**WQU MSc in Financial Engineering**

**Course: Computational Finance**

**Groupwork Assignment Submission 1 M3**

Pricing a European up-and-out call option held with a Risky Counterparty.

by

**Group members**:

[**Tewodros Ayalew (ttaayyll32@gmail.com)**](https://masters.wqu.org/user/view.php?id=2308&course=99)[**Nicholas Munu (nicholasmunu@yahoo.com)**](https://masters.wqu.org/user/view.php?id=2482&course=99)[**Nishan Agarwal (agarwalnishan5@gmail.com)**](mailto:Nishan%20Agarwal%20(agarwalnishan5@gmail.com))[**Akshay Kulkarni (akshaypk2012@gmail.com)**](https://masters.wqu.org/user/view.php?id=14760&course=99)[**Dhruv Agrawal (dhruva1@stanfordalumni.org)**](https://masters.wqu.org/user/view.php?id=21612&course=99)

**Contents**

**1.Introduction**

**2. Task 1**

**3. Task 2**

**4. Task 3**

**4. Summary and Conclusion**

**5. References**

# Introduction

In this project we are expected to price a European up-and-out call option held with a risky counterparty. This is a call option whose payoff becomes 0 if the share price gets too high over the lifetime of the option. This limits the final payoff of the option, which subsequently makes it cheaper that a vanilla call option.

The .py file is attached in the solution submitted for checking.

## **Task 1**: Simulate paths for the underlying share and for the counterparty’s firm value using sample sizes of 1000, 2000… 50000. Do monthly simulations for the lifetime of the option.

|  |
| --- |
| import numpy as np  from scipy.stats import norm   import matplotlib.pyplot as plt  import math   import random |

We import the necessary packages.

|  |
| --- |
| def terminal\_value(S\_0, risk\_free\_rate, sigma, Z,T):     return S\_0\*np.exp((risk\_free\_rate  - sigma\*\*2/2)\*T + sigma\*np.sqrt(T)\*Z)  def option\_payoff(S\_T,K):     return np.maximum(S\_T - K,0) |

Since we are working within the Black Scholes Merton model, we are going to be simulating our assets using a lognormal distribution for the stock’s value and the counterparty firm’s value.

We are going to be using the terminal\_value function to transform our initial stock and counterparty firm value into terminal values for a given risk free rate, volatility (sigma), random path array (Z), and time to maturity. The option\_payoff function finds the payoff for a call option at terminal time for given terminal stock price and strike price.

|  |
| --- |
| #Monte Carlo  risk\_free = 0.08 T = 1 strike = 100 S\_0 = 100 BarrierLevel=150        #up and out barrier for the option is 150 sigma = 0.3 sigma\_firm = 0.25 debt = 175 recovery\_rate = 0.25 correlation =  0.2 V\_0 = 275 #assuming the value of the counterparty in the beginning is 275 |

The relevant variables are initialized:

* Risk-free rate, *𝑟isk\_free*, of 8%
* Initial stock value, 𝑆\_0 of $100
* Initial firm value, 𝑉\_0 of $275
* Strike price of option, *strike* of $100
* BarrierLevel of option, *BarrierLevel* of $150
* Time to maturity of option, T, of 1 year
* Stock volatility, sigma, 30%
* Counterparty Firm volatility, sigma\_firm 25%
* Option strike price, strike, of $100
* Counterparty Firm debt, debt, of $175
* Recovery rate from counterparty in event of default, recovery\_rate, of 25%
* Counterparty Firm debt, debt, of $175
* Counterparty firm and stock correlation, *correlation* of 0.2

|  |
| --- |
| np.random.seed(0)  moption\_estimates = [None]\*50 #monte carlo option estimates not incorporating default risk  moption\_std = [None]\*50 #standard deviation of monte carlo option estimates  cva\_estimates = [None]\*50 #cva estimates  cva\_std = [None]\*50 # standard deviation of cva estimates  CVAadjustedOptionval = [None]\*50 #Option value incorporating default risk  CVAadjustedOptionval\_std = [None]\*50 #standard deviation of OptionValue with default risk |

In the above code, we have fixed the seed for result replication. We initialize the arrays of size 50 for monte carlo option estimates, cva estimates, and estimates of option value incroporating default risk along with each of their respective standard deviations.

|  |
| --- |
| for i in range (1,51):          size = i\*1000 #sample sizes set ranging from 1000 to 50,000 with increments of 1000     #in each loop this is the number of paths simulated     maxST = np.zeros(size)  #max stock price for each simulation         stock\_val = np.zeros(size)     firm\_val = np.zeros(size)                     option\_val = np.zeros(size) |

We vary the size from 1000 to 50000 in increments of 1000 for simulating the paths for the underlying share and for the counterparty’s firm value.

We run the loop for each sample size (the i’s). The value of stock, firm, maxST ( maximum stock value for each simulation path) and option value without default risk of counterparty is initialized to zero for each simulation.

|  |
| --- |
| for month in range(1,12\*T+1): #simulation from month 1 to month 12\*T                  corr\_matrix = np.array([[1,correlation],[correlation,1]])         norm\_matrix = norm.rvs(size = np.array([2,size]))         corr\_norm\_matrix = np.matmul(np.linalg.cholesky(corr\_matrix),                                           norm\_matrix)                     #The statement above creates a 2 ×50 000 array of standard          #normal random variables with a given correlation                      stock\_val = terminal\_value(S\_0,risk\_free, sigma,                                    corr\_norm\_matrix[0,],month/12.0)                           firm\_val = terminal\_value(V\_0,risk\_free, sigma\_firm,                                     corr\_norm\_matrix[1,],month/12.0) |

We run a loop for every month where

1) We generate the normal random path matrix adjusted for correlation between the stock and firm’s value (by multiplying the cholesky decomposition of the correlation matrix between stock and firm of dimension (2,2) and the randomly generated normal matrix of dimensions (2,size))

2) We calculate the stock and firm’s final value using the terminal value function initialized in the beginning by passing in the appropriate parameters

## **Task 2:** Determine Monte Carlo estimates of both the default-free value of the option and the Credit Valuation Adjustment (CVA).

|  |
| --- |
| for j in range(0,size):             if(stock\_val[j] > maxST[j]):                     maxST[j] = stock\_val[j] |

Here we update maxST to the maximum value for the respective stock price path (in case this month’s value is higher than the maxST on record)

|  |
| --- |
| for j in range(0,size):         if(maxST[j] > BarrierLevel):             option\_val[j] = 0       #if stock price has exceeded the barrier level then option value is 0              else             option\_val[j] = np.exp(-risk\_free\*T) \* option\_payoff (stock\_val[j] , strike)                  #finds the discounted call payoff |

Here we come out of the month loop and calculate the option\_val (option value without the default risk of counterparty) for each simulation by running a loop through each simulation and:

1. If maximum value of stock is greater than Barrier level then setting option\_val to 0
2. Otherwise setting option\_val to the discounted call payoff with given terminal values of stock and strike price

|  |
| --- |
| amount\_lost = np.exp(-risk\_free\*T)\*(1-recovery\_rate)\*(firm\_val < debt)\*option\_val     #estimates for CVA based on firm value  cva\_estimates[i-1] = np.mean(amount\_lost) cva\_std[i-1] = np.std(amount\_lost)/np.sqrt(size) |

In amount\_lost we calculate the CVA estimate for that particular simulation using the formula

𝔼Q[𝑒−𝑟𝑇(1−δ)𝑋T𝕀{𝑉T<𝐷}]

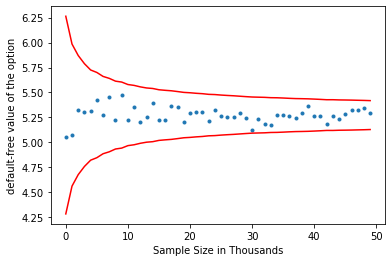
Where r is risk free rate, T is time to maturity, 𝑋𝑇 is the value of our portfolio at time 𝑇(here portfolio is call option), and 𝕀{𝑉T<𝐷}] is an indicator which is 1 if value of firm at time T is less than debt and zero otherwise. (This is essentially our original CVA equation, but with all the τ's replaced by capital 𝑇's, since we are assuming that the only possible default time is capital 𝑇). We calculate the **CVA estimate** to be the mean of the amount\_lost across simulations and also calculate its standard deviation.

|  |
| --- |
| #default-free value of the option     moption\_estimates[i-1] = np.mean(option\_val)     moption\_std[i-1] = np.std(option\_val)/np.sqrt(size) |

We calculate **the estimate of the default free value of the option** to be the mean of the default free value of the option for each simulation and also calculate its standard deviation.

We show the plots of default free value of option and CVA estimate after coming out of the loop across size of simulations below.

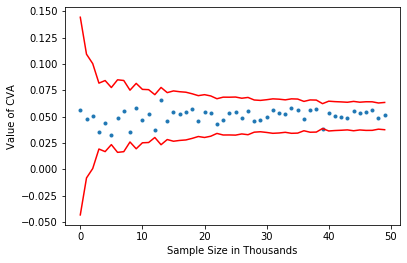
|  |
| --- |
| #Plotting the graph for default-free value of the option plt.plot(moption\_estimates,'.') plt.plot(np.mean(moption\_estimates)+np.array(moption\_std)\*3, 'r') plt.plot(np.mean(moption\_estimates)-np.array(moption\_std)\*3, 'r') plt.xlabel("Sample Size in Thousands") plt.ylabel("default-free value of the option") |



The above is the plot for the **default free value of the option** along with 3 standard deviation error bounds – we can see that the mean of the data points comes to be around **5.3**

￼ ￼

|  |
| --- |
| #Plotting the graph for CVA plt.plot(cva\_estimates,'.') plt.plot(np.mean(cva\_estimates)+np.array(cva\_std)\*3, 'r') plt.plot(np.mean(cva\_estimates)-np.array(cva\_std)\*3, 'r') plt.xlabel("Sample Size in Thousands") plt.ylabel("Value of CVA") |



The above is the plot for the **CVA** along with 3 standard deviation error bounds – we can see that the mean of the data points comes to be around **0.05.**

**Task 3:** Calculate the Monte Carlo estimates for the price of the option incorporating counterparty risk, given by the default-free price less the CVA.

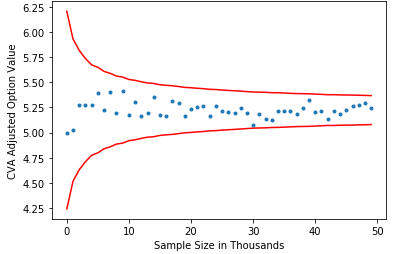
|  |
| --- |
| #Monte Carlo estimates for the price of the option incorporating counterparty risk, given by the default-free price less the CVA     CVAadjustedOptionval[i-1] = moption\_estimates[i-1] - cva\_estimates[i-1]     CVAadjustedOptionval\_std[i-1] = np.std(option\_val - amount\_lost)/np.sqrt(size) |

Within the loop for sizes of simulations and outside the month loop we calculate CVA Adjusted option value (the price of the option incorporating counterparty risk) to be the default free value of the option minus the CVA estimate.

We show the plot of CVA Adjusted option value after coming out of the loop across size of simulations below.

.

|  |
| --- |
| #Plotting the graph for CVA adjusted option value plt.plot(CVAadjustedOptionval,'.') plt.plot(np.mean(CVAadjustedOptionval)+np.array(CVAadjustedOptionval\_std)\*3, 'r') plt.plot(np.mean(CVAadjustedOptionval)-np.array(CVAadjustedOptionval\_std)\*3, 'r') plt.xlabel("Sample Size in Thousands") plt.ylabel("Value of CVA") |



The above is the plot for the **CVA Adjusted Option Value** (the price of the option incorporating counterparty risk) along with 3 standard deviation error bounds – we can see that the mean of the data points comes to be around **5.25**.

**Conclusion and Summary**

Using the above code for simulating the option value we can conclude that the **default free value of the option is 5.3, CVA estimate is 0.05, and CVA Adjusted Option Value (**the price of the option incorporating counterparty risk**) is 5.25**

**References**

Hilpisch, Y. (2014) *Python for Finance*, *The FEBS journal*. doi: 10.1111/febs.12952.

Kenton Will (2019) *Value at Risk (VaR)*, *Corporate Finance &Accounting*. Available at: https://www.investopedia.com/terms/v/var.asp (Accessed: 27 September 2019).

Michell Cory (2019) *Up-And-Out Option Definition*, *Options & Derivatives Trading*. Available at: https://www.investopedia.com/terms/u/up-and-outoption.asp (Accessed: 26 September 2019).

Ming-Long Wang, A. and Shen, S.-Y. (2005) ‘On Pricing of the Up-and-Out Call-A Boundary Integral Method Approach’, *Asia Pacific Management Review*, 10(3), pp. 205–213.

Risk.net (no date) *Credit valuation adjustment (CVA) news and analysis articles - Risk.net*. Available at: https://www.risk.net/topics/credit-valuation-adjustment-cva (Accessed: 27 September 2019).

Saunders Anthony &Allen Linda (2010) *Credit Risk Measurement In and Out of the Financial Crisis*. Third, *New Approaches to Value at Risk and Other Paradigms*. Third.