Price a European 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

April. 8, 2020

Keywords: Exotic Options, European up-and-out call option, Black-Scholes-Merton model and Geometric Brownian Motion.

Abstract

This report presents the results of the simulation of a European up-and-out call option over twelve months. The Black-Scholes-Merton model will be used to implement Monte Carlo estimations using Python code. The estimations will simulate paths for the underlying share and the counterparty's firm value using varying sample sizes, and determine estimates of both the default-free value of the option and the Credit Valuation Adjustment (CVA). Following from these it is simple to price the option incorporating counterparty risk, given by the default-free price less the CVA.

1 Introduction

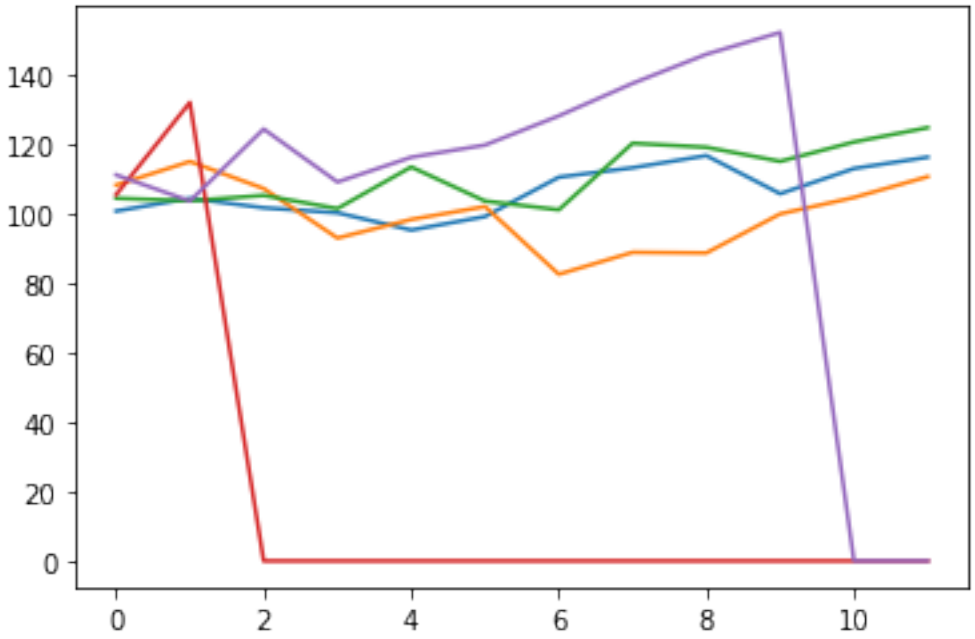
A European up-and-out call option is type of a Barrier Option, these options are consider path-dependant because barrier option's payoff is based on the underlying stock's price path. The payoff of an up-and-out option at maturity T is given by,

$$Payoff_T = (S_T - K)^+ given, max_{t\in[0,T]} S_t < L$$

Where, where K is the strike of the option, L is the barrier level, and S_t is the share price at time t. This is a type of call option whose payoff is reduced to 0 if the share price crosses the barrier level.

The seminal work of Merton [1] pioneered the formula for pricing barrier options. This led pricing formulas under the geometric Brownian motion (GBM) framework for one-asset barrier options by Rich [2] and multi-asset barrier options by Wong and Kwok [3]. Although the return dynamics of underlying shares are not sufficiently well described by the GBM process proposed by Black and Scholes [4], we are going to assume both the stock and counterparty firm values follow GBM with constant drift and volatilities and default only occurs at maturity.

Possible paths for the up-and-out barrier option payoff would look similar to the chart below, where two of the paths have crossed the barrier and resulted in zero payoff, but the other three have not crossed the barrier and behave like a vanilla option.



2 Simulated Paths for the Underlying Share and Counterparty Firm

In order to simulate paths for the underlying share and the firm, a matrix of correlated prices is generated based on the requirement for a correlation of 0.2.

[4]:
```
def share_path(S_0, risk_free_rate, sigma, Z, dT):
    return S_0*np.exp(np.cumsum((risk_free_rate-sigma**2/2)*dT + sigma*np.
 ↪sqrt(dT)*Z,1))
def generate_share_and_firm_price(S0, v_0, risk_free, sigma, sigma_firm, corr,␣
 ↪T, sample_size = 10, timesteps = 12):
    corr_matrix = np.array([[1, corr], [corr, 1]])
    norm_matrix = stats.norm.rvs(size = np.array([sample_size, 2, timesteps]))
    corr_norm_matrix = np.matmul(np.linalg.cholesky(corr_matrix), norm_matrix)

    share_price_path = pd.DataFrame(share_path(S0, risk_free, sigma,␣
 ↪Z=corr_norm_matrix[:,0,], dT=1/timesteps))
    share_price_path = share_price_path.transpose()
    firm_price_path = pd.DataFrame(share_path(v_0, risk_free, sigma_firm,␣
 ↪Z=corr_norm_matrix[:,1,], dT=1/timesteps))
    firm_price_path = firm_price_path.transpose()
    return [share_price_path,firm_price_path]
```

A Pearson test can be used to verify that the correlation is very close to the 0.2 target value. In Python this can be achieved as follows.

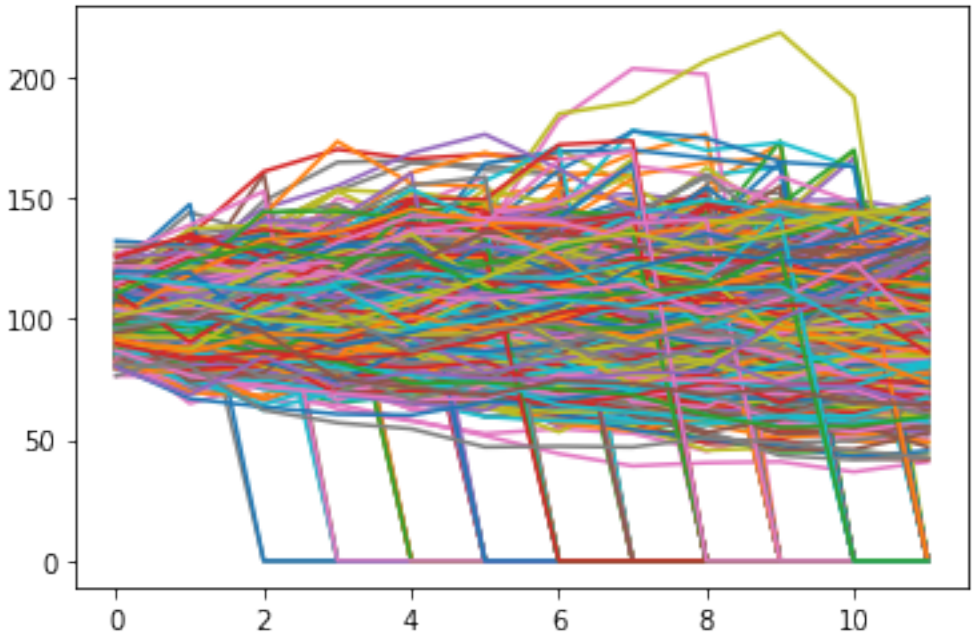
[5]:
```
#Testing share and firm price correlation
sample_size = 20
test = generate_share_and_firm_price(S0, v_0, risk_free, sigma, sigma_firm,␣
 ↪corr, T, sample_size, timesteps = 20000)
share_ret = np.log(test[0])
for i in range(sample_size):
    test[0]['sharelog'] = np.log(test[0][i])
    test[1]['firmlog'] = np.log(test[1][i])
    print(stats.pearsonr(test[0]['sharelog'].diff().dropna(), test[1]['firmlog'].
 ↪diff().dropna()))
```

```
(0.2035028479587627, 6.07336614624919e-186)
(0.1986599648970965, 4.04999708397334e-177)
(0.2016759847597355, 1.3775160789695309e-182)
(0.20364055709605317, 3.381815103189066e-186)
(0.19385401550478024, 1.3726860317349184e-168)
(0.19892212605701942, 1.3666646645127986e-177)
(0.18344242259502544, 7.064732561303709e-151)
(0.20519864966486748, 4.356484560250648e-189)
(0.193927020114363, 1.022531700201078e-168)
(0.19751037260327703, 4.659785935918803e-175)
(0.20393895669991588, 9.495454795149592e-187)
(0.20458792743312265, 5.953557128978809e-188)
(0.2042028358117276, 3.0829601588859696e-187)
(0.19512700379639908, 7.943982338391081e-171)
(0.19603082201925184, 2.003660281095667e-172)
(0.19820752618175266, 2.630853001655849e-176)
(0.20299979761482267, 5.137198920374959e-185)
(0.1966663418793928, 1.4903037901828931e-173)
(0.20319824398313788, 2.214204369119104e-185)
(0.20765530182748704, 1.0810080941108714e-193)
```

The payoff calculation for the up-and-out barrier option is defined as follows.

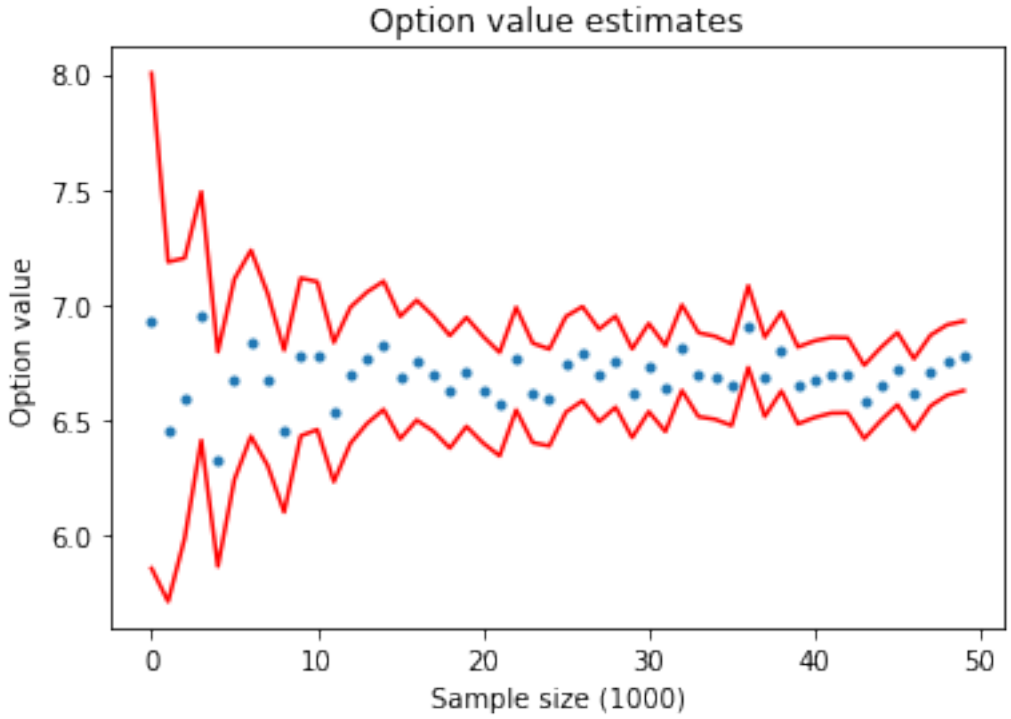
[11]:
```
# define payoff for up-and-out call option
def payoff(S_t, K, L):
    stopped_S = S_t.iloc[-1].where((S_t < L).all(), 0)
    return np.maximum(stopped_S - K, 0).to_numpy()
payoff(share_price_12_months, K, L)
```

The simulated payoff paths over twelve months are plotted below, with an arbitrary starting value of 200 for the firm value.

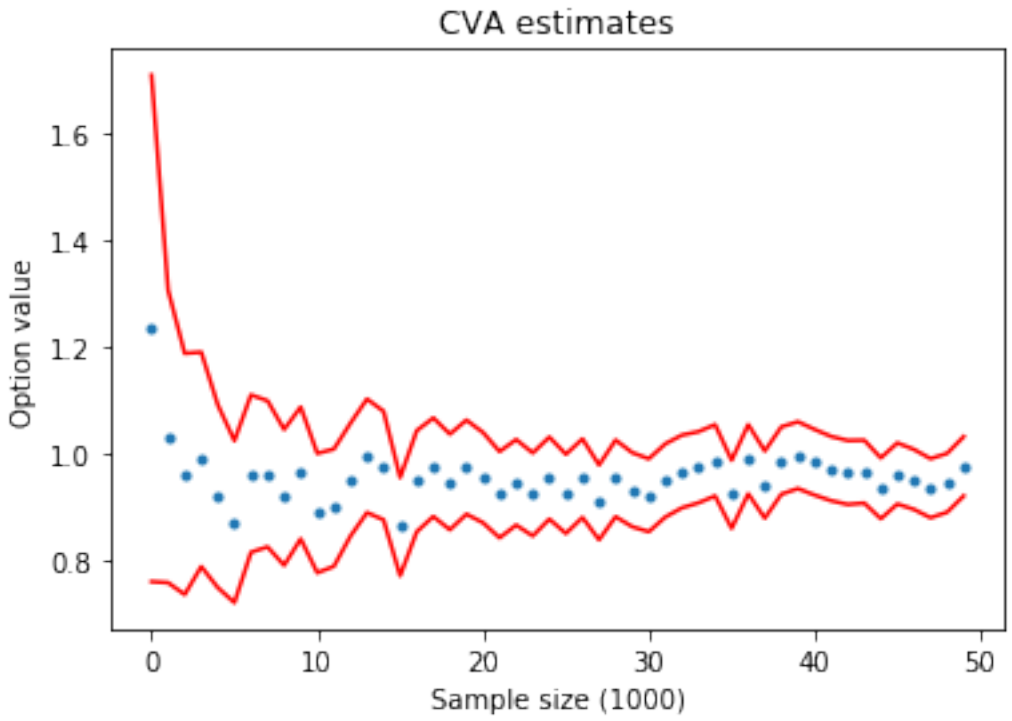


3 Monte Carlo Estimations

3.1 Default-Free Value Estimation



3.2 Credit Valuation Adjustment Estimation



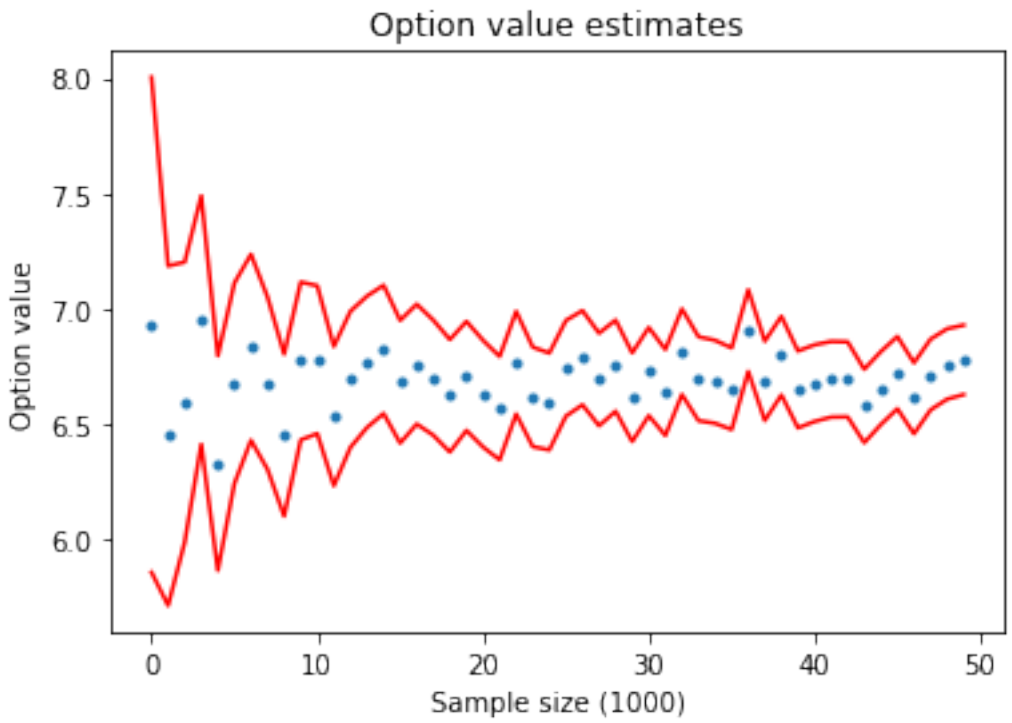
4 Incorporating Counterparty Risk

To estimate the price of the option incorporating counterparty risk, the CVA is subtracted from the default-free price; both of these have already been calculated at this point.

[20]:
```
option_cva_adjusted_prices = []
option_cva_adjusted_std = []
for sample_size, paths in share_price_paths.items():
    payoffs = payoff(paths, K, L)
    option_price = np.exp(-risk_free*T)*payoffs
    term_firm_vals = firm_val_paths[sample_size].iloc[-1].to_numpy()
    amount_lost = np.exp(-risk_free*T)*(1-recovery_rate)*(term_firm_vals <␣
 ↪debt)*payoffs

    option_cva_price = option_price - amount_lost

    option_cva_adjusted_prices.append(option_cva_price.mean())
    option_cva_adjusted_std.append(option_cva_price.std()/np.sqrt(sample_size))
```



[1]:
```
import QuantLib as ql
from collections import namedtuple
import math
```

[2]:
```
today = ql.Date(15, ql.February, 2020);
settlement= ql.Date(19, ql.February, 2020);
ql.Settings.instance().evaluationDate = today;
term_structure = ql.YieldTermStructureHandle(
    ql.FlatForward(settlement,0.04875825,ql.Actual365Fixed())
    )
index = ql.Euribor1Y(term_structure)
```

```
-------------------------------------------------------------------------------
--
    Model Price    Market Price     Implied Vol      Market Vol       Rel Error
-------------------------------------------------------------------------------
--
        0.00871         0.00949         0.10531         0.11480        -0.08263
        0.00968         0.01008         0.10634         0.11080        -0.04018
        0.00867         0.00871         0.10652         0.10700        -0.00448
        0.00653         0.00625         0.10665         0.10210         0.04442
        0.00357         0.00334         0.10680         0.10000         0.06773
-------------------------------------------------------------------------------
--
Cumulative Error :         0.12288
```

5 Conclusion

References

[1] Merton, Robert C. "Theory of rational option pricing." The Bell Journal of economics and management science (1973): 141-183.

[2] Rich, Don R. "The mathematical foundations of barrier option-pricing theory." Advances in futures and options research 7 (1994).

[3] Wong, Hoi Ying, and YueâĂŘKuen Kwok. "Multi-asset barrier options and occupation time derivatives." Applied Mathematical Finance 10.3 (2003): 245-266.

[4] Black, Fisher, and Myron Scholes. "The pricing and Corportate Liabilities." Journal of Political Economy 81 (1973).

[5] Yousuf, M. "A fourth-order smoothing scheme for pricing barrier options under stochastic volatility." International Journal of Computer Mathematics 86.6 (2009): 1054-1067.