Valutazione Prodotti Finanziari

Docente: R. Daluiso

02/2023

Master's Degree in Quantitative Finance

CVA

Group 5

Pietro Maironi da Ponte
Tommaso De Witt
Antonio Prestidonato
Mattia Sigona
Saverio Taschetta Latteri

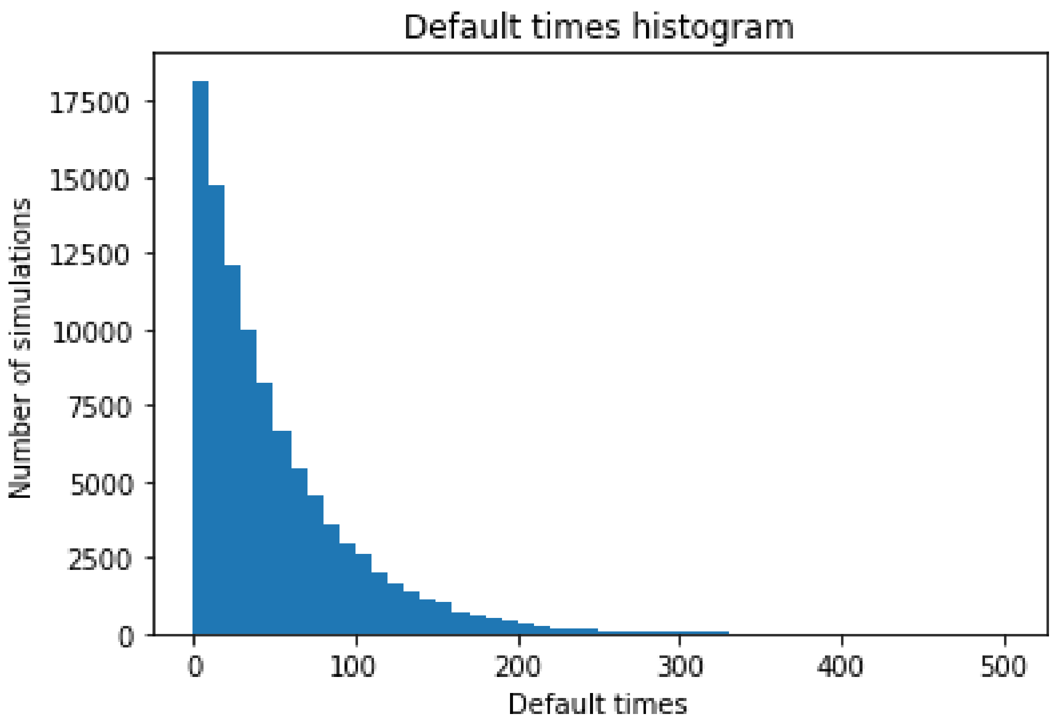
Summary

1 Introduction

Consider an underlying X with no dividends with Bachelier dynamic subject to risk neutral measure Q
$dX_t = \sigma^X dW_t^X$
where σ^X is a positive constant and W^X is a standard brownian motion under Q.
Suppose that the counterpart has a constant and deterministic default intensity $\lambda > 0$ and a fraction of expected loss given default lgd, with which there is no collateralization deal. Hypothesize that we want to stipulate a forward on X: the contract foresees that at a future date T the bank will receive X_T paying a fixed strike K
Assuming that the bank hasn't contracts with the counterpart we'll calculate unilateral Credit Valuation Adjustment for unit of underlying and we'll implement that to approximate numerically CVA at t=0 under the following values:
$\lambda = 4\% y^-1, lgd = 60\%, X_0 = 500\text{EUR}, T = 2y$
After that we'll evaluate a confidence interval of 98% for the CVA simulating 100000 indipendent realizations of the underlyng and default times

2 CVA Pricing

This code is performing a credit valuation adjustment (CVA) calculation for an option contract, using both analytical and Monte Carlo simulation methods. The first part of the code sets up the necessary variables and imports required libraries such as math, numpy, scipy, and matplotlib. It then defines the risk-free rate (rf), probability of default (PD), loss given default (lgd), underlying asset price at time 0 (S0), time to maturity (T), volatility (σ), strike price (K), dividend yield (q), and default intensity (lambda$_)$.
The next section defines a lambda function (bachelierIntegrand) that represents the integrand in the Bachelier formula for pricing European call options. It then uses the scipy.integrate.quad() function to numerically integrate the Bachelier formula over the range of [0,3] with respect to T, and obtains the result for the CVA.
CVA=-0.9940953582112317
The second part of the code simulates default times using the inverse exponential distribution method. It generates a random uniform distribution of numbers and then applies the inverse exponential function to obtain a distribution of default times. It then plots a histogram of the default times using the matplotlib library. The final section of the code adds axis labels and a title to the histogram plot, and displays the plot using the show() function.



3 Confidence Interval

This code is simulating a credit valuation adjustment (CVA) for a European call option contract by estimating the expected value and 98% confidence interval of the CVA using Monte Carlo simulation.

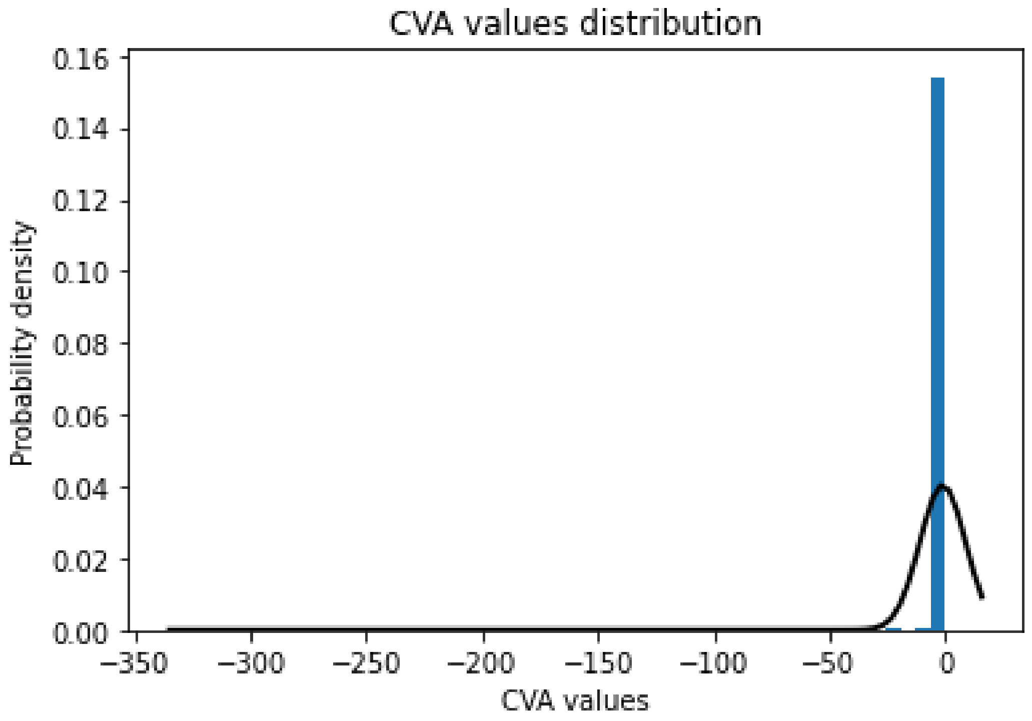
The first line defines a function named 'underlying$_{p}rice'thattakesintheinitialstockprice(S0), volatility(\sigma)$, time to default (t), and a standard normal random variable (z) as inputs. The function uses the Black-Scholes model to calculate the underlying asset price at the default time given the inputs.

The second section initializes a vector X with all the initial stock prices (S0) using the numpy.full() function. It then uses a for loop to simulate the stock price at the time of default for every simulation by generating a random normal variable (z), selecting the default time (t) from the previously generated distribution of default times, and then evaluating the underlying price at the time of default using the 'underlying$_{p}rice'function.ThesimulatedstockpricesarestoredintheXvector.$

The third section estimates the value of the option (V) for every simulation by taking the maximum of the difference between the simulated stock price at the time of default and the strike price (K) and zero.

The fourth section initializes a vector for CVA (Credit Valuation Adjustment) and computes the CVA for every simulation by checking if the default time is within 3 years and if the value of the option is positive. If both conditions are satisfied, the CVA is computed as -0.6 times the value of the option.

Finally, the code estimates the expected value of the CVA by taking the mean of the CVA vector, and calculates the 98% confidence interval for the CVA using the standard deviation of the CVA and the number of simulations. The expected value and confidence interval are then printed to the console using the print() function.
With the last chart I plot a histogram and a normal distribution curve to visualize the distribution of CVA (Credit Valuation Adjustment) values.


CVA values distribution
Probability density
CVA values