Option Calculator

Assignment 17 - ANF 20247

Gianmarco Mulazzani - 3026084

Introduction

The aim of this project is to create an user-friendly interface to compute option prices given certain parameters selected by the user. In particular, this application has been developed using Pyhton and the Tkinter library. The idea is to allow the user to choose between European and American plain vanilla call and put options. Then the application automatically computes the theoretical price and the greeks. In particular, Binomial Tree Model and Black-Scholes Model are both implemented for European derivatives while for American ones only the Binomial Tree Model is used. The structure of this report is organized as follows: the first two sections present some aspects related both to the underlying theory of the above-mentioned models and their implementation in Pyhton, focusing on the functions defined in the file `Options.py`. Moreover, the third section includes the investigation of some practical aspects of the program and the code related to the graphical aspects and the usage of Tkinter. Then, after some final remarks, an appendix has been included for the computation of the Greeks.

In the end, the following picture represents an interface that inspired the one created (see `http://www.math.columbia.edu/~smirnov/options13.html`).

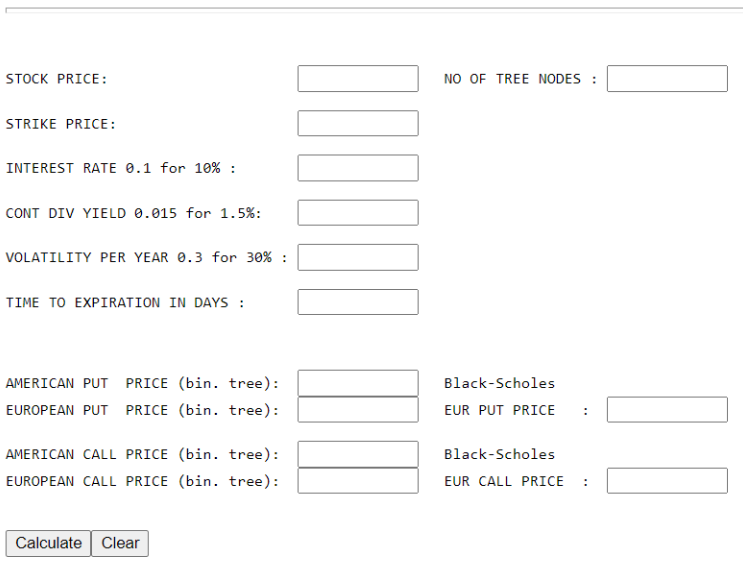

STOCK PRICE:
NO OF TREE NODES :
STRIKE PRICE:
INTEREST RATE 0.1 for 10% :
CONT DIV YIELD 0.015 for 1.5%:
VOLATILITY PER YEAR 0.3 for 30% :
TIME TO EXPIRATION IN DAYS :
AMERICAN PUT PRICE (bin. tree):
EUROPEAN PUT PRICE (bin. tree):
Black-Scholes
EUR PUT PRICE :
AMERICAN CALL PRICE (bin. tree):
EUROPEAN CALL PRICE (bin. tree):
Black-Scholes
EUR CALL PRICE :
Calculate
Clear

Figure 1: Graphical User Interface for the Option Calculator by M. Smirnov

Background Theory: Financial Issues

The aim of this paragraph is not to go through the whole theory discussed in class, but instead to clarify and clearly state some of the main formulas used in the code. Moreover, an appendix with the derivation of the greeks is included. Besides this, the aim of this section is also to remark the choice of the models, in fact Black-Scholes model is allowed to be used only for European options while the Binomial Model is implemented for both European and American Options. This is done also to check how the two models comparatively work in the case of European options. Let's first begin with the Black-Scholes Model used to price European plain vanilla options. The closed-form formulas used are the following:

$$C_0 = Se^{-qT}\mathcal{N}(d_1) - Ke^{-rT}\mathcal{N}(d_2) \tag{1}$$

$$P_0 = Ke^{-rT}\mathcal{N}(-d_2) - Se^{-qT}\mathcal{N}(-d_1) \tag{2}$$

$$\text{with } \ d_1 = \frac{ln\left(\frac{S}{K}\right) - \left(r - q + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}} \ \text{ and } \ d_2 = d_1 - \sigma\sqrt{T} \tag{3}$$

Where respectively (1) is used for call options while (2) refers to put options. As regards the greeks, the following holds for call options (check **Appendix A** for puts and general derivations):

$$\Delta = \frac{\partial C_0}{\partial S} = e^{-qT}\mathcal{N}(d_1) \tag{4}$$

$$\mathcal{V} = \frac{\partial C_0}{\partial \sigma} = Ke^{-rT}\mathcal{N}'(d_2)\sqrt{T} \tag{5}$$

$$\Theta = \frac{\partial C_0}{\partial T} = qSe^{-qT}\mathcal{N}(d_1) - rKe^{-rT}\mathcal{N}(d_2) - \frac{\sigma}{2\sqrt{T}}Ke^{-rT}\mathcal{N}'(d_2) \tag{6}$$

$$\rho = \frac{\partial C_0}{\partial r} = TKe^{-rT}\mathcal{N}(d_2) \tag{7}$$

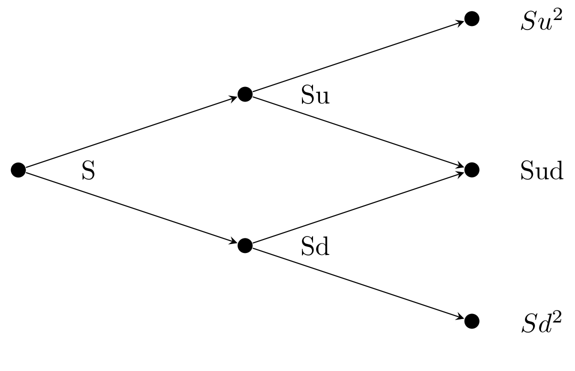
$$\Gamma = \frac{\partial \Delta}{\partial S} = \frac{1}{S\sigma\sqrt{T}}e^{-qT}\mathcal{N}'(d_1) \tag{8}$$

The second model implemented is the Binomial Tree. The reference used is the work by Cox, Ross and Rubinstein[1]. In particular, the dynamics of the underlying asset is based on what seen in class, but it is important to state the formulas used for the parameters involved in this model. Thus, the following holds:

$$u = e^{\sigma\sqrt{\Delta t}} \quad \text{and} \quad d = \frac{1}{u} = e^{-\sigma\sqrt{\Delta t}} \quad \text{with} \quad \Delta t = \frac{T}{\#nodes} \tag{9}$$

$$q = \frac{e^{(r-div)\Delta t} - d}{u - d} \tag{10}$$

Now, consider the first two steps of the lattice model for the underlying security:



Note that at each node it should be assigned the related payoff and the price of the option using common backward recursion formulas[2]. However, the aim is not to focus on these but instead present how greeks are performed in this context. Let's consider again the case of a call plain vanilla option, then it holds:

[1]Cox, John C., Stephen A. Ross, and Mark Rubinstein. "Option pricing: A simplified approach." Journal of financial Economics 7.3 (1979): 229-263.

[2]Recall the distinction between European options where $V_X(t) = E_t^{\mathcal{Q}}\left[\frac{V_X(t+\Delta t)}{1+r\Delta t}\right]$, while for the America option also the immediate payoff must be taken into account and thus it is needed to solve a maximization problem between $X(t)$ and the continuation value (CV).

$$\Delta = \frac{C_1(u) - C_1(d)}{S_1(u) - S_1(d)} \tag{11}$$

$$\Theta = \frac{C_2(ud) - C_0}{2\ \Delta t\ 365} \tag{12}$$

$$\mathcal{V} = \frac{C_0(\sigma + 1\%) - C_0(\sigma - 1\%)}{100\ 2\%} \tag{13}$$

$$\rho = \frac{C_0(r + 1\%) - C_0(r - 1\%)}{100\ 2\%} \tag{14}$$

Notice that there is a slight abuse of notation in the above formulas. Thus, to be more precise consider that $C_t(u)$ indicates the price of the option at time t given an upward trend and similarly for the other terms of above equations. While $C_0(r + 1\%)$ indicates the price of the option at time 0 when an increase of 1% of the interest rate is taken into account. In this way, it is possible to apply central difference estimators for the computations. In the end, the calculation of Gamma requires a further step. Firstly, two Deltas are computed at 2 Δt and then the value of Gamma is retrieved as follows:

$$\Delta_1(u) = \frac{C_2(uu) - C_2(ud)}{S_2(uu) - S_2(ud)} \tag{15}$$

$$\Delta_1(d) = \frac{C_2(ud) - C_2(dd)}{S_2(ud) - S_2(dd)} \tag{16}$$

$$\Gamma = \frac{\Delta_1(u) - \Delta_1(d)}{S_1(u) - S_1(d)} \tag{17}$$

Background Theory: Coding Issues

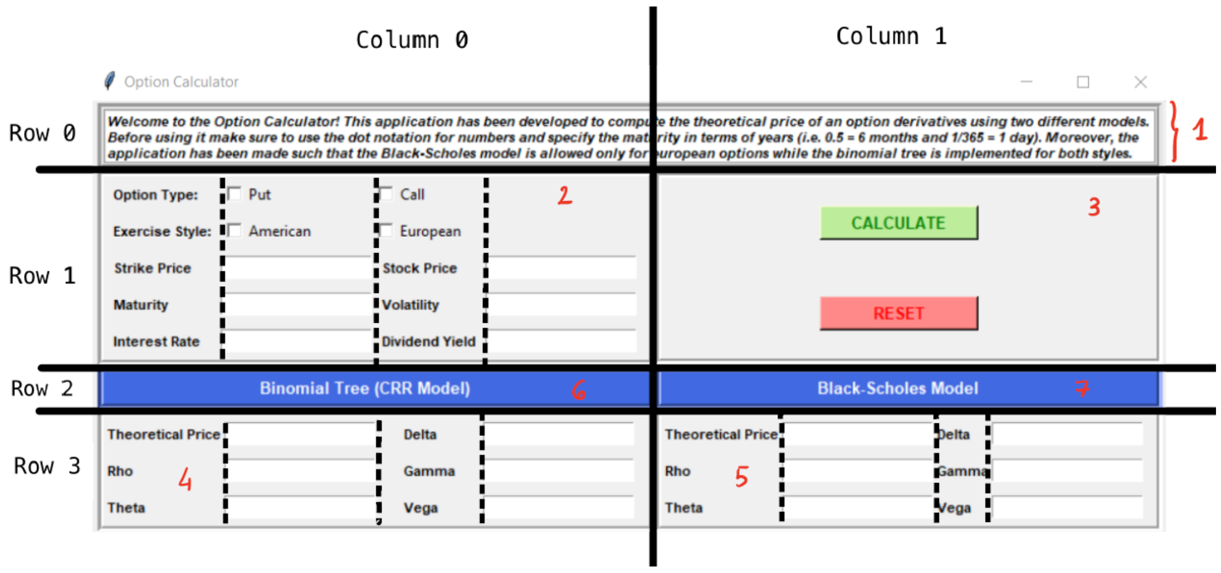
In this section the above theoretical concepts are contextualized in a Python script. The idea is to create an independent file from which the user can retrieve the relevant formulas that can be used also in other projects. In particular, these functions are defined in the file `Options.py` and they are called: `BinomialTree`, `GreeksBinomial` and `BlackScholes`. They respectively implement a Binomial Tree Model, compute Vega and Rho in the Binomial Tree Model and implement the Black-Scholes Model. Starting from the `BlackScholes` function, its implementation in Python is quite simple since it is just necessary to write the formulas using standard mathematical notation, the only element is that for specific statistics functions the `scipy` package is needed. Moreover, to distinguish the two cases of a put and call option an `if` loop has been used. As regards the `BinomialTree` function, the code used is less immediate. In fact, in order to implement the backward recursions, it has been necessary to initialize some zero matrices and then populate them using nested `for` loops. To be more precise, these matrices are `lattice`, `payoff` and `price` and they include the lattice dynamics of the underlying asset, the payoff at each node[3] and the price of the option respectively. The only element of attention is to correctly specify the indices over which the loop must run, and this requires to correctly specify the dimension of the matrix which is a square matrix: $(nodes + 1)$`x`$(nodes + 1)$, where *nodes* indicate the number of nodes chosen, which is 150 but it could be easily changed. To conclude, the last function implemented is `GreeksBinomial` which is used to compute Rho and Vega in the case of the Binomial Model. This computation is separated from the previous function because it has been used within the `GreeksBinomial` itself.

Practical Implementation

This application has been developed using the Python programming language and specifically the Tkinter (`ttk` for short) package that permits to create Graphical User Interfaces (GUI) in a simple way. In particular, this Python's

[3]For put options it is: $X(t) = (K - S(t))^+$ and for call options it is: $X(t) = (S(t) - K)^+$

extension works using a hierarchical approach, meaning that you need to specify the objects, called widgets, from the most external ones. To better understand the underlying structure consider the following picture:



Welcome to the Option Calculator! This application has been developed to compu e the theoretical price of an option derivatives using two different models. Before using it make sure to use the dot notation for numbers and specify the mat rity in terms of years (i.e. 0.5 = 6 months and 1/365 = 1 day). Moreover, the application has been made such that the Black-Scholes model is allowed only for uropean options while the binomial tree is implemented for both styles.

Figure 2: GUI of the Option Calculator: underlying structure

In the code, the main framework is called `root` and inside it, seven different frames (called `frame` in `ttk`) have been defined (indicated by red numbers in Figure 2). Then, inside each of this frame, different widgets have been collocated. Precisely, the following holds:

- First Frame (`frame1`): a text object identified by the `label` widget is defined in order to include a brief description on the functioning of the application.
- Second Frame (`frame2`): all the inputs required to perform the computations are included in this frame. In particular, the `label` widget is used to define the name of the variable, while `entry` is used to create a small empty window where the user can type the value (notice that this is saved as a string, thus it will be necessary to convert it into a number using `float`). Moreover, the widget `CheckButton` is used to create some check boxes for the variables: option type and exercise style.
- Third Frame (`frame3`): two buttons (called `button` in `ttk`) are created, thus the user can choose between *Calculate* or *Reset* to perform the tasks desired. Note that two specific functions have been assigned to these buttons so they are activated as soon as the user clicks on them.
- Fourth Frame (`frame4`): all the outputs related to the Binomial Tree Model are included in this frame. They include the theoretical price and the greeks. Again, `label` and `entry` widgets have been used.
- Fifth Frame (`frame5`): all the outputs related to the Black-Scholes Model are included in this frame. They include the theoretical price and the greeks. Again, `label` and `entry` widgets have been used.
- Sixth Frame (`frame6`): this is just an auxiliary frame created to insert the title indicating the model used. In this case: `Binomial Tree (CRR Model`.
- Seventh Frame (`frame7`): this is just an auxiliary frame created to insert the title indicating the model used. In this case: `Black-Scholes Model`.

It is outside the scope of this report to explain in detail Tkinter, but it is obvious that all the above elements can be modified in terms of font, colour and size to make the interface aesthetically good. Moreover, each element must be positioned in the general framework that can be simply seen as a grid (see Figure 2, solid and dashed lines), in fact the method used is `grid`. Besides this, an important note is to go a bit deeper on the functions used. As already mentioned, two functions are defined:

- Calculate Function (called `OptionPricing`): this simply activate the functions defined in the previous paragraph given the parameters inserted by the user. In particular, based on the choice of the exercise style, only `BinomialTree` and `GreeksBinomial` functions are retrieved for American options, while all the three functions defined above are used for European options.
- Reset Function (called `cancel`): this is used to cancel out all the inputs inserted by the user and also all the outputs if any.

Conclusion

Running the code called `Option_Calculator.py`, the interface illustrated above should appear. Then, the user insert all the inputs (make sure to insert everything correctly) and using the button `Calculate` the results should be printed out. An interesting fact is that for European options can be easily seen how the Binomial Model performs in comparison to the Black-Scholes one, and results are encouraging. Obviously, different improvements could be introduced to make the interface more complete, such as: the usage of more models like Jump Diffusion Model, the possibility to price also more complex options and finally the interest rate could be inserted automatically by the system that retrieves it from some official source.

Appendix A

The scope of this Appendix is to explain how the computations of the greeks have been carried out in the Black-Scholes Model. Firstly, let's state the greeks formulas for a plain vanilla European Put Option:

$$\Delta = \frac{\partial C_0}{\partial S} = -e^{-qT}\mathcal{N}(-d_1) \tag{18}$$

$$\mathcal{V} = \frac{\partial C_0}{\partial \sigma} = Ke^{-rT}\mathcal{N}'(d_2)\sqrt{T} \tag{19}$$

$$\Theta = \frac{\partial C_0}{\partial T} = -qSe^{-qT}\mathcal{N}(-d_1) + rKe^{-rT}\mathcal{N}(-d_2) - \frac{\sigma}{2\sqrt{T}}ke^{-rT}\mathcal{N}'(d_2) \tag{20}$$

$$\rho = \frac{\partial C_0}{\partial r} = -TKe^{-rT}\mathcal{N}(-d_2) \tag{21}$$

$$\Gamma = \frac{\partial \Delta}{\partial S} = \frac{1}{S\sigma\sqrt{T}}e^{-qT}\mathcal{N}'(-d_1) \tag{22}$$

In particular, consider the Black-Scholes Model and the case of an European Call Option. The following steps describe the derivation of the Delta and show some crucial steps that are behind the computation also of other derivatives. Given the formulas already presented above, consider the following:

$$\Delta = \frac{\partial C_0}{\partial S} = \frac{\partial}{\partial S}[Se^{-qT}\mathcal{N}(d_1) - Ke^{-rT}\mathcal{N}(d_2)] \tag{23}$$

$$= \frac{\partial}{\partial S}[Se^{-qT}\mathcal{N}(d_1)] - \frac{\partial}{\partial S}[Ke^{-rT}\mathcal{N}(d_2)] \tag{24}$$

In equation 4, the first term is the derivative of a product and also the derivative, with respect to S, of $\mathcal{N}(d_1)$ can be solved using the chain rule:

$$\frac{\partial \mathcal{N}(d_i)}{\partial S} = \frac{\partial \mathcal{N}(d_i)}{\partial d_i}\frac{\partial d_i}{\partial S} \tag{25}$$

$$\text{where } \frac{\partial \mathcal{N}(d_i)}{\partial d_i} = \frac{1}{\sqrt{2\pi}}e^{-d_i^2/2} = \mathcal{N}'(d_i) \tag{26}$$

Thus, equation 3 can be written as:

$$\Delta = e^{-qT}\mathcal{N}(d_1) + Se^{-qT}\mathcal{N}'(d_1)\frac{\partial d_1}{\partial S} - Ke^{-rT}\mathcal{N}'(d_2)\frac{\partial d_2}{\partial S} \tag{27}$$

A useful result, which behind also the computation of all other derivatives, is the following:

$$Se^{-qT}\mathcal{N}'(d_1)\frac{\partial d_1}{\partial S} - Ke^{-rT}\mathcal{N}'(d_2)\frac{\partial d_2}{\partial S} = 0 \tag{28}$$

$$\text{given } \frac{\partial d_2}{\partial S} = \frac{\partial}{\partial S}[d_1 - \sigma\sqrt{T}] = \frac{\partial d_1}{\partial S} \text{ follows} \tag{29}$$

$$Se^{-qT}\mathcal{N}'(d_1) - Ke^{-rT}\mathcal{N}'(d_2) = 0 \tag{30}$$

$$Se^{-qT}\frac{1}{\sqrt{2\pi}}e^{-d_1^2/2} = Ke{-}rT\frac{1}{\sqrt{2\pi}}e^{-d_2^2/2} \tag{31}$$

Writing everything as exponential and substituting into d_1 and d_2 their original formulas, it gets the following equations:

$$exp\left\{ln(S) - qT - \frac{[ln(S/K) + (r+q+\sigma^2/2)]^2}{2\sigma^2 T}\right\} = exp\left\{ln(K) - rT - \frac{[ln(S/K) + (r+q-\sigma^2/2)]^2}{2\sigma^2 T}\right\} \tag{32}$$

Just focus on the argument of the exponential function and let's prove that they are equal:

$$2\sigma^2 Tln(S) - 2q\sigma^2T^2 - (ln(S/K))^2 - (r+q+\sigma^2/2)^2 - 2ln(S/K)(r+q+\sigma^2/2) = \tag{33}$$

$$2\sigma^2 Tln(K) - 2r\sigma^2T^2 - (ln(S/K))^2 - (r+q-\sigma^2/2)^2 - 2ln(S/K)(r+q-\sigma^2/2) \tag{34}$$