Assignment n.3 Group n.2

Andreotti P., Hanna K., Nuttini E., Rigamonti M., Zhou F.

April 2024

Contents

1 A brief introduction on option pricing

In this assignment the option pricing process is addressed with different techniques according to the *Black76* model. The assumptions of this model are the following:

- Frictionless market: absence of transaction costs;
- Constant risk-free rate r;
- Absence of arbitrage;
- The underlying forward price follows a **geometric Brownian motion** (martingale process);

In particular we are given the task of pricing a **European Call Option**:

$$C(t_0, T; K)$$

where:

- t_0 is the time when you are pricing;
- T is the time at maturity of the contract;
- K is the **strike price**.

From now on $C(t_0, T) = C(t_0, T; K)$ for simplicity. A Call Option is a non-linear derivative that gives you the right to buy the underlying at T and its pay-off is the following:

$$C(T, T) = (F(T, T) - K)^+ = \max(0, F(T, T) - K)$$

where $F(T, T)$ is the price of the underlying at time T. Under non arbitrage assumption the following holds true:

$$C(t_0, T) = B(t_0, T)E_0\left[(F(T, T) - K)^+\right]$$

It is of paramount importance to recall that $F(t, T)$ is a random process and the dynamic of it can be written as follows (geometric Brownian motion):

$$dF(t, T) = F(t, T)\sigma dW_t$$

where:

- σ is the volatility of the underlying;

- dW_t is the stochastic increment distributed as $dW_t \sim N(0, dt)$ and it satisfies the following: $\int_{t_0}^{T} dW_t = W_T - W_0$, with $W_0 = 0$

Therefore, under log-normal distribution:

$$F(T,T) = F_0 \exp\left(-\frac{\sigma^2}{2}(T - t_0) + \sigma W_T\right)$$

By exploiting the *equivalence in law* the latter can be re-written equivalently as:

$$F(T,T) = F_0 \exp\left(-\frac{\sigma^2}{2}(T - t_0) + \sigma\sqrt{T - t_0 g}\right)$$

where $F_0 = F(t_0, T)$ and $g \sim N(0,1)$ is a standard normal. This fact will be exploited during Monte Carlo.

The *Black76* model has closed form solution:

$$C(t_0, T) = B(t_0, T)\left[F_0 N(d_1) - K N(d_2)\right]$$

where $N(\cdot)$ are cumulative distribution functions of standard normal distributions and $d_{1,2}$ are defined as follows:

$$d_1 = \frac{log(\frac{F_0}{K}) + \frac{\sigma^2}{2}(T - t_0)}{\sigma\sqrt{T - t_0}}$$

$$d_2 = d_1 - \sigma\sqrt{T - t_0}$$

The main advantage of using a closed form solution is having an immediate solution if you are dealing with a case within the model assumptions. Most real use cases do not satisfy them, hence alternative ways like Monte Carlo and Binomial Tree are required for the pricing. Both techniques require just one input parameter: σ.

For the assignment $t_0 = 0$, so the time domain is $t \in [0, T]$.

1.1 As of Equivalence in law

The main aspect to highlight here is the fact that the following equations may seem equivalent:

$$F(T,T) = F_0 \exp\left(-\frac{\sigma^2}{2}(T - t_0) + \sigma W_T\right)$$

$$F(T,T) = F_0 \exp\left(-\frac{\sigma^2}{2}(T-t_0) + \sigma\sqrt{T-t_0 g}\right)$$

but in the first case, W_T is the realization of the entire random process, meaning that you need the whole dynamic of dW_t in order to get $\int_{t_0}^{T} dW_t = W_T$.

On the other hand, when exploiting the equivalence in law property, you can directly access the effect of the whole path just by simulating the last instance, this means that you just need to sample $g \sim N(0,1)$ at time $t = T$.

This is of paramount importance as you gain much more speed-up.

1.2 Euler method

The geometric Brownian motion dynamic can be addressed in the Euler fashion as follows. The main advantage of Euler method is the fact that you don't need the explicit behavior of the underlying in order to simulate it, but you only require the initial condition.

Given a time domain discretization in $N+1$ sub-intervals Δt such that: $t_n \in \{t_0, t_1, \ldots, t_{N-1}, t_N\}$, $t_{n+1} - t_n = \Delta t$, $n = 0, \ldots, N-1$, of course $t_0 = 0$ and $t_N = T$, the Euler schema reads:

$$F(t_{n+1}, t_N) - F(t_n, t_N) = F(t_n, t_N)\sigma(W_{t_{n+1}} - W_{t_n})$$

Each iteration, given the initial value $F(t_0, t_N)$, is updated as:

$$F(t_{n+1}, t_N) = F(t_n, t_N)\left[1 + \sigma(W_{t_{n+1}} - W_{t_n})\right]$$

where $\Delta W_n = W_{t_{n+1}} - W_{t_n} \sim N(0, \Delta t)$.

The accuracy of Euler method depends on how small Δt is, and the choice of it is in trade-off against the computational requirement.

2 Monte Carlo Pricing

The main advantage of Monte Carlo method is its versatility, meaning that it can be used in most of the real use cases. The rationale behind why Monte Carlo works is due to the *law of large numbers*:

$$\begin{aligned}
\frac{C(t_0,T)}{B(t_0,T)} &= E_0\left[(F(T,T)-K)^+\right] \\
&= E_0\left[F_0 \exp\left(-\frac{\sigma^2}{2}(T-t_0)+\sigma\sqrt{T-t_0 g}\right)-K\right]^+ \\
&= \lim_{M\to\infty}\frac{1}{M}\sum_{i=1}^{M}\left[F_0 \exp\left(-\frac{\sigma^2}{2}(T-t_0)+\sigma\sqrt{T-t_0 g_i}\right)-K\right]^+
\end{aligned}$$

where each $g_i \sim N(0,1)$, this is the direct Monte Carlo formulation.

From the given code and initial datum, we implemented slightly different functions called *monte_carlo_european_pricer_* which allows us to simulate the distribution of $F(t,T)$ up to delivery T and to compute the value of a Call option on $F(t,T)$ with strike price K. In particular we built:

- *monte_carlo_european_pricer* for full path simulation using the whole dynamics of dW_t;
- *monte_carlo_european_pricer_dir* for direct simulation using the equivalence in law of g;
- *monte_carlo_european_pricer_Euler* for full path simulation using Euler method.

Additionally, inside the functions we execute the Monte Carlo pricing routine and compute its error with respect to the closed form solution. The Monte Carlo method is slower than the closed form solution since it involves a great number of full path simulations, and this significantly slows down the process. Moreover, the convergence to the correct result is fast in the beginning but is progressively slower when increasing the number of simulations. We compared the results of the full path function with our direct simulation version, and we saw that directly simulating the final distributions was extremely faster, in our case achieving a 40846% speedup. Also with Euler method it seems to consistently gain roughly 180% speedup.

We report here a plot of the error with respect to the number of iterations for both the Full Path Monte Carlo method and the Direct one:

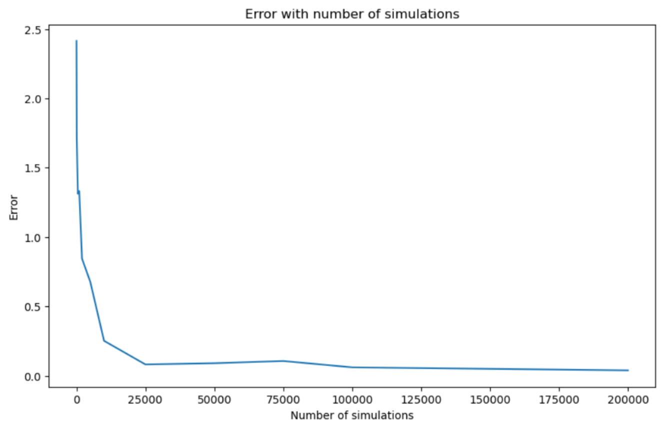


Figure 1: Error in Full Path Monte Carlo method

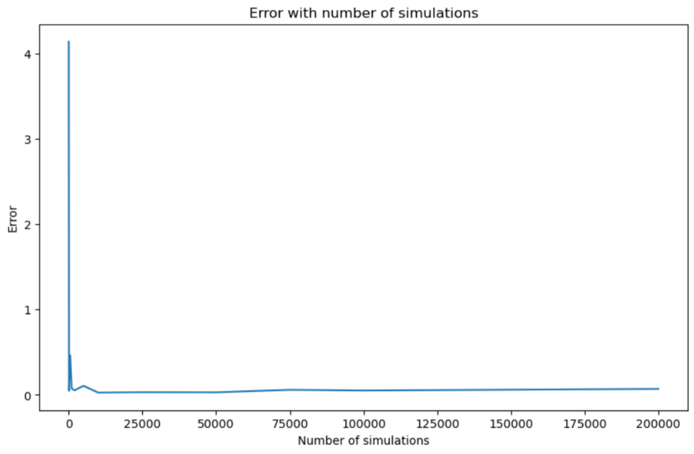


Figure 2: Error in Direct Monte Carlo method

We have to check if the order of convergence of the Monte Carlo method is $O(N^{-\frac{1}{2}})$. We do it by plotting a line with slope $-\frac{1}{2}$ on the logarithmic plot:

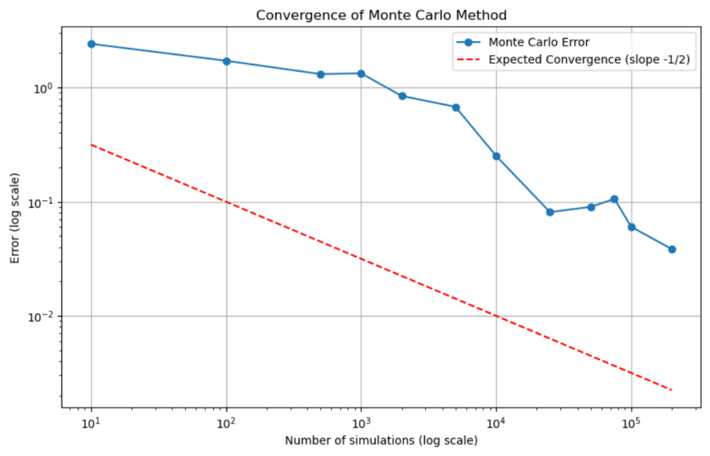


Figure 3: convergence of Full Path Monte Carlo method

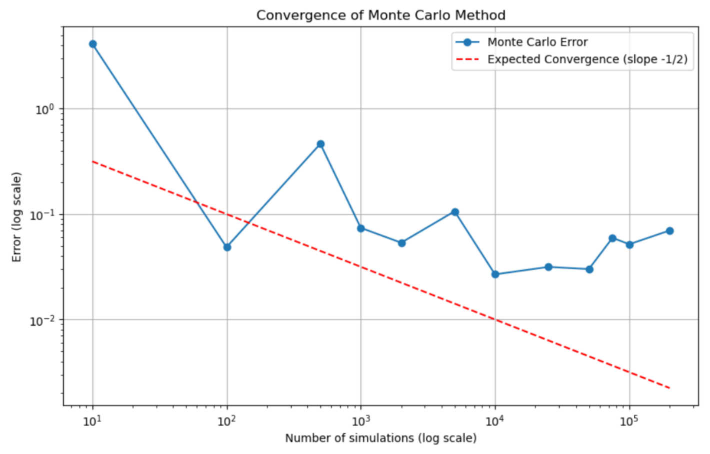


Figure 4: convergence of Direct Monte Carlo method

2.1 Extra - Time convergence analysis

We also want to study how time complexity scales with the number of simulations, so we use the previous results, which are sure to converge, and time the execution of the algorithm.

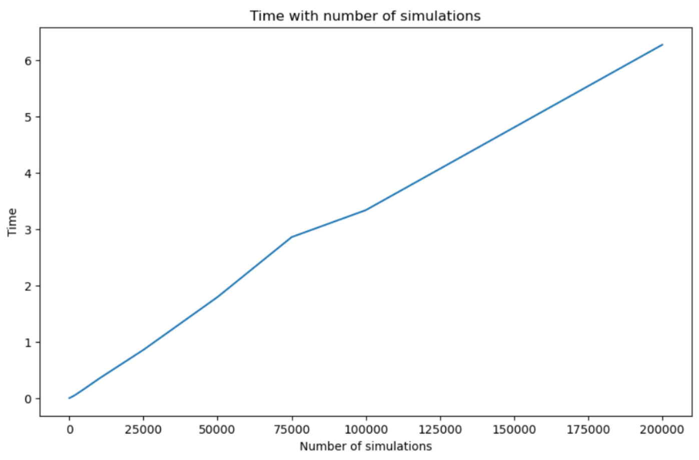


Figure 5: time convergence of Monte Carlo method

And we can see that the convergence of the method is linear $O(N)$.

Also we report the timing of the direct simulation version of the Monte Carlo method, to show the difference in times:

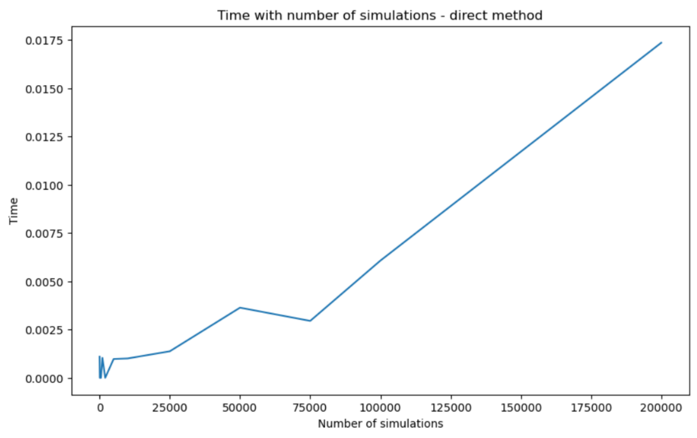
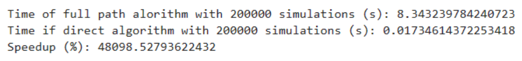


Figure 6: time convergence of direct Monte Carlo method

the speedup is calculated with a ratio of the timing of the two algorithms in the final point. The speedup that we obtain is of 48098%.



```
Time of full path alorithm with 200000 simulations (s): 8.343239784240723
Time if direct algorithm with 200000 simulations (s): 0.01734614372253418
Speedup (%): 48098.52793622432
```

Figure 7: Speedup

3 Binomial Tree Pricing

The Binomial Tree method, or CRR (Cox - Ross - Rubinstein) method, exploits a recombining tree of arbitrary complexity in order to price an option. Like Monte Carlo, it requires only one parameter in order to be applied: σ, the volatility of the underlying.
Given σ, CRR uses the following quantities:

- $u = e^{\sigma\sqrt{\Delta t}}$, meaning that the price of the underlying increases with probability q;
- $d = \frac{1}{u}$, meaning that the price of the underlying decreases with probability $1 - q$;
- $q = \frac{1-d}{u-d}$

The quantity Δt depends on the complexity of the graph. Let be N the number of time steps you are considering, then the total number of nodes follows this relationship:

$$\frac{N^2 + 3N + 2}{2}$$

This means that you can *a priori* estimate the complexity of the algorithm that scales quadratically $O(N^2)$

CRR works in a recursive-backward substitution fashion. Given N time steps, you have $N + 1$ leaves. Starting from the final leaves, you compute all the maturity values of the underlying ranging from F_0u^N to $F_0u^{-N} = F_0d^N$, you basically get $N + 1$ values for $F(t_N, t_N)$ and the corresponding $C(t_N, t_N) = (F(t_N, t_N) - K)^+$. Then proceed backwards by merging adjacent leaves in order to get the values of $F(t_i, t_N)$ for all the nodes like: $F(t_{N-1}, t_N) = qF_0u^N + (1 - q)F_0u^{N-2}$. As you have all $F(t_N, t_N)$, you can start pricing the option by positioning in t_{N-1} as:

$$\begin{aligned} C(t_{N-1}, t_N) &= B(t_{N-1}, t_N)E_{N-1}\left[(F(t_N, t_N) - K)^+\right] \\ &= e^{-r\Delta t}\left[qC(t_N, t_N)_u + (1 - q)C(t_N, t_N)_d\right] \end{aligned}$$

where $C(t_N, t_N)_u = (F_0u^N - K)^+$ and $C(t_N, t_N)_d = (F_0u^{N-2} - K)^+$. Repeat until $C(t_0, t_N)$.

With the Binomial Tree method we observe a convergence to the exact solution which is faster than the Monte Carlo method. However, since this method has a time complexity that scales with $O(N^2)$, it takes significantly

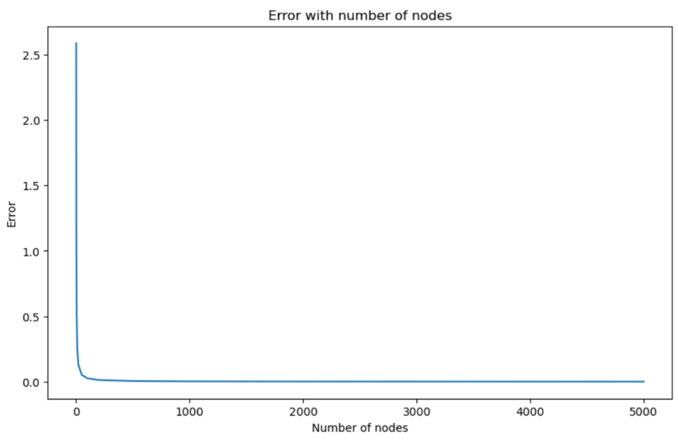


Figure 8: Error in Binomial Tree method

longer to compute when the number of nodes is great. Here we provide a plot of the error as a function of the number of nodes in the tree:

And here we analyze the convergence of the method. We plot the log-log convergence and we check with a linear regression that the line in the log-log space has a slope of -1. From this we deduce that the Binomial Tree method converges with an order of approximately $O(N^{-1})$, and we plot the results:

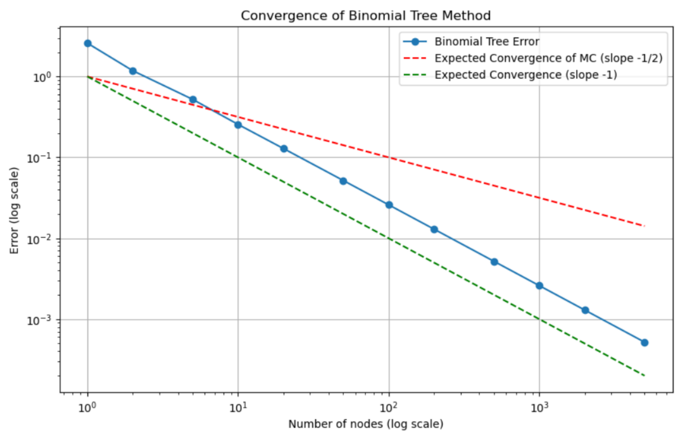
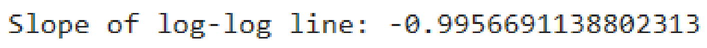


Figure 9: convergence of Binomial Tree method

Here is the result of the regression:



```
Slope of log-log line: -0.9956691138802313
```

Figure 10: Slope of log-log line

3.1 Extra - Time convergence analysis

We also want to study how time complexity scales with the number of nodes, so we use the previous results, which are sure to converge, and time the execution of the algorithm.

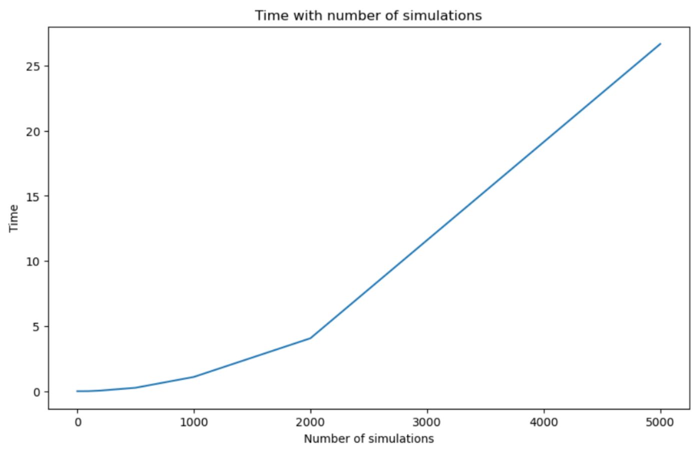


Figure 11: time convergence of Binomial Tree method

And we can see that the convergence of the method is quadratic $O(N^2)$.