Professor Perazzi Elena

Teaching assistant Hayes Joshua

Assignment 1

Group U

Marius Pécaut 330684
Timothée Pottié 327549
Cyprien Tordo 345618

Introduction

This report presents our solutions to Assignment 1 for the course FIN-503 – Adv. Derivatives. The accompanying Jupyter Notebook `GroupU_A1.ipynb` contains the Python implementations supporting the analysis.

We consider a non-dividend paying stock with current value $S = 100$. Under the risk-neutral measure $\mathbb{Q}$, the stock follows a jump-diffusion process :

$$\frac{dS}{S} = (r - \lambda^Q \gamma)dt + \sigma dW_t + \gamma dN_t \tag{1}$$

where we assume $r = 0.04$, $\sigma = 0.2$, $\lambda^Q = 0.2$ and $\gamma = -0.08$ to be constant parameters.

In the first part, we will determine the value of a call option at different expiration dates $T = 0.02, 0.08, 0.25, 0.5$ (corresponding approximately to 1 week, 1 month, 3 months and 6 months), across different strike prices K : $\frac{K}{S} = 0.8, 0.9, 1.0, 1.1$.

In a second part, we will compute the implied volatility of the call prices. That is, for every expiration T, we will plot the Black-Scholes implied volatility of the option as a function of moneyness $\frac{K}{S}$.

1 Methodology

1.1 Call prices using Merton's argument

Based on the derivations presented in Lecture 2, we can find a closed form for the underlying's value at maturity S_T using Merton's argument. The major subtleties with what was done in the lecture is the inverted sign of γ and the discarding of q.

Using the risk-neutral dynamics of our underlying asset (1), we start by applying Itô to $d(\log(S_t))$:

$$d(\log(S_t)) = (r - \lambda^Q \gamma - \frac{\sigma^2}{2})dt + \sigma dW_t^Q + \log(1+\gamma)dN_t$$

Integrating the above expression, we find:

$$\log\left(\frac{S_T}{S_t}\right) = (r - \lambda^Q \gamma - \frac{\sigma^2}{2})(T-t) + \sigma(W_T - W_t) + \log(1+\gamma)(N_T - N_t)$$

Extracting S_T leads to:

$$S_T = S_t(1+\gamma)^{(N_T-N_t)} e^{(r-\lambda^Q\gamma-\frac{\sigma^2}{2})(T-t)+\sigma(W_T-W_t)} \tag{2}$$

With that, Merton's argument (as covered in Lecture 2) writes as:

$$Call_{Merton} = \sum_{j=0}^{\infty} \mathbb{P}(N_T - N_t = j) \cdot \underbrace{\mathbb{E}_t^Q \left[e^{-r(T-t)} \left(S_t(1+\gamma)^j e^{(r-\lambda^Q\gamma-\frac{\sigma^2}{2})(T-t)+\sigma(W_T-W_t)} - K \right)^+ \right]}_{Call_{BS}(S=S_t(1+\gamma)^j, K, T, \sigma, r, q=\lambda\gamma)}$$

This formula is essentially a weighted average of Black-Scholes prices with "adjusted inputs":

- The spot price is taken as $S_t(1+\gamma)^j$
- The dividend rate is taken as $\lambda\gamma$
- The rest of inputs are not modified

The average is weighted by the probability that we observe exactly j jumps between current date t and maturity T, that is $\mathbb{P}(N_T - N_t = j) = \frac{e^{-\lambda(T-t)}(\lambda(T-t))^j}{j!}$.

The python implementation of this pricing is pretty straightforward, using the two following functions. For simplification, we put ourselves at $t = 0$, thus we only need T as a time parameter.

- `BS_call_price(S0, K, T, sigma, r, q)`
  This function computes the price of a European call option under the Black–Scholes model. It takes the standard option parameters as inputs, computes the intermediate terms $d1$ and $d2$, and then evaluates the Black–Scholes closed-form expression.
- `Merton_call_price(S0, K, T, sigma, r, q, lam, gamma, max_jumps)`
  This function prices a European call option under Merton's jump-diffusion model. As explained previously, it sums over possible numbers of jumps in $[0, T]$, weighting each case by its Poisson probability. For each scenario, the call to the Black-Scholes pricing function is done with carefully injecting the "adjusted inputs" that we described earlier (to incorporate jump risk).

Note that we should theoretically consider an infinite number of jumps in the time interval. However, the probability of j jumps occurring in the finite interval quickly converges to 0 as j tends to infinity. Therefore, we limit the maximum number of jumps to 10.

1.2 Implied volatility

We now ask ourselves: what is the volatility that we would have needed to inject in the classic Black-Scholes model in order to end up with the observed prices (i.e. the prices calculated with Merton, since we assume the jump-diffusion process to reflect the "true" dynamics)? Naturally this doesn't lead to a single constant, but rather an "implied volatility surface" $\sigma_{imp}(T, \frac{K}{S})$ with dependence on time to maturity and moneyness.

However, there is no closed-form formula that allows us to directly solve for volatility from a given option price. The Black–Scholes formula is explicit in the option price given volatility, but trying to invert it involves the cumulative normal distribution in a nonlinear way. We cannot simply rearrange terms to isolate volatility. An iterative numerical method (a root-finder) is needed: the algorithm searches for the volatility that makes the model price equal to the observed price.

To tackle this task, we define a new function:
`implied_vol(price, S0, K, T, r, q, sigma_lower, sigma_upper)`
This function inverts the Black–Scholes formula: given a call price, it finds the volatility that would reproduce that price under the BS model. It first checks whether the input price is outside arbitrage-free bounds (returning 0 or NaN accordingly). The core of the function is a root-finding routine (`brentq`), which searches for the volatility where the difference between the BS price (previously defined function) and the target price is zero. We use default values for bounds of the search.

2 Results

2.1 Outputs

After setting the values of fixed parameters and creating lists of values to iterate through for maturity and strike, we generate two DataFrames, structured as 2-dimensional tables with different maturities on rows and different strikes on columns.

The first DataFrame stores call prices (obtained by Merton's formula); the second one stores implied volatilites of latter prices. Looping through values for T and K, we progressively fill these tables with calls to the previously defined functions.

Below is a transcript of those results. Prices have been rounded to two decimals and volatilities to four decimals.

	Strike K			
Maturity T	80.0	90.0	100.0	110.0
0.02	20.06	10.07	1.18	0.00
0.08	20.26	10.36	2.45	0.13
0.25	20.83	11.52	4.55	1.18
0.50	21.83	13.22	6.72	2.83

Table 1: Call prices (.2f) using Merton's formula

	Strike K			
Maturity T	80.0	90.0	100.0	110.0
0.02	0.3140	0.2504	0.2021	0.2006
0.08	0.2215	0.2074	0.2029	0.2016
0.25	0.2062	0.2043	0.2032	0.2025
0.50	0.2045	0.2038	0.2033	0.2029

Table 2: Implied volatilites (.4f) of latter prices

We can now plot implied volatility for each maturity T, as a function of moneyness $\frac{K}{S}$. Note that the four plots currently have different y-axis scales for readability.

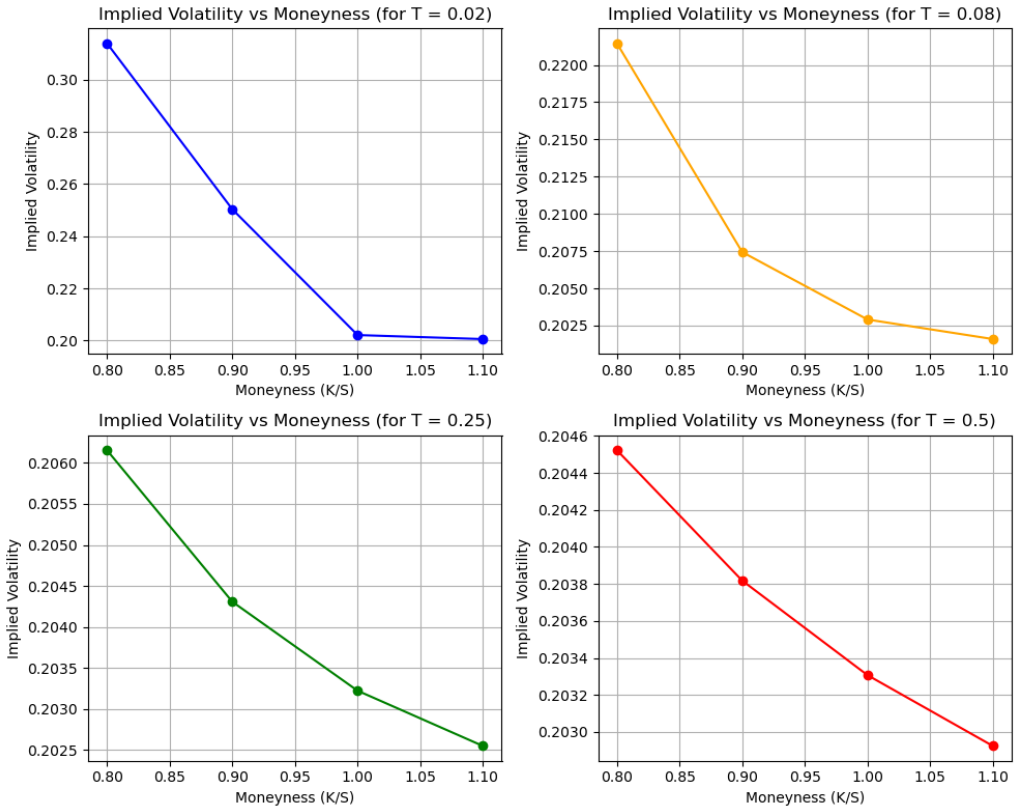

Figure 1: Implied Volatility vs Moneyness for fixed maturities

It is also helpful to display the four curves on a single plot, to better evaluate how maturity plays an important role

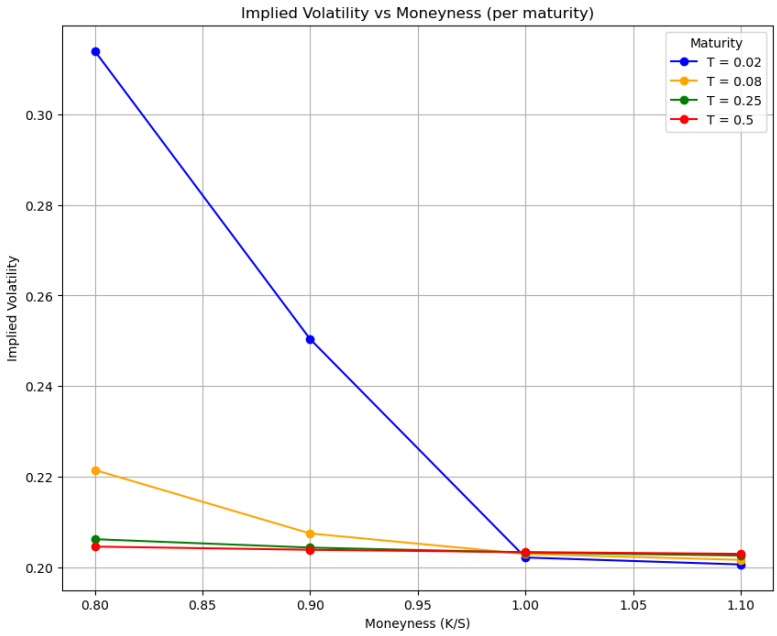

Figure 2: Implied Volatility vs Moneyness for fixed maturities - Comparison

We can even plot our result as a (discretized) 3D surface, to visually combine the dependence on both maturity and moneyness.

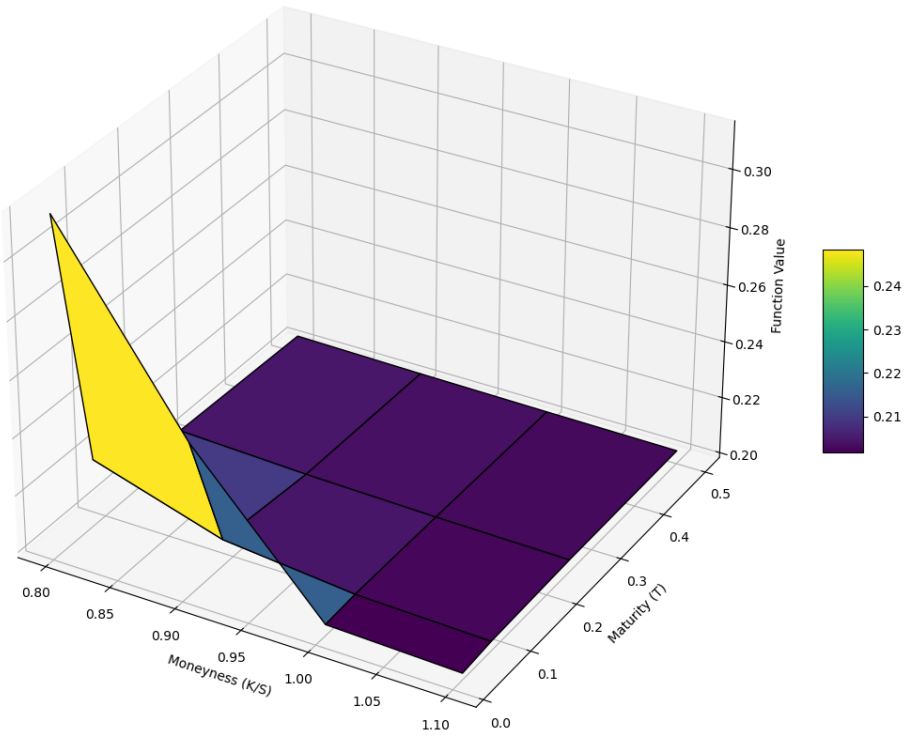

Figure 3: Volatility Surface

2.2 Discussion

We can observe that ITM calls ($\frac{K}{S} < 1$) have higher implied volatilities than OTM calls.

To explain this, recall that in the jump-diffusion model that we consider, $\gamma = -0.08$, i.e. the underlying's price is subject to downward jumps of 8%. Therefore, ITM calls are more exposed to jump risk. For OTM calls that are already unlikely to end up in the money, a downward jump just makes them even less valuable. They're less sensitive to jump risk, so implied vol is lower. An ITM call, in contrast, has more value at stake: a downward jump can wipe out part of its intrinsic value, so the risk premium built into its price is higher. Investors demand more volatility compensation, so implied vol is higher.

Short maturities magnify this effect: the blue curve ($T = 0.02$) is sharp, while the red curve ($T = 0.5$) is much flatter. This is because, over short horizons, a single jump can drastically change the option's value (the ITM call's value is more "vulnerable"), whereas over long horizons the impact of individual jumps is averaged out by diffusion dynamics.

This observation is actually coherent with real-world option markets. In practice, implied volatility surfaces are not flat (as predicted by Black–Scholes): they typically display smiles or skews, with higher implied volatilities for options that are either ITM or OTM, especially at short maturities. The most common example is OTM puts (serving as downside protection) being particularly expensive due to crash aversion. In other contexts, like ours, investors believe in a possibility of sudden upward or downward jump moves, and therefore may also pay a premium for ITM or OTM calls.

Conclusion

This concludes our assignment in which we explored how accounting for jump-diffusion in the underlying's dynamics translates into a distortion of the implied volatility surface relative to the Black–Scholes benchmark. One could argue that the Merton jump-diffusion model is therefore "more accurate" than the Black–Scholes model because it naturally generates implied volatility smiles and term structures consistent with observed data. Black–Scholes, by construction, cannot account for these effects, as it assumes a continuous diffusion process and predicts flat implied volatility across strikes and maturities.