Advanced Derivatives

FIN-503 2024

HW 2 : Monte Carlo Simulation of Option Pricing

Authors:
Edouard Bueche 319539
Jonas Dunand 315725
Guillaume Ferrer 311582

Instructor:
Elena Perazzi

To begin, we are going to compute the undiscounted price of the call option on our stocks using Black-Scholes-Merton undiscounted equation (fix t=0):

$$\tilde{C}(S_0, K, \sigma_{BS}(K,T), T) = S_0 e^{(r-q)T} N(d_1) - K N(d_2) \tag{1}$$

$$d_1 = \frac{\log(\frac{S_0}{K}) + (r - q + \frac{\sigma^2}{2})T}{\sigma\sqrt{T}} \qquad d_2 = d_1 - \sigma\sqrt{T} \tag{2}$$

The fig.1 shows the call price in function of the strike price with maturity T = $\frac{105}{365} \approx 0.288$.

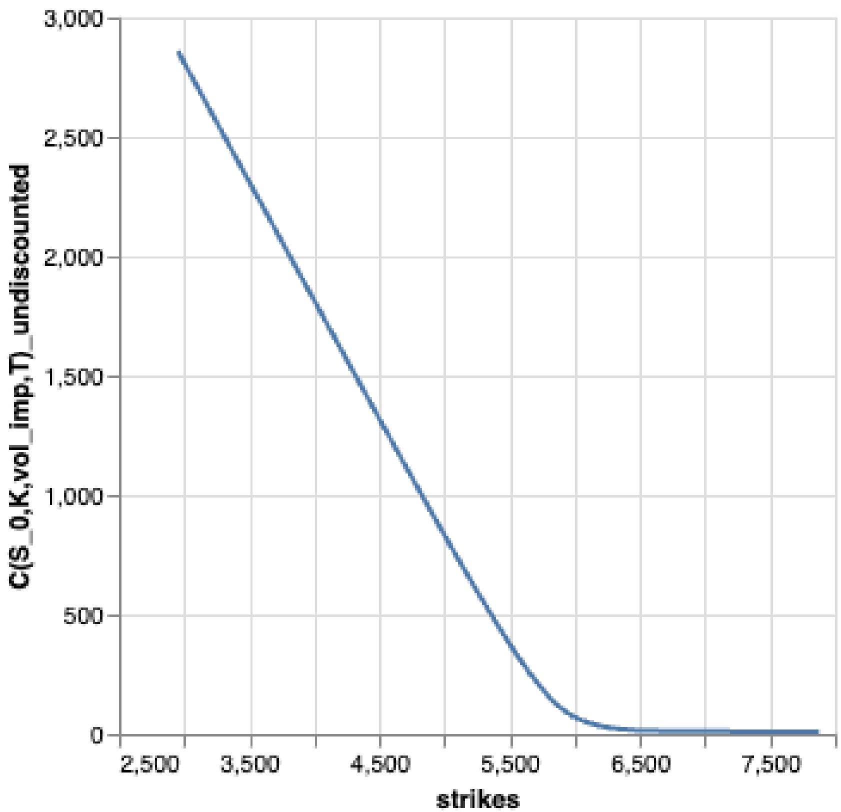


Figure 1: Undiscounted call price of SPX stock.

From the Breeden-Litzenberger formula we know that:

$$\frac{\partial C}{\partial K} = \Phi(K; S_0) - 1 \qquad \frac{\partial^2 C}{\partial K^2} = \phi(K; S_0) \tag{3}$$

The first derivative of the call option w.r.t. the strike is the shifted implied CDF and the second order derivative the PDF of the stock distribution.

Since we have discrete prices, basic analysis tells us that we can approximate this derivative by the finite difference:

$$\frac{\partial C}{\partial K} \approx \frac{C(K+\Delta K; S_0) - C(K; S_0)}{\Delta K} \tag{4}$$

As we are interested to model the stock's price, we want the cdf of the stock at time T, it shown on fig.2

$$\Phi_{implied}(K; S_0) = \frac{\partial C}{\partial K} + 1 \tag{5}$$

We have fixed the first value to 0 and the last to 1.

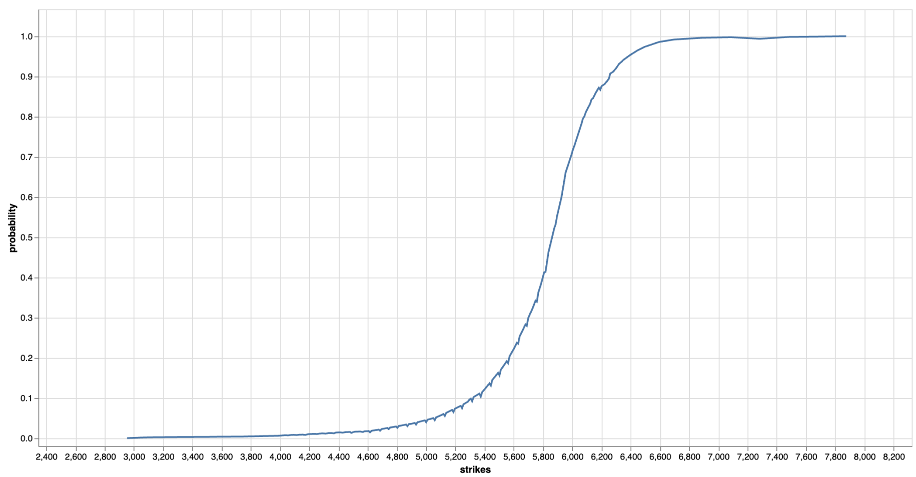


Figure 2: Implied cumulative density of the SPX stock at maturity T.

We have worked with that CDF for the rest of the homework, but in the code we also did 2 differents version: Smooting the curve or erasing some values to make the CDF strictly increasing.

Now, since the stock are correlated, we account for that correlation, the solution is to use the Copula method and then when we have the underlying variable correlated, we can use the inverse of the implied cdf to find the stock's price at maturity.

$$\begin{pmatrix} S_{SPX}(T) \\ S_{AMZN}(T) \end{pmatrix} = \begin{pmatrix} \Phi^{-1}_{implied,SPX}(\Phi_{normal}(x)) \\ \Phi^{-1}_{implied,AMZN}(\Phi_{normal}(y)) \end{pmatrix} \qquad \begin{pmatrix} x \\ y \end{pmatrix} \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}\right) \tag{6}$$

where x and y have been drawn from a bivariate normal distribution shown in fig.3 with correlation $\rho = 0.5$.

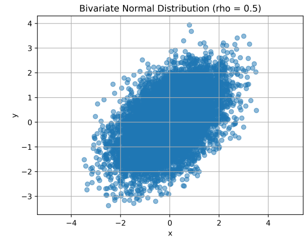
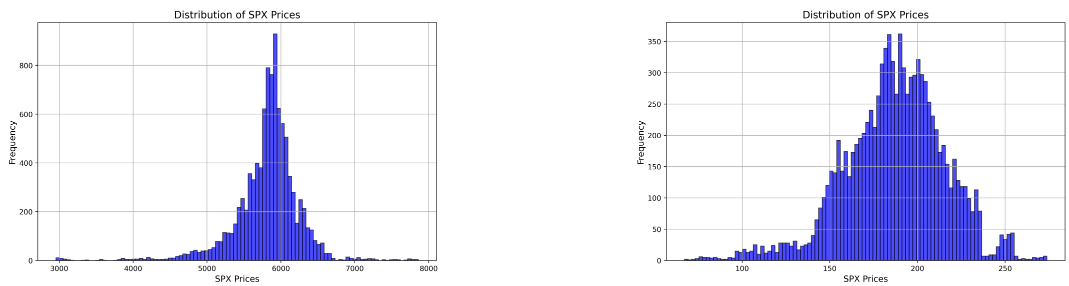


Figure 3: Bivariate normal distribution

We can now generate the stock's price for SPX and AMZN. The prices of for both stocks are shown in fig.4



(a) SPX Prices Distribution

(b) AMZN Prices Distribution

Figure 4: SPX and AMZN Price Distributions

As it can be seen the AMZN stock price is much more volatile than the SPX index prices.

We are left computing the discounted expected payout of the option. In the absence of arbitrage, FTAP guarantees the existence of a risk-valuation neutral measure under which the discounted price is a martingale $\mathbb{Q}$.

$$O_t(T) = e^{-rT}\mathbb{E}^{\mathbb{Q}}_{t=0}\left[\left(\frac{S_{SPX,T}}{S_{SPX,0}} - \frac{S_{AMZN,T}}{S_{AMZN,0}}\right)^+\right] \approx \frac{e^{-rT}}{gen}\sum_{i=1}^{gen}\left(\frac{S_{SPX,T}(i)}{S_{SPX,0}} - \frac{S_{AMZN,T}(i)}{S_{AMZN,0}}\right)^+ \tag{7}$$

To find the price we computed $gen = 10000$ bivariate and evaluated the price. In order to find the standard error, this process was done 100 times.

```python
def compute_option_price(nbr_samples = 10000, corr = 0.5)
    expected_option_price = []
    #Iterate to compute the std of our simulation
    for i in range(100):
        # Generate copulas
        x_spx, x_amzn = Generate_copulas(corr,nbr_samples)
        # Compute the prices of the stocks using the correlated data
```

```python
        spx_prices = compute_price_T_inter(x_spx,spx_implied_cdf)
        amzn_prices = compute_price_T_inter(x_amzn,amzn_implied_cdf)
        # Evaluate the risk-valuation expectation and discount it to today
        expected_option_price.append(np.mean(np.maximum(spx_prices/S_SPX_0-amzn_prices/S_AMZN_0,0)*np.exp(-r * T)))
    return np.mean(expected_option_price), np.std(expected_option_price)
```

Which outputs:

Expected Option Price	0.054606 \$
Simulation Standard Error	0.000858

Table 1: Code output