Simulate Asset Price Evolutions and Reprice Risky up-and-out Call Option

Ng, Joe Hoong
ng_joehoong@hotmail.com

Nguyen, Dang Duy Nghia
nghia002@e.ntu.edu.sg

Ansari, Zain Us Sami Ahmed
zainussami@gmail.com

Thorne, Dylan
dylan.thorne@gmail.com

May. 10, 2020

Keywords: European Options, Non-constant interest rate, Local volatility, LIBOR forward rates, Zero-coupon bond and Stochastic Volatility.

Abstract

This report presents the results of the simulation of a European up-and-out call option over twelve months, but the difference between this submission and submission 1 is the implementation of a non-constant interest rate and local volatility. We use LIBOR forward rate model to simulate interest rates. We also use discount factor to value the option without default risk and then the value of the option with counterparty default risk.

1 Introduction

In this paper, we go beyond the constant risk-free continuously-compounded rate. There are several models to implement stochastic interest rates. The short rate models which describe instantaneous continuously-compounded interest rate at time t, r_t include the Vasicek model introduced in [1], Mamon in [2] advances this model by presenting approaches in obtaining the closed-form solution using the Vasicek model. Under the Vasicek model the short rate dynamics is given by,

$$dr_t = \alpha(b - r_t)dt + \sigma dW_t$$

Where W_t is Brownian motion, b is the level to which the short rate will tend in the long run, α is the rate at which the short rate will tend towards b and σ is the volatility of the short rate. The stochastic differential equation can be solved to give r_t

$$r_t = e^{-\alpha t}[r_0 + b(e^{\alpha t} - 1) + \int_0^t \sigma e^{\alpha s} dW_s]$$

Cox, Ingersoll and Ross also presented a stochastic differential equation for short term rates in [4] given by

$$dr_t = \alpha(b - r_t)dt + \sigma\sqrt{r_t}dW_t$$

This model prevents the short from becoming 0 but offers no closed form solution

The Hull-White model described in [3] also presents stochastic differential equation for the short term rate,

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

where $\theta(t) = \alpha b$ which is a non constant term, allowing the mean reversion level to vary.

The model we implement in this assignment is the LIBOR forward rate model to simulate interest rates. The initial values for the LIBOR forward rates need to be calibrated to the market forward rates which can be deduced through the market zero-coupon bond prices. This continuously compounded interest rate is given by,

$$e^{r_{ti}(t_{i+1}-t_i)} = 1 + L(t_i, t_{i+1})(t_{i+1} - t_i)$$

Most of code implemented in this submission is derived from Module 6 [8] and Module 7 [9] of the course.

We initialize most variables as given by the question.

- Option maturity is one year
- The option is struck at-the-money
- The current share price is \$100
- The up-and-out barrier for the option is \$150
- The risk-free continuously compounded interest rate is 8%
- The volatility for the underlying share is 30%
- The volatility for counterparty's firm value is 25%
- The counterparty's debt, due in one year, is \$175
- The current firm value for the counterparty is \$200
- The correlation between the counterparty and the stock is constant at 0.2
- The recovery rate with the counterparty is 25%

2 LIBOR Forward Rates, Stock Paths, and Counterparty Firm Values

The zero coupon bond prices are provided as observed, which means that the interest rate can be determined in the calibration process, along with the other parameters needed for the model. The following code sets bounds for the parameters and finds optimal values within those bounds.

```
#minimizing F
bnds = ((0,1),(0,0.2),(0,0.2), (0.00,0.10))
opt_val = scipy.optimize.fmin_slsqp(F, (0.3, 0.05, 0.03, 0.05), bounds=bnds)
opt_alpha = opt_val[0]
opt_b = opt_val[1]
opt_sigma = opt_val[2]
opt_r0 = opt_val[3]
```

Running this code results in the following values for the model parameters:

- Optimal alpha: 0.273
- Optimal b: 0.069
- Optimal sigma 0.028
- Optimal r0: 0.075

3 Discount Factor and Value of the Up-and-Out Call Option

Since this is a barrier option, it is necessary to use the 'mask' function to zero the payoffs that would exceed the barrier. This is done in the payoff function.

```
# define payoff for up–and–out call option

def payoff(S_t, K, L):
    stopped_S = S_t.mask(S_t > L, 0)
    return np.maximum(stopped_S - K, 0).to_numpy()

# Estimate the default−free value of the option:

option_estimate = []
option_std = []

for month, path in share_prices.items():
    payoffs = payoff(path, K, L)
    option_price = np.exp(-risk_free*T)*payoffs
    option_estimate.append(option_price.mean())
    option_std.append(option_price.std()/np.sqrt(sample_size))
```

This calculates the mean value for the call option mean risk-free price as 1.200.

4 Conclusion

References

[1] Vasicek, O. (1977). An equilibrium characterization of the term structure, Journal of financial economics 5(2): 177-188.

[2] Mamon, R. S. (2004). Three ways to solve for bond prices in the vasicek model, Advances in Decision Sciences 8(1): 1-14.

[3] Hull, J. and White, A. (2001). The general hull-white model and supercalibration, Financial Analysts Journal pp. 34-43.

[4] Cox, J. C., Ingersoll Jr, J. E. and Ross, S. A. (1985). An intertemporal general equilibrium model of asset prices, Econometrica: Journal of the Econometric Society pp. 363-384.

[5] Albrecher, H., Mayer, P., Schoutens, W. and Tistaert, J. (2007). "The Little Heston Trap", Wilmott (1): 83–92.

[6] Cox, John. "Notes on option pricing I: Constant elasticity of variance diffusions." Unpublished note, Stanford University, Graduate School of Business (1975).

[7] MScFE630 Computational Finance Module 5: Monte Carlo Methods for Risk Management

[8] MScFE630 Computational Finance Module 6: Pricing Interest Rate Options

[9] MScFE630 Computational Finance Module 7: Calibration