POLITECNICO
MILANO 1863

Financial Engineering 2023/24

Assignment 1 - Group 17

Alessandro Torazzi, Matteo Torba, Giovanni Urso, Chiara Zucchelli

Contents

Assumptions

Having at disposition only the ttm-zero-rate, we considered the short-rate as constant and deterministic, as also suggested by the discount price already given in the `runAssign1_Group17` script.

a Price a European Call Option

We computed the price of the European Call Option via `blkprice` Matlab function (case 1), via the Cox Ross Rubinstein (CRR) tree approach and via the Monte-Carlo approach (case 3).
In the first case, the price was obtained through the function `EuropeanOptionClosed`, which was already given.
In the second case, we wrote the function `EuropeanOptionCRR`: this function works for both put and call options, thanks to the $flag$ variable. In both cases, we build an upper triangular matrix where each column represents an instant of time and the elements in the row are the different possible states of the world. The function determines the final payoff in the last column and, through a backward method, it computes the expectation under the risk neutral probabilities in the columns before, up to the first one. By discounting the initial node of the tree, which is in the position $(1, 1)$ of the matrix, we obtain the option price.
In the last case, we wrote the function `EuropeanOptionMC` which computes the option price via a Monte-Carlo simulation.
For these computations we arbitrarly set the value of the discretized time steps of the CRR tree equal to 100 and the number of Monte-Carlo simulations equal to 10^6. We also computed the execution time of each method by adding the Matlab functions `tic` and `toc` inside of the function `EuropeanOptionPrice`. Since the European Call Option given subscribes 1 million of contracts, we also multiplied the prices obtained by the notional value.
The results obtained are represented in the following table:

Method	Price of the European Call Option (EUR)	Execution time (seconds)
blkprice	$C_{Black} = 39763.7764$	0.001045
CRR	$C_{CRR} = 39849.4913$	0.000132
Monte-Carlo	$C_{MC} = 39790.5100$	0.091899

We observed that the prices computed via the CRR tree approach and via the Monte-Carlo simulation are very closed to the price obtained through the black's formula. We have further analized the error of the methods in point c.
We can notice that the execution time of the Monte-Carlo method is bigger than the execution time of the CRR approach: to execute 10^6 simulations we need more time than to built a tree with 100 timesteps. On the other hand, the Monte-Carlo price difference from the price obtained via

Black's formula is smaller than the difference between CRR price and Black's price, therefore the greater execution time guarantees more precision in this case.

b Choice of M to reach the threshold

To select the value of M_{CRR}, the minimal number of intervals in CRR approach to have an error below a threshold of $1bp$, we used the function `PlotErrorCRR.m` to compute the errors vector. We computed the error of the CRR tree method as:

$$Error_{CRR} = |C_{Black} - C_{CRR}| \tag{1}$$

and we built a vector of errors by varying M as $M = 2^m$ for $m = 1, ..., 10$. Finally, we used the function `findM` to determine M_{CRR}, obtaining:

$$M_{CRR} = 16$$

Therefore, by building a tree with 16 time steps we can reach the threshold required for the error.

In a similar way, we computed M_{MC}, the number simulations in a Monte-Carlo method to have an error below the threshold, using the function `PlotErrorMC` to compute the errors vector. The error of the Monte-Carlo method is computed as:

$$Error_{MC} = \frac{\sqrt{\frac{1}{M-1}\dot{\sum}_{i=1}^{M}(C_i - C_{MC})^2}}{\sqrt{M}} \tag{2}$$

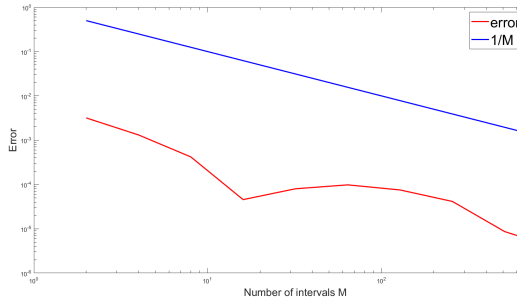
which is an estimation of the unbiased standard deviation of the MC price thanks to the Central Limit Theorem; the numerator is the sample standard deviation of the price computed by the Monte-Carlo method with M simulations and the denominator is the square root of the number of simulations. We did this computation for $M = 2^m$ with $m = 1, ..., 20$ to build a vector of errors with 20 elements. Finally, we computed the errors vector and we used the function `findM` to determine M_{MC}, obtaining:

$$M_{MC} = 524288$$

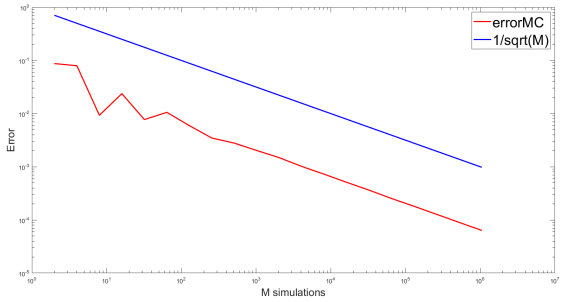
Thus, by applying the Monte-Carlo method with 524288 simulations, we reach the threshold of $1bp$ for the bid ask spread.

c Errors vector representation

To analize the errors, we considered a European Call Option which subscribes one contract.
In order to show the trend of the numerical errors for a Call rescale with M, we used the functions `PlotErrorCRR` and `PlotErrorMC` The obtained graphs are shown in the figures below:



(a) Error of the CRR method



(b) Error of the Monte-Carlo method

In the first case we noticed, despite soma small inflection points, that for a European Call Option the CRR numerical error rescales with 1/M. In the second case, we had that the Monte-Carlo numerical error rescales with M as $1/\sqrt{M}$.

Comparing the two methods, we can observe that as M increases in the two cases, the CRR method's error decreases faster than the MC method's error.

d Pricing of a European Call Option with European barrier (up&in)

We are asked to price a knock-in European barrier Call Option by using CRR, Monte-Carlo or a closed formula. For the first two numerical methods we just modified the previous pricing function by multipling the payoffs by the indicator function of hitting the barrier.
In order to find a closed formula, we found a replicative strategy, using plain vanilla options. The euro barrier payoff can be obtained with the following portfolio:

- a long position on a european call having the *barrier* as strike;
- a long position on an amount of $barrier - strike$ digital options with the same strike as above.

For the pricing of the call we used the closed pricer described before and we implemented a pricer for the digital option, recalling that the value of a single option is just $N(d_2)$, the probability of being in the money (ITM), and where:

$$d_2 = \frac{ln(\frac{F_0}{KI})}{\sigma\sqrt{T}} - \frac{\sqrt{T}\sigma}{2}$$

We also represented graphically the Payoff of the European Bierrier Option, as shown in Figure 3.

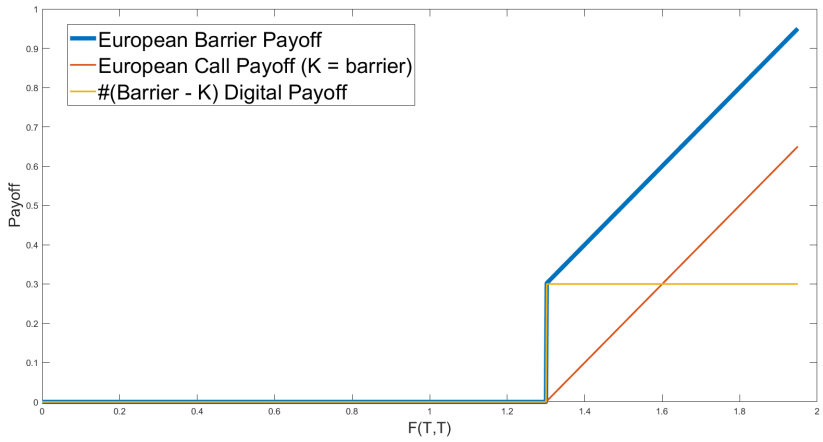


Figure 2: The blue payoff is the one of the european barrier option. It can be obtained by summing the red and the yellow ones, respectively of a call and of the correct amount of digital.

Also in this case, we assumed that the European Call Option with European Barrier subribes 1 million of contracts, hence the prices obtained were multiplied by the Notional value of 10^6. The results, obtained by setting arbitrarily the discretiztion (1000 time intervals) and simulation parameters (10^5 simulations), are as follows:

Method	Price of the European Call Option with European barrier up&in (EUR)
Closed formula	$KI_{exact} = 2092.6526$
CRR	$KI_{CRR} = 2182.3931$
Monte-Carlo	$KI_{MC} = 2116.5845$

We observe that the results obtained are closed to the exact formula price, however both the numerical methods underestimate the result obtained with the replicating portfolio. We could try to solve this problem by augmenting the number of time intervals or the number of simulations, increasing also the computational time.

We can also compare the price of the knock-in European barrier Call Option with the price of the European Call Option computed in point a. As we could have expected, the European barrier Call Option has a lower price: it guarantees a lower payoff at the maturity, since we add the condition of hitting the barrier, therefore by absence of arbitrage we expect its price to be lower.

e Vega of the European Call Option with European barrier

The Vega calculations for the CRR and Monte-Carlo pricing techniques are perforfemed by numerical approximation of the partial derivative of the price with respect to the volatility. We used the

central difference approximation method:

$$\frac{\partial KI}{\partial \sigma} \approx \frac{KI(\sigma + h) - KI(\sigma - h)}{2h}$$

which ensures that the appoximation error order is $o(h^2)$.
In order to find the exact Vega, we used the well known formula for the Call discussed in class and we added the one for the digital obtained after some computation:

$$\frac{\partial D}{\partial \sigma} = -B(0,T)F_0\phi(d_2)\left(\frac{ln(\frac{F_0}{K_{digital}})}{\sigma^2\sqrt{T}} - \frac{\sqrt{T}}{2}\right)$$

In the derivative estimation we set the $h = 10\%\sigma$ in order to avoid numerical problems and we arbitrarly set the discretization for the underlying and hyperparameters in CRR and Monte-Carlo, trading computational time for precision. We analized the execution times of the three methods which are represented in the following table:

Method	Execution time (seconds)
Closed formula	0.003169
CRR	12.482510
Monte-Carlo	0.230507

We observe that among the two numerical methods the CRR approach executes the computations in a longer time: if we increase the number of time steps, building the CRR tree seems to be a longer process than a Monte-Carlo simulation.

We can graphically represent the results obtained in Figure 4:

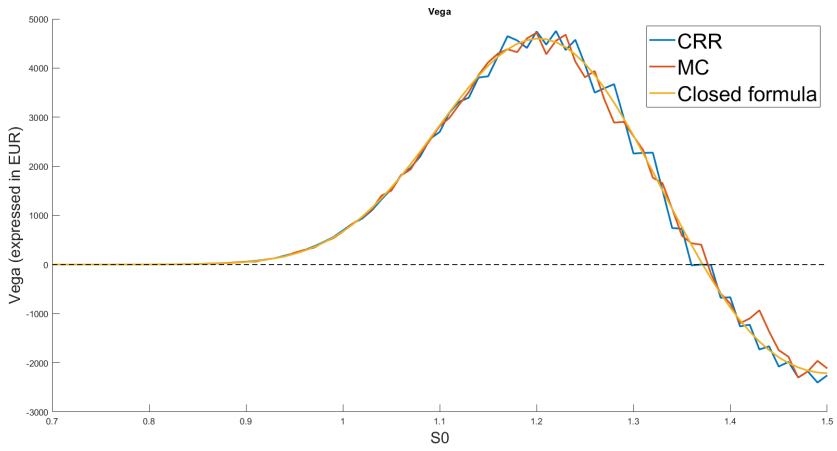


Figure 3: Vega of the barrier option computed via the CRR approach (blue line), via the MC approach (orange line) and via the closed formula (yellow line)

We observe that the Vega for this option is not positive for all values of the underlying, as in the plain vanilla case. This is due to the discontinuity of the payoff: for high values of the underlying (i.e. $S_0 \geq 1.38$) the derivative becomes negative because an increment in volatility incereases the probability of being out of the money (OTM).

In conclusion, both the two numerical methods seem to approximate well the Vega of the option, however the MC approach gives back the result in a considerably lower computational time.

f Antithetic variables technique

Using the Monte-Carlo method, we generated N/2 values through the randomic g function, and for each value we considered his negative as well, thus obtaining N/2 pairs of anthitetic variables, for a total of N values. Once computed the option price for both the halves of the N vector (`callPrices` and `callPricesAV`), we calculated the unbiased standard deviation of the final price as in point b. In this way, the negative correlation among the pairs of antithetic variables reduced sensibly the error, with respect to the one computed in point b. To visualize this change, we repeated the calculation for some values of N and then we plotted the results, comparing them with the errors calculated for the 'regular' Monte-Carlo method.

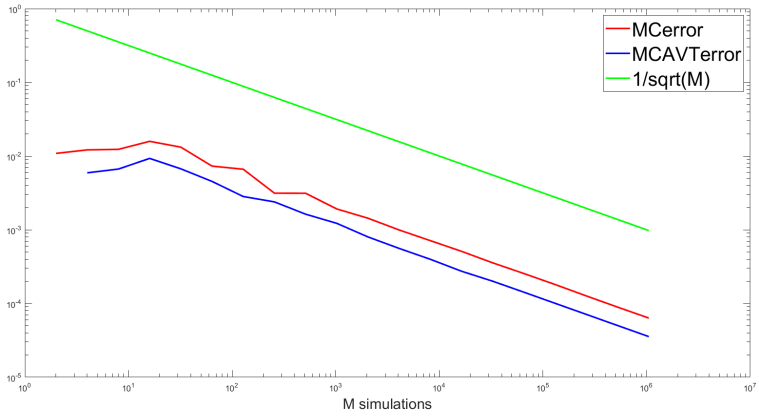


Figure 4: In blue the error computed using the Antithetic Variables tecnique, in red the error computed using the regular Monte-Carlo method

As we expected, Figure 5 shows that the error obtained with the antithetic variables is smaller of the error obtained in the 'regular' Monte-Carlo method for any number of simulations tested and it follows the slope of $1/\sqrt{M}$. We also observe that for bigger values of M, the scaling factor of the error by passing from the 'regular' Monte-Carlo method to the antithetic variables technique tends to be constant and by doing the computations in Matlab we found that it is equal to 0.56074 (since it is smaller than 1, the error is reduced).

g Pricing of a Bermudan option

For the Bermudan Call we consider the time interval between two consecutive exercise dates equal to 1/12 of a year (and also between the value date and the first exercise date). In particular, we computed the CRR tree in a way that the number of total intervals was a multiple of the number of intervals in each month.
Differently from how we have done for the European option, we discounted [1] the Bermudan call at each step. In the exercise dates, the node price would be computed as the maximum between the regular discounted tree value given by the following nodes, and the option's intrinsic value at that date: the intrinsic value is equal to obtaining the stock at the strike price, hence we retrieved the underlying price from the forward and we computed the option payoff in the case of the early exercise. We obtain the following price:

$$C_{Bermudan} = 40465.8856 EUR$$

The execution time is 0.06503. As expected, being that the Bermudan Call has more privileges than a Plain Vanilla Call, its price was higher than the one of the European Call computed on point a.

h Pricing of a Bermudan option with varying dividend yield

To observe the European and Bermudan Call options beahviour with respect to variations in the dividend yield, we computed their prices for some yield values ranging from 0 to 0.06.

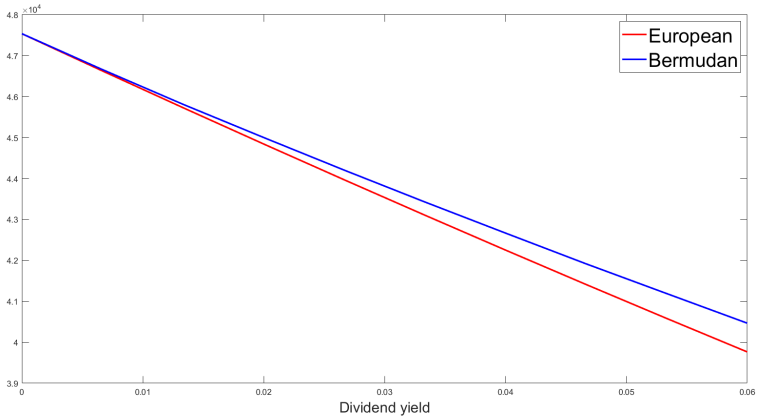


Figure 5: Price of a European Call Option (red line) and of a Bemudan Call Option (blue line) as the dividend yield varies between 0% and 6%

[1] As aforementioned, for our calculations of the Bermudan option price we considered a constant and deterministic short-rate.

Once plotted the results we observed, as expected, that the prices for the two options overlapped when the dividends approached zero: that was due to the fact that, without dividends, the Bermudan Call option, similarly to an American Call option, would behave as a submartingale, so it would never be exercised before maturity. Therefore, as the dividend yield tend to 0, the two call option should tend to the same price as well.