Quant Scholarship Program

Report: Introduction to Option Pricing

Krzysztof Głowacz

April 2023

Exercise 1

Implement Black-Scholes formulas for option prices and deltas (in any programming language or spreadsheet). Prepare a short report

Exercise 1 was based on implementing Black-Scholes formulas for option prices and deltas. I have decided to implement them in Python.

- *Check your implementation: verify that the difference of the deltas for put and call for the same stock price (S) is 1 (for call – put) and the difference of their values is $S - K$ (put-call parity).*

  Let's take a look at the Black-Scholes formulas:

$$BS_{call} = S_0\Phi(d_1) - Ke^{-rT}\Phi(d_1 - \sigma\sqrt{T})$$

$$BS_{put} = Ke^{-rT}\Phi(\sigma\sqrt{T} - d_1) - S_0\Phi(-d_1)$$

$$\frac{\partial BS_{call}}{\partial S_0} = \Phi(d_1)$$

$$\frac{\partial BS_{put}}{\partial S_0} = -\Phi(-d_1)$$

  ,where:

$$d_1 = \frac{ln\frac{S_0}{K} + (r + \frac{\sigma^2}{2})T}{\sigma\sqrt{T}}$$

  Hence the difference of the deltas for put and call for the same stock price (call-put):

$$\frac{\partial BS_{call}}{\partial S_0} - \frac{\partial BS_{put}}{\partial S_0} = \Phi(d_1) - (-\Phi(-d_1)) = \Phi(d_1) + (1 - \Phi(d_1)) = 1$$

  And the difference of their values:

$$BS_{call} - BS_{put} = S_0 - Ke^{-rT}$$

  This equation matches the put-call parity: $Call - Put = S - K$, after multiplying it by the discount factor: e^{-rT}.

  To test my implementation I used the pytest module to write some parametrized tests. I prepared a few sets of input parameters and checked, whether the results of running the functions computing the option values and deltas, matches the expected values (based on the equations presented above). The differences in the values were rounded to avoid round-off error:

```python
@pytest.mark.parametrize(
    "initPrice, K, drift, time, volatility, expectedResult",
    [
        (100, 120, 0.05, 1, 0.2, (1.0, round(100-(120*math.exp(-0.05*1)), 10))),
        (100, 50, 0.05, 1, 0.2, (1.0, round(100-(50*math.exp(-0.05*1)), 10))),
        (100, 120, 0.35, 1, 0.2, (1.0, round(100-(120*math.exp(-0.35*1)), 10))),
        (100, 120, 0.15, 3.5, 0.2, (1.0, round(100-(120*math.exp(-0.15*3.5)), 10))),
        (100, 120, 0.35, 0.4, 0.6, (1.0, round(100-(120*math.exp(-0.35*0.4)), 10))),
        (100, 150, 0.2, 2.333, 0.2, (1.0, round(100-(150*math.exp(-0.2*2.333)), 10))),
        (50, 40, 0.1, 5, 0.1, (1.0, round(50-(40*math.exp(-0.1*5)), 10))),
        (50, 30, 0.5, 3, 0.1, (1.0, round(50-(30*math.exp(-0.5*3)), 10))),
        (50, 10, 0.999, 0.5, 0.1, (1.0, round(50-(10*math.exp(-0.999*0.5)), 10))),
        (50, 500, 0.7, 2, 0.1, (1.0, round(50-(500*math.exp(-0.7*2)), 10)))
    ]
)
def testCallPutParity(initPrice, K, drift, time, volatility, expectedResult):
    resultTuple = callPutParityDifferences(initPrice, K, drift, time, volatility)
    assert expectedResult == (resultTuple[0], round(resultTuple[1], 10))
```

Figure 1: Prepared parametrized tests.

After running those tests the result is the following:

```
============================================= test session starts =============================================
platform win32 -- Python 3.11.0, pytest-7.2.2, pluggy-1.0.0
rootdir: C:\Users\Kris\Documents\Studia\Quant Scholarship\Source_files\Assignment_5
plugins: anyio-3.6.2
collected 10 items

Black_Scholes_formulas.py ..........                                                                 [100%]

============================================== 10 passed in 0.98s =============================================
```

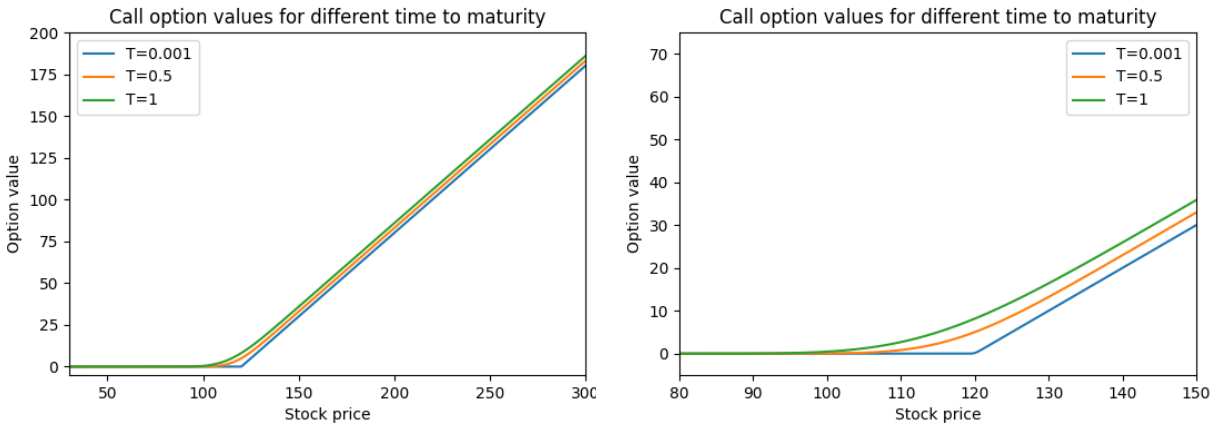
Figure 2: Result of tests execution.

- *In a spreadsheet, plot values of a call option (on one chart) as a function of stock price:*
  - for different times to maturity: 0.001 (almost at expiry), 0.5 and 1 year, for volatility being 10%
  - include spots "left from" and "right to" the option strike ($0 < S_L < K < S_H$)

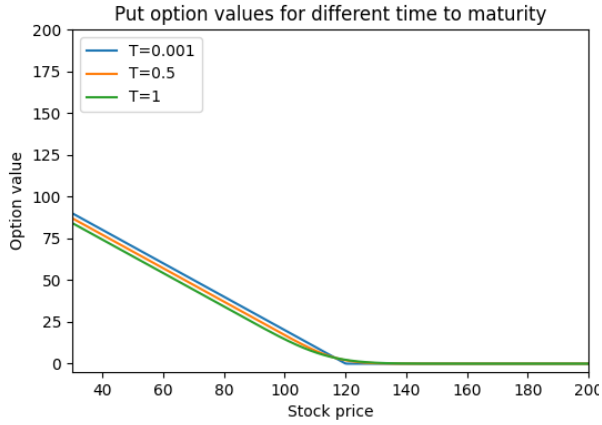
  To plot that chart I used a Python module - matplotlib. The parameters values are the following:

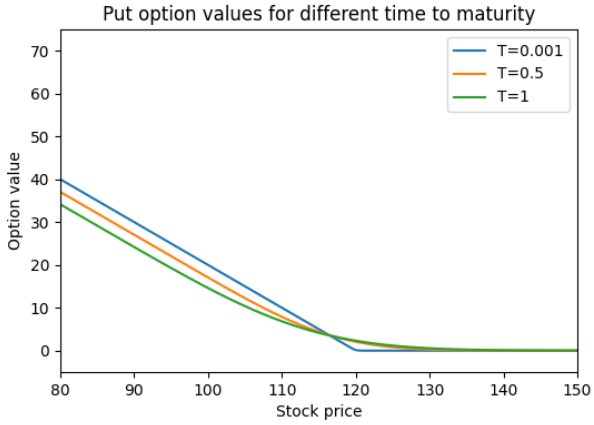
$$K = 120, r = 0.05, volatility = 0.1$$

  In each of the following points, both charts present the same functions. The chart on the right is "zoomed" and centering around the K spot. The parameters values remain the same.

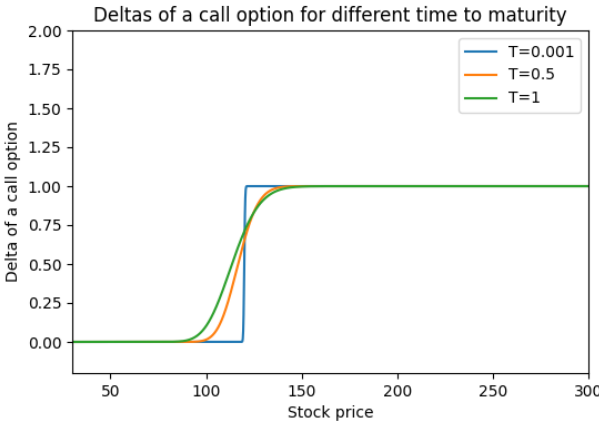


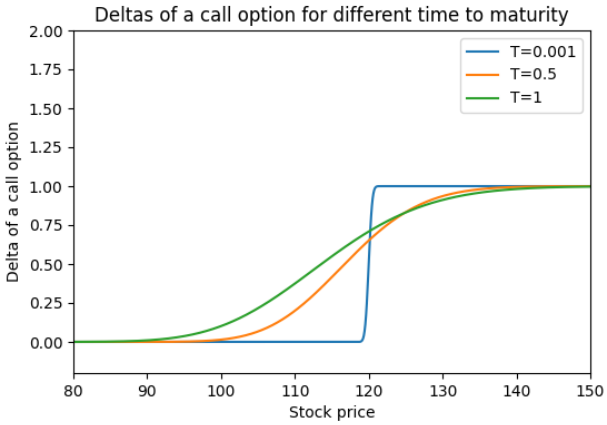
- *Repeat the previous point for values of a put option*



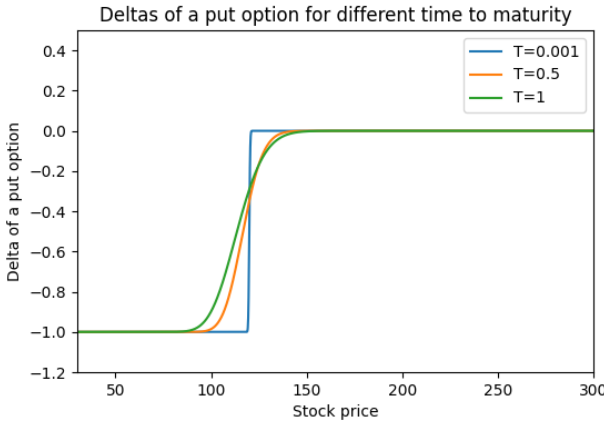


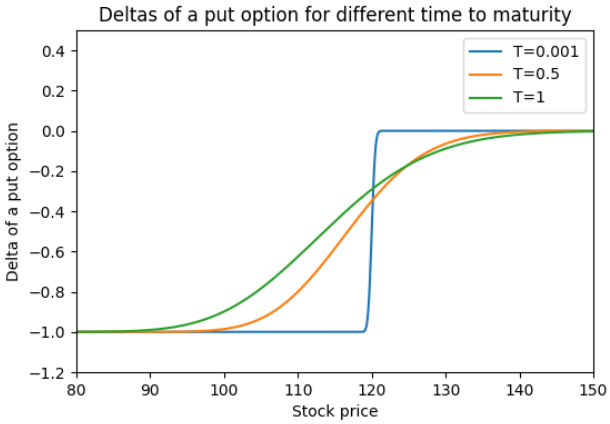
- *Repeat the previous point for deltas of a call option*





- *Repeat the previous point for deltas of a put option*

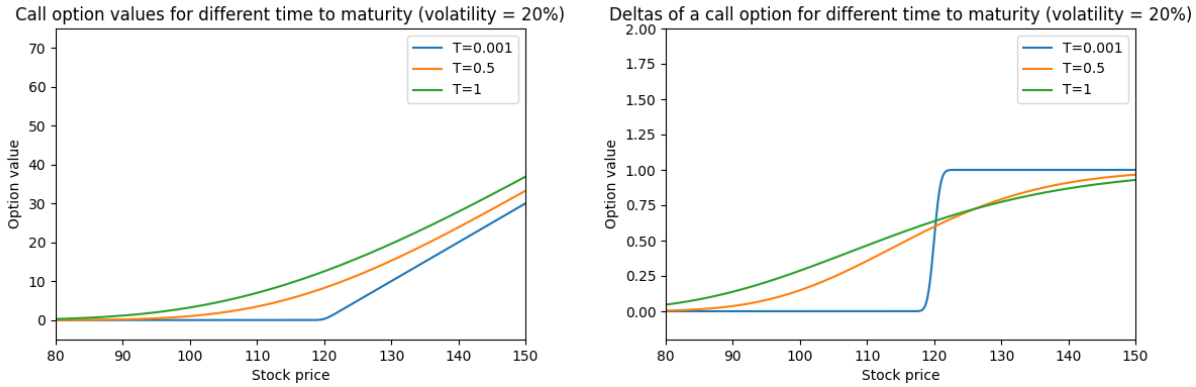




Exercise 2

Repeat exercises 1b. (plotting values of a call option) and 1d. (plotting deltas of a call option) for a different volatility level (20%). What has changed compared to the case where $\sigma = 10\%$ and why?

The parameters used in that exercise were the same as previously. The only change was the change of the volatility level - from 10% to 20%. The charts are shown below.



We can see that in general, the relationship between the three functions graphs at each chart (which differ only in time to maturity) remains the same compared to the corresponding charts with $\sigma = 10\%$.

However, having compared those charts to the corresponding charts from the previous points, we can observe, that the graphs computed for the volatility set to 20% are much "smoother" and less centered around the spot K than the graphs with volatility set to 10%. Call option values for the corresponding time to maturity and stock price are in that scenario higher than in the previous one. This is because the higher volatility means, that larger price movements are more likely to happen, which increases the probability of the option ending up in the money. Therefore, if the volatility increases, the option prices will also increase. Graphs presenting deltas of a call option with volatility set to 20% are also less centered around the spot K than in the case with $\sigma = 10\%$. This means that for stock prices higher than K, deltas of a call option are smaller, and for stock prices lower than K - higher.

To understand this let's think about delta as the partial derivative of the call option value with respect to the initial stock price of the asset. In general, the partial derivative measures the sensitivity of the function to changes in one of its variables, while holding the other variables constant. When a partial derivative is equal to one, an increase in the variable will result in a proportional increase in the function by the same amount. Thus, $\frac{\partial BS_{call}}{\partial S_0} = 1$ means that there is a perfect correlation between the option value and the price of the asset. On the call option values it is represented by the linear growth of the graphs. By analogy, when $\frac{\partial BS_{call}}{\partial S_0} = 0$ the call option value is not affected by changes in the asset's price.

Now it is easier to understand, that with $\sigma = 20\%$ option prices are in general less correlated with the price of the underlying asset than in the scenario with $\sigma = 10\%$. This observation is more visible on the graphs with a longer time to maturity. The reason here is the higher volatility itself, so a higher probability of larger price changes. Option values cannot be that correlated to the price of the asset, because the higher volatility, the more scenarios with a different initial stock price of the asset can now end up in the money. But to achieve this, they need some time (the more time they have, the more likely it is to happen), so the amount of time to maturity also plays a crucial role here.

Exercise 3

How would you calculate a value of a contract paying 1 unit of cash, if the stock price on maturity is above some level K (i.e. $S_T > K$), 0 otherwise. Explain your reasoning.

The value of such a contract would be based on the probability of the option ending up in the money. In the Black-Scholes formula, this term: $\Phi(d_1 - \sigma\sqrt{T})$, represents the risk-adjusted probability that the option will be exercised. To calculate the value of the contract, we could multiply this probability by the amount of money, that potentially would be paid, so by 1. We obtain:

$$V_{contract} = 1 \cdot \Phi(d_1 - \sigma\sqrt{T})$$

However, this formula refers to the future scenario, so we have to include the change in the value of money over time. To achieve this let's multiply the formula by a discount factor: e^{-rT}.
Hence, the value of the contract could be calculated using the following formula:

$$V_{contract} = e^{-rT} \cdot \Phi(d_1 - \sigma\sqrt{T})$$

I have implemented this formula in Python and ran two simulations. Here are the results:

```
Parameters for the calculation:
Initial stock price: 60
K: 50
Risk free rate: 0.05
Time to maturity: 1
Volatility: 0.2
Calculated value of the option is: 0.814



Parameters for the calculation:
Initial stock price: 100
K: 120
Risk free rate: 0.1
Time to maturity: 0.5
Volatility: 0.1
Calculated value of the option is: 0.027
```

Figure 3: Calculated contract values.

Such a contract is an example of a binary option in which the payoff is either some fixed amount of money or nothing at all.