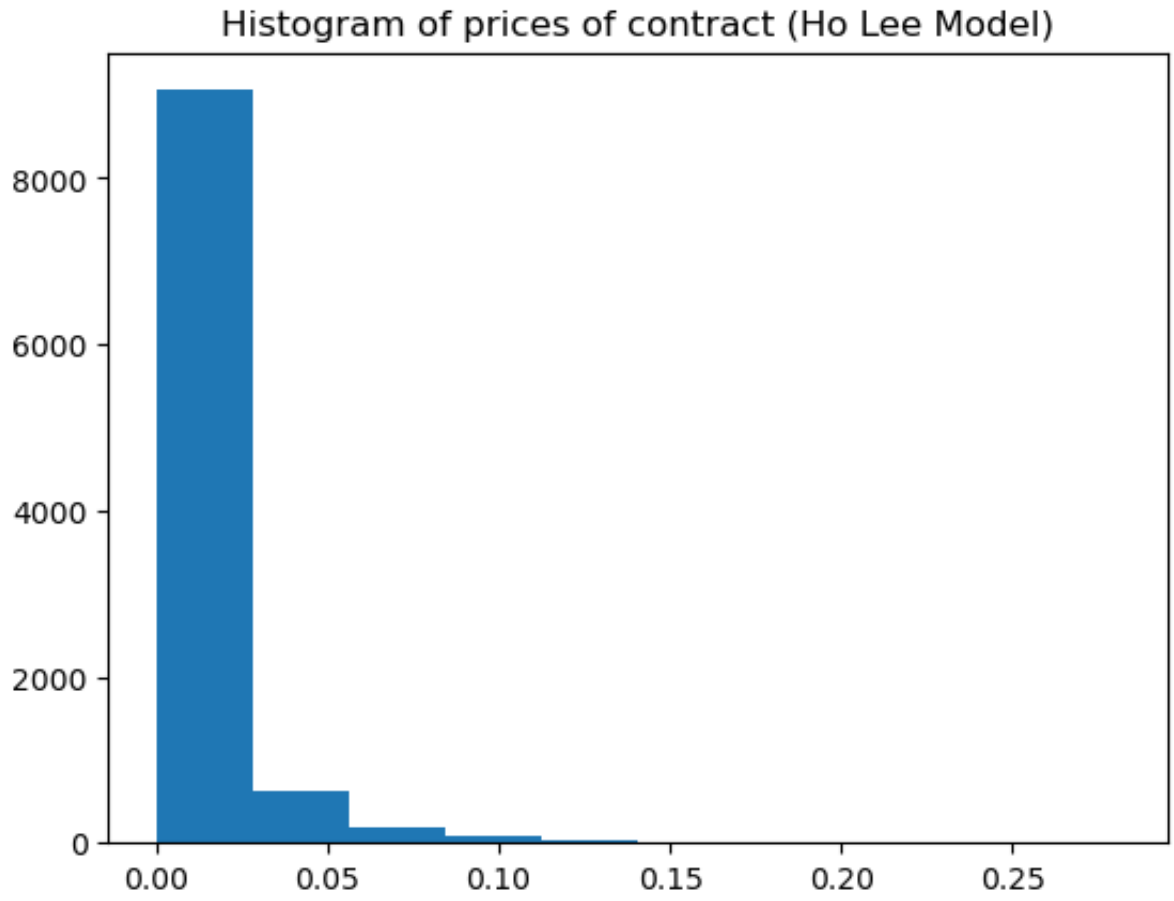
Out[15]:
```
(0.007451218423250279, 0.01775405464243452)
```



In [16]:
```
# Pricing of the contract using Ho Lee model for simulating short rate
pricing_exotic_option_HL(params)
```

```
Price of the contract: 0.007701571021378513
```

Out[16]:
```
(0.007701571021378513, 0.018081566404472112)
```


Histogram of prices of contract (Ho Lee Model)
8000
6000
4000
2000
0
0.00
0.05
0.10
0.15
0.20
0.25

In [ ]: