Pricing of Lookback options using a jump diffusive asset with a non-homogenous stochastic intensity parameter

Bocconi University

Pricing of Lookback options using a jump diffusive asset with a non-homogenous stochastic intensity parameter – Application on Italian Stock Banco BPM

Stockholm School of Economics

Pricing of Lookback options using a jump diffusive asset with a non-homogenous stochastic intensity parameter – Application on Stock Index S&P 500

Double Degree Thesis, Bocconi & SSE, MSc in Finance
Master Thesis
Alessandro Michele de Renzis Sonnino
Under the supervision of Anna Battauz (Bocconi) and Tobias Sichert (SSE)

Pricing of Lookback options using a jump diffusive asset with a non-homogenous stochastic intensity parameter – Application on Italian stock Banco BPM

Bocconi University, Laurea Magistrale in Finance
Supervised by professor Anna Battauz

Ringraziamenti

Desidero esprimere la mia più sincera gratitudine a tutte le persone che, in modi diversi, hanno contribuito alla realizzazione di questo lavoro.

In primo luogo, il mio ringraziamento più profondo va alla mia famiglia. Ai miei genitori e a mio fratello, che rappresentate una parte insostituibile del mio cuore: vi voglio un bene immenso, e senza il vostro affetto e sostegno costante non sarei la persona che sono oggi.

Un sentito grazie va alla professoressa Anna Battauz, che con grande disponibilità, competenza e gentilezza ha saputo accompagnarmi in questo percorso. La sua guida è stata fondamentale non solo per lo sviluppo della tesi, ma anche per la mia crescita personale. Le sono profondamente grato per il rapporto professionale — e umano — che si è instaurato.

Ringrazio inoltre l'Università Bocconi, che è stata la mia casa per cinque anni. Mi ha offerto un ambiente stimolante e ricco di opportunità, che mi ha permesso di crescere sotto ogni punto di vista. Le esperienze vissute, seppur spesso impegnative, mi accompagneranno per sempre.

Un ringraziamento speciale va ai miei amici di sempre: Nane, Vittoria, Eugenia e Lilian. Siete una costante preziosa nella mia vita, e vi sono infinitamente riconoscente per esserci sempre stati.
Ringrazio con affetto anche i miei amici di Milano: Edoardo, Sara, Elena, Margherita, Cristiana, Maria Antonietta, Pietro, Elisa, Arianna e Ginevra. In questi anni mi avete regalato sorrisi, leggerezza e amicizia sincera.
Un grazie di cuore va anche ai miei amici di Firenze: Alessandro, Filippo, Angelica, Ione, Sibilla, Francesca e Maria Vittoria, che ad ogni ritorno mi avete fatto sentire a casa.
Infine, un pensiero speciale ai miei amici di Stoccolma: anche se ci conosciamo da meno tempo, avete saputo conquistare un posto importante nel mio cuore. Vi voglio bene.

Table of Contents

Introduction **1**
Theoretical review of key concepts **3**
Jump diffusion model **3**
Counting process and *stochastic differential* equation 3
Comparison of first and second moment with Black-Scholes asset 7
No-Arbitrage Analysis 9
Hawkes Processes **11**
Stationarity and Kernel of the non-homogenous counting process 11
Exponential and power-law kernel 12
Monte Carlo Technique **14**
General Idea and pricing of European call 14
Euler discretization 15
Statistical properties of Monte Carlo estimator 16
Numerical Application **18**
Calibration of plain vanilla European call closed-end formula to market prices **19**
L-B FGS-B and Gradient Descent algorithms 24
Pricing European Calls using Merton Model **28**
Generation of matrix of simulated jump-diffusive prices 29
Pricing Fixed Lookback Calls using Merton Model **32**
Pure Monte Carlo Estimator 32
Control Variate estimator to reduce variance 33
Extension to a non-homogenous, stochastic intensity parameter **37**
Ogata's thinning algorithm 39
Simulation in the $\mathbb{P}$ probability measure 40
Pricing of Lookback option using counting process with stochastic intensity 43
Sensitivity Analysis 47
Results **51**
Conclusions **53**
Bibliography **54**

1. Introduction

In 1963, the paper "*The pricing of options and corporate liabilities"* published by Fischer Black and Myron Samuel changed the shapes of the quantitative finance world by introducing the first closed-end formula to price European call and put options. This led to the renowned Black Scholes model which, despite many rigid assumptions it is built upon, still is an important benchmark for traders and financial engineers. One of the main assumptions of the model is for the diffusive asset $S(t)$ to be modelled as a Geometric Brownian Motion with the following Stochastic Differential Equation:

$$dS(t) = S(t)(\mu * dt + \sigma * dW(t))$$

Where $dW(t)$ is a standard Brownian motion, $S(t)$ is the risky asset, μ is the drift under the $\mathbb{P}$ probability, σ is the diffusion parameter.

Coupled with the risk free asset $B(t)$, the diffusive stock $S(t)$ creates a complete and free of arbitrage market.

Out of the many simplistic assumptions of the model, the specification of the stochastic process for $S(t)$ is the one which has been most frequently challenged by practitioners and academics. One of the problems frequently reported regards the path followed by $S(t)$: since the stock $S(t)$ is modelled in a continuous trajectory, the possibility of massive jumps, either upward or downward, which have been seen in empirical data, has been minimized.

In this regard, Robert Cox Merton published in 1976 a paper titled 'Option prices when underlying stocks' returns are discontinuous' in which he tried to tackle this limit.

In his model, still comprehensive of a risk-free asset $B(t)$ and a diffusive asset $S(t)$, the underlying's stochastic differential equation also includes an extra component to include the possibility of sudden jumps in process. These jumps are modelled as a compound Poisson process, with a constant intensity parameter λ, and they introduced potential discontinuities in the path of the stock.

This is crucial, as now we have introduced a probability of a non-local change in price which has found strong evidence in data - think of the many outliers found in the empinrical distribution of stock price compared to the log-normal kernel, or of the massive downward jumps experienced in a breakout of a bad news.

The more accurately we model the stock price dynamics in a realistic way, the more efficient and precise the pricing of given exotic options will be.

The purpose of this analysis is thus to go further in the modelling of the stock dynamics by specifying a slightly different probability distribution for the likelihood of jump occurrence. In the original Merton framework, the Poisson distribution specified for the jumps had a single, constant intensity parameter lambda. Despite being reasonable, data have shown in the last years that a clustering of jumps have formed: the occurrence of a jump makes it more likely for another jump to occur in the future, thus changing the intensity parameter. In order to introduce a self-exciting behaviour in the intensity parameter, we are going to specify the counting process of jumps as a Hawkes process. This process shows this precise property: the non-homogenous intensity parameter changes over time.

After developing the mathematical model, we are going to use it to price path-dependent options traded in over-the-counter market. The growth of the market for exotic options has been extraordinary in recent years, as they allow financial institutions to tailor the financial derivative to their needs and allow much more flexibility on the conditions and computations of the final payout. Lookback options are a popular example of path-dependent options, as the strike price is defined as an arithmetic mean of the stock price itself occurred in different points in time during the life of the option. The Pricing of such products requires a solid mathematical framework to perform payout simulations, as they do not present a closed formula yet.

In the first section of the work, we are going to review all the mathematical and theoretical tools necessary for the analysis. The original Merton framework will be discussed, highlighting the derivation of the price $S(t)$using Ito's lemma and a non-arbitrage analysis of the model. Then, the main properties of Hawkes processes will be reviewed, analyzing the first moments and the different kernels proposed by the literature for the model. Finally, a review of Monte Carlo simulations will be carried out, proposing an example of a simple algorithm to perform simulations to price European options.

2. Theoretical review of key concepts

2.1 Jump Diffusion Model

2.1.1 Counting Process and stochastic differential equation

In 1973 Merton proposed a different *stochastic differential equation* for the risky asset in order for it to include sudden, non-local changes in its price trajectory.

This new framework represented an imporant step in mathematical modeling, as it allowed the path for the stock price to be much more realistic and, thus, more effective for pricing purposes. The *SDE* proposed by Merton has the following structure.

$$dS(t) = S(t-)(\mu\, dt + \sigma dW(t) + dJ(t))$$

Where $S(t)$ indicates the stock price at time t, $S(t-)$ indicates the stock price at time $t^- = t - dt$, so the price one infinitesimal instanct before time t. $W(t)$ is a Standard Brownian motion, μ and σ are the drift and diffusion parameter, respectively, and $J(t)$ is the jump process introduced by Merton.

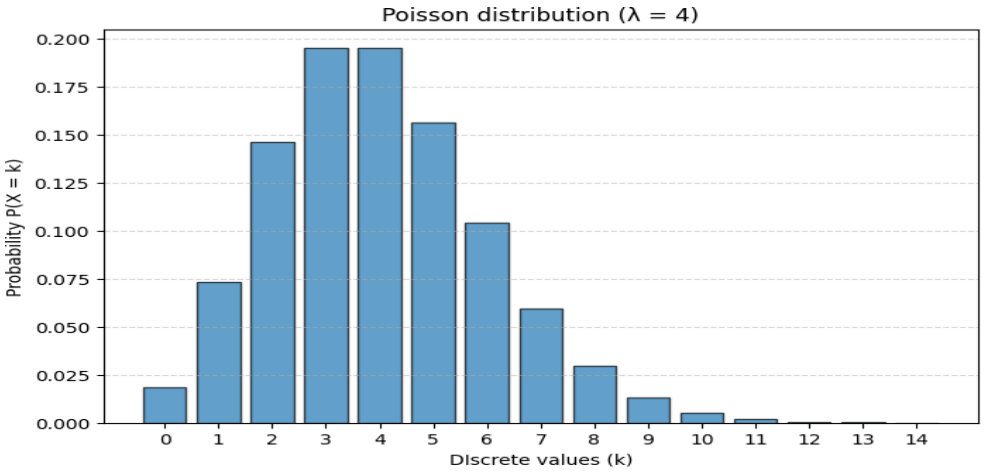
Figure 1 shows some trajectories created using the *SDE* above, to include the presence of visible discontinuities in the stock price path.

A jump process is a different type of stochastic process that anaylizes the likelihood of witnessing a random number of jumps in a given time period. A very simple case of a jump process is a Poisson process, that models the discrete number of jumps to occur in a given time interval. The problem of using a Poisson process for this purpose is that you are only modelling the number of jumps, and not the size of the latters. To overcome this problem, Merton chose to use a **Compound Poisson process** as jump process for the model. So, the process $J(t)$ can be described as:

$$J(t) = \sum_{j=1}^{N(t)} (Y_j - 1)$$

$N(t)$ is defined as a counting process, which is a vital component of the process $J(t)$ as it models the number of jumps to occur within the time interval $[0, t]$. To understand the counting process, we can look at *Figure 1*:

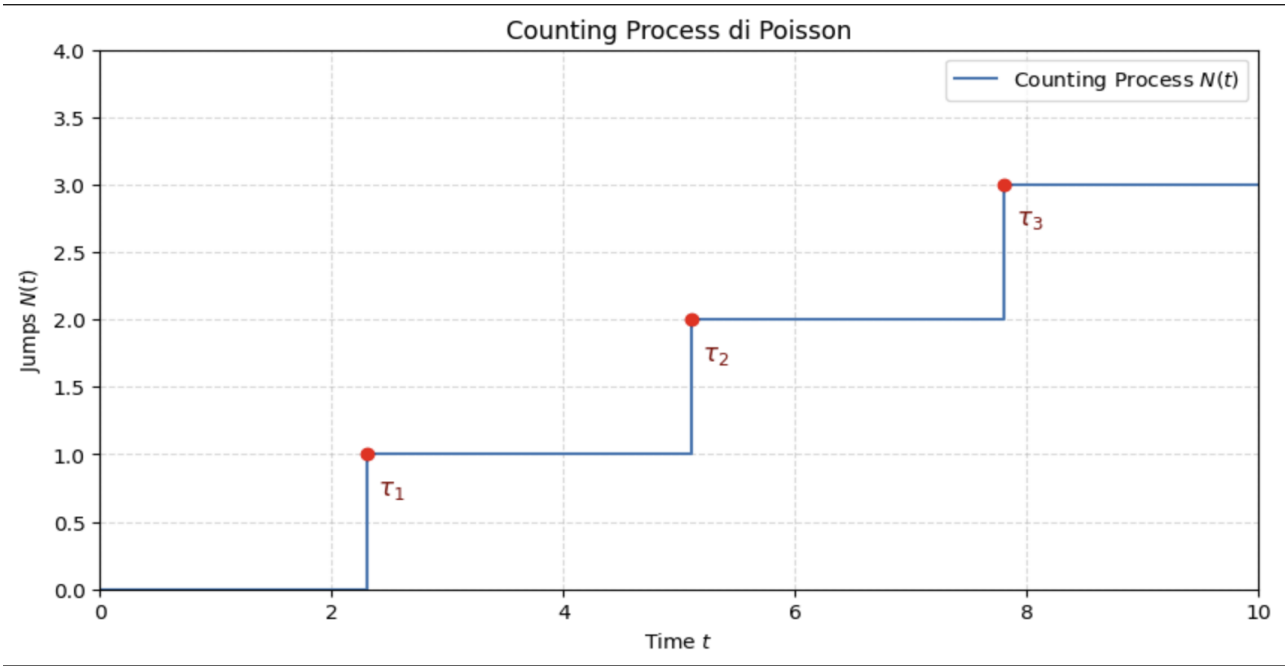
Figure 1



This distribution has been created using a Poisson distribution, and it shows the probabilities estimated for each number of discrete jumps to occur in a given time period.

Aside from the overall number of jumps in a time period, we also may be interested in assessing the times in which jumps occur τ_i over the period. A visualization in this sense may come from the following graph:

Figure 2



In *Figure 2*, we see a path for a counting process simulated over a time interval: in this simulation, we see that 3 jumps have occurred, and $\tau_1 = 2.2, \tau_2 = 4.6, \tau_3 = 7.9$.

This is the visualization of the counting process, and we can see that it is right-continuous, since the value $N(t)$ includes the jump occurred in time t. The arrival of a jump at a random time τ_i generates a discontinuity from the left in the counting process as, again, it gets included in the value for $N(\tau_i)$. Furthermore, since the jump time is random and completely independend from all the other sources of randomness of the model - $W(t)$ and Y_i – there is no chance at time $t^- = t - dt$ to unveil the occurence of a jump at time t.

To better understand how this counting process has been generated, we now dwell on the statistical properties of the underlying process used to model the occurrence of the jumps: the Poisson process.

The *probability distribution function (PDF)* for this discrete Poisson random variable is the following:

$$P[N(t) = n] = e^{-\lambda t} * \frac{(\lambda t)^n}{n!}, \forall n \in N$$

The graphical representation for this *PDF* is the one shown in *Graph 1*, where we selected an arbitrarily intensity parameter λ equal to 4. Another useful property of this distribution for the counting process is that it can also be shown that:

$$P[\tau_{j+1} - \tau_j \leq t] = 1 - e^{\lambda t}$$

Which describes the waiting times between jumps, as this is also a random variable. The key parameter in this discrete distribution is $\boldsymbol{\lambda}$. The intensity parameter under the pyhsical probability $\mathbb{P}$ measures the

probability of occurrence of a jump between the infinitesimal time interval $[t^-, t]$ – thus, it can be rewritten as $P[\Delta N(t) = 1\ in\ [t^-, t]] = \lambda$. The higher this intensity parameter, the higher the likelihood of witnessing a jump in our time interval. Since the parameter λ refers to the infinitesimal time interval dt, we need to scale up the time factor when we extend the parameter to the whole interval $[0, t]$. So, we have that the first two moments of this process are characterized as follows:

$$E[N(t)] = \lambda t$$

$$Var[N(t)] = \lambda t$$

The counting process $N(t)$ works well to model the number, or the frequencies of jumps, but in order to model the size of the random jumps Merton used a series of different variables Y_j. These random variables $Y_j, \forall j: 1 \dots N(t)$ are positive, IID random variables coming from the same specified distribution. In this framework, $W(t), N(t)$ and Y_j are assumed to be independent from each other under the physical probability measure $\mathbb{P}$. This specification makes the process $J(t)$ a Compound Poisson process, as the frequencies and number of jumps come from a Poisson distribution, but the sizes of jumps do not. In our specific case, the size of the j_{th} jump is $(Y_j - 1)$.

For instance, let us consider the case where $N(T) = 3$ – so we would have 3 jumps. The compound Poisson process would thus be $J(T) = (Y_1 - 1) + (Y_2 - 1) + (Y_3 - 1)$, where T is the maturity time.

After having analyzed the extra component of Merton's *stochastic differential equation*, we now focus on the price path for $S(t)$ in this new framework.

Focusing on the path price for the stock $S(t)$ means solving the *SDE* proposed for the asset, i.e. it means to get the right expression for the stock price S at each time t assuming the infinitesimal differential $dS(t)$ follows that equation indicated above. To do that, we use Ito' lemma to derive the solution for $S(t)$ from the indirect solution for $\ln(S(t))$. **Ito's lemma** is a powerful tool in stochastic calculus that, given a target process $Y = f(X)$, and a know process X, allows us to derive the *differential equation* for $dY(t)$, so for a linear function of the original know process $dX(t)$.

(Ito's lemma). Given f: R x R → R continuously differentiable with respect to the first argument and twice continuously differentiable with respect to the second argument. Then,

$$df(t, S(t)) = \frac{df(t, S(t^-))}{dt} + \frac{df(t, S(t^-))}{dS} S(t^-)(\mu dt + \sigma dW(t)) + \frac{1}{2}\frac{d^2 f(t, S(t^-))}{dS^2} S(t^-)^2 \sigma^2 dt + (f(t, S(t)) - f(t, S(t^-))) * \Delta N(t)$$

We know use this lemma to derive the process for the log price $\ln(S(t))$ – so, our $f = \ln(x)$, and, once we find the solutions for the log price, we exponentiate to get the solutions for the price $S(t)$.

Using the expression above, we can write:

$$dln(S(t)) = (\mu - \frac{\sigma^2}{2}) dt + \sigma dW(t) + (\ln(S(t)) - \ln(S(t^-))) * \Delta N(t)$$

The last term is the most interesting to be dealt with, since it heavily depends on whether, at a given time t, a jump occurs or not. If it does occur, and so $\Delta N(t) = 1$, the term becomes $\ln(S(t^-) * Y_j) - \ln(S(t^-))$, which in

term can be written as $\ln\left(\frac{S(t^-)*Y_j}{S(t^-)}\right)$. If a jump does not occur, and so $\Delta N(t) = 0$, the whole term will be equal to 0. Simply speaking, the only relevant cases where the last term is going to impact on the solutions of $S(t)$ are when a jump is going to occur. So, we can rewrite our expression as:

$$dln\big(S(t)\big) = \left(\mu - \frac{\sigma^2}{2}\right)dt + \sigma dW(t) + \big(\ln(\mathrm{Y_j})\big) * \Delta N(t)$$

Taking the integral on both sides, to sum all the infinitesimal changes, we get:

$$\int_0^T dln\big(S(t)\big) = \int_0^T \left(\mu - \frac{1}{2}\sigma^2\right)dt + \int_0^T \sigma dW(t) + \int_0^T \ln\big(Y_j\big) * \Delta N(t)$$

$$\ln\left(\frac{S(T)}{S(0)}\right) = \left(\mu - \frac{1}{2}\sigma^2\right)T + \sigma dW(T) + \sum_{j=0}^{N(T)} \ln(Y_j)$$

The last integral becomes a sum operator because, again, we are implicitly considering only the cases in which $\Delta N(t) = 1$, so when a jump occurs. Thus, the integral, an infinitesimal summation, becomes a discrete summation of all the $\ln\big(Y_j\big), \forall j: 1 \dots N(T)$. The rest of the equation evolves just as in the regular Black Scholes diffusive stock. The final solution for $S(T)$ is thus:

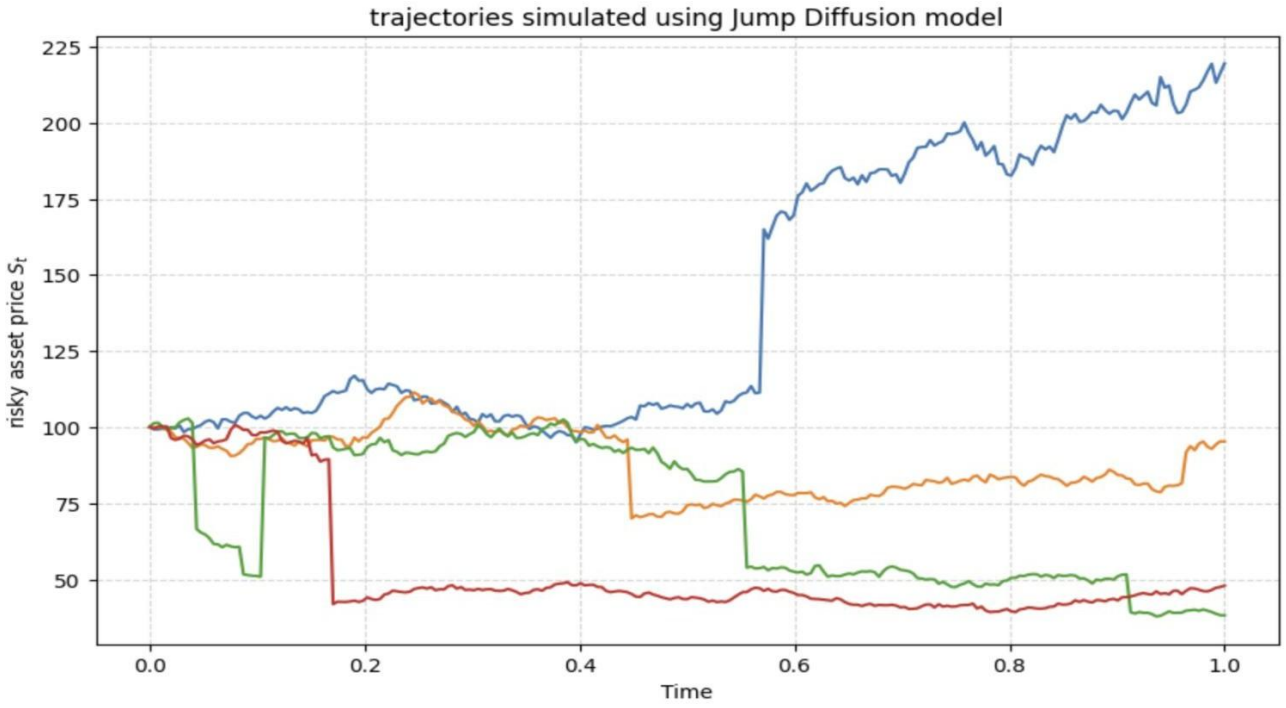
$$S(T) = S(0) * e^{\left(\mu - \frac{1}{2}\sigma^2\right)*T + \sigma W(T)} * e^{\sum_{j=0}^{N(T)} \ln(Y_j)}$$

And, because of the property of the exponential $e^{a+b} = e^a * e^b$, and because $e^{\ln(Y_j)} = Y_j$,

$$\boldsymbol{S(T) = S(0) * e^{\left(\mu - \frac{1}{2}\sigma^2\right)*T + \sigma W(T)} \left(\prod_{j=0}^{N(T)} Y_j\right)}$$

For each future time t, we can find the expression of the jump-diffusive asset $S(t)$ using this expression. We see that the main difference between the "classic" solution within the Black and Scholes framework is the product term, which sums up the impact of the – random – jumps. The two different sources of randomness (the "continuous" one, $W(t)$, and the "discontinuous" one, Y_j) are completely independent from one another – otherwise we could try to predict the occurrence of the jumps, which is the opposite of the interpretation of *random jumps*. An example of trajectories for $S(t)$ simulated using this framework is the shown in *Figure 3*.

Figure 3



Assuming an initial price of $S(0) = 100$ and T = 1, we can clearly see the sudden jumps in each simulated trajectory for the asset price.

2.1.2 Comparison of First and Second moment with the Black-Scholes asset

Now that we have defined the equation for the price of the jump diffusive asset at any time t, it is time to dwell more on the first two moments for this variable: namely, the **expected value** and the **variance** of $S(t)$ under the physical probability measure $\mathbb{P}$. The computations for these measures are strongly simplified by the notion that the three main sources of randomness - $W(t)$, $N(t)$ and Y_j – are independent. In fact, using the expectations operator we can see that:

$$E^{\mathbb{P}}[S(t)] = E^{\mathbb{P}}[S(0)e^{\mu t - \frac{1}{2}\sigma^2 t + \sigma W(t)} * \prod_{j=0}^{N(T)} Y_j \ ]$$

Because of $\sigma W(t) \sim N(0, \sigma^2 t)$, and $E^{\mathbb{P}}[e^{\sigma W(t)}] = e^{\frac{1}{2}\sigma^2}$ - given by the expected value of a log-normal variable, we can rewrite the expectation and simplify to reach:

$$E^{\mathbb{P}}[S(t)] = S(0)e^{\mu t}E^{\mathbb{P}}[\prod_{j=0}^{N(T)} Y_j \ ]$$

The first part of this equation is precisely the case of the asset following a Geometric Brownian motion, as in the Black Scholes model; the presence of the jumps has an impact on the expected value of the risky asset through the sum of the random jumps. So, the term to look at now is the $E^{\mathbb{P}}[\prod_{j=0}^{N(T)} Y_j \ ]$.

We can rewrite this using the well-known Law of Total Probability

$$E^{\mathbb{P}}[\prod_{j=0}^{N(T)} Y_j\ ] = \sum_{n\in N} E^{\mathbb{P}}\left(\prod_{j=0}^{N(T)=n} Y_j \ \middle| N(T)=n\right) * \mathbb{P}[N(T)=n]$$

Exploiting the fact that Y_j and $N(T)$ are independent, and knowing that $N(T)$ is distributed according to a homogenous Poisson distribution – whose probabilities for each value of n are known, and that the sum of expectations is equal to the expectation of the sum - $\sum E[X] = E[\sum X]$ - we can rewrite this as:

$$E^{\mathbb{P}}[\prod_{j=0}^{N(T)} Y_j\ ] = \sum_{n\in N} (\mu_Y)^n * \frac{(\lambda t)^n}{n!} * e^{-\lambda t}$$

And this expression results in:

$$E^{\mathbb{P}}[\prod_{j=0}^{N(T)} Y_j\ ] = e^{\lambda t(\mu_Y - 1)}$$

Finally, the first moment of the jump-diffusive asset can be expressed as:

$$E^{\mathbb{P}}[S(t)] = S(0)e^{\mu t}e^{\lambda t(\mu_Y-1)}$$

So, the jump-diffusive asset has a slightly different mean compared to the traditional Black-Scholes asset, and this difference is determined by the instantaneous intensity of the Poisson process governing the number of jumps λ and by the mean of the specified variable governing the size of the single jump μ_Y. As λ is a non-negative parameter, it all depends on μ_Y: if $\mu_Y > 0$, jump diffusive asset will be on average higher than BS asset; if $\mu_Y < 0$, the jump diffusive asset will be on average lower than BS asset. In fact, the implied drift rate for a BS asset to have the same expected value as a jump-diffusive asset for a time t would be
$\mu^* = \mu + \lambda(\mu_Y - 1)$.

Intuitively, the jump-diffusive asset has a higher variance than the standard BS asset. As the jumps are unforecastable, and massive in their impact, it is understandable that the asset subject to these jumps will have an equal or higher variance than the standard, jumps-free asset. To show it more rigorously, we know that the variance of the asset is equal to:

$$Var^{\mathbb{P}}[S(t)] = E^{\mathbb{P}}[S^2(t)] - E^{\mathbb{P}}[S(t)]^2$$

Considering the case of a BS asset with the specified drift rate equal to μ^*, in such a way that $E^{\mathbb{P}}[S^*(t)] = E^{\mathbb{P}}[S(t)]$, where $S^*(t)$ is the BS asset, we can see that claiming that $Var^{\mathbb{P}}[S(t)] > Var^{\mathbb{P}}[S^*(t)]$ is the same as claiming that $E^{\mathbb{P}}[S^2(t)] > E^{\mathbb{P}}[S^{*2}(t)]$, since the first moments of the two assets are matched by construction – we chose the drift rate μ^* for the BS asset.
Since the expression for the second moment can be reduced to:

$$E^{\mathbb{P}}[S^2(t)] = S^2(0)e^{(2\mu+\sigma^2)t} * e^{\lambda t\left(E^{\mathbb{P}}\left[Y_j^2\right]-1\right)}$$

We can solve that the expression above where we compare the two second moments can be simplified into the following inequality – by working out the algebra of the two second moments: one for the jump-diffusive, one for the BS asset.

$$E^{\mathbb{P}}\left[\left(Y_j - 1\right)^2\right] > 0$$

The expression above is always true, except for one interesting exception: when $Y_j = 1$.

If we have that $Y_j = 1$, the jump has no impact on the risky asset, as $S(\tau_j) = S(\tau_j^-)Y_j$. Since Y_j is a random variable by itself, we must talk in probability terms: if and only if $\mathbb{P}[Y_j \neq 1] > 0$, then the risk-diffusive asset will have a greater variance than the BS one. If $\mathbb{P}[Y_j = 1] = 1, \forall j: 1 \dots N(t)$, then the variance of the jump-diffusive and of the BS one will be equal, as the jump will leave the jump-diffusive asset unaffected.

2.1.3. No-arbitrage analysis

In this section, we look at the market, within the continuous time framework, generated by the jump-diffusive asset $S(t)$ and $B(t)$. Referring to the first and second fundamental theorems of asset pricing, we will show that this market is free of arbitrages but incomplete.
(First Fundamental Theorem of Asset Pricing): A financial market with finite time horizon T and price processes of one risky asset and one risk-less asset is free of any arbitrage opportunities if and only if we can find an ***Equivalent Martingale Measure (EMM)*** *such that:*

- *The EMM, also denoted as $\mathbb{Q}$ to claim it is a risk-neutral measure, is equivalent to $\mathbb{P}$.*
- *The discounted price process of the risky asset is, under this risk-neutral probability, a martingale.*

(Second Fundamental Theorem of Asset Pricing): A financial market with finite time horizon T and price processes of one risky asset and one risk-less asset is complete, meaning that any contingent claim X(T) can be replicated with a trading strategy of the two securities if and only if we can find a unique Equivalent Martingale Measure for the specified market

Hence, in order to carry out our *No-Arbitrage analysis,* we must look for a way to get an equivalent martingale measure $\mathbb{Q}$ under which the price process of the discounted jump-diffusive asset is a martingale and verify whether we can find an infinite number of measures of probabilities that allow this or only one. In the former case, the market would be arbitrage-free but incomplete; in the latter, it would be both arbitrage-free and complete.
A martingale is defined as a stochastic process whose conditional expected value of the infinitesimal change is equal to 0, i.e. a process that stays, on average, constant over time. More rigorously, we can define a martingale $M(t)$ under the risk neutral probability $\mathbb{Q}$, where $\mathcal{F}_t$ indicates the set of information contained at time t*:*

$$E^{\mathbb{Q}}\left(dM(t)|\mathcal{F}_t\right) = 0$$

We are interested in modelling the discounted price process of the risky asset $\tilde{S}(t) = e^{rt}S(t)$. Using the Ito's lemma presented in the section above, taking as our target function $f(t,x) = e^{-rt}x$, we can easily derive the stochastic differential equation for the discounted price, and this is equal to:

$$d\tilde{S}(t) = (\mu - r)dt + \sigma dW^{\mathbb{P}}(t) + \Delta J(t)$$

On this occasion, we write $W^{\mathbb{P}}(t)$ to mark that this is a standard Brownian motion under the physical probability $\mathbb{P}$, and so it is not the one suitable with the no-arbitrage analysis we are trying to carry out, as we need a Brownian motion under the new, risk-neutral, probability measure $\mathbb{Q}$. The same logic is applied to the counting process $N(t)$. In fact, also this process is a Poisson process under the physical probability, and we need to find the risk-neutral counterparty for both $N^{\mathbb{P}}(t)$ and $W^{\mathbb{P}}(t)$. To "shift" the probability measure we use another fundamental theorem, called the **Girsanov's Theorem**:

(Girsanov's Theorem): Given parameters θ and $\varphi > 0$, there exists a probability measure $\mathbb{Q}$ equivalent to $\mathbb{P}$ ($\mathbb{Q}\sim\mathbb{P}$) such that under $\mathbb{Q}$:

- $W^{Q}(t) = W^{\mathbb{P}}(t) + \theta dt$ *is a Standard Brownian Motion*
- $N^{\mathbb{Q}}(t)$ *is a Poisson process with intensity* $\lambda^{\mathbb{Q}} = \varphi\lambda^{\mathbb{P}}$

With all processes being independent under the $\mathbb{Q}$ probability measure.

Thus, Girsanov's theorem is the tool that we use to find the risk neutral "counterparts" for both the Brownian Motion and the Poisson process that are suitable for our analysis. The search for an Equivalent Martingale Measure $\mathbb{Q}$ is basically reduced to the search of the parameters θ and $\varphi > 0$ that would allow us to specify and define the two processes under the $\mathbb{Q}$ measure. If we can find such parameters, we can apply Girsanov's theorem and verify the existence of a probability measure under which $\tilde{S}(t)$ is indeed a martingale.
By construction, we will find the two parameters that guarantee us that the last condition is verified. By imposing the martingale requirement to $\tilde{S}(t)$, we will see if and how many parameters exist that automatically satisfy the first fundamental theorem of asset pricing. Our goal would be thus to find:

$$\theta \; and \; \varphi > 0 \; s.t. E^{\mathbb{Q}}\left((\mu - r)dt + \sigma\big(dW^{\mathbb{Q}}(t) - \theta dt\big) + \Delta J(t)\middle|\mathcal{F}_t\right) = 0$$

Given that $E^{\mathbb{Q}}(\Delta J(t)|\mathcal{F}_t) = (\mu_Y - 1)\lambda^{\mathbb{Q}}dt$, and that $E_t^{\mathbb{Q}}\left[dW^{\mathbb{Q}}(t)\right] = 0$, we can rewrite the expression above as:

$$(\mu - r)dt + \sigma\,(-\theta dt) + (\mu_Y - 1)\lambda * \varphi dt = 0$$

To solve this equation means to find the two parameters under which the discounted price of the jump-diffusive asset is a martingale. This is in turn equal to:

$$\mu - r - \sigma\theta + ((\mu_Y - 1)\lambda * \varphi = 0$$

By isolating the parameter θ on the left-hand side of the equation we can find that:

$$\boldsymbol{\theta} = \frac{\mu + (\mu_Y - 1)\boldsymbol{\varphi}\lambda - r}{\sigma}$$

Thus θ being a function of φ. This means that for any possible parameter chosen $\varphi > 0$ we can find a unique parameter θ, and, by using those two parameters, $\tilde{S}(t)$ would be a martingale by construction. We have infinite possible vectors of parameters (θ, φ) that make our discounted price a martingale, so we have infinite EMM, and the second fundamental theorem does not apply: Market is free of arbitrage but incomplete.

It is important to stress that the pair of parameters chosen for our EMM will then be the parameters used to get the risk-neutral counterparties for $W^{\mathbb{Q}}(t)$ and $N^{\mathbb{Q}}(t)$, according to the formulas provided within the Girsanov's Theorem.

2.2. Hawkes Processes

The novelty of this paper lies in the usage of a different counting process compared to the classical case of a Poisson distribution for $N(t)$.

2.2.1. Stationarity and Kernel of the non-homogenous counting process

The process, as we have already highlighted in the previous section, shows a constant, homogenous intensity parameter λ. This parameter is invariant in time and is not impacted by the occurrence of a jump. This represents the main difference with the other counting process we are going to describe now: The Hawkes Process.
Initially used to model the occurrence of seismic events, a Hawkes process is a counting process which can accurately model the occurrence of jumps for the jump-diffusive asset. The main feature of these processes is that the intensity parameter is not constant but, on the contrary, is a function of time $\lambda(t)$. This is not to be seen as a *deterministic* function of time, since the parameter is driven by a *stochastic* nature. Moreover, this parameter presents a self-exciting nature: the intensity parameter is positively impacted by the occurrence of past jumps by a function named *kernel* according to the expression below:

$$\lambda(t) = \lambda(0) + \sum_{m=0}^{M} \varphi(t - t_m)$$

Where M is the total number of events occurred before time t, and t_m indicates the time of each event. The formula expressed above is the specific case of a univariate Hawkes process – the one that will be of interest for us, as we are going to model only one asset, but a more general formula can be found for a Multivariate Hawkes process, where $\boldsymbol{N(t)} = \{N^i(t)\}_{i=1}^{D}$, to jointly model D different counting processes.

$$\boldsymbol{\lambda}(t) = \boldsymbol{\mu} + \boldsymbol{\Phi} * d\boldsymbol{N}(t)^1$$

Where $\boldsymbol{\Phi(t)} = \{\varphi^{ij}(t)\}_{i,j=1}^{D}$ is a matrix-valued kernel that requires three conditions to be met:

- $\varphi^{ij}(t) \geq 0, \forall 1 \leq i, j \leq D$; meaning that all the kernels, for each process included in the multivariate distribution, must be non-negative.
- If $t < 0, \varphi^{ij}(t) = 0, \forall 1 \leq i, j \leq D$.
- Each component $\varphi^{ij}(t)$ belongs to the space of L^1-integrable functions.

A more direct way of visualizing the multivariate process is using the following equation:

$$\lambda_i(t) = \lambda(0)_i + \sum_{j=1}^{D} \sum_{k:t_{j,k}<t} \varphi_{ij}(t - t_{j,k}), \forall i: 1, \dots, D^2$$

The notion of kernel introduced is particularly important as it represents the implicit "memory" of the counting process itself; if a jump occurs, the intensity parameter will increase making it more likely to witness other jumps in the immediate future. This increase is due to the kernel, as it cannot be negative, but its effects tend to decay over time. The amplitude and the subsequent speed of decay depend on the specification of the kernel; in other words, it depends on the functional form we set for the kernel $\varphi(t)$.

[1] Hawkes Processes in finance, Bacry et al. (2015)

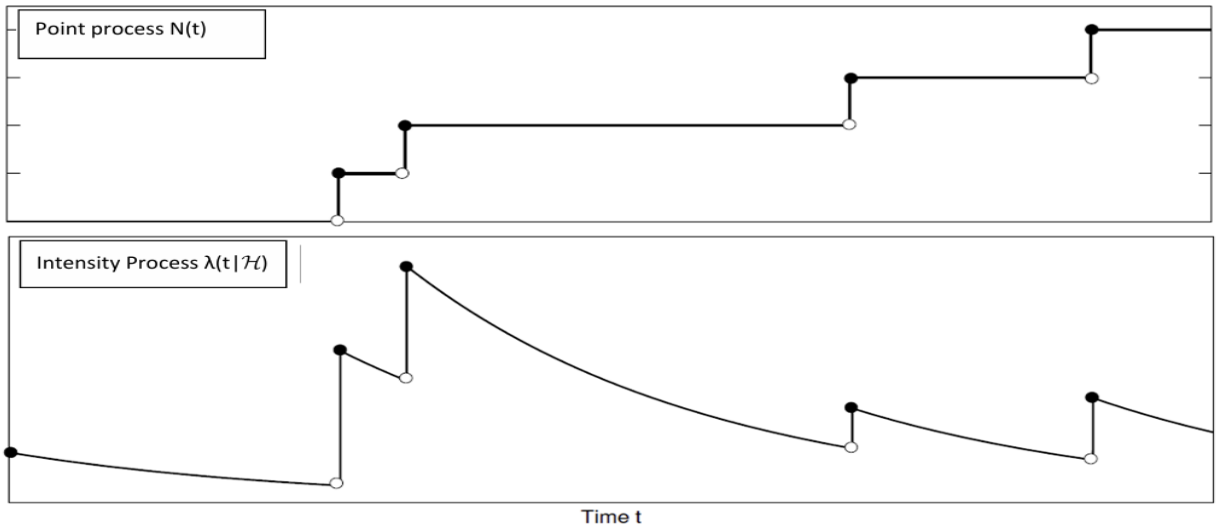
[2] tick.hawkes — tick 0.6.0 documentation

The main assumption we will impose and will rely on is the stationarity of the counting process. Stationarity is ensured by looking at one important measure in linear algebra: the spectral radius of a matrix. The Spectral radius is the biggest eigenvalue for a given matrix *A*. In our case, our matrix $\boldsymbol{\Phi(t)}$ is a matrix of functions $\{\varphi^{ij}(t)\}_{i,j=1}^{D}$. So, we indicate the L^1-norm of a function as $||\varphi|| = \int |\varphi(t)|dt$, and we will write the matrix containing the norms of each function as $||\boldsymbol{\Phi(t)}|| = \{||\varphi^{ij}||\}_{i,j=1}^{D}$. The spectral radius will thus be the biggest eigenvalue of this matrix of norms.

(Stationarity) The process N(t) has asymptotically stationary increments and $\lambda(t)$ is asymptotically stationary if the matrix $||\boldsymbol{\Phi(t)}||$, where $\boldsymbol{\Phi(t)}$ is the Kernel, has spectral radius smaller than 1 - Stability Condition.

This property insures that, cumulatively, there is not "too much memory" allowing the process to still be stationary and not containing a unit root.

Figure 4[3]



In *Figure 4* we can see an example of a simulated Hawkes Process. The upper graph shows the counting process itself, whereas the lower graph shows the modelling for the stochastic intensity parameter $\lambda(t)$. We clearly see that whenever jumps occur - $\Delta N(t) = 1$, a jump in the stochastic intensity occurs as well, triggered by the kernel of the expression. Slowly, the positive effect on the intensity parameter decays over time, and it jumps back up as soon as another jump is witnessed. It is worth noting that in this example the exogenous parameter λ_0 is set to be a deterministic function of time, whereas in our case we simply have taken a constant starting value.

2.2.2. Exponential and Power-Law kernel

The two main functional forms for the kernel $\varphi(t)$ are exponential kernel and power-law kernel**.** The first specification consists of using an exponential function to model our kernel, and, thanks to the time-decaying property of the exponential distribution, it is among the most common within the finance and mathematics world. The "reaction" function of this kernel – in the univariate case - is the following[4]:

$$\varphi(t) = \alpha e^{-\beta t}$$

[3] Simulation, Estimation and Application of Hawkes Processes, Kataryna Obral, June 2016, pag. 15

[4] A Brief Introduction to Hawkes Processes | by Hey Amit | We Talk Data | Medium

The first parameter α in the function measures the initial intensity of the jump, the overall strength of the interaction, whereas the second parameter β measures the decaying effect over time of past events. The following graphs show this functional relationship in function of time with different parameters:

Figure 5

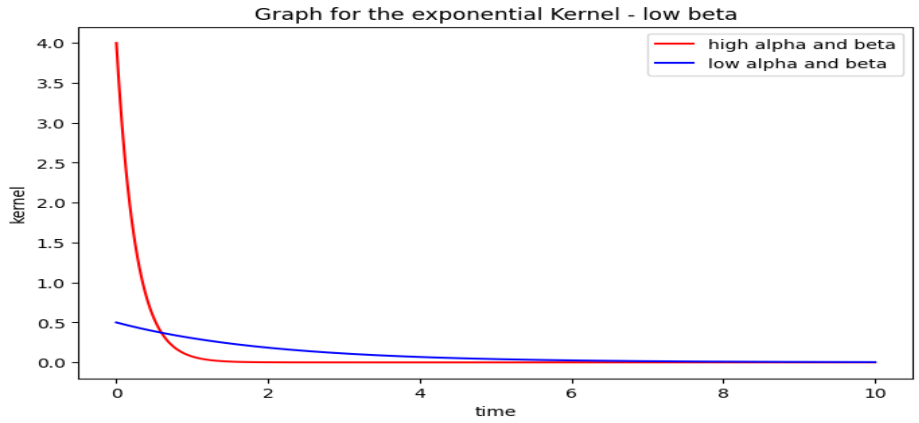


Figure 5 shows two different kernel functions. The red one presents high parameters and has, thus, a very strong but ephemeral impact: it does not have a long-lasting memory, indicating that, in the occurrence of a jump at time 0, the subsequent increase in the intensity parameter will not last long. The blue one presents low parameters, in both alpha and beta. The impact is visibly limited in size, but it is much more long-lasting compared to the previous case – driven by the parameter β.

The exponential specification is very much used also for another interesting and useful property: under this specification, the Hawkes process is in fact *Markovian* and thus useful for simulation and estimation. In fact, the intensity $\lambda(t)$ can be further recast in Markovian form as a *stochastic differential equation:*

$$d\lambda(t) = \beta(\lambda_0 - \lambda(t))dt + \alpha dN(t)^5$$

This representation is very useful as it shows that the infinitesimal difference in time t is not relying on the whole history of the process, which is indeed the Markovian property. The first component of the *SDE* $-\beta\lambda(t)dt$ refers to the time decaying feature of the process, whereas the second one $\alpha dN(t)$ is related to the sudden jump in the intensity parameter in the occurrence of a jump ($\Delta N(t) = 1$). This equation for the stochastic intensity parameter is only viable under the exponential kernel.
The second specification often used to model this process is the power-law kernel. In this specification, the kernel has the following form:

$$\varphi(t) = \frac{\alpha\beta}{(1+\beta t)^{1+\gamma}}$$

This specification allows for the intensity parameter to have more memory since each jump that occurred has a more long-lasting effect on the stochastic parameter compared to the exponential kernel. In this case, the intensity is not a Markovian variable and the intensity parameter $\lambda(t)$ depends on the whole history of $N(t)$, and not just the last realization.

[5] Hawkes Processes in finance, Bacry et al. (2015)

2.2 Monte Carlo Technique

2.3.1. General Idea and Pricing of European Call

Monte Carlo is a powerful probabilistic technique to carry out **simulations** that we will exploit to carry out the pricing of lookback options. According to the traditional mathematical finance definition, the price of an option can be written as a conditional expectation of the future payoff and, thus, as an integral.
For instance, the price of a call option at time 0 can be written as:

$$C(0) = E_0^{\mathbb{Q}}[e^{-rT} * max\{S_T - K; 0\}]$$

Because the only random variable contained in the expression is S_T, and because this is a continuous random variable, we know that the first moment of a random variable can be expressed as an integral:

$$E[g(X)] = \int_{-\infty}^{\infty} g(x) f_X(x) dx$$

The price of the call can, then, be expressed as the following integral:

$$C(0) = e^{-rT} \int_K^{\infty} (s - K) * f_{S(T)}(s) ds$$

In other words, the price of the call is the infinitesimal summation of the future payoff of the call - $(s - K)$ – weighted by the probability of having that value $S(T) = s$ - $f_{S(T)}(s)$ is the probability distribution function of the $S(T)$ – for each infinitesimal difference in the value of the underlying asset - ds. Taking the integral from the strike price K to infinity implied that we are only considering the cases where the call is in the money, rightfully.

Monte Carlo technique is a simple way to estimate this unknown quantity by *discretizing* the integral shown above. Instead of taking *infinitesimal* differences in *s,* we **simulate** *N* different values for $S(T)$. From this simulated prices of the underlying, we compute the implied payoff for the European call – which will be positive only if the call is in the money, is $S(T) > K$. Finally, we compute the price of the call for that specific simulation by discounting the payoff back to time 0, and we carry the same analysis N times. The Monte Carlo estimate for the price of the call would be the average of all these N simulated prices. Analytically, the MC price can be represented in the following expression:

$$\widehat{C(0)} = \frac{1}{N} * \sum_{i=1}^{N} e^{-rT} (S_i(T) - K)^+$$

Where $S_i(T)$ is the simulated stock price at time T in the i-th simulation.
The next important point would lie in how to accurately simulate $S(T)$. In order to do, we need to have specified a mathematical framework to model the stochastic dynamic of the underlying within $[0, T]$. In the most common case, a Geometric Brownian motion for the asset will be assumed.
As we only need to model the final value of the underlying – because it is a plain-vanilla, European call, we will

exploit the notion that:

$$S(T) = S_0 e^{\left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma W(t)}$$

Since $W(t) \sim N(0,t)$, we can generalize and say that $W(t) \sim t^{0.5}Z,\ Z \sim N(0,1)$. Thus, it is easy to simulate different value for $S(T)$; we simply must simulate different values for the standard normal and then plug the simulated values into the expression below:

$$S_i(T) = S_0 e^{\left(\left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma t^{0.5} * \boldsymbol{Z_i}\right)}$$

It is much easier to simulate values for a normal standard. Simulate a value from a *Uniform* distribution $U[0:1]$, and use the Inverse cumulative function for the standard normal $F^{-1}(u)$ to get the relative quantile for that simulated level of probability. This is the so-called *Inverse Function Method,* and it is feasible because the *CDF* $F(x)$ of the standard Normal is invertible – it does not have constant traits on some interval.

Figure 6

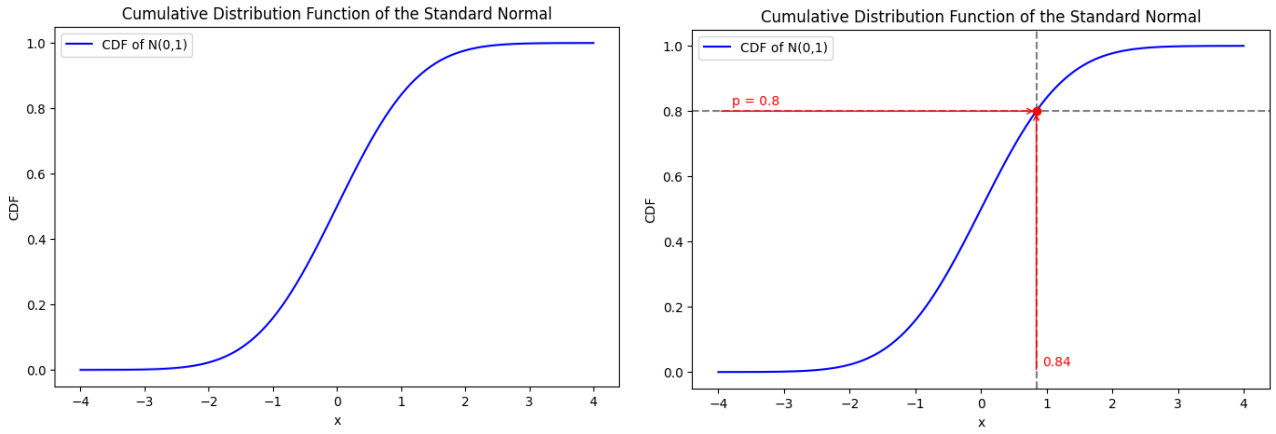


Figure 6 shows the Inverse Function Method to simulate values for $N(0,1)$ from the *Cumulative distribution function* of the variable.

2.3.2. Euler Discretization

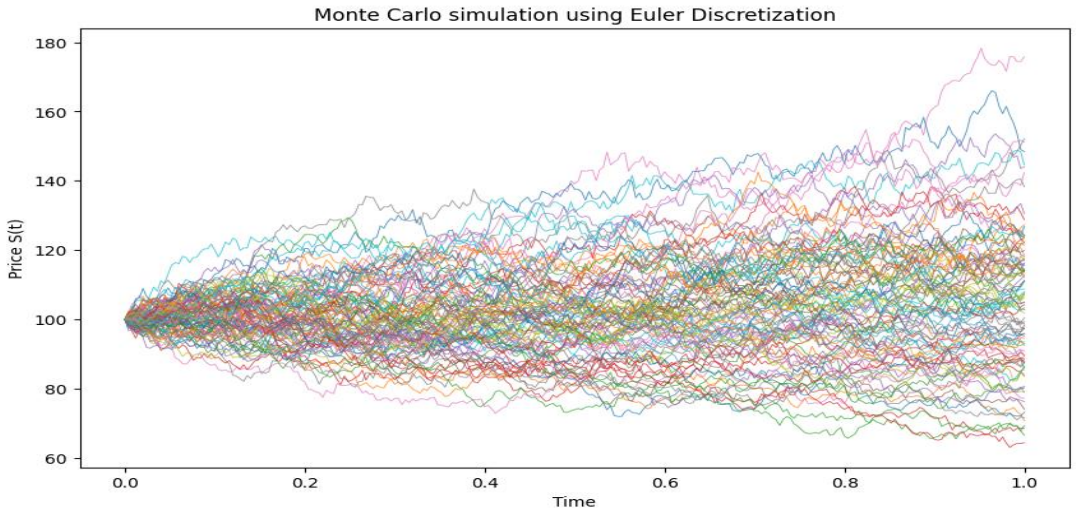
In this "simple" exercise, we are only interested in simulating the prices of the asset at maturity of the call option $S(T)$. Because this is plain vanilla, the payoff only depends on this variable. In the later section we would be dealing with the pricing of lookback options, which is an exotic product whose payoff depends on the *whole path* of the stock, and not just the price at one point in time. We will thus use the well-known **Euler discretization** to simulate the whole path:

$$S(t + \Delta t) = S(t) e^{\left(\mu - \frac{1}{2}\sigma^2\right)\Delta t + \sigma(\Delta t)^{0.5} Z_{i6}}$$

In this exercise we would need to specify both the number of simulations N and the number of time steps for each simulation M. Visually, we would thus end up with a matrix of simulated values.

[6] Numerical Methods II, prof. Mike Giles, Oxford University Mathematical Institute

Figure 7



In *Figure 7* we can see an application of the Euler discretization to carry out 100 simulations of the stock price, including 250 time-steps between today and a year from today.
In the pricing of the lookback call, we will use these simulated paths to compute the maximum and the minimum of the asset's price within the life of the option, which will be crucial in computing the payoff of the fixed / floating lookback option. The payoff of the floating lookback call is:

$$S(T) - \min_{0 \le t \le T} S(t)$$

Whereas the payoff of the fixed lookback call is:

$$\max(\max_{0 \le t \le T} S(t) - K, 0)$$

2.3.3. Statistical Properties of Monte Carlo estimator

Monte Carlo produces an **Unbiased** and **Consistent** estimator of the true variable we are trying to estimate by discretization and simulation. Unbiased means that the estimator is, on average, equal to the true parameter:

$$E\left[\widehat{C(0)}\right] = C(0)$$

Consistency is another statistical property relying on the strong law of large numbers. In fact, it basically claims that the estimator converges to the true value as the sample size increases to infinity:

$$\widehat{C(0)} \to_{n \to \infty} C(0)$$

These two properties justify the usage of the Monte Carlo estimator, guaranteeing that the prices obtained have statistical meaning and are thus reliable and useable.

Another useful property of the Monte Carlo estimator is that, thanks to the existence of the Central limit theorem, we can build an interval of confidence around the point estimate of the price of the option – we can use the quantiles of

a standard normal or of a T student. The formula for the interval is the usual:

$$[\widehat{C(0)} \pm \frac{\sigma_f}{n^{0.5}} * z_{\frac{\delta}{2}}]$$

Where the first term of the radius $\frac{\sigma_f}{n^{0.5}}$ is the standard error of our estimator, and the second $z_{\frac{\delta}{2}}$ is the quantile of the standard normal for the desired significance level δ. σ_{f_f} is the standard deviation of the estimator, which is naturally unknown and must be thus replaced with the sample standard deviation $\widehat{\sigma_n}$.
This measures the volatility of the different simulated call prices among the *vector* of simulated call prices by the Monte carlo estimator, and can be thus computed as:

$$\widehat{\sigma_n} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(C_i - \widehat{C_0}\right)^2}$$

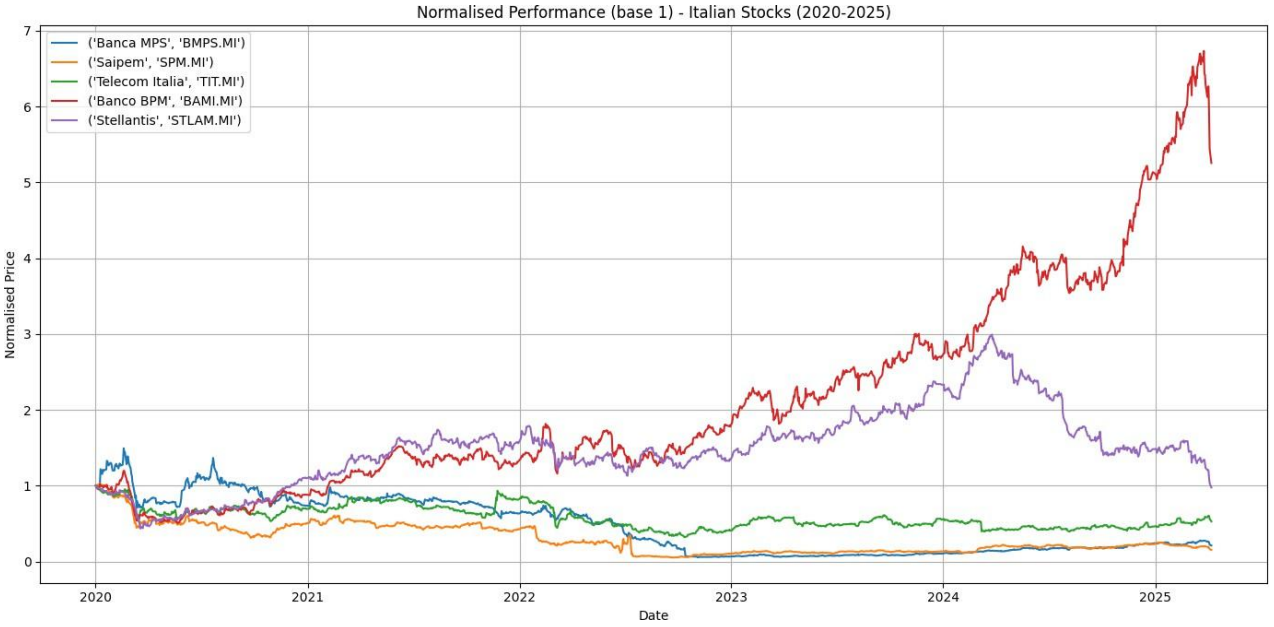
This parameter measures how dispersed the individual simulated call prices are and thus how variable our estimator is.

3. Numerical Application

In this section we analyze the algorithms to carry out pricing of specific path-dependent options using the augmented Jump diffusion model as framework for the risky asset.

First, we need to choose one specific stock for our analysis, which will be taken as the underlying asset of the options considered. We selected a small batch of stocks which have shown empirically a *jumpy* and *erratic* behavior; these stocks are plotted in *Figure 8*:

Figure 8



As it was stated in section 1, the price path is subject to two different sources of randomness: the exposure to $W(t)$ and to $J(t)$. The first exposure determines variation in the price dynamic in a *continuous* fashion, whereas the second one introduces big variations within the form of *discountinuous jumps.* In practice, it can be troublesome to distinguish these sources: a daily return of *+5%* can either indicate a sudden jump in the price or an increase in the diffusion coefficient which makes the path much more volatile. In statistics, many tests have been developed to test whether a jump has occurred on a certain date or not, but they require access to *high-frequency-data* to model the Realized Variance and build the test statistic. For instance, the Barndorff-Nielsen and Shephard (2006) test proposes a test statistic with the realized variance to assess the null hypothesis H_0 that there is no jump at a given time t[7]. Since having access to High-Frequency data would be troublesome and outside of the scope of this paper, we will carry out a much simpler rule of thumb to assess the presence of jumps.

For each stock, we compute the first differences in prices $\Delta P_t = P_t - P_{t-1}$ in absolute terms for each stock. We then compute the mean and standard deviation of these first differences $\mu_{\Delta P}$ and $\sigma_{\Delta P}$. We then build a *z_score* to which is the series of first differences normalized for the two moments computed:

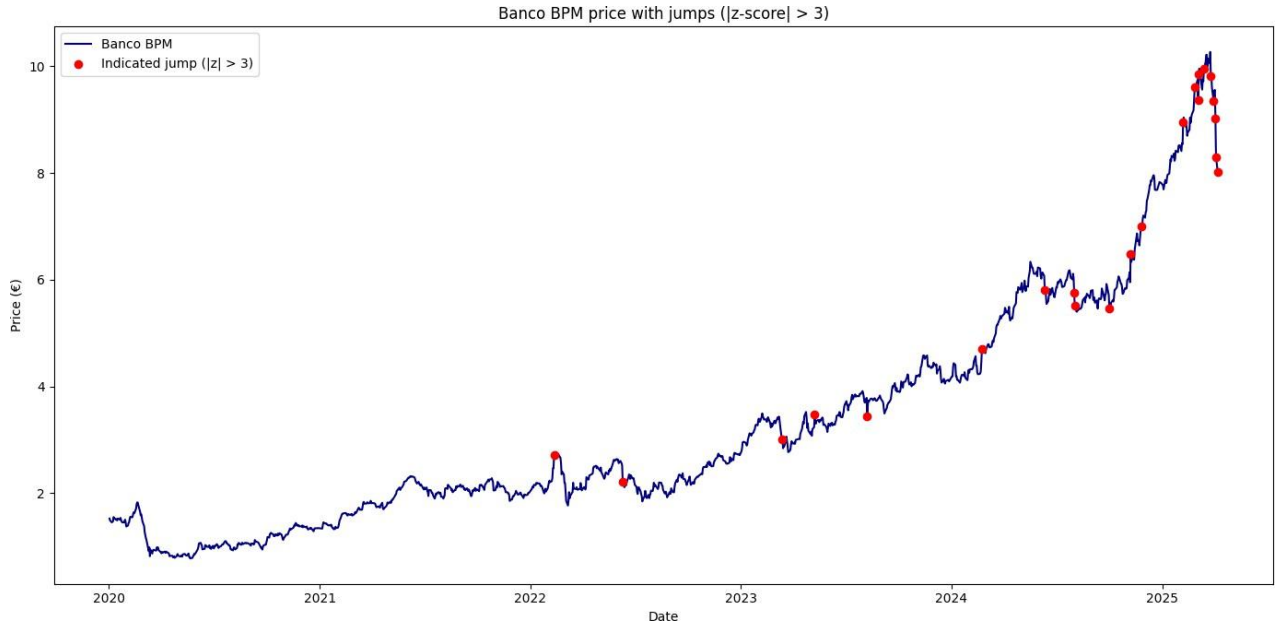
$$z_i = \frac{\Delta P_i - \mu_{\Delta P}}{\sigma_{\Delta P}}$$

And we will consider as a *jump-day* all the days in which $|z_i| > 3$.

[7] R: Barndorff-Nielsen and Shephard (2006) tests for the presence…

Of the 5 stocks considered, **Banco BPM** is the one with the most *jump days*, as it counts a total of 22 jumps, as shown in *Figure 9*

Figure 9



So, we will take this stock as our underlying.

3.1 Calibration of plain-vanilla European call closed-end formula to market prices

Before dealing with the pricing using a stochastic intensity parameter for the jump component, we will develop an initial framework to find the input parameters for the traditional Merton model: once we have these input parameters, we will extend the model to include a varying and stochastic intensity parameter λ. The procedure of finding the best input parameters for our pricing model is called Calibration: we look for parameters that make out theoretical prices as close as possible to market prices, so that our pricing process will be as realistic as possible. Within the Merton model, we mainly have four input parameters that need to be calibrated:

- σ: the diffusion coefficient for the jump-diffusive asset
- λ: the intensity parameter for the jump component
- a: the parameter taken for the mean of the jump size Y_j
- b: the second parameter taken for the variance of the jump size Y_j

Initially, we will need to find the market data for plain-vanilla call prices on Banco BPM across a range of strikes and maturities. Then, for each collected market price, we will compute the corresponding theoretical call prices using the closed formula within the Merton model. Finally, we will calibrate the pricing model by applying optimization algorithm to minimize a chosen loss function measuring the discrepancy between market and theoretical prices.

Market prices of calls for the underlying were taken from *Borsa Italiana*[8], and we downloaded the mid prices for all the options with strikes ranging from *5€* to *15€*. We selected both the options with both maturity June 2025 and September 2025; this resulted in approximately *40* cross-data on market prices. The risk-free rates

[8] Opzioni su Azioni - Borsa Italiana

r used as the drift parameter in the risk neutral world has been extracted from the Italian bonds' yield curve. Since there are no yield for maturity of 2 or 5 months, we bootstrapped those yields from the curve using linear interpolation[9]: the yields are thus equal to $r_{Jun} = 2.22\%$ and $r_{Sept} = 2.13\%$. The initial price $S0 =$ 8.9€. All these market data have been taken Tuesday, 8^{th} April 2025.

Now, let us dwell into the closed-end formula to find the corresponding theoretical prices of European calls found above. As shown in the previous section, we will use the Merton framework, where:

$$S(T) = S(0) * e^{\left(\mu - \frac{1}{2}\sigma^2\right)*T + \sigma W(T)} \left(\prod_{j=0}^{N(T)} Y_j\right)$$

Under the Physical probability measure $\mathbb{P}$. Naturally, we are now interested in the price dynamics under the risk neutral $\mathbb{Q}$ measure – as this is the common framework in option pricing, where the discounted price S_t is a martingale. So we need the expression for $S(T)$ under the $\mathbb{Q}$ probability measure. As we have previously seen, there are infinite possible risk-neutral measures (*EMMs)* we can use, since there are infinite vectors of (θ, φ) that kills the drift of S_t under $\mathbb{Q}$ probability measure. We select the specific case of $\varphi = 1$, meaning that the intensity parameter of the jump is the same in both probability measures $\mathbb{P}$ and $\mathbb{Q}$. This implicitly means that investors do not require any premium for the jump risk, they are themselves risk neutral with respect to jump risk. The *stochastic differential equation* for the asset under $\mathbb{Q}$ will thus be:

$$\frac{dS(t)}{S(t)} = (r - (\mu_Y - 1)\lambda^Q)dt + \sigma dW^Q(t) + \Delta J(t)$$

Briefly, the term $(\mu_Y - 1)\lambda^Q dt$ has been added within the drift term because, under $\mathbb{Q}$, we need to ensure that no arbitrage condition holds. That term represents the potential extra-return that is achievable under $\mathbb{Q}$ thanks to the presence of jump, since μ_Y is the expected value and λ is the intensity parameter of these – since we assume $\varphi = 1$, the two intensity parameters are equal. In fact, $E_t^Q[\Delta J(t)] = (\mu_Y - 1)\lambda^Q dt$, where $(\mu_Y - 1)$ in the conditional mean of the size of the jump in an infinitesimal time interval and $\lambda^Q dt$ is the probability of occurrence of a jump in the same infinitesimal time interval.

Following the same procedure followed in section "Counting process and *stochastic differential equation",* thus exploiting the Ito's lemma to get the *SDE* for the logarithmic price under $\mathbb{Q}$, we can come up with the following expression:

$$S(T) = S(0)e^{\left(r - (\mu_y - 1)\lambda^Q - \frac{1}{2}\sigma^2\right)T + \sigma W^Q(T)} \prod_{j=1}^{N(T)} Y_j$$

Under Merton's assumption, $Y_j \sim LogNormal(a, b^2)$, so $Y_j \sim e^{X_j}$ where $X_j \sim N(a, b^2)$. The parameters a and b are, respectively, the mean and the standard deviation of the natural logarithm of the lognormal variable Y_j, as $X_j = \ln(Y_j)$.

The price of a European call option can be derived in the following way.

We can rewrite the expression above as follows:

$$S(T) = S(0)e^{\left(r - \frac{1}{2}\sigma^2\right)T + \sigma W^Q(T) + \sum_{j=1}^{N(T)} X_j - (\mu_Y - 1)\lambda T}$$

We now write the same expression of the stock price conditioning on a precise number of jumps occurred in the period $[0; T]$ n. So, we will rewrite the expression of the stock price at time T knowing that in the period there have been n jumps – This way, we will treat $N(T)$ as a fixed value and not as a random variable.

$$S(T)|_{N(T)=n} = S(0)e^{\left(r-\frac{1}{2}\sigma^2\right)T+\sigma W^Q(T)+\sum_{j=1}^{n} X_j-(\mu_Y-1)\lambda T}$$

We notice that the second $\sigma W^Q(T)$ and third term $\sum_{j=1}^{n} X_j$ are both Normal variables. In fact, $\sigma W^Q(T) \sim N(0, \sigma^2 T)$ and $\sum_{j=1}^{n} X_j \sim N(na, nb^2)$. Since the sum of two Normal Distributions is still a Normal distribution, we have that $\sigma W^Q(T) + \sum_{j=1}^{n} X_i \sim N(na, nb^2 + \sigma^2 T)$. Rewriting the variance of this new normal as $\sigma^2 T$ and using the Standardization rule for the Normal distribution - $\frac{Ni-\mu}{\sigma} = Z_i$, where $Z_i \sim N(0,1)$, we can rewrite the expression above as:

$$S(T)|_{N(T)=n} = S(0)e^{\left(r-\frac{1}{2}\sigma^2\right)T+na+\sigma_n\sqrt{T}Z-(\mu_Y-1)T\lambda}$$

Further simplifying, and setting $r_n = r - (\mu_Y - 1)\lambda + \frac{nlog(\mu_Y)}{T}$, we can rewrite this as:

$$S(T)|_{N(T)=n} = S(0)e^{(r_n-\frac{1}{2}\sigma^2)T+\sigma_n\sqrt{T}Z}$$

Conditioning on $N(T) = n$, we can thus see the price of the jump-diffusive asset at time T is indeed the same, under the $\mathbb{Q}$ probability measure, as the price of a standard Black Scholes asset with drift r_n and diffusion σ_n. These parameters, as shown above, are indeed adjusted versions of the original drift and diffusion parameters to account for the number (n) and size of jumps occurred in the period.

The price of the European Call will thus be equal to:

$$C(0) = E^Q[(S(T) - K)^+ e^{-rT}] =$$

$$\sum_{n\in N} E^Q\left[\left(S(0)e^{\left(r_n-\frac{1}{2}\sigma^2\right)T+\sigma_n\sqrt{T}Z} - K\right)^+ e^{-rT}\middle| N(T) = n\right] * \frac{e^{-\lambda^Q T}(\lambda^Q T)^n}{n!}$$

Which can be interpreted as an average of all the expected payoffs of the call for each possible realization of number of jumps n. The second term $\frac{e^{-\lambda^Q T}(\lambda^Q T)^n}{n!}$ is in fact the probability of having exactly n jumps,

$\mathbb{Q}(N(T) = n)$, so we are taking the expected payoff from the call at time T $\forall n$: 1 ..., weighted by the risk neutral probability of occurrence of that n number of jumps. One last adjustment is needed before getting to the final closed-end formula. In the expression above, the drift rate r_n is different from the discount rate r used within the expected payoff formula. To price that expected payoff with the Black Scholes Formula, we need to make sure that there two rates coincide. Thus, by taking e^{-rT} out of the expectation operator, and by both multiplying and dividing by $e^{r_n T}$, we can rewrite the expression as:

$$\sum_{n\in N} E^{Q}[\left(S(0)e^{\left(r_n-\frac{1}{2}\sigma^2\right)T+\sigma_n\sqrt{T}Z}-K\right)^{+} e^{-r_nT}] * \frac{e^{-\lambda' T}(\lambda' T)^n}{n!}$$

Where $\lambda' = \lambda^{\mathbb{Q}} * \mu_Y$ is a slightly adjusted intensity parameter. This adjustment has been carried out to make sure that the discount and drift rates are the same. The mathematics of this is outside the scope of this paper.

Finally, we can use the Black Scholes formula to price those conditional payoffs, as, thanks to the specification of r_n and σ_n, we are back within the traditional Geometric Brownian Motion dynamic for the underlying asset.

So, the theoretical price of a plain-vanilla European call using a jump diffusive asset as in the Merton model is an *average* of all the Black-Scholes call prices for each different value of n weighted by the risk neutral probability of having precisely those n jumps.

$$C(0) = \sum_{n\in N} bsc(S(0), r_n, \sigma_n) * e^{-\lambda' T} \frac{\lambda' T^n}{n!}$$

Now that we have the closed-end formula, we set up a Python algorithm to find the corresponding theoretical prices for all the calls for which we gathered the market prices, to see how the model fits the cross-data and to calibrate the main parameters of the model: σ, a, b, λ. The algorithm used is the following:

```
#These are the formulas to compute the closed price
def black_scholes_call(S, K, T, r, sigma):
    d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    return S * stats.norm.cdf(d1) - K * np.exp(-r * T) * stats.norm.cdf(d2)

def merton_jump_call_price(S, K, T, r, sigma, lamb, a, b, N=50):

    k = np.exp(a + 0.5 * b**2) - 1  # mu_y - 1
    lamb_tilde = lamb * (1 + k)     # adjusted intensity parameter

    price = 0
    for n in range(N):
        poisson_weight = np.exp(-lamb_tilde * T) * (lamb_tilde *T)**n /
math.factorial(n)
        sigma_n = np.sqrt(sigma**2 + (n * b**2) / T)
        r_n = r - lamb * k + n * np.log(1 + k) / T
        price += poisson_weight * black_scholes_call(S, K, T, r_n, sigma_n)

    return price
```

With this algorithm we will compute the price of a European plain vanilla call using the Merton closed-end formula. Now, that we have our theorical prices, we need to compare them with market prices and define a loss function which we will try to minimize. This loss function will be a measure of the discrepancy between the two prices. Our first candidate for loss function is the following:

$$Loss_\theta = \sum_{j}^{Short} \left(lnC^i_{Merton,\vartheta} - lnC^i_{Mkt}\right)^2 + \sum_{i}^{Medium} \left(lnC^i_{Merton,\theta} - lnC^i_{Mkt}\right)^2 + \sum_{i}^{Long} \left(lnC^i_{Merton,\theta} - lnC^i_{Mkt}\right)^2$$

Where ϑ is the vector containing all four parameters we are calibrating $\vartheta = (\sigma, \lambda, a, b)$. The algorithm to compute this loss function is described in the following section.

```
def loss_function_log_scaled(params, calib_data, S0, N=50, eps=1e-6):
    sigma, lamb, a, b = params
    loss = 0.0
    for d in calib_data:
        C_model = merton_jump_call_price(
            S=S0,
            K=d["K"],       # array di strike
            T=d["T"],       # maturity fissa
            r=d["r"],
            sigma=sigma,
            lamb=lamb,
            a=a,
            b=b,
            N=N
        )
        diff = np.log(C_model + eps) - np.log(d["C"] + eps)
        loss += np.dot(diff, diff)

    return float(loss)
```

Our goal is now to set up an optimization problem to find the optimal parameters ϑ^* so that the loss function is minimized; in other words, we wish to find the parameters of input ϑ^* such that the discrepancy between theoretical prices produced by the pricing model and the market prices is minimized, so that in later uses these parameters will be used. In mathematical terms, our problem is:

$$\underset{\vartheta}{\text{argmin}}\ Loss_{\vartheta}$$

Where $Loss_{\text{P}}$ is essentially a function defined on four variables $f: \mathbb{R}^4 \to \mathbb{R}, f(a, b, \lambda, \sigma)$. We are trying to find the minimum in a static optimization problem where the target function is defined in four dimensions. Naturally, there are some constraints that must be met by our optimum: $b > 0, \sigma > 0\ and\ \lambda \geq 0$. Note that whereas b and σ must be positive, as they both refer to standard deviations, the intensity parameter for the jump must be non-negative: it is plausible to have $\lambda = 0$, in which case we would simply go back to the traditional Black-Scholes model. Thus, our constrained optimization model has the following form:

$$\underset{(a|b|\lambda|\sigma)}{\text{argmin}} \sum_{j}^{Short} \left(lnC^i_{Merton,(a,b,\lambda,\sigma)} - lnC^i_{Mkt}\right)^2 + \sum_{i}^{Medium} \left(lnC^i_{Merton(a,b,\lambda,\sigma)} - lnC^i_{Mkt}\right)^2 + \sum_{i}^{Long} \left(lnC^i_{Merton(a,b,\lambda,\sigma)} - lnC^i_{Mkt}\right)^2$$

s.t

$$a \in \mathbb{R}$$

$$b > 0$$

$$\sigma > 0$$

$$\lambda \geq 0$$

3.1.1 L-BFGS-B and Gradient Descent algorithms

To carry out this minimization problem, we have tried two different algorithms, with slightly different features. The first one used is **L-BFGS-B**[10]**.** This second-order, quasi-Newton algorithm tries to find the optimum by using both the gradient and the Hessian matric of the target function. The main idea of this class of optimization algorithm is that starting from an initial point of the multivariate function $f(\boldsymbol{x})$, they use the Taylor expansion of second order to approximate a quadratic function from that point, and they minimize that function g. From the new point they repeat this procedure until they reach the local minimum. For each iteration, an estimation of the Hessian matrix used for the Taylor expansion is necessary:

$$g(x) = f(q) + (x-q)^T \nabla f(g) + \frac{1}{2}(x-q)^T B(x-q)^{11}$$

Where q represents the point of the previous iteration, $\nabla f(g)$ the gradient of the target function in the point of the previous iteration and B the estimated Hessian matrix. This is the common setting for any second-order minimization algorithm, where they use both the first derivative (Gradient) and the second derivative (Hessian matrix) in the process. What is specific about L-BFGS-B is the way in which this algorithm updates the Hessian matrix. Let us call B' the Hessian matrix of the previous iteration. The matrix B for the new iteration is obtained under the following criterion:

$$B \; s.t. \min_B ||W(B^{-1} - B'^{-1})W||_F \quad ^{12}$$

Where $||.||_F$ is the Frobenius norm for a matrix. The matrix W in inserted to guarantee that B is positive definite, so that our minimization problem does not become a maximization one, and we take the inverse of the Hessian for numerical efficiency.

This algorithm yields a minimum in a rather short period of time and proposes the following optimal parameters:

a^*	*0.103*
b^*	*0.0448*
σ^*	*0.01*
λ^*	*4.95*

The Python code to carry out the algorithm is presented below: note that some constraints in values have been artificially imposed on the parameters, to reduce the computing time and complexity of the algorithm. Nevertheless, these numerical constraints can be relaxed.

[10] Una breve introduzione all'algoritmo di ottimizzazione BFGS

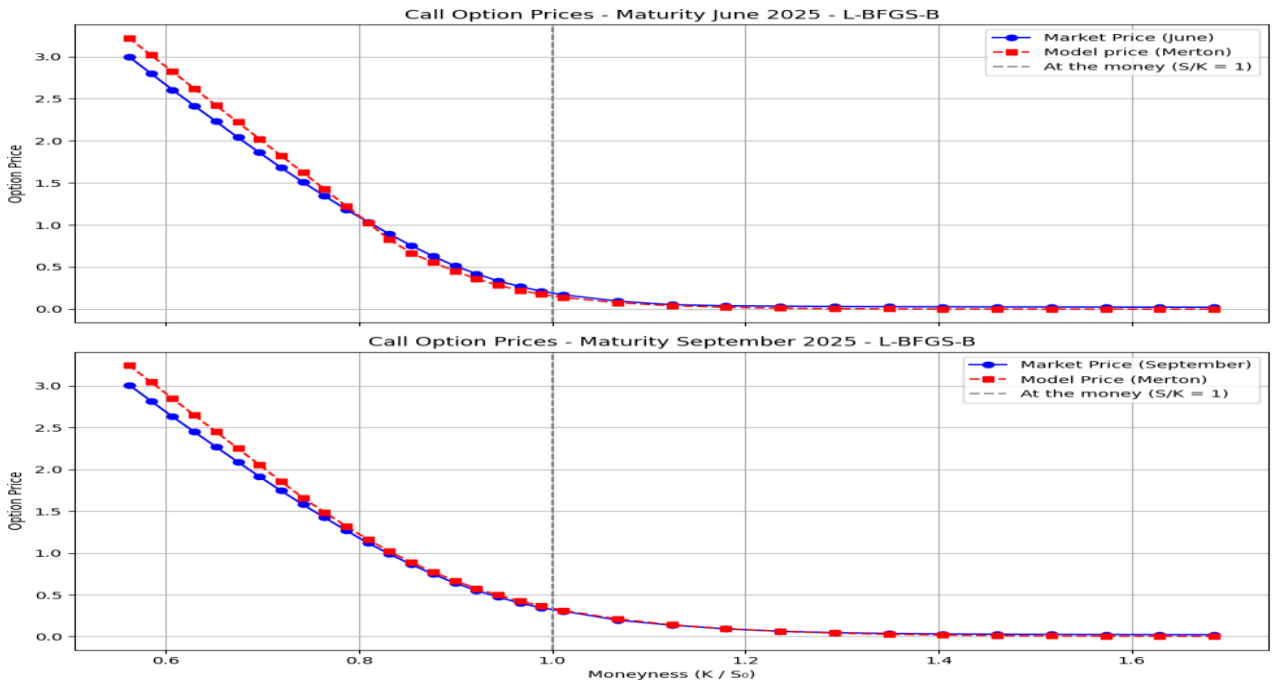
[11] *Understanding scipy.minimize part 1: The BFGS algorithm*

[12] *Understanding scipy.minimize part 1: The BFGS algorithm*

```
def calibra_merton(df, S0):
    # Initial guess per i parametri: sigma, lambda, a, b
    initial_guess = [0.25, 1.0, -0.1, 0.2]
    # Bounds: (min, max)
    bounds = [
        (0.01, 1.0),      # sigma > 0
        (0.0, 5.0),       # lambda ≥ 0
        (-1.0, 1.0),      # a: mean of log-jump (it can be negative)
        (0.01, 1.0)       # b > 0
    ]
    # Minimization of Loss function
    result = minimize(
        loss_function,
        x0=initial_guess,
        args=(df, S0),
        bounds=bounds,
        method='L-BFGS-B',
        options={'disp': True, 'maxiter': 500}
    )
    # results
    if result.success:
        sigma_opt, lambda_opt, a_opt, b_opt = result.x
        print("✅ Solution found")
        print(f"sigma:   {sigma_opt:.4f}")
        print(f"lambda: {lambda_opt:.4f}")
        print(f"a:       {a_opt:.4f}")
        print(f"b:       {b_opt:.4f}")
        return result.x
    else:
        print("❌ Solution not found")
        print(result.message)
        return None
```

The resulting theoretical prices compared with market prices are graphed in *Figure 10.*

Figure 10



We see that theoretical prices are well calibrated for *ATM* and *OTM* options, whereas they tend lie above the market prices when they are *ITM* – for call options with strike price K much lower than the current underlying's stock price $S0$.

The second possibility for the minimization problem is in the Gradient Descent algorithm. This algorithm follows a similar logic to the previous one, but it differs as it is first-order algorithm: it relies only on the information conveyed by the gradient of the function, ignoring the Hessian matrix.[13] Starting from an initial point x_0 and having specified the learning rate α of the process, the algorithm iterates toward the minimum according to the following expression: $x^i = x^{i-1} - \alpha \nabla f(x^{i-1})$, where x^i is the vector after the iteration and x^{i-1} is the vector before the current iteration[14]. The logic is to move in the opposite direction of the sign of the target function's gradient to reach a vector x^* where $\nabla f(x^*) = 0$, indicating the presence of a minimum.

The main problem with both Gradient Descent and L-BFGS_B methods is that they look for a local minimum. Since these iterative methods stop when they find a local minimum, they may miss out "true" target of the optimization problem: the global minimum. Hence, the assumption of convexity of the target function is often taken for granted in application of these algorithms, so that a local minimum corresponds inevitably to a global minimum. To make sure that the results yielded by the algorithm make sense, we will make this assumption as well.

The Python code to define the needed function is reported below:

```
def gradient_descent(loss_fn, theta0, args=(), eta=0.01, max_iter=1000, tol=1e-
6, verbose=True):
    theta = np.copy(theta0)
    loss_history = []
    for i in range(max_iter):
        grad = numerical_gradient(loss_fn, theta, 1e-5, *args)
        theta -= eta * grad
        theta[0] = np.clip(theta[0], 0.01, 1.0)     # sigma
        theta[1] = np.clip(theta[1], 0.0, 5.0)      # lambda
        theta[2] = np.clip(theta[2], -1.0, 1.0)     # a
        theta[3] = np.clip(theta[3], 0.01, 1.0)     # b
        loss = loss_fn(theta, *args)
        loss_history.append(loss)

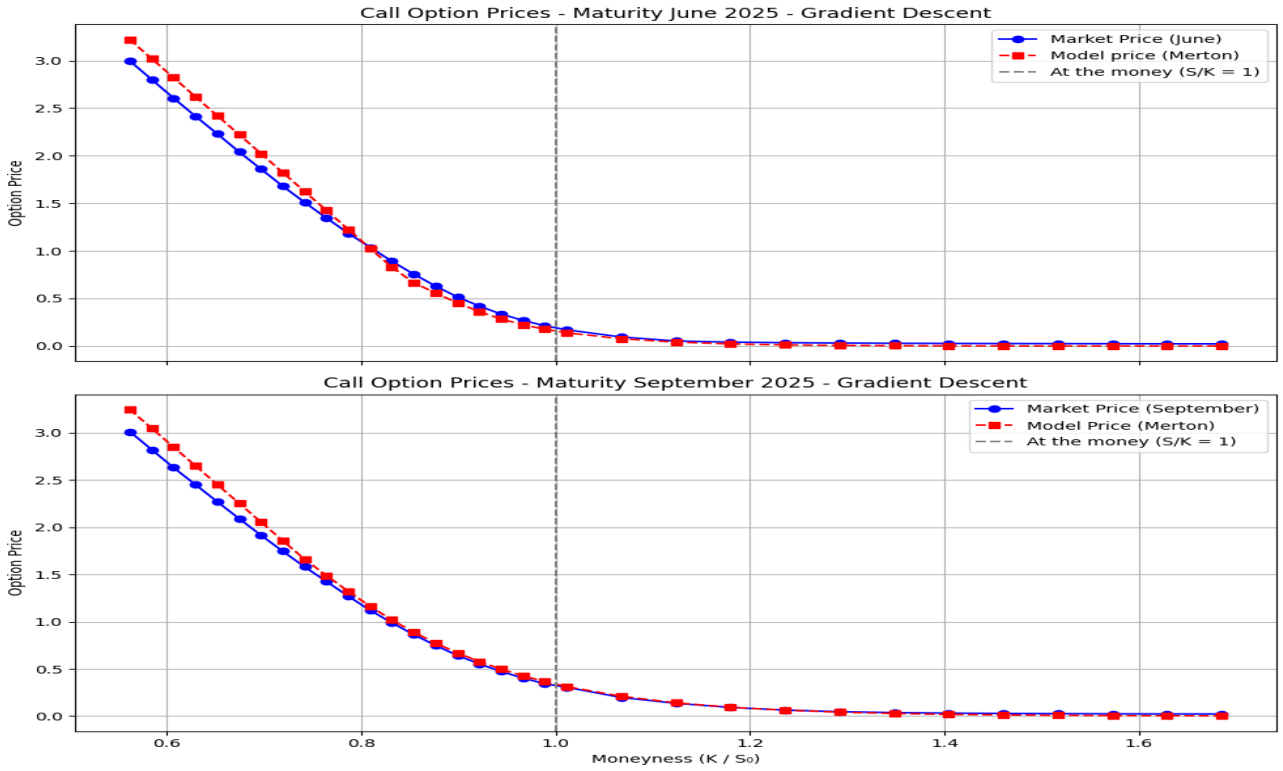
        if verbose and i % 50 == 0:
            print(f"[Iter {i}] Loss: {loss:.6f}   Params: {theta}")
        if i > 0 and abs(loss_history[-2] - loss_history[-1]) < tol:
            print("✅ Convergenza raggiunta")
            break
    return theta, loss_history
```

The theoretical prices computed with the optimal parameters yielded by the Gradient descent are graphed in *Figure 11*

[13] Cos'è l'algoritmo di ottimizzazione: una guida completa

[14]

Figure 11



We see similar features with the prices yielded by the L-BSGF-B: It works very well for *ATM* and *OTM* options, whereas it generates substantial overpricing for *ITM* options. Comparing the two minimized Loss Function, we see that L-BSGF-B slightly outperforms the Gradient Descent algorithm, with a Loss Function of 0.669. Thus, we will use parameters calibrated using this method for further use.

3.2. Pricing European Calls using Merton Model

Having obtained the optimal calibrated parameters, we must now use these coefficients to simulate paths for the stock price according to Merton's stochastic differential equation. Using the Euler discretization method with the calibrated parameters as inputs it is indeed possible to obtain a matrix containing *M* simulated paths for the stock price with jumps. Since Merton's *stochastic differential equation* to describe the continuous dynamics of the asset price is the following:

$$\frac{dS(t)}{S(t^-)} = \mu dt + \sigma dW(t) + \Delta J(t)$$

We can shift to the risk neutral probability measure $\mathbb{Q}$ selecting one of the infinite possible *Equivalent martingale measure.* We will select the one where $\varphi = 1$ – indicating no jump risk premium, since $\lambda^{\mathbb{Q}} = \lambda^{\mathbb{P}}$ - and $\vartheta = \frac{\mu+(\mu_Y-1)\lambda-r}{\sigma}$. With these two parameters we are guaranteed that the discounted stock price is a martingale process, as they *kill* the drift of the discounted price under the *EMM*, and we can rewrite the *stochastic differential equation* by applying the Girsanov's Theorem as:

$$\frac{dS(t)}{S(t^-)} = rdt + \sigma(dW^{\mathbb{Q}}(t) - \vartheta dt) + \Delta J(t)$$

Since $\sigma\vartheta = \mu + (\mu_Y - 1)\lambda - r$, we can rewrite this expression as:

$$\frac{dS(t)}{S(t^-)} = (r - (\mu_Y - 1)\lambda)dt + \sigma dW^{\mathbb{Q}}(t) + \Delta J(t)$$

By discretizing the infinitesimal differences, we can express this as:

$$\frac{\Delta S(t)}{S(t)} = (r - (\mu_Y - 1)\lambda)\Delta t + \sigma\sqrt{\Delta t} * Z + \Delta J(t)$$

The important component in this expression we need to look at for numerical analysis is $\boldsymbol{\Delta J(t)}$**.** The discrete variation of the compound Poisson process can take on two different values:

$$\Delta J(t) = \begin{cases} Y_j - 1 & \Delta N(t) = 1 \\ 0 & \Delta N(t) = 0 \end{cases}$$

Δ

Meaning that if a jump occurs at time t then the impact in size to the asset price is measured by the other lognormal random variable $Y_j \sim LogN(a, b^2)$.

Using the solution provided in the previous sections for the price $S(t)$ governed by the Merton's dynamics we can write the discretization Euler algorithm used to simulate *M* paths for the stock:

$$\begin{cases} S(t) = S(t-\Delta t)e^{\left(r-(\mu_Y-1)\lambda-\frac{1}{2}\sigma^2\right)\Delta t+\sigma\sqrt{\Delta t}*Z} & if\ \Delta N(t) = 0 \\ S(t) = S(t-\Delta t)e^{\left(r-(\mu_Y-1)\lambda-\frac{1}{2}\sigma^2\right)\Delta t+\sigma\sqrt{\Delta t}*Z} * (Y_j * \Delta N(t)) & if\ \Delta N(t) = 1 \end{cases}$$

)

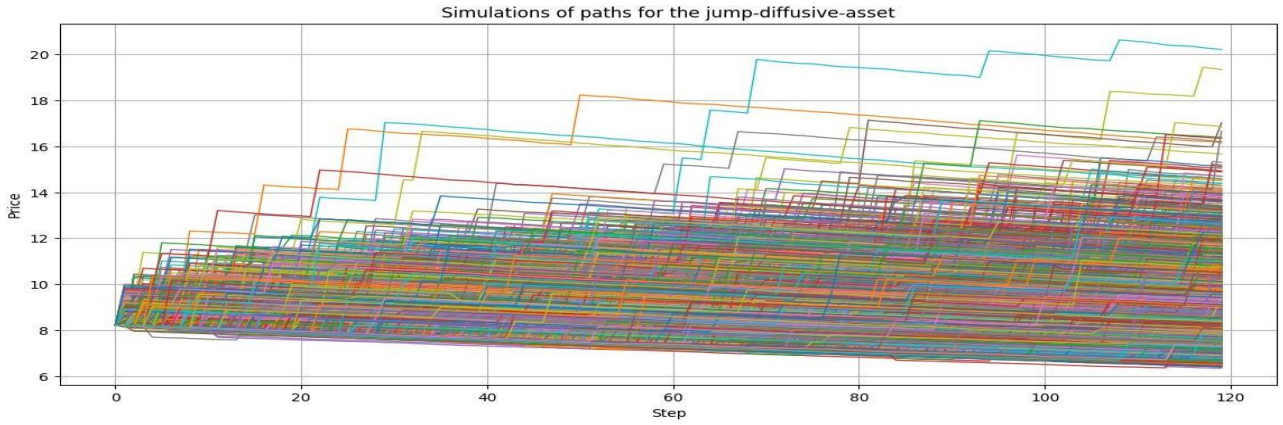
In fact, the last term is included to consider the potential jump occurred in Δt where the intensity parameter of the Poisson is $\lambda * \Delta t$. If a jump occurs in t then this will be reflected in the stock price at time t by the last multiplication term, and the size of this jump is determined by the realization of the lognormal variable Y_j.

By setting the proper Python function, we can create a matrix $A \in \mathbb{R}^{MxN}$ where the M rows are the number of simulated paths and the N columns are the discretized points in time in $[0, T]$ such that $\Delta t = T/N$.

3.2.1 Generation of matrix of simulated paths

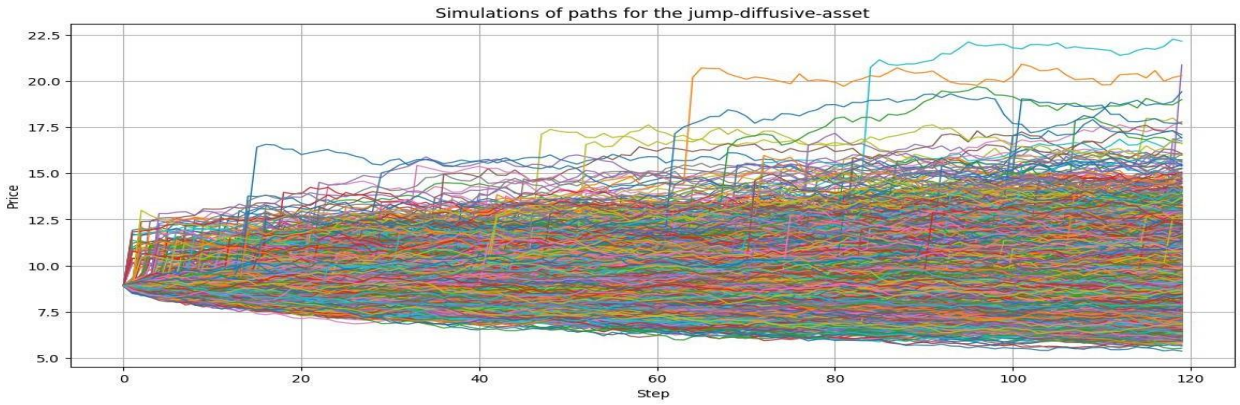
Using the optimal calibrated parameters obtained by the minimization problem we obtain the following 10,000 simulated paths for the 119 trading days between the 8^{th} of April and 20^{th} of September (maturity of the option) and we plot it in *Figure 12*

Figure 12



In this plot we can see that jumps are mostly upward and that there is very little *continuous* variability: most of the variations of the stock price within the time framework are due to the *discontinuous* and sudden jumps rather than volatility induced by the Brownian motion risk exposure. This is due to the calibrated parameter $\sigma = 0.01$ which is indeed a very low volatility coefficient. Forcing the volatility parameter to be equal to 0.2 we would see that the resulting paths are much more volatile in a continuous fashion as in *Figure 13*:

Figure 13



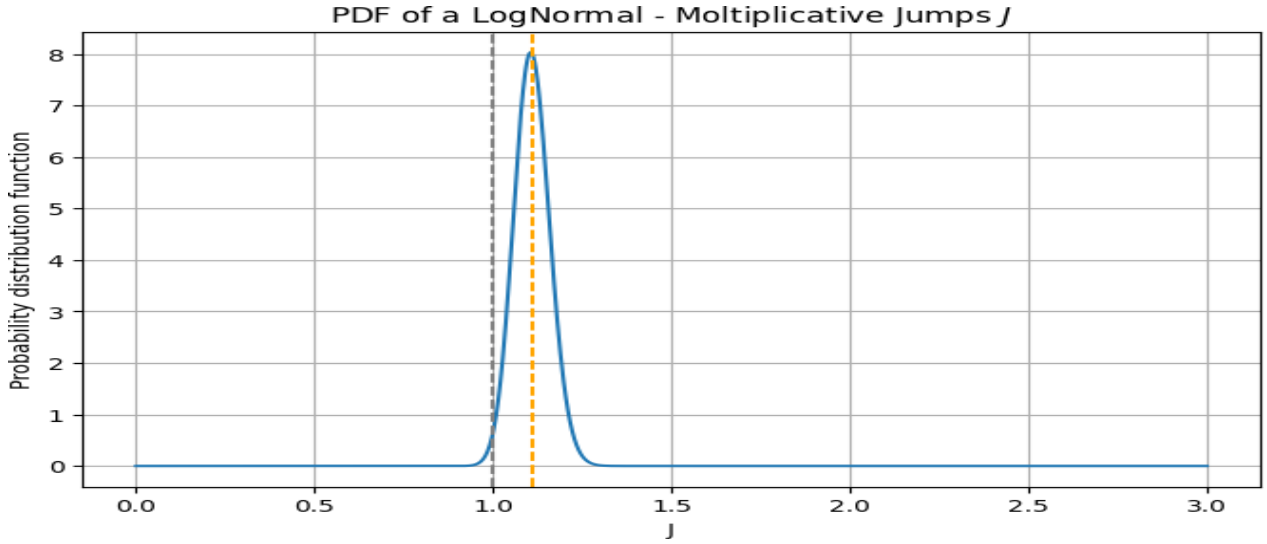
Given the properties of the log-normal distribution we can also study the jumps simulated for Banco BPM. The expected value for the size of the jump is indeed $E[Y_j] = e^{a-\frac{1}{2}b^2} = e^{0.103-\frac{1}{2}0.04^2} = 1.1$. On average, the jump occurred in the stock price is positive will have a 1.1x upward impact, as $S(\tau_i) = S(\tau_i^-) * Y_{\tau_i}$ and

$$E\left[S(\tau_i)|\mathcal{F}_{\tau_i^-}\right] = S(\tau_i^-) * \mu_{Y_{\tau_i}}$$

Taking the initial price $S0 = 8.2$ as a reference, the average impact of the jump would bring the price up to $8.2 * 1.1 = \mathbf{9}$.

We can also compute the probability of having a downward jump, which is easily achievable by taking the integral from $-\infty$ to 1 of the *probability distribution function* of the lognormal distribution Y. In *Figure 14* we show the PDF of this distribution highlighting with a grey vertical bar the right extreme of the integral to compute $\Pr[Y < 1] = \int_{-\infty}^{1} f_Y(j)dj$, where $f_Y(j)$ is the PDF of the lognormal.

Figure 14



We see graphically that the integral would be small, and it is indeed equal to $0.0106 \sim 1.06\%$.

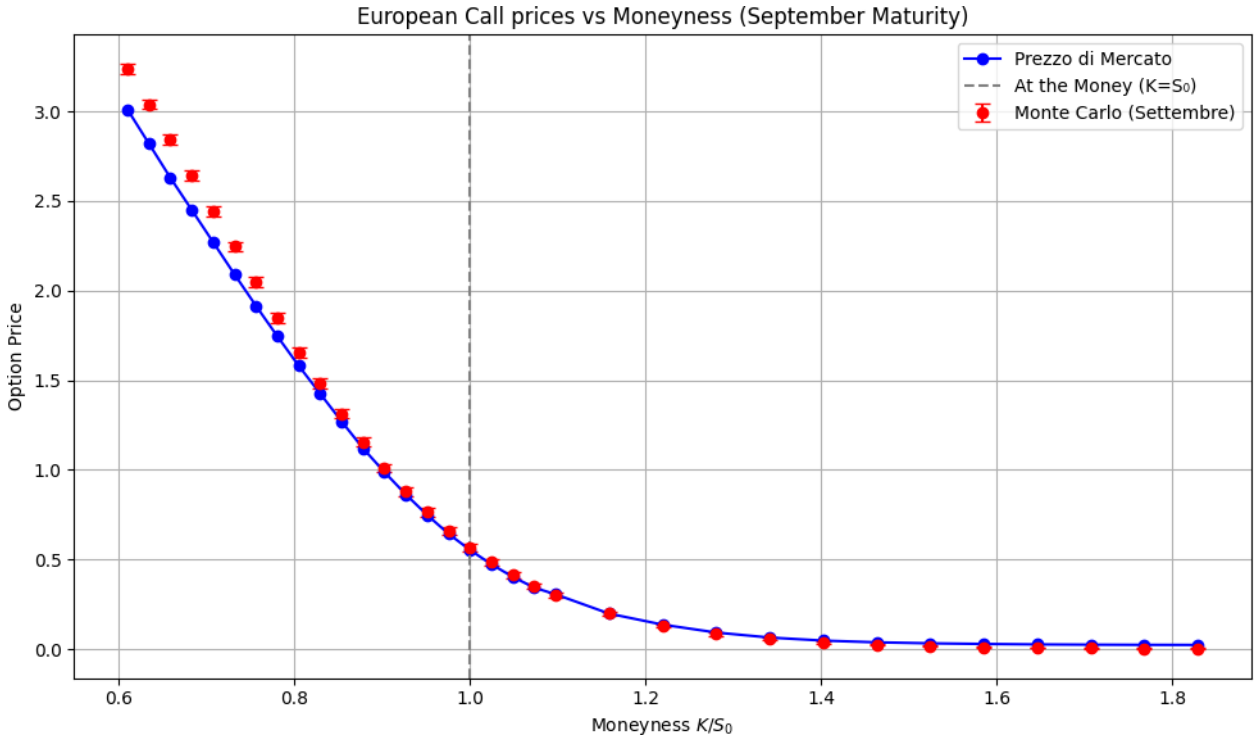
We know have the matrix of simulated prices for the underlying. To compute the Monte Carlo estimator of the European plain vanilla call we need only the last column of the simulated prices since the payoff of this derivative is not path dependent. As explained in the previous sections, the formula for this consistent and unbiased estimator for the call price is the following:

$$C_{MC} = \frac{1}{M}\sum_{i=1}^{M}\left(S(T)^i - K\right)^+ e^{-rT}$$

Where $S(T)^i$ are all the M rows of the last column contained in the matrix of the simulations. We will also compute the standard error of this estimator and build an interval of confidence around the point estimate.

In *Figure 15* we can see the prices computed using Monte Carlo, including an interval of confidence, for the call option on Banco BPM for different values of strikes, plotted against the market prices for each call option. We see that the outcome of this numerical exercise is coherent with the theoretical prices for the European calls computed in the previous section: whereas the application prices the ATM and OTM call correctly, it has a significant overpricing for the ITM calls. The radius of the confidence interval is very small thanks to the big amount of simulations computed ($M = 10{,}000$)

Figure 15


European Call prices vs Moneyness (September Maturity)
Prezzo di Mercato
At the Money (K=S₀)
Monte Carlo (Settembre)
Option Price
Moneyness K/S_0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.6
0.8
1.0
1.2
1.4
1.6
1.8

3.3. Pricing Fixed Lookback Calls using Merton Model

3.3.1. Pure Monte Carlo estimator

We will now dig deeper into the pricing of the Fixed Lookback call using the Merton Model we have been developing in the previous sections. Initially, we will keep the intensity parameter constant as originally postulated by Merton, and only subsequently will we carry out the numerical exercise assuming that the counting process is a Hawkes process to allow a stochastic intensity parameter. The Fixed lookback call option has a path-dependent payoff of the following form:

$$payoff_T = \max\{(\max_t S(t) - K); 0\}$$

The main difference between the fixed and the floating lookback lies in the strike: in the fixed lookback, the strike is, indeed, fixed, and the payoff is given by the difference between the maximum price observed in the specific time horizon and the pre-specified strike K. The pricing of this product, rather than the floating counterpart, seems more logical so we can compare the call prices with the European calls for a given strike.

To carry out the pricing, we still use the same matrix $A \in \mathbb{R}^{MxN}$ which contains all the simulated prices. In contrast with the previous exercise, we now need to look at the whole matrix and not just the last column, as every simulated price matters to form the final payoff. The logic of the following algorithm is the following: for every row $i = 1 \dots M$ we save the maximum price achieved in that simulation and compute the payoff of the fixed lookback in accordance with the expression above. As usual, we then discount the payoff and take the average. This is the Python code to compute the pure Monte Carlo price and build an interval of confidence.

```
def monte_carlo_lookback_fixed(paths, K, r, T, alpha=0.95):
    S_max = np.max(paths, axis=1)
    payoffs = np.maximum(S_max - K, 0)
    discounted = np.exp(-r * T) * payoffs

    M = len(discounted)
    price = np.mean(discounted)
    se = np.std(discounted, ddof=1) / np.sqrt(M)

    from scipy.stats import t
    t_quantile = t.ppf(1 - (1 - alpha)/2, M-1)
    ci_lower = price - t_quantile * se
    ci_upper = price + t_quantile * se
    return price, se, (ci_lower, ci_upper), payoffs
```

The price for an At the Money ($K = 8.2$) fixed Lookback call option on Banco BPM with maturity on September 2025 is the following:

Price	0.9524
Standard Error	0.012
Lower Bound (95%)	0.929
Upper Bound (95%)	0.975

We see that the price of the Lookback is significantly higher than the one for ATM European call ($C(0) = 0.57$) and this is perfectly logical: the payoff of a fixed lookback is greater or equal than the payoff of a European call, since it does not take the final price of the stock at time T but rather it takes the maximum in the whole period $[0; T]$. Intuitively, the payoff of the lookback cannot be smaller than the one of the European call. Hence, since the price of an option is a conditional expected value of the payoff, weighted by the probabilities of each outcome, the higher the payoff the higher the price.

The only issue we can find in this procedure is the radius of the confidence interval; it is not extremely wide, but still it can be improved. To do so, we introduce a **Variance reduction technique** used to reduce the uncertainty regarding a Monte Carlo estimator for the same number of simulations carried out. The three main techniques we can use are the following: Control Variate, Antithetic Variate and Quasi Monte Carlo. For this analysis we will carry out the first technique for the following reasons: Antithetic Variate technique is helpful because for a one simulation of Z you save both Z and $-Z$, but it may be problematic because of the asymmetry of the stock distribution because of the presence of jumps (we have argued that jumps are asymmetric as the expected impact is 1.2x). Quasi-Monte Carlo technique is also very helpful as it allows the matching of the first moment by changing the weights given to each simulation but is very burdensome when dealing with path-dependent options like the one we are pricing.

3.3.2. Control Variate estimator to reduce variance

The logic of the Control Variate variance reduction technique is the following. Given our target variable Y, we wish to estimate, by Monte Carlo, the unknown expected value $E[Y]$. We can exploit the information we have about another variable X, whose $E[X]$ is instead known, to reduce the error in the estimation of the *unknown* quantity Y. The Control Variate Estimator has the following structure:

$$Y_i(b) = Y_i - b(X_i - E[X])$$ [15]

$$\widehat{Y_n}(b) = \frac{1}{n}\sum_{i=1}^{n} Y_i(b)$$

Instead of taking the values of Y_i, we apply a simple correction and take $Y_i(b)$ to make the estimator instead. This estimator will have the benefit of having a reduced variance and standard error, whilst still preserving the properties of unbiasedness and consistency. The last two properties heavily rely on the choice of the variable X: this estimator must be unbiased and consistent as well.

$$E\left[\widehat{Y_n}(b)\right] = E[\frac{1}{n}\sum_{i=1}^{n} Y_i - b(X_i - E[X])] = E[Y] - b(E[X] - E[X]) = E[Y]$$

This expression only holds if $E[\bar{X}_n] = E[X]$, when the average of the sample mean of X is truly the true parameter $E[X]$. If a bias is present in the X variable, it will be kept in the Control Variate Monte Carlo estimator as well. The same logic can be used to assess the consistency of the Control Variate MC estimator, and in this case the property is preserved only if $\lim_{n\to\infty} \bar{X}_n = E[X]$.

So, the estimator is unbiased, and we can prove that the variance of this new estimator is lower than the one of the *pure* Monte Carlo as long as the two variables X and Y have a correlation different from zero, $|\rho_{XY}| \neq 0$. In the estimator formula we see the usage of a coefficient b to link the variable X and Y, and we need to

establish a criterion, a rule to set the optimal parameter b^*. The most immediate and natural logic would be to pick the coefficient b^* such that the variance of each replication $Y_i(b)$ is minimized:

$$\min_b Var(Y_i(b))$$

The variance of the individual replication can be rewritten as follows:

$$Var(Y_i(b)) = Var(Y_i - b(X_i - E[X])) = Var(Y_i) + b^2 Var(X_i - E[X]) - 2bCov(Y_i; X_i - E[X])$$

Hence, the variance of the single replication of control variate estimator is a function of b; in particular, we can easily see that it is a upward parabola. Hence, to find the optimal parameter b^* we can take the first derivative of the variance function and set it equal to 0.

$$\frac{\partial Var(Y_i(b))}{\partial b} = 0$$

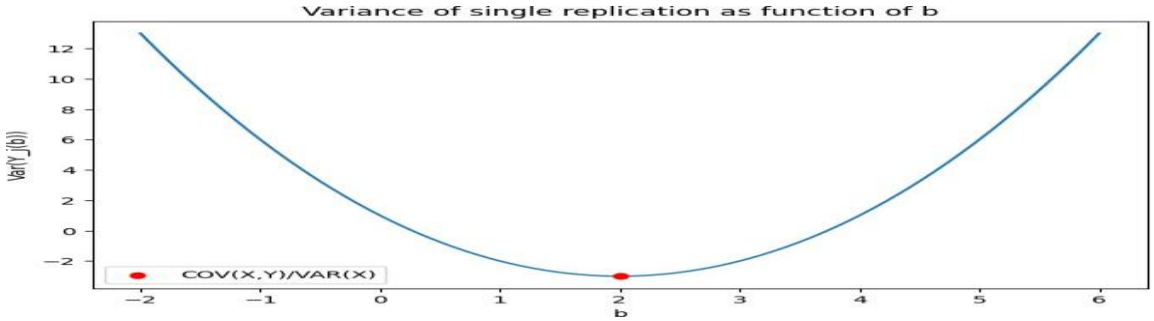
$$\frac{\partial(b^2\sigma_X^2 - 2b\sigma_X\sigma_Y\rho_{XY} + \sigma_Y^2)}{\partial b} = 0$$

$$2b\sigma_X^2 - 2\sigma_X\sigma_Y\rho_{XY} = 0$$

$$b^* = \frac{2\sigma_X\sigma_Y\rho_{XY}}{2\sigma_X^2} = \frac{Cov(X,Y)}{Var(X)}$$

This outcome can be visualized in *Figure 16*

Figure 16



The optimal coefficient b^* is thus the *OLS* coefficient between the series of Y_i taken as the target variable and X_i as the independent variable. To get this coefficient it will be enough to run a simple univariate regression between these two variables including the intercept.

The control variate estimator has variance $Var\left[\widehat{Y_n}(b^*)\right] = Var\left[\frac{1}{n}\sum_{i=1}^{n} Y_i(b^*)\right] = \frac{1}{n^2} nVar\left(Y_i(b^*)\right) = \frac{Var(Y_i(b^*))}{n}$

where $Var(Y(b^*))$ is the variance of the individual, single simulation according to the Control

Variate Estimator. Using the same properties of the variance operator, we can come up that the variance of the pure MC estimator is $\frac{Var(Y_i)}{n}$, where $Var(Y_i)$ is, again, the variance of the individual simulation. Hence, to compare the variances of the two estimators we need to assess whether and when:

$$Var\left(Y_i(b^*)\right) < Var(Y_i)$$

If this inequality is met, then we have an actual advantage by using the control variate rather than the pure Monte Carlo. By using the decomposition of $Var(Y_i(b^*))$ already used above we can further rewrite this as $\sigma_Y^2(1-\rho_{XY}^2)$. Thus,

$$\sigma_Y^2(1-\rho_{XY}^2) < \sigma_Y^2$$

If and only if

$$|\rho_{XY}| > 0$$

To conclude, we can say that it is reasonable to use the control variate technique when we use a variable which is correlated, either positively or negatively, with the target variable.

Now, we can look for a control variate for the pricing of the fixed lookback within the Merton Model. Again, this variable must be both correlated with our target and must also have a known expected value $E[X]$. In our case, we selected as X variable the price of the European plain vanilla call options using the Merton jump-diffusive asset. The choice is reasonable because the two prices are extremely highly correlated ($\rho_{XY} = 0.99$) and because, as we have shown in the previous sections, the expected value of the price of the European calls is indeed known and given by the closed end formula:

$$C_0 = \sum_{n\in N} bsc(S(0), r_n, \sigma_n) * e^{-\lambda' T} \frac{\lambda' T^n}{n!}$$

Again, we will deal with an ATM option, whose strike is $K = 9$. The optimal b^* computed by the linear regression between X and Y is close to unity, equal to 1.06. Nevertheless, we will not be using this coefficient for our estimator to avoid adding bias in the estimate.
Since the true regression parameter β_{OLS} is not known, we will use the sample counterpart of the same coefficient $\widehat{\beta_{OLS_n}} = \frac{\widehat{Cov(X,Y)}}{\widehat{Var(X)}}$. From traditional econometric theory, we know from the Gauss-Markov theorem that, under the *OLS Assumptions*, this estimator is the *B.L.U.E* - Best linear unbiased estimator – and is also consistent with the true population parameter. Despite all these nice statistical properties, we find that using $\hat{Y}_n(\hat{b}_n)$ rather than $\hat{Y}_n(b^*)$ introduces a positive bias in the estimate for $E[Y]$. In fact:

$$E\left[\hat{Y}_n(\hat{b}_n)\right] - E[Y] = E[\hat{Y}_n - \hat{b}_n(\hat{X}_n - E[X])] - E[Y] = -E[\hat{b}_n(\hat{X}_n - E[X])] + E[Y] - E[Y]$$

If $\hat{b}_n$ and $\hat{X}_n$ were independent, then the bias would be zero. Because $E[X*Y] = E[X]*E[Y]$ if the two variables are independent, we could indeed *break* the expectations operator in $E[\hat{b}_n] * E\,[\hat{X}_n - E[X]] = 0$, as the second term is, by unbiasedness of the control variate, equal to 0. Because we estimate $\hat{b}_n$ from the whole sample, we use the same data to both compute $\hat{b}_n$ and $\hat{X}_n$, making them not independent from each other and inducing a bias. To avoid this bias, we simply need to come up with an algorithm that allows $\hat{b}_n$ and $\hat{X}_n$ to be independent from each other. A procedure to do so would be to divide the dataset into two sub-datasets composed by n_1 and n_2 datapoints, respectively, and use the two datasets for the two different purposes. We would use the first n_1 observations of X and Y to compute the coefficient $\hat{b}_{n_1}$, and the remaining n_2 simulations of X and Y to compute the estimator:

$$Y_i(\hat{b}_{n_1}) = Y_i - \hat{b}_{n_1}(X_i - E[X])$$
$$\hat{Y}_{n_2}(\hat{b}_{n_1}) = \frac{1}{n_2}\sum_{i=1}^{n_2} Y_i(\hat{b}_{n_1})$$

By using a portion of the dataset only to compute the $\hat{b}_{n_1}$ and the rest to compute the estimate guarantees that

$\hat{b}_{n_1}$ and X_n are independent, since they are computed on different datapoints, different simulations. By using this procedure, we avoid adding a bias in our estimate. The following Python code is the algorithm used to price the *At-The-Money* Fixed Lookback call using this procedure, taking the prices of the European plain vanilla as control variate to reduce the variance of the estimator.

```
def stima_control_variate_unbiased(payoffs_Y, payoffs_X, call_theo, r, T, alpha=0.95,
split_ratio=0.3):
    M = len(payoffs_Y)
    n1 = int(M * split_ratio)
    # Split dataset
    Y1, Y2 = payoffs_Y[:n1], payoffs_Y[n1:]
    X1, X2 = payoffs_X[:n1], payoffs_X[n1:]
    # linear regression
    X1_reg = sm.add_constant(X1)
    model = sm.OLS(Y1, X1_reg).fit()
    beta_hat = model.params[1]
    # formula
    Y2_star = Y2 - beta_hat * (X2 - call_theo)
    # Discount the payoffs
    discounted = Y2_star
    N = len(discounted)

    # compute price, standard error and interval of confidene
    price = np.mean(discounted)
    se = np.std(discounted, ddof=1) / np.sqrt(N)
    t_quantile = t.ppf(1 - (1 - alpha)/2, N - 1)
    ci_lower = price - t_quantile * se
    ci_upper = price + t_quantile * se
    print(f"🎯 Price lookback (Control Variate Unbiased): {price:.4f}")
    print(f"📉 Standard Error: {se:.6f}")
    print(f"D CI {int(alpha*100)}%: ({ci_lower:.4f}, {ci_upper:.4f})")

    return Y2_star, price, se, (ci_lower, ci_upper)
```

By using this algorithm, we come up with the following values:

Price	0.964
Standard Error	0.0049
Lower Bound (95%)	0.9541
Upper Bound (95%)	0.973

As expected, the point estimate of the control variate estimator is very close to the estimate already obtained with the pure Monte Carlo technique, but the standard error of this estimate is more than 10 times smaller than before: the interval of confidence is much narrower, indicating more precision and confidence in the price provided. Again, the price of the At-the-money Lookback is higher than the corresponding price for the At-the-money European Call, and, as explained above, this makes perfect sense.

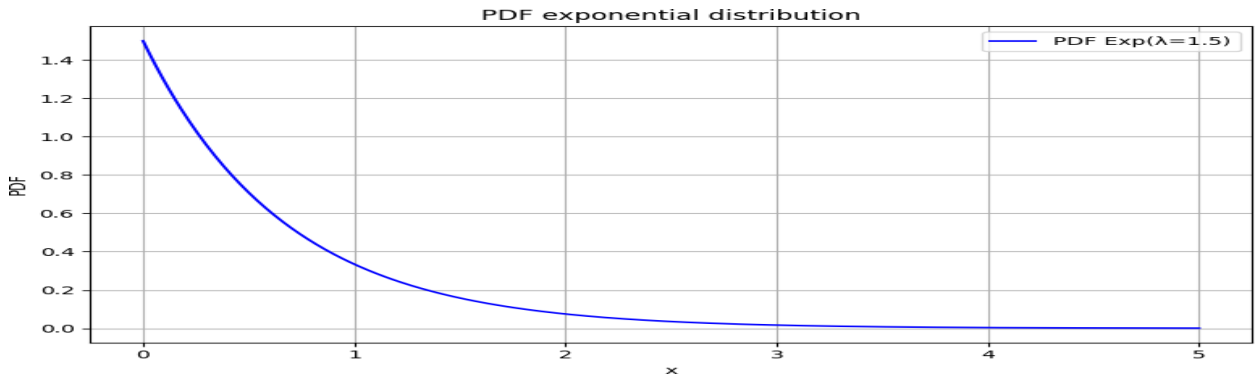
3.4. Extension to a non-homogenous, stochastic intensity parameter λ

After having developed a solid procedure to price path-dependent, lookback options using the risk-neutral framework proposed by Merton, we now extend our model to relax the assumption of constant intensity parameter. As we have stressed in the previous sections, the classical framework sees the counting process $N(t)$ governed by a Poisson distribution with parameter $\lambda > 0$: the *Probability distribution function* of the number of jumps occurred within time t can be expressed as:

$$\Pr[N(t) = n] = \frac{e^{-\lambda t}(\lambda t)^n}{n!}$$

Where the waiting periods between jumps $\tau_j - \tau_{j-1}$ are independent random variables distributed according to an exponential distribution with parameter λ. The important feature of this process is that each event occurs independently from each other, and there is no statistical relationship between different events (jumps) at different times. This distribution is decaying overtime, according to the following graph:

Figure 17



While this is a natural and intuitive assumption, it does not match the real dynamics of stock prices. The occurrence of a jump is often followed, empirically, by other jumps. A reasonable explanation for this phenomenon is information asymmetry among investors. As presented by Chen, Clements and Urquhart[16], there is substantial clustering in price and volatility jumps in high frequency, intraday data, which lasts between 2.5 and 6 hours on average. As investors are differently informed about firm's financial status or, in general, macro-finance news, there may be a mistiming of actions between investors, causing multiple jumps at different times after a certain news is released. If a company were to publish negative quarterly results, the most aware and informed investor would act immediately, causing an immediate price variation. Other investors may come behind, inducing a subsequent jump within the next hours or even days. Statistical tests to verify the need for an introduction of self-exciting behavior across jumps is beyond the purpose of the paper. Many tests have been carried out in other papers, as in the one cited above, proving the existence of a self-exciting behavior. Our goal is to assess how the pricing of exotic, path-dependent options changes when we relax the assumption of independence between jumps in a Merton framework. We will thus use a univariate Hawkes process – since we are not dealing with rainbow options, otherwise we would have needed a multivariate process, as counting process $N(t)$ for our model. We will choose the exponential specification for the kernel of the stochastic intensity parameter λ, as it easily managed by the two parameters α and β. Our intensity parameter will thus have the following specification:

[16] Modelling variance and price jump clustering using the Marked Hawkes Process; Chen, Clements and Urquhart, Journal of Financial Econometrics, 2023

$$\lambda(t) = \lambda(0) + \sum_{m=0}^{M} \varphi(t - t_m)$$

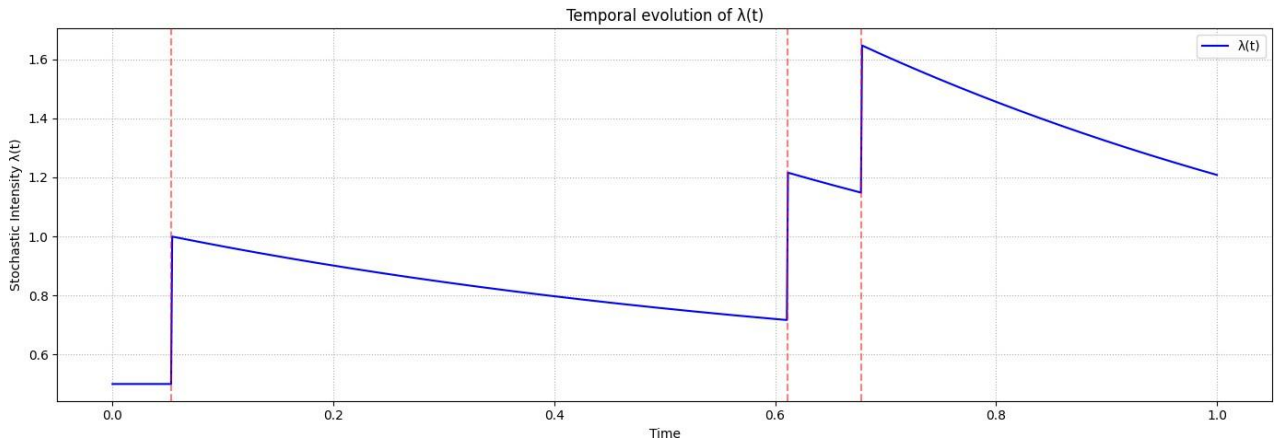
$$with \qquad \varphi(t - t_m) = \alpha e^{-\beta(t-ti)}$$

Where M is the total number of events occurred before time t, and t_m indicates the time of each event. The algorithm to simulate a path of this stochastic intensity parameter is the following:

```
def simulate_hawkes(mu, alpha, beta, T, dt=0.01):
    t = 0
    jumps = []
    lambda_t = mu
    history = []
    while t < T:
        u = np.random.uniform()
        s = -np.log(u) / lambda_t
        t += s
        lambda_bar = mu + np.sum(alpha * np.exp(-beta * (t - np.array(history))))
        d = np.random.uniform()
        if d <= lambda_bar / lambda_t:
            if t <= T:
                history.append(t)
                jumps.append(t)
            lambda_t = lambda_bar
        else:
            lambda_t = lambda_bar
    return np.array(jumps)
```

And an example of a simulated path for $\lambda(t)$ is in *Figure 18*

Figure 18



The red vertical lines represent the occurrence of a jump in the price (meaning that $\Delta N(t_j) = 1$ in those periods), adding up to a total of 3 jumps, $N(1) = 3$. Other than having a jump in the price, we see that we also have a jump in the intensity parameter itself: thanks to the specification of the kernel, the parameter λ has a self-exciting memory of past jumps, meaning that if a jump occurs at a given time, the probability of observing another jump in the following periods increases. The parameters α and β control the dynamics of this process: α indicates the strength of the jump in the intensity parameter in the occurrence of a jump induced by the intensity parameter at the previous date ($\Delta N(t_j) = 1$); high values for α indicate high jumps.

Parameter β indicates the strength of the decay once the jump has occurred. The function of λ as a function of time must be non-increasing in the non-jump periods to maintain the stationarity condition – otherwise the probability of jumps would keep on increasing. High values for β indicate that the decay is slower and takes more time.

3.4.1. Ogata's thinning algorithm

We must now get deeper in how we were able to simulate and generate this process for the stochastic intensity parameter. A well-known algorithm used is called the Ogata's Algorithm[17], introduced in 1981 as a thinning algorithm, and used in the previous code provided. The logic of this algorithm is to use a "*faster*" *homogenous* Poisson process, by imposing a higher intensity parameter lambda, and to later remove candidate jumps that are not aligned, in probabilistic terms, with the actual intensity parameter governed by the Hawkes Process. We will thus set an upper bound M, which moves over time as well in occurrence of jumps, and we will use this parameter to simulate the value $E = \tau_{i+1} - \tau_i$, that represents the waiting period between two jumps and can be sampled from an exponential distribution with intensity parameter M. The precise steps of the algorithm are the following[18]:

1. We start from initial time t, we set a small value $\varepsilon = 10^{-5}$, and we set our initial upper bound M to be equal to $M = \lambda(t + \varepsilon)$. This value will be kept constant until the next candidate time τ_i for a jump.
2. We sample the next candidate point $E = \tau_{i+1} - \tau_i \sim Exp(M)$ and we set the new time index equal to $t = t + E$. All the periods in between these two periods will not be considered explicitly by the algorithm. The important thing to keep in mind is that the M will remain constant in the interval, whilst our actual intensity parameter $\lambda(t)$ will slowly decay because of the presence of the coefficient β.
3. The new time t is our next candidate point to have a jump. We must now remember that this point has been generated using the *homogenous intensity parameter M,* which is indeed higher than the acutal intensity parameter we are trying to simulate. Thus, we need a criterion to decide whether or not this same candidate jump time *would have been generated by the lower intensity parameter λ(t) eventually*. The algorithm thus proceeds in generating a random point U^* from a Uniform distribution $U[0: M]$: the candidate point will be kept as a jump time only if this randomly generated value U^* is lower than the actual intensity parameter at the candidate time $\lambda(t)$. If $U^* < \lambda(t)$, then we accept the candidate point as a true jump, and we will have a jump there. Otherwise, the candidate point is rejected and there will be no jump at that time.
4. Regardless of that, by any candidate jump point t we will update our new maximum bound $M = \lambda(t + \varepsilon)$. Now the procedure starts over, generating a random value for the exponential and updating the time index accordingly.

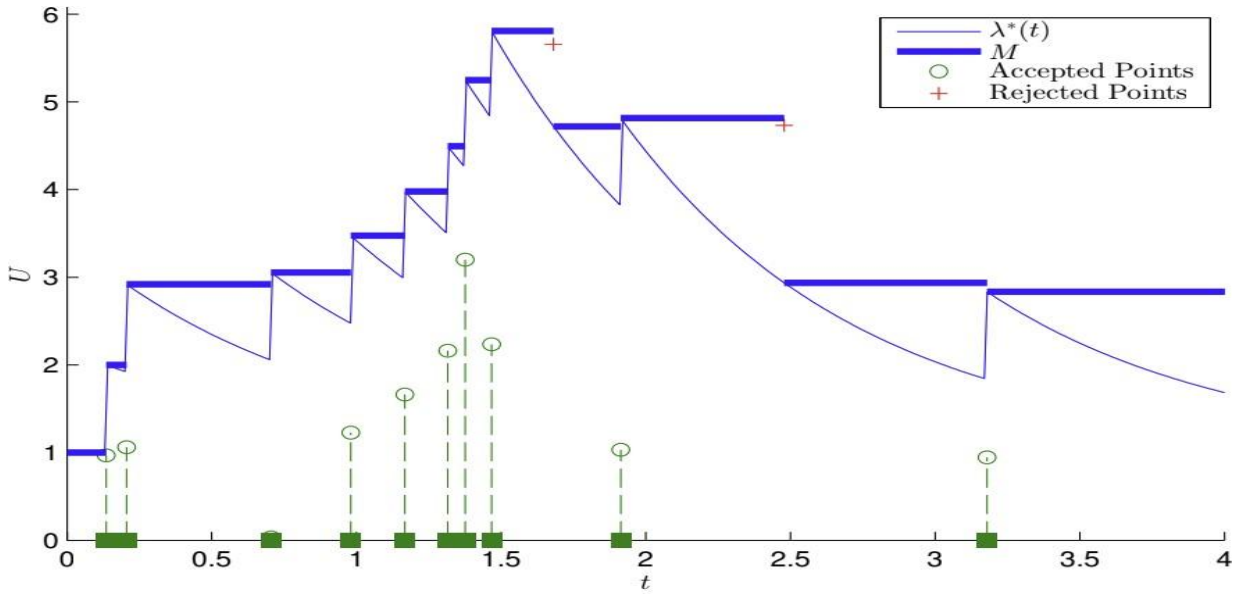
The final output of this algorithm is to generate a simulated path for λ overtime and also, accordingly, a realization of the Hawkes process $N(t)$. The jumps counted by the $N(t)$ must, always, coincide with the jumps *implied* by the path of λ: whenever we see a jump in the intensity parameter, it means that an actual jump for the prices process has occurred, and this is counted by $N(t)$.

Figure 19 helps in the visualization of the whole process:

[17] Hawkes processes: Simulation, Estimation and Validation; Patrick Laub, 2014; The University of Queensland, Australia

[18] Hawkes processes: Simulation, Estimation and Validation; Patrick Laub, 2014; The University of Queensland, Australia

Figure 19[19]



(b) Hawkes process

In the graph, the blue thick horizontal line indicates the upper bound M explained above; we see that it gets updated any time we draw a candidate jump point. The blue thin line represents the intensity parameter $\lambda(t)$; we see that it gets updated upward only in proximity of green void circles, whereas it keeps decaying in proximity of the red crosses. This is because some candidates points are accepted as jump times – with the green void circle, as we see that the sampled value from the Uniform distribution is smaller than intensity parameter, $U(t^*) < \lambda(t^*)$, where t^* is the accepted jump time; On the contrary, we can spot also two candidates points that have been rejected – with the red cross, because we see visually that the sampled Uniform lies above the respective intensity parameter, $U(t^{**}) > \lambda(t^{**})$, where t^{**} is the rejected jump time. The occurrence of a jump triggers in fact a spike in the intensity parameter, while the rejected points do not have any Impact on the parameter. Nevertheless, the latter have an impact on the M because, as stressed in the previous description, the upper bound gets updated iteratively.

3.4.2. Simulation under the $\mathbb{P}$ probability measure

Before getting into the pricing algorithms and simulations, it is crucial to find optimal parameters α^* and β^* of the Hawkes process to later be used in our Monte Carlo simulations. There have been some attempts in the literature to try and statistically calibrate these two parameters to have the best fit for the model. This calibration would be troublesome and expensive for our analysis, so we will avoid an actual calibration and deploy another, simpler, method to determine the correct parameters to use. We will create an algorithm to simulate the path for BAMI.IM in the historical time frame observed $[01-01-2020; 08-04-2025]$ by discretizing the *stochastic differential equation* proposed by Merton – using the Euler's discretization scheme presented in the introduction. We will allow the intensity parameter to vary stochastically following the dynamics of a Hawkes process with exponential kernel, and we will find the parameters α^* and β^* that make our simulated path as close as possible to the actual path of the Banco BPM stock.

A crucial notion that needs to be dealth with in this section is the probability measure we will employ to carry out this simulation. As we have stressed in previous sections, the risk-neutral measure $\mathbb{Q}$ is employed in

[19] Hawkes processes: Simulation, Estimation and Validation; Patrick Laub, 2014; The University of Queensland, Australia

option pricing because it guarantees that the discounted stock price is a martingale and no arbitrages are introduced in the market formed by risky security and the risk-free asset. Under this probabilistic measure, the coefficients introduced in Girsanov's Theorem (θ, φ) are set so that the drift of the price process is equal to 0, to reflect the fact that investors are assumed to be risk neutral and thus require no risk premium for their investments. In the framework we will develop now, there is no need for this assumption as we are not trying to build an arbitrage-free market to correctly price options, rather we want to define a stochastic dynamic for the stock price to accurately replicate the path that the stock followed within the real world, where investors are far from neutral to risk. Thus, it is justified to develop our framework within the physical probability measure $\mathbb{P}$, where the SDE by Merton is:

$$\frac{dS(t)}{S(t^-)} = \mu dt + \sigma dW(t) + \Delta J(t)$$

Where μ is the mean return of the stock price. Using Ito's lemma to solve this differential equation – as shown in the previous sections, we can rewrite the discretized expression of the evolution of $S(t)$ as:

$$\begin{cases} S(t) = S(t-\Delta t)e^{\left(\mu-\frac{1}{2}\sigma^2\right)\Delta t+\sigma\sqrt{\Delta t}*Z} & if\ \Delta N(t) = 0 \\ S(t) = S(t-\Delta t)e^{\left(\mu-\frac{1}{2}\sigma^2\right)\Delta t+\sigma\sqrt{\Delta t}*Z} * (Y_j * \Delta N(t)) & if\ \Delta N(t) = 1 \end{cases}$$

Where $N(t)$ is the Hawkes process.

We will thus try to find the optimal parameters α^* and β^* by making simulations with different pairs of values to see which pair of parameters gives us the most realistic sets of simulations.

The Python code to generate this algorithm and simulate this process is the following:

```
def simulate_hawkes_series(mu, alpha, beta, trading_days, dt, rng=None):
    if rng is None:
        rng = np.random.default_rng()
    T = len(trading_days)
    lambda_series = np.zeros(T)
    lambda_t = mu
    jump_times = []
    for t in range(T):
        lambda_series[t] = lambda_t
        if rng.random() < lambda_t * dt:
            jump_times.append(t)
            lambda_t += alpha
        else:
            lambda_t = mu + (lambda_t - mu) * np.exp(-beta * dt)
    return lambda_series, np.array(jump_times)

def simulate_jump_diffusion_hawkes(S0, avg_ret, sigma, a, b, mu, alpha, beta, trading_days, dt):
    lambda_series, jump_times = simulate_hawkes_series(mu, alpha, beta, trading_days, dt)
    N = len(trading_days)
    S = np.zeros(N)
    S[0] = S0
    k = np.exp(a + 0.5 * b**2) - 1
    for t in range(1, N):
        Z = np.random.normal()
        #drift = (r - lambda_series[t-1] * k - 0.5 * sigma**2) * dt      drift dotto la probabilità Q
        drift = (avg_ret - 0.5 * sigma**2) *dt
        jump = np.random.lognormal(a, b) if t in jump_times else 1.0
```

```
    S[t] = S[t-1] * np.exp(drift + sigma * np.sqrt(dt) * Z) * jump
return S, lambda_series, jump_times
```

Using the vector of parameters $(\alpha^*, \beta^*) = (0.001,20)$ produces simulations can be conducted to the empirical observed path for Banco BPM stock price. Two examples of paths are *Figure 20* and *Figure 21.*

Figure 20

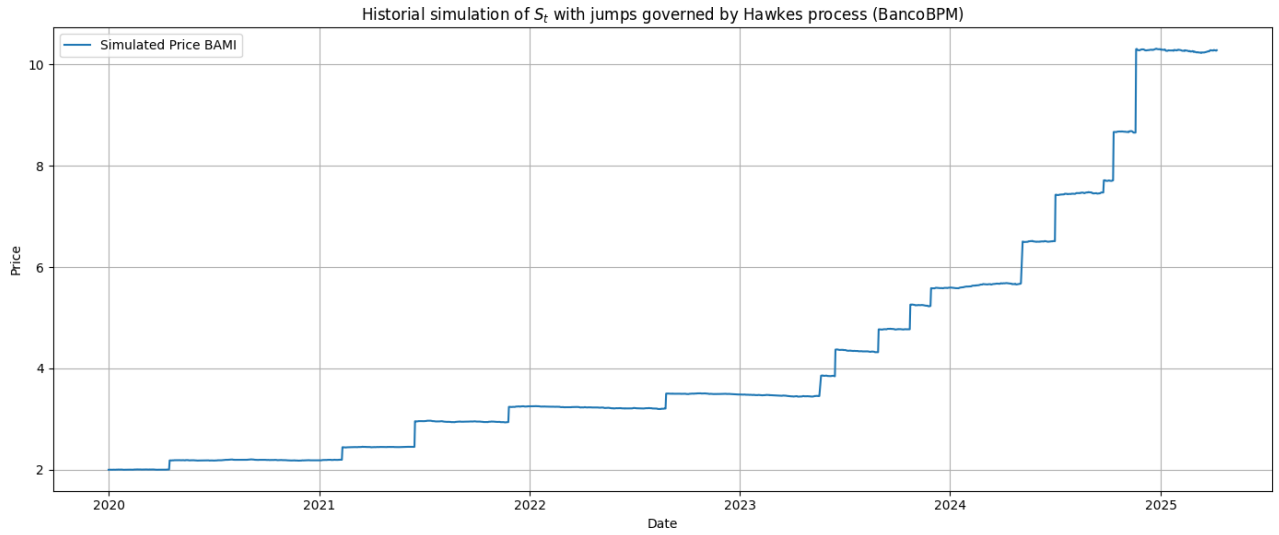
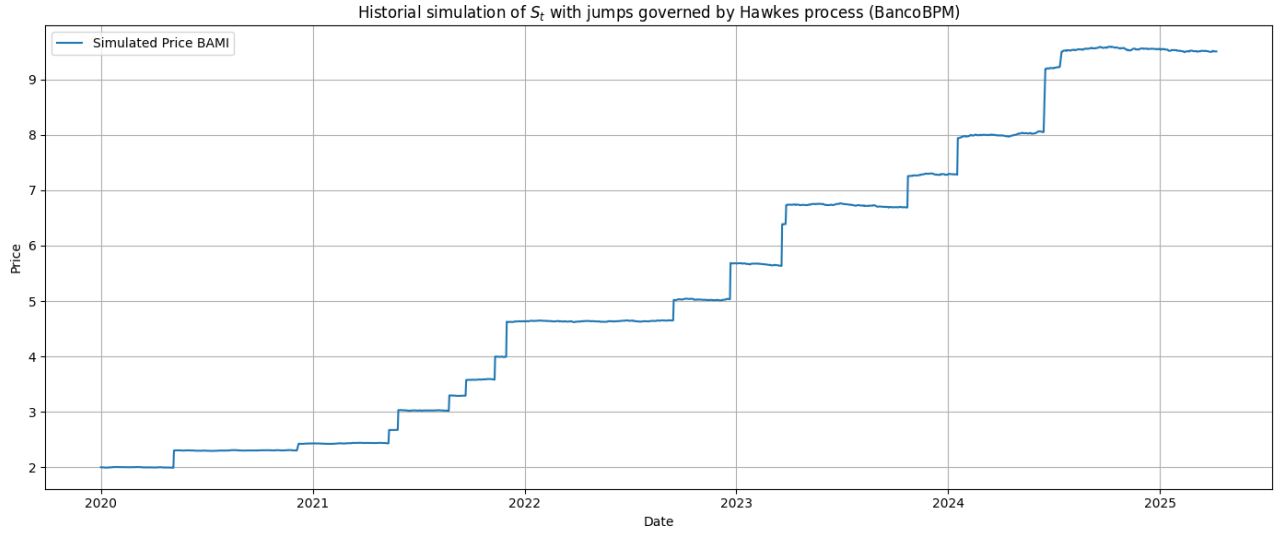


Figure 21



The parameters of α and β have been chosen in such a way to guarantee the stationarity of the process of $\lambda(t)$: $\frac{\alpha}{\beta} < 1$. All the optimal parameters already computed in previous sections for $(\sigma, \lambda_0, a, b)$ were kept for this simulation. The optimal lambda parameter employed in the analysis within classical Merton framework has been taken as the *initial* parameter intensity λ_0, so that, in case of no jumps, the stochastic intensity parameter is constant and equal to the initial homogenous intensity parameter. We stress again that the reason why the stock price simulated shows very little variability is to be conducted to the small value of σ^*, probably because all the empirical variability has been absorbed by the Jump component. To present a complete overview of the algorithm, plots for the stochastic intensity parameter and Hawkes process inherent in the simulated path in *Figure 20* are presented below:

Figure 22

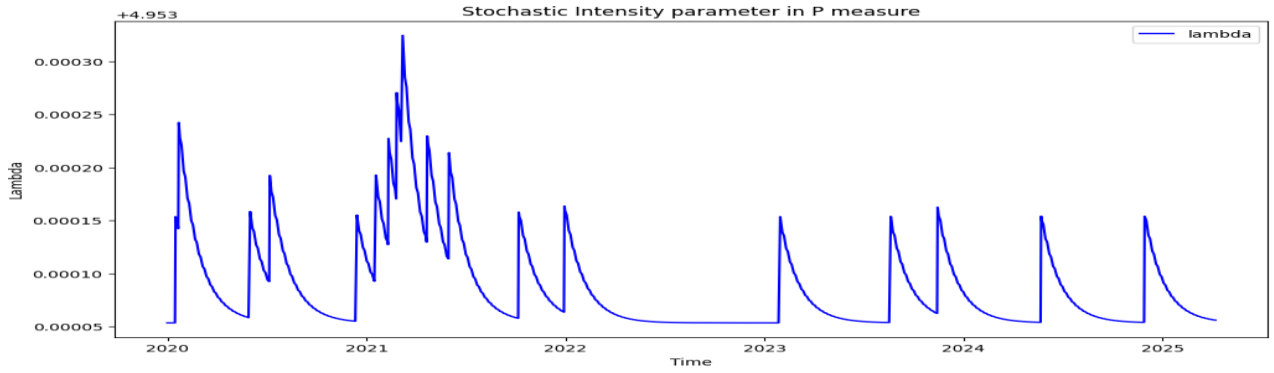
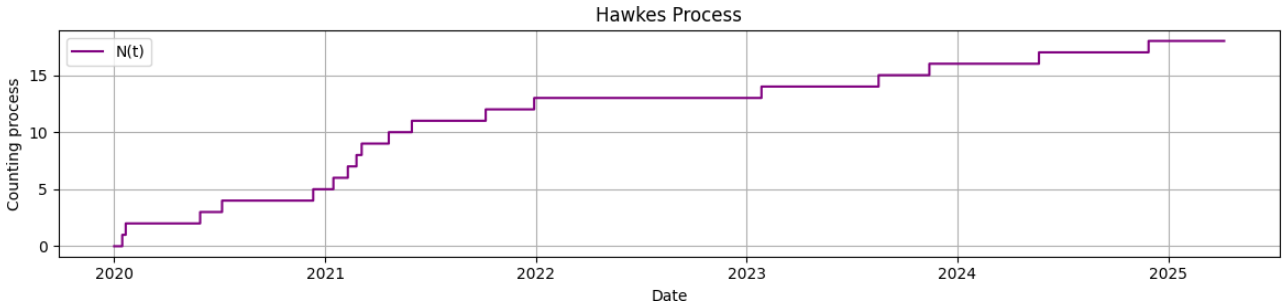


Figure 23



We see that the number of jumps presented in $N(t)$ is coherent the one simulated by the algorithm. Hence, we will employ these parameters for our Hawkes process used in the risk-neutral simulations to price our Lookback option:

$\boldsymbol{\alpha^*}$	**0.001**
$\boldsymbol{\beta^*}$	**20**

The "optimal" parameters found describe very little dynamics of the stochastic intensity parameter. As $\alpha^*$is very low, the impact of the occurrence of a jump to the intensity parameter will be very limited, close to 0. The high value of β^* indicates low persistence of the impact: once a jump occurs, it decays very quickly. The sum of these two components makes us think that the data may not be calling for a Hawkes process to model the counting process, and this can also be explained by the already very high value estimated for $\lambda(0) = 4.9$ by our optimization algorithm. We employ this as our *background* intensity, which is a very high starting point.

3.4.3. Pricing of Lookback Options using the counting process with stochastic intensity

Having found the right parameters to specify an exponential Hawkes process, we now engage in the final purpose of our analysis: the pricing of lookback options using a Merton model with stochastic intensity parameter. To do so, we will again employ Monte Carlo estimator based on M simulations of the option's underlying asset.

The first intuitive difference from the simulation run in the previous section is that we must now switch back to the risk neutral probability measure $\mathbb{Q}$. The no-arbitrage condition is essential to accurately price options, hence we will simulate from the discretized stochastic differential equation that describes the dynamics of the stock price S under the risk-neutral dynamics. The EMM used ensures the property that the discounted price S_t is a martingale in this probability measure, meaning that the stock, in this framework, must have a

drift equal to 0. To avoid arbitrage, we must employ a *pure jump martingale process under* $\mathbb{Q}$ in the stochastic differential equation, in the form of $CJ_t = J_t - (\mu_Y - 1)\lambda^{\mathbb{Q}}t$. The only source of uncertainty in this peculiar process is the jump exposure, but by subtracting the expected value of the jumps themselves we ensure that the drift of this process is equal to 0 under $\mathbb{Q}$ - thus implying the martingale property.

In fact, $E^{\mathbb{Q}}[\Delta J_t|\mathcal{F}_t] = E^{\mathbb{Q}}[(Y_j - 1)] * Q[occurence\ of\ a\ jump\ in\ dt] = (\mu_Y - 1) * \lambda^{\mathbb{Q}}dt$

Where $(\mu_Y - 1)$ indicates the average impact of a jump, given that a jump occurs in dt and $\lambda^{\mathbb{Q}}dt$ indicates the probability of jump occurring in dt, as shown in previous sections. Subtracting this measure from the jump process makes sure that the drift is *corrected* to account for the presence of jumps and still to ensure the no arbitrage condition. Finally

$$\frac{dS(t)}{S(t^-)} = rdt + \sigma dW^{\mathbb{Q}}(t) + \Delta CJ(t)$$

$$\frac{dS(t)}{S(t^-)} = rdt + \sigma dW^{\mathbb{Q}}(t) + \left(\Delta J(t) - (\mu_Y - 1)\lambda^{\mathbb{Q}}dt\right)$$

$$\frac{dS(t)}{S(t^-)} = \left(r - (\mu_Y - 1)\lambda^{\mathbb{Q}}\right)dt + \sigma dW^{\mathbb{Q}}(t) + \Delta J(t)$$

As previously indicated by page 20, the solutions, for each time t, of this stochastic differential equation can be found using Ito's lemma and are equal to:

$$S(T) = S(0)e^{\left(r-(\mu_y-1)\lambda_t-\frac{1}{2}\sigma^2\right)T+\sigma W^Q(T)}\prod_{j=1}^{N(T)} Y_j$$

Note that to derive this solution we have chosen a specific EMM. Out of the infinite possible solutions of (θ, φ) we have chosen the vector with $\varphi = 1$. Since we are trying to introduce a stochastic intensity parameter, we will have a vector of parameters with different intensities in different times. The main novelty with respect to the Merton model lies in this: in each cycle of the *for* loop, we must update the intensity parameter and take, at each time t, the parameter of that period according to the Hawkes process. If a jump occurs at time τ_i, we know that λ_{τ_i} will also experience a jump driven by the parameter α. So, the intensity parameter for that cycle will be higher and thus the reducing impact on the drift of S stronger: when a jump occurs, the time deterministic component will push the stock price down, with a subsequent gradual recovery to lower values of λ – driven by parameter β, assuming no further jumps push the intensity parameter even higher. In this new framework, the number of jumps $N(T)$ we should witness in every simulation should be higher compared to the Merton's case, as we have introduced a self-exciting behavior of jumps while keeping the initial intensity parameter fixed.

Using the following code, and the function defined above – having changed the code to get the drift under the $\mathbb{Q}$ probability $(r - (\mu_Y - 1)\lambda^{\mathbb{Q}})$ rather than μ under the $\mathbb{P}$ probability, we can generate $10{,}000$ simulations of the stock price.

```
S_full = np.zeros((M, N))

for m in range(M):

    S, lambda_series, jump_times = simulate_jump_diffusion_hawkes(

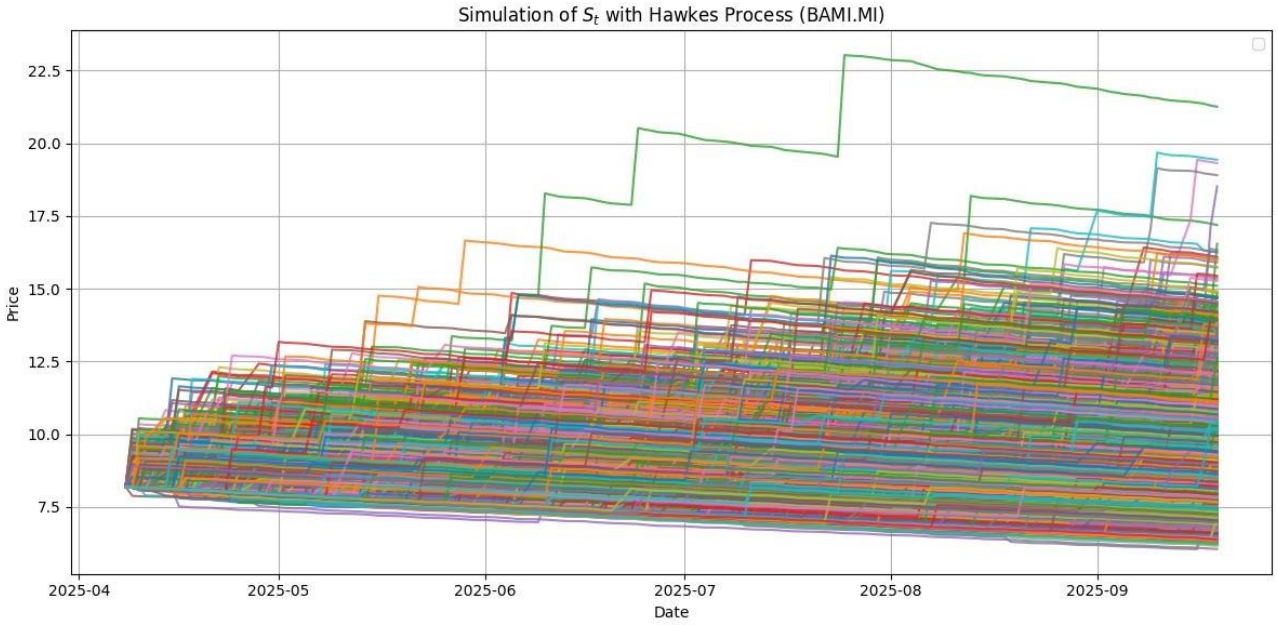
        S0, r, sigma, a, b, mu, alpha, beta, trading_days, dt

    )

    S_full[m,:] = S
```

It is very important to stress that for each of the 10,000 simulations carried out we call the defined function to get a vector of simulated lambdas. Since lambda is stochastic, to get a specific path of $\lambda(t)$ one must run a simulation. If we used the same vector of simulated values for all the 10,000, the paths for S would not be independent because they would all be using the same Hawkes process, with the same number of jumps per period, and the resulting Monte Carlo estimator would be biased. Hence, it is crucial to stress that we simulated both the jump times and the lambda vector for each of the M simulations, so that each has its own vector of stochastic intensity parameters. The simulated matrix is shown in *Figure 24*

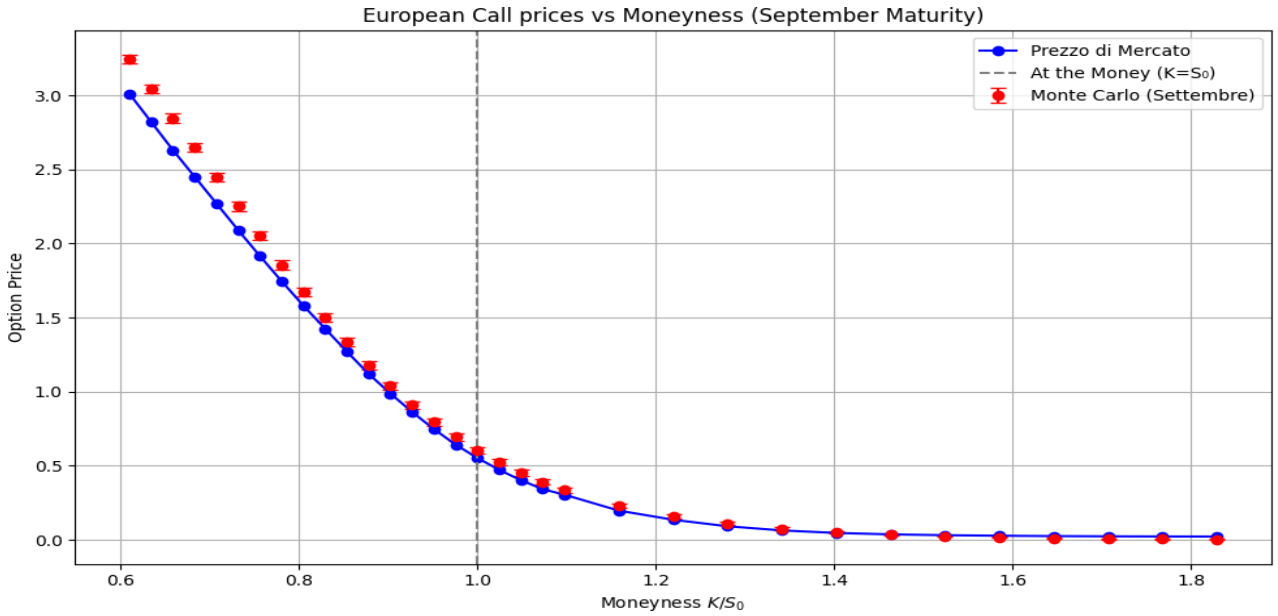
Figure 24



As in the previous simulation matrix, we have taken the September maturity. So, we are simulating the stock price for all the trading days between the 8^{th} of April and 20^{th} of September.

For starters, we use this Pandas Data frame of M simulations to run initial Monte Carlo simulations to price European calls for different values of spot moneyness K/S in this new mathematical framework. The code is the same as the one used in previous sections and will not be reported here. The plot is displayed in *Figure 25:*

Figure 25



We see that Monte Carlo estimator works quite well in fitting the market prices. The price for the ATM call is equal to 0.56. As in the previous section, we now compute the price of the Fixed lookback option ATM. The price for a fixed lookback option ATM on Banco BPM at date 8^{th} of Aprile 2025 is:

Price	1.02
Standard Error	0.0126
Lower Bound (95%)	0.997
Upper Bound (95%)	1.046

We first note that, despite the parameters α and β being indicative of a small difference in the stochastic intensity parameter path, the price of the lookback in this new framework is nevertheless higher than the price of the same product within the Merton model: The change represents a 7.3% increase in the derivative's price $(\frac{1.02}{0.95} - 1)$.

To reduce the variance of the Monte Carlo estimator, we also run the same code defined above to carry out a control variate estimator to price the lookback. Again, the OLS coefficient beta has been computed in such a way to guarantee that β_{OLS} and $\bar{X}_n$ are independent and that unbiasedness of the estimator remains untouched. The price for the fixed lookback using the control variate estimator, where the variable X is still the simulated prices for the European call for a jump diffusive asset – with $E[X]$ given by the closed-end formula to price these products, is:

Price	0.99
Standard Error	0.005
Lower Bound (95%)	0.98
Upper Bound (95%)	1.00

Also, in this pricing exercise we see that the price of the Fixed lookback is higher than the European call and this, as explained earlier, makes perfect sense. Furthermore, we also notice that the price of the Lookback from the control variate within the stochastic intensity parameter model is higher than the price computed previously with the homogenous intensity parameter. This is perfectly coherent with the framework used

since the intensity parameter, thanks to the addition of the kernel to include some memory, is greater than or equal to the homogenous intensity parameter used in the Merton framework. As we have set the optimal homogenous parameter lambda as $\lambda(0)$ in the formula:

$$\lambda(t) = \lambda(0) + \sum_{m=0}^{M} \varphi(t - t_m)$$

and $\lambda(0)$ is the same homogenous parameter used earlier, it is intuitive that $\lambda(t) > \lambda(0)$, $\forall t$. Since $\alpha, \beta \in \mathbb{R}^+$, the occurrence of a jump increases the parameter until this effect decays to 0 and $\lambda(t) = \lambda(0)$. The positive difference in prices between the different mathematical models used is to be conducted to the self-exciting nature of jumps introduced by the Hawkes process.

However, it is true that in the no-arbitrage framework imposed by the risk-neutral measure $\mathbb{Q}$ a higher parameter lambda has a negative effect on the drift of the simulated paths, but numerically this is more than compensated by the higher frequency of jumps. This increases the simulated payoff for the fixed lookback (as the $\max\{S_t\}$ increases), and, thus, the no-arbitrage price.

Finally, we see that also in this algorithm the control variate technique was successful in reducing the standard error of the estimator and making the estimate much more reliable.

3.4.4. Sensitivity analysis

Now that we have come up with a pricing model and both a punctual and interval estimate of the price for this exotic derivative, it is crucial now to carry out a brief analysis on the sensitivities of this option price. The Sensitivities are called *greeks* and represent the partial derivative of the model option price with respect to the different input variables used in the model. The main ones are:

$$\Delta = \frac{\delta C}{\delta S}, \gamma = \frac{\delta^2 C}{\delta S^2}, \upsilon = \frac{\delta C}{\delta \sigma}$$

Delta, the first partial derivative, measures the infinitesimal change in the option price because of an infinitesimal change of the underlying stock price. These measures are crucial in risk management or portfolio management view, as they allow dynamic management of the *PnL* of an options portfolio. Usually, traders set their portfolio to be *delta neutral* by a procedure called *delta hedging*. Essentially, the goal of the trader is to have a portfolio of options on a given underlying asset in such a way he is hedged against any fluctuation of the underlying price, because the market value of his portfolio is neutral with respect to this risk factors. For a trader to efficiently hedge his portfolio, he must have at least an estimate of the delta of the options he has in his portfolio here lies the importance of numerically computing the sensitivities of a given exotic options. The trader, with his pricing model, must estimate the sensitivities to accurately measure the risks and exposures embedded in each product, so that he dynamically manages and hedges his portfolio.

The main technique used to estimate a Greek is the Finite Difference Approximation:

$$\frac{\partial C}{\partial S} = \frac{\widehat{C_n}(S^* + h) - \widehat{C_n}(S^* - h)}{2h} \; as \; h \to 0^+$$

Where C_n is the Monte Carlo estimator for the option using n simulations. It is basically a procedure to discretize the infinitesimal variation and estimate the partial derivative of the function $C(S, \sigma, T-t, r)$ with respect to S in the point $S = S^*$, by computing the slope of the secant line between $C(S^* + h)$ and $C(S^* - h)$. This technique is called **Central-Difference Estimator (CDE)** and its logic is to estimate the derivative in a symmetric way around the point S^* in which you want to compute the derivative. Another valid alternative is the **Forward-Difference Estimator (FDE)**, with the following expression:

$$\frac{\partial C}{\partial S} = \frac{\widehat{C_n}(S^* + h) - \widehat{C_n}(S^*)}{h} \; as \; h \to 0^+$$

The logic of this estimator is very similar to the previous one, with the difference that the derivative is not estimated symmetrically around the point S^* but, rather, it only takes one step in one direction and computed the derivative as the slope of the secant line between the function in that step and S^* itself. Note that the FDE is essentially an estimation of the derivative of a function in a given point, and it tries to discretize the classic definition of a derivative:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

In our case we are dealing with a partial derivative, so we need to consider $f(x)$ as a multivariate function defined in more variables and h as a discrete variation in one of the variables over which the function is defined.

Both estimators are unfortunately biased and non-consistent, so the estimates produced will not be extremely reliable statistically. Nevertheless, we will use the CDE estimator as the bias of this estimator is positive but grows linearly in h, whereas the bias of FDE grows quadratically in h.

$$Bias\left[\widehat{\Delta_{CDE}}(n,h)\right] = constant * h$$

$$Bias\left[\widehat{\Delta_{FDE}}(n,h)\right] = constant * h^2$$

An important decision to make before estimating numerically these partial derivatives is the length of h. This is a very important choice as it represents a trade-off between Bias of the estimator and variance of the estimator. The variances of the two estimators are:

$$Var\left[\widehat{\Delta_{CDE}}(n,h)\right] = Var\left[\frac{\widehat{C_n}(S^* + h) - \widehat{C_n}(S^* - h)}{2h}\right] = \frac{1}{4}h^{-2}Var[\widehat{C_n}(S^* + h) - \widehat{C_n}(S^* - h)]$$

$$Var\left[\widehat{\Delta_{FDE}}(n,h)\right] = Var\left[\frac{\widehat{C_n}(S^* + h) - \widehat{C_n}(S^*)}{h}\right] = h^{-2}Var[\widehat{C_n}(S^* + h) - \widehat{C_n}(S^*)]$$

So, we see analytically that increasing h is negative for the bias, since we are discretizing for a higher interval, but it is positive for the variance, as now the estimate is less noisy. Choosing the optimal h^* is a result of a trade-off between the variance and the bias of the estimator, so it is to be kept in mind that the smallest h is not always the optimal. We should choose to have a low value for h and a high value for n so that the increased number of simulations can help reduce the variance of $\widehat{\Delta_{CDE}}(n,h)$.

We then want to compute the price for the lookback fixed option for these two levels of underlying stock price, $C_n(S^* + h)$ and $C_n(S^* - h)$. Essentially, for both levels of initial stock price we simulate the whole matrx of simulated paths using the same function defined above. The essential requisite for this simulation technique is that all other inputs (including $K, \sigma, T - t, r$) must be kept the same. We are conditioning the derivative only on the underlying stock price, making it, in fact, a partial derivative. The code used to compute the option prices are the following:

```
C_up = monte_carlo_lookback_fixed(S_full, r=r_sept, K = S0, T=T_sep)

C_up = C_up[0]

print(C_up)
```

Finally, we take the difference between the two option prices, and we divide this value by $2h$ using the formula presented above to compute a delta of $\hat{\Delta}_{DE}(10{,}000, 0.08) = 0.001$. The value estimated is reasonable if we look at the context of the derivative: we are evaluating the $\hat{\Delta}_{DE}$ of the exotic option at the beginning of his life and has time to maturity $T = 5/12$. Given that the payoff is computed as the difference between the maximum price achieved during the life of the option and the contractual strike, it is intuitive that an infinitesimal change in the underlying will not have much impact on the price of the option, given that 5 months are still left to maturity. From this framework we can also compute the gamma of this product, which refers to the second derivative of the option price with respect to underlying price $\frac{\delta^2 C}{\delta S^2}$

We thus want to estimate the second derivative of a function in a given point, which would be included in the main diagonal of the Hessian matrix for a multivariate function (like the function for the option price). The purpose would be to analyze the curvature of our function in the point, to assess whether it goes in the same direction of the $\hat{\Delta}_{DE}$ or if it goes in the opposite direction. A positive gamma means a positive curvature and convexity of the function: this is positive for the PnL of our portfolio as, intuitively, the Δ increases when S increases and it decreases when S decreases. The Δ, being the first derivative, only captures the linear, first order impact of a variation of S, and γ helps us in having a bigger picture.

The numerical approximation by finite differences of the second derivative can be computed as follows:

$$\gamma = \frac{(\widehat{C_n}(S^* + h))' - (\widehat{C_n}(S^*))'}{h} \; as \; h \to 0$$

Where $(\widehat{C_n}(S^* + h))'$ represents the first derivative of $\widehat{C_n}(S^* + h)$ with respect to S. In fact, the second derivative can be seen as the derivative of the first derivative. Then:

$$\gamma = \frac{\frac{\widehat{C_n}(S^* + h) - \widehat{C_n}(S^*)}{h} - \frac{\widehat{C_n}(S^*) - \widehat{C_n}(S^* - h)}{h}}{h} = \frac{\widehat{C_n}(S^* + h) + \widehat{C_n}(S^* - h) - 2\widehat{C_n}(S^*)}{h^2}$$

[20]

Using the following Python code we were able to compute the gamma in $S^* = 8.2$:

[20] Differenze finite: introduzione al metodo numerico e primi esempi

```
gamma_CDE = (C_up + C_down - 2*C) / h**2

print(gamma_CDE)
```

And it was equal to 22.8. This indicates a strong positive curvature of the option price with respect to the underlying, implying that the delta moves in the right direction for whatever change in the underlying price.

The last important sensitivity that needs to be computed is vega $\upsilon = \frac{\delta C}{\delta \sigma}$, the sensitivity of the option price with respect to the volatility parameter. The procedure to compute this partial derivative is the same as in the case for the Δ. The only difference is that we now keep S fixed and the infinitesimal variation h refers to the volatility parameter only:

$$\upsilon = \frac{\Delta C}{\Delta \sigma} = \frac{C_n(\sigma^* + h) - C_n(\sigma^* - h)}{2h}$$

We employ a similar algorithm as before by setting:

```
h = 0.08

sigma_up = sigma_opt + h

sigma_down = sigma_opt - h
```

And we compute a $\upsilon = 0.002$. As in the case for delta, the vega is very close to zero and the reasoning behind this value is similar to the one for the delta. To sum up, these are the sensitivities estimated:

Δ	**0.0007**
γ	**2.97**
υ	**0.002**

3.4.5. Results

In this concluding section we summarize all our results in terms of prices and greeks for the exotic derivative we have dealt with during the paper. We have priced a floating lookback exotic derivative in two different mathematical frameworks: the original Merton model with constant intensity parameter and an augmented Merton model with stochastic intensity parameter whose counting process is governed by a Hawkes process. Since the novelty is the pricing of the latter exotic derivative, we will report prices and greeks for this product for different sets of parameters

All prices reported consist of the point estimate using the Control-Variate Monte Carlo estimator

We will report the quantities for different pairs of parameters:

	Params	***Params_2***	***Params_3***	***Params_4***	***Params_5***
Price	**0.99**	**0.974**	**1.026**	**1.003**	**0.99**
Lower-bound	0.98	0.965	1.015	0.99	0.98
Upper-bound	1	0.98	1.037	1.014	1
Delta	**0.0007**	**0.001**	**0.002**	**0.0003**	**0.001**
Lower-bound	-0.227	-0.21	-0.23	-0.22	-0.22
Upper-bound	0.228	0.21	0.24	0.23	0.26
Gamma	**2.97**	**-0.4**	**2.6**	**-3.46**	**-5.6**
Lower-bound	-6.3	-9.3	-7.25	-13.1	-15
Upper-bound	12.2	8.4	12.45	6.2	3.8
Vega	**0.002**	**0.0008**	**0.002**	**0.0014**	**0.002**
Lower-bound	-0.34	-0.21	-0.23	-0.22	-0.22
Upper-bound	0.35	0.21	0.24	0.23	0.226

The values reported in the first column refer to the parameter highlighted in the other parts of the work, so $\lambda_0 = 4.95, \alpha = 0.001, \beta = 20, \sigma = 0.01, a = 0.103, b = 0.0448$.

λ_0	4.95
α	0.001
β	20
σ	0.01
a	0.103
b	0.0448

We are interested in assessing how prices and greeks would change for a variation in some relevant parameters. For instance, an important parameter of our pricing model is *baseline intensity* λ_0. Since this value has a relevant impact on the whole stochastic process of $\lambda(t)$, we recompute all the measures above for $\lambda^*_0 = \lambda_0 - 0.5 = 4.95 - 0.5 = 4.45$ and $\lambda^{**}_0 = \lambda_0 + 0.5 = 4.95 + 0.5 = 5.45$, while keeping all other parameters constant. The second column (***Params_2)*** contains all the metrics when baseline intensity is λ^*_0 and the third column (***Params_3)*** is filled with value for λ^{**}_0.

The price of the fixed lookback is higher when the baseline intensity parameter is higher, and it is lower when the respective baseline parameter is lower. This is perfectly coherent with what illustrated previously in the paper and, thus, reasonable: when λ_0 is lower, the whole process for the intensity parameter will be, on average, lower. In the occurrence of no jumps in the underlying, and thus no memory in the intensity parameter, the $\lambda(t)$ will converge to the lower value λ_0^*. Thus, the likelihood of having jumps which, on average, have a positive impact on the underlying – as shown in previous section by the computation of the expected value of the log-normal jump – will thus be lower, and so are the simulated payoffs of the derivative. Finally, also the price of the derivative will be lower. The same logic applies for the price of the derivative when baseline intensity is λ_0^{**} and explains why it is higher.

We now want to test how price and greeks would be by changing the parameters governing the Hawkes process: α and β. The fourth column (***Params_4***) includes all the relevant metrics computed by changing parameters in the following way: $\alpha^* = \alpha + 0.0001 = 0.001 + 0.0001 = 0.0011$ and $\beta^* = \beta - 5 = 20 - 5 = 15$. These adjustments are aimed in assessing how price and greeks would change if the jumps on the underlying had a stronger, upward impact on the intensity parameter. A higher α^* implies a bigger jump in the intensity in occurrence of a jump, and a lower β^* indicates a lower decay of the intensity. The fifth column (***Params_5***) includes all metrics with $\alpha^{**} = \alpha - 0.0001 = 0.001 - 0.0001 = 0.0009$ and $\beta^{**} = \beta + 5 = 20 + 5 = 25$. The logic is now assessing the relevant metrics if the jumps on the underlying had a weaker impact on the intensity parameter. Note that the values for the fourth and fifth columns are computed using the original baseline intensity parameter λ_0. Overall, we could argue that variation in parameters of the Hawkes process do not imply a significant change in any of the computation.

4. Conclusions

In this thesis, a pricing model for fixed lookback options was developed based on an augmented Merton jump-diffusion framework, enriched with a stochastic intensity governed by a Hawkes Process. The intention was to augment the original model proposed by Merton in 1973 by introducing a self-exciting behavior in jumps which has been studied and observed empirically.

Information asymmetry among investors has been signaled as a possible cause of this clustering of jumps, but nevertheless it is plausible that more jumps are triggered by the occurrence of a single jump. Because of high levels of macro uncertainty, by the beginning of April the world has witnessed large fluctuations in the major global stock indices. We can consider the historical daily $+9.5\%$ of the $S\&P500$ registered on the $9^{th}$of April a jump, as the size of this increase is so massive in a short period of time that, in mathematical terms, it is hardly driven by the continuous risk exposure of a Geometric Brownian Motion. This jump does not stand alone, as in the previous days the same index also had extreme daily fluctuations, -6.65% on the 3^{rd} of April and -6% on the $4^{th}$of the same month. These values are well below the average daily return of the index and levels of standard deviation. So, we aimed at quantifying how strong this self-exciting behavior really is in a relatively "*jumpy*" stock like Banco BPM ($BAMI.MI$) by computing the parameters of the exponential kernel of the Hawkes process and we also aimed at quantifying how much of a difference this introduction could trigger on a path-dependent exotic option.

The results, in this sense, are mixed. The parameters for Hawkes process α and β imply a very stationary stochastic process, which barely parts from its background intensity parameter. The low value found for α implies a low upward force in case of a price jump, and the high value β implies a strong decaying motion after a price jump. This would imply that there is actual no need to use a Hawkes process as a counting process and introducing stochastic features to the intensity parameter, if the parameter itself moves very little from the homogenous parameter previously used in the Poisson process within the Merton model. Naturally, more statistically sophisticated methods and procedures could have been used to find these optimal parameters, but a full calibration would have been troublesome and would have shifted the focus of the thesis.

Nevertheless, the difference in the prices computed by the two mathematical models is somewhat different. The traditional Merton model prices the *At-the-money* fixed lookback at 0.964€, whereas the Merton model with this stochastic intensity parameter prices the same product at 1.02€, implying a difference of 6%. This difference is not huge, but it is still not insignificant.

To complete the analysis also the greeks computed with the augmented Merton framework has been computed by finite difference methods. Despite being delta neutral, this exotic derivative priced using Hawkes process is highly positive in gamma, and thus a very positive curvature at the initial price of Banco BPM. It is also vega neutral, implying that a change in the volatility parameter would not affect the price of the option.

In conclusion, we would argue that, despite the clustering of jumps in stock prices is empirically observed in the market, because of the low-impact parameters computed for the Hawkes process, the observed price data for Banco BPM between $01-01-2020$ and $08-01-2025$ do not appear to support the need for incorporating a Hawkes process as the jump intensity component within the Merton model. Potential improvements for the analysis include a deeper search for a global minimum for the loss function minimization problem and a solid and robust statistical procedure to calibrate parameters α and β.

Bibliography

Option prices when Underlying stocks returns are discontinuous; Robert C. Merton; Journal of Financial Economics; January 1976

Topics in Quantitative Finance, lecture notes for the graduate course Applied numerical Finance; Anna Battauz; Bocconi University; 2024-2025, 83 – 99

Hawkes Processes: Simulation, Estimation, and Validation; Patrick Laub, supervised by Phil Pollet; The University of Queensland, Australia; 2014; 9-22

Implementing Derivatives Models; Les Clewlow and Chris Strickland; 1998

Option pricing and hedging in jump diffusion models; Yu Zhou; Uppsala Universitet; 2010

Hawkes Processes in finance; Emmanuel Bacry, Iacopo Mastromatteo, Jean-Francois Muzy; Ecole Polytechnique, Université de Corse; 2015

Control Variates Techniques for Monte Carlo simulation; Roberto Zvechman; Operations Research Department; 2003

Exponential Kernels with Latency in Hawkes Processes: Application in Finance; Marcos Costa Santos Carreira; January 2021

Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-constrained Optimization; Ciyou Zhu, Richard Byrd, Jorge Nocedal; Northwestern University, University of Colorado at Boulder; 1-3

Numerical Methods II; Mike Giles; Oxford University Mathematical Institute

Monte Carlo – 1st Class; Anna Battauz; Bocconi University

Variance Reduction Techniques; Anna Battauz; Bocconi University

Estimating the sensitivities; Anna Battauz; Bocconi University

Modelling price and variance jump clustering using the Marked Hawkes Process; Chen, J., Clements; Oxford University Press; Journal of Financial Econometrics; 2023

Simulation, Estimation and applications of Hawkes Processes; Katarzyna Obral; University of Minnesota; June, 2016; 3-29

Simulation Efficiency and an Introduction to Variance Reduction methods; Martin Haugh; Colombia University; 2017

Acheroui, I., Bacry, E., Gaïffas, S., & Poulenard, A. (2020). tick.hawkes — tick 0.6.0 documentation. École Polytechnique. Retrieved from tick.hawkes — tick 0.6.0 documentation

Amit, H. (2020, August 26). A brief introduction to Hawkes processes. Medium. Retrieved from A Brief Introduction to Hawkes Processes | by Hey Amit | We Talk Data | Medium

Chen, J. (2023). Monte Carlo simulation: What it is, how it works, history, 4 key steps. Investopedia. Retrieved from Monte Carlo Simulation: What It Is, How It Works, History, 4 Key Steps

Borsa Italiana. (n.d.). Opzioni su azioni. Retrieved April 8, 2025, from https://www.borsaitaliana.it

TradingView. (n.d.). Curva dei rendimenti dei titoli di Stato dell'Italia: IT10Y. Retrieved April 8, 2025, from https://it.tradingview.com/symbols/TVC-IT10Y/yield-curve/

AI Post. (2023). Ottimizzazione locale e globale: trovare la soluzione migliore in grandi set di dati. Retrieved from https://blog-ai.it/ottimizzazione-locale-globale

HashDork. (2023). Cos'è l'algoritmo di ottimizzazione: una guida completa. Retrieved from Cos'è l'algoritmo di ottimizzazione: una guida completa

HashDork. (2023). Esempi di algoritmo di discesa del gradiente. Retrieved from bing.com/ck/a?!&&p=676c876358d5963008eb14d617bd7c0c6b7c3d9abeaebeb773a52a0240be34 a1JmltdHM9MTc0OTUxMzYwMA&ptn=3&ver=2&hsh=4&fclid=324cb3bb-1ea9-6480-3428-a60a1f746526&psq=Esempi+di+algoritmo+di+discesa+del+gradiente+-+HashDork&u=a1aHR0cHM6Ly93d3cuaGFzaGRvcmsuY29tL2l0L2VzZW1waW8tZGktYWxnb3Jp dG1vLWRpLWRpc2Nlc2EtZGVsLWdyYWRpZW50ZS8&ntb=1

HashDork. (2023). Una breve introduzione all'algoritmo di ottimizzazione BFGS. Retrieved from https://it.python-3.com/?p=4397

AI Post. (2023). BFGS in poche parole: un'introduzione ai metodi quasi-Newton. Retrieved from https://ichi.pro/it/bfgs-in-poche-parole-un-introduzione-ai-metodi-quasi-newton-37043576956164

Bing. (n.d.). Video. Retrieved 2025, from Understanding scipy.minimize part 1: The BFGS algorithm

Barndorff-Nielsen, O. E., & Shephard, N. (2006). Econometrics of testing for jumps in financial economics. Journal of Econometrics, 131(1–2), 377–398.

Investing.com. (n.d.). BTP 5 anni prezzi storici. Retrieved June 10, 2025, from https://it.investing.com/rates-bonds/italy-5-year-bond-yield-historical-data

Investing.com. (n.d.). Storico Banco BPM | Dati storici. Retrieved 2025, from Storico Banco Bpm | Dati storici - Investing.com

Numerical Lab. (2023). Differenze finite: introduzione al metodo numerico e primi esempi. Retrieved from Differenze finite: introduzione al metodo numerico e primi esempi

Pricing of Lookback options using a jump diffusive asset with a non-homogenous stochastic intensity parameter – Application on stock index S&P 500

Stockholm School of Economics, MSc in Finance
Supervised by professor Tobias Sichert

Table of Contents

Introduction **1**

Theoretical review of key concepts **3**

Jump diffusion model **3**

Counting process and *stochastic differential* equation 3

Comparison of first and second moment with Black-Scholes asset 7

No-Arbitrage Analysis 9

Hawkes Processes **11**

Stationarity and Kernel of the non-homogenous counting process 11

Exponential and power-law kernel 12

Monte Carlo Technique **14**

General Idea and pricing of European call 14

Euler discretization 15

Statistical properties of Monte Carlo estimator 16

Numerical Application **18**

Calibration of plain vanilla European call closed-end formula to market prices **19**

L-BFGS-B and Gradient Descent algorithms 24

Pricing European Calls using Merton Model **27**

Generation of matrix of simulated jump-diffusive prices 28

Pricing Fixed Lookback Calls using Merton Model **30**

Pure Monte Carlo Estimator 30

Control Variate estimator to reduce variance 31

Extension to a non-homogenous, stochastic intensity parameter **36**

Ogata's thinning algorithm 38

Simulation in the $\mathbb{P}$ probability measure 39

Pricing of Lookback option using counting process with stochastic intensity 46

Sensitivity Analysis 51

Results **55**

Volatility Smile fitting **57**

Conclusions **67**

Bibliography **68**

1. Introduction

In 1963, the paper "*The pricing of options and corporate liabilities"* published by Fischer Black and Myron Samuel changed the shapes of the quantitative finance world by introducing the first closed-end formula to price European call and put options. This led to the renowned Black Scholes model which, despite many rigid assumptions it is built upon, still is an important benchmark for traders and financial engineers. One of the main assumptions of the model is for the diffusive asset $S(t)$ to be modelled as a Geometric Brownian Motion with the following Stochastic Differential Equation:

$$dS(t) = S(t)\big(\mu * dt + \sigma * dW(t)\big)$$

Where $dW(t)$ is a standard Brownian motion, $S(t)$ is the risky asset, μ is the drift under the $\mathbb{P}$ probability, σ is the diffusion parameter.
Coupled with the risk-free asset $B(t)$, the diffusive stock $S(t)$ creates a complete and free of arbitrage market.
Out of the many simplistic assumptions of the model, the specification of the stochastic process for $S(t)$ is the one which has been most frequently challenged by practitioners and academics. One of the problems frequently reported regards the path followed by $S(t)$: since the stock $S(t)$ is modelled in a continuous trajectory, the possibility of massive jumps, either upward or downward, which have been seen in empirical data, has been minimized.
In this regard, Robert Cox Merton published in 1976 a paper titled 'Option prices when underlying stocks' returns are discontinuous' in which he tried to tackle this limit.
In his model, still comprehensive of a risk-free asset $B(t)$ and a diffusive asset $S(t)$, the underlying's stochastic differential equation also includes an extra component to include the possibility of sudden jumps in process. These jumps are modelled as a compound Poisson process, with a constant intensity parameter λ, and they introduced potential discontinuities in the path of the stock.
This is crucial, as now we have introduced a probability of a non-local change in price which has found strong evidence in data - think of the many outliers found in the empirical distribution of stock price compared to the log-normal kernel, or of the massive downward jumps experienced in a breakout of a bad news.
The more accurately we model the stock price dynamics in a realistic way, the more efficient and precise the pricing of given exotic options will be.
The purpose of this analysis is thus to go further in the modelling of the stock dynamics by specifying a slightly different probability distribution for the likelihood of jump occurrence. In the original Merton framework, the Poisson distribution specified for the jumps had a single, constant intensity parameter lambda. Despite being reasonable, data have shown in the last years that a clustering of jumps has formed: the occurrence of a jump makes it more likely for another jump to occur in the future, thus changing the intensity parameter. To introduce a self-exciting behavior in the intensity parameter, we are going to specify the counting process of jumps as a Hawkes process. This process shows this precise property: the non-homogenous intensity parameter changes over time.
After developing the mathematical model, we are going to use it to price path-dependent options traded in over-the-counter market. The growth of the market for exotic options has been extraordinary in recent years, as they allow financial institutions to tailor the financial derivative to their needs and allow much more flexibility on the conditions and computations of the final payout. Lookback options are a popular example of path-dependent options, as the strike price is defined as an arithmetic mean of the stock price itself observed in different points in time during the life of the option. The Pricing of such products requires a solid mathematical framework to perform payout simulations, as they do not present a closed formula yet.
In the first section of the work, we are going to review all the mathematical and theoretical tools necessary for the analysis. The original Merton framework will be discussed, highlighting the derivation of the price $S(t)$ using Ito's lemma and a non-arbitrage analysis of the model. Then, the main properties of Hawkes processes will be reviewed, analyzing the first moments and the different kernels proposed by the literature for the model. Finally, a review of Monte Carlo simulations will be carried out, proposing an example of a simple algorithm to perform simulations to price European options. At the end of the numerical section, we will perform the pricing of lookback options with a self-exciting intensity parameter and compute some sensitivities. In the last chapter of the thesis, we will carry out an analysis of the implied volatility skew generated by the model when pricing plan vanilla options and compare it to the market implied volatility skew observed in a given period.

2. Theoretical review of key concepts

2.1 Jump Diffusion Model

2.1.1 Counting Process and stochastic differential equation

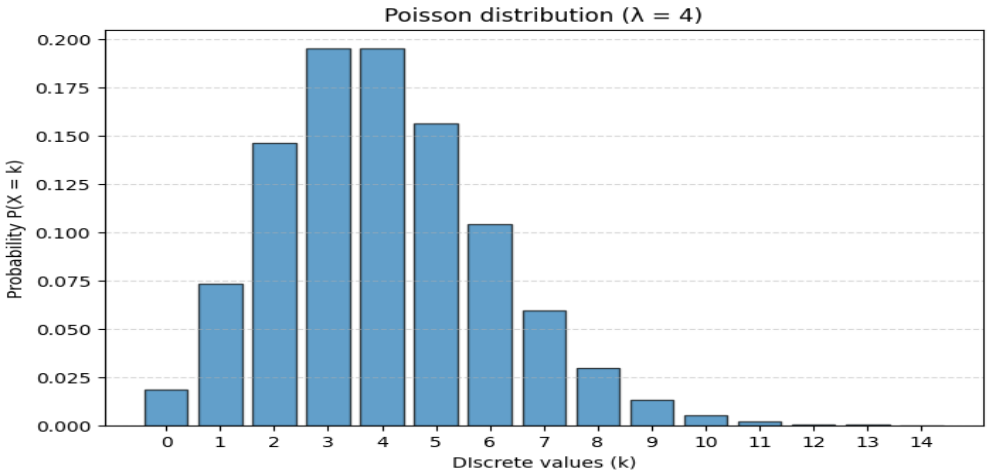
In 1973 Merton proposed a different *stochastic differential equation* for the risky asset in order for it to include sudden, non-local changes in its price trajectory.
This new framework represented an important step in mathematical modeling, as it allowed the path for the stock price to be much more realistic and, thus, more effective for pricing purposes. The *SDE* proposed by Merton has the following structure.

$$dS(t) = S(t-)(\mu\, dt + \sigma dW(t) + dJ(t))$$

Where $S(t)$ indicates the stock price at time t, $S(t-)$ indicates the stock price at time $t^- = t - dt$, so the price one infinitesimal instanct before time t. $W(t)$ is a Standard Brownian motion, μ and σ are the drift and diffusion parameter, respectively, and $J(t)$ is the jump process introduced by Merton.
Figure 1 shows some trajectories created using the *SDE* above, to include the presence of visible discontinuities in the stock price path.
A jump process is a different type of stochastic process that analyzes the likelihood of witnessing a random number of jumps in a time period. A very simple case of a jump process is a Poisson process, that models the discrete number of jumps to occur in a time interval. The problem of using a Poisson process for this purpose is that you are only modelling the number of jumps, and not the size of the latter. To overcome this problem, Merton chose to use a **Compound Poisson process** as jump process for the model. So, the process $J(t)$ can be described as:

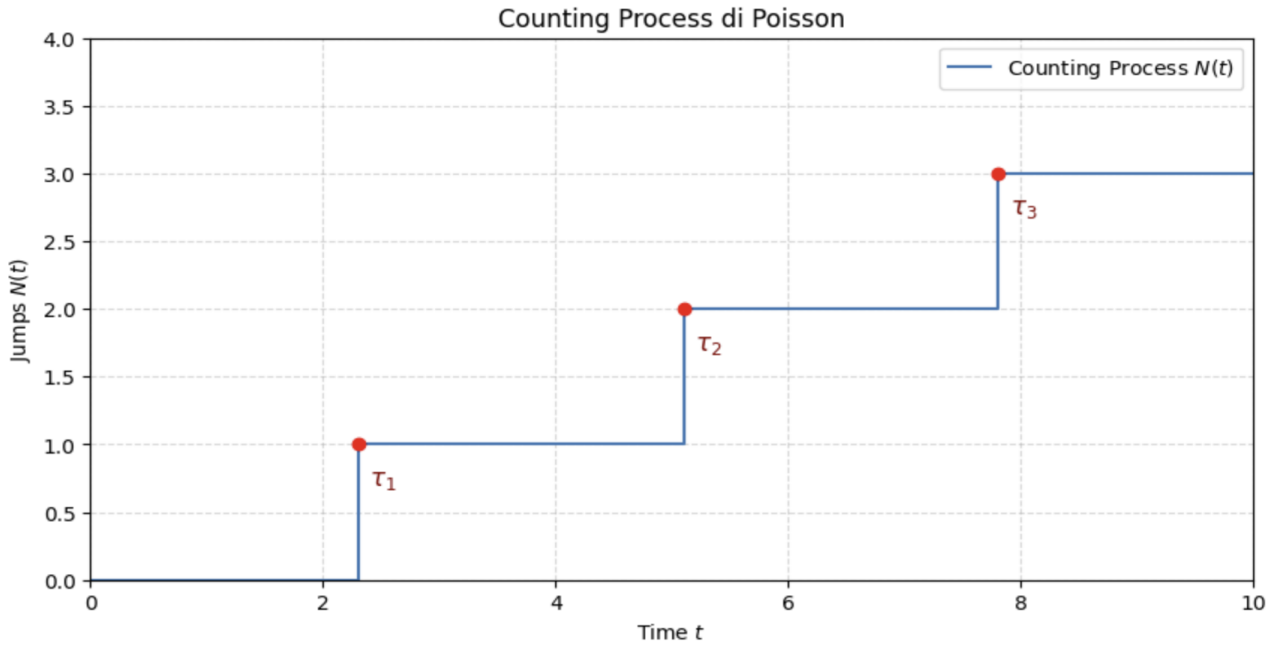
$$J(t) = \sum_{j=1}^{N(t)} (Y_j - 1)$$

$N(t)$ is defined as a counting process, which is a vital component of the process $J(t)$ as it models the number of jumps to occur within the time interval $[0, t]$. To understand the counting process, we can look at *Figure 1*:
Figure 1



This distribution has been created using a Poisson distribution, and it shows the probabilities estimated for each number of discrete jumps to occur in a given time period.
Aside from the overall number of jumps in a period, we also may be interested in assessing the times in which jumps occur τ_i over the period. A visualization in this sense may come from the following graph:

Figure 2



In *Figure 2*, we see a path for a counting process simulated over a time interval: in this simulation, we see that 3 jumps have occurred, and $\tau_1 = 2.2, \tau_2 = 4.6, \tau_3 = 7.9$.
This is the visualization of the counting process, and we can see that it is right-continuous, since the value $N(t)$ includes the jump occurred in time t. The arrival of a jump at a random time τ_i generates a discontinuity from the left in the counting process as, again, it gets included in the value for $N(\tau_i)$. Furthermore, since the jump time is random and completely independend from all the other sources of randomness of the model - $W(t)$ and Y_i – there is no chance at time $t^- = t - dt$ to unveil the occurence of a jump at time t.
To better understand how this counting process has been generated, we now dwell on the statistical properties of the underlying process used to model the occurrence of the jumps: the Poisson process.
The *probability distribution function (PDF)* for this discrete Poisson random variable is the following:

$$P[N(t) = n] = e^{-\lambda t} * \frac{(\lambda t)^n}{n!}, \forall n \in N$$

The graphical representation for this *PDF* is the one shown in *Graph 1*, where we selected an arbitrarily intensity parameter λ equal to 4. Another useful property of this distribution for the counting process is that it can also be shown that:

$$P\left[\tau_{j+1} - \tau_j \leq t\right] = 1 - e^{\lambda t}$$

Which describes the waiting times between jumps, as this is also a random variable. The key parameter in this discrete distribution is $\boldsymbol{\lambda}$. The intensity parameter under the pyhsical probability $\mathbb{P}$ measures the probability of occurrence of a jump between the infinitesimal time interval $[t^-, t]$ – thus, it can be rewritten as $P\left[\Delta N(t) = 1 \; in \; [t^-, t]\right] = \lambda$. The higher this intensity parameter, the higher the likelihood of witnessing a jump in our time interval. Since the parameter λ refers to the infinitesimal time interval dt, we need to scale up the time factor when we extend the parameter to the whole interval $[0, t]$. So, we have that the first two moments of this process are characterized as follows:

$$E[N(t)] = \lambda t$$
$$Var[N(t)] = \lambda t$$

The counting process $N(t)$ works well to model the number, or the frequencies of jumps, but in order to model the size of the random jumps Merton used a series of different variables Y_j. These random variables $Y_j, \forall j: 1 \dots N(t)$ are positive, IID random variables coming from the same specified distribution. In this framework, $W(t), N(t)$ and Y_j

are assumed to be independent from each other under the physical probability measure $\mathbb{P}$. This specification makes the process $J(t)$ a Compound Poisson process, as the frequencies and number of jumps come from a Poisson distribution, but the sizes of jumps do not. In our specific case, the size of the j_{th} jump is $(Y_j - 1)$.

For instance, let us consider the case where $N(T) = 3$ – so we would have 3 jumps. The compound Poisson process would thus be $J(T) = (Y_1 - 1) + (Y_2 - 1) + (Y_3 - 1)$, where T is the maturity time.
After having analyzed the extra component of Merton's *stochastic differential equation*, we now focus on the price path for $S(t)$ in this new framework.

Focusing on the path price for the stock $S(t)$ means solving the *SDE* proposed for the asset, i.e. it means to get the right expression for the stock price S at each time t assuming the infinitesimal differential $dS(t)$ follows that equation indicated above. To do that, we use Ito' lemma to derive the solution for $S(t)$ from the indirect solution for $\ln(S(t))$.
Ito's lemma is a powerful tool in stochastic calculus that, given a target process $Y = f(X)$, and a know process X, allows us to derive the *differential equation* for $dY(t)$, so for a linear function of the original know process $dX(t)$.

(Ito's lemma). Given f: R x R → R continuously differentiable with respect to the first argument and twice continuously differentiable with respect to the second argument. Then,

$$df\big(t,S(t)\big) = \frac{df\big(t,S(t^-)\big)}{dt} + \frac{df\big(t,S(t^-)\big)}{dS} S(t^-)\big(\mu dt + \sigma dW(t)\big) + \frac{1}{2}\frac{d^2 f\big(t,S(t^-)\big)}{dS^2} S(t^-)^2 \sigma^2 dt + \Big(f\big(t,S(t)\big) - f\big(t,S(t^-)\big)\Big) * \Delta N(t)$$

We know use this lemma to derive the process for the log price $\ln(S(t))$ – so, our $f = \ln(x)$, and, once we find the solutions for the log price, we exponentiate to get the solutions for the price $S(t)$.
Using the expression above, we can write:

$$dln\big(S(t)\big) = \left(\mu - \frac{\sigma^2}{2}\right) dt + \sigma dW(t) + \Big(\ln\big(\mathrm{S(t)}\big) - \ln\big(\mathrm{S(t^-)}\big)\Big) * \Delta N(t)$$

The last term is the most interesting to be dealt with, since it heavily depends on whether, at a given time t, a jump occurs or not. If it does occur, and so $\Delta N(t) = 1$, the term becomes $\ln\big(S(t^-) * Y_j\big) - \ln\big(S(t^-)\big)$, which in term can be written as $\ln\left(\frac{S(t^-)*Y_j}{S(t^-)}\right)$. If a jump does not occur, and so $\Delta N(t) = 0$, the whole term will be equal to 0. Simply speaking, the only relevant cases where the last term is going to impact on the solutions of $S(t)$ are when a jump is going to occur. So, we can rewrite our expression as:

$$dln\big(S(t)\big) = \left(\mu - \frac{\sigma^2}{2}\right) dt + \sigma dW(t) + \big(\ln(\mathrm{Y_j})\big) * \Delta N(t)$$

Taking the integral on both sides, to sum all the infinitesimal changes, we get:

$$\int_0^T dln\big(S(t)\big) = \int_0^T \left(\mu - \frac{1}{2}\sigma^2\right) dt + \int_0^T \sigma dW(t) + \int_0^T \ln\big(Y_j\big) * \Delta N(t)$$

$$\ln\left(\frac{S(T)}{S(0)}\right) = \left(\mu - \frac{1}{2}\sigma^2\right) T + \sigma dW(T) + \sum_{j=0}^{N(T)} \ln(Y_j)$$

The last integral becomes a sum operator because, again, we are implicitly considering only the cases in which $\Delta N(t) = 1$, so when a jump occurs. Thus, the integral, an infinitesimal summation, becomes a discrete summation of all the $\ln\big(Y_j\big), \forall j: 1 \dots N(T)$. The rest of the equation evolves just as in the regular Black Scholes diffusive stock. The final solution for $S(T)$ is thus:

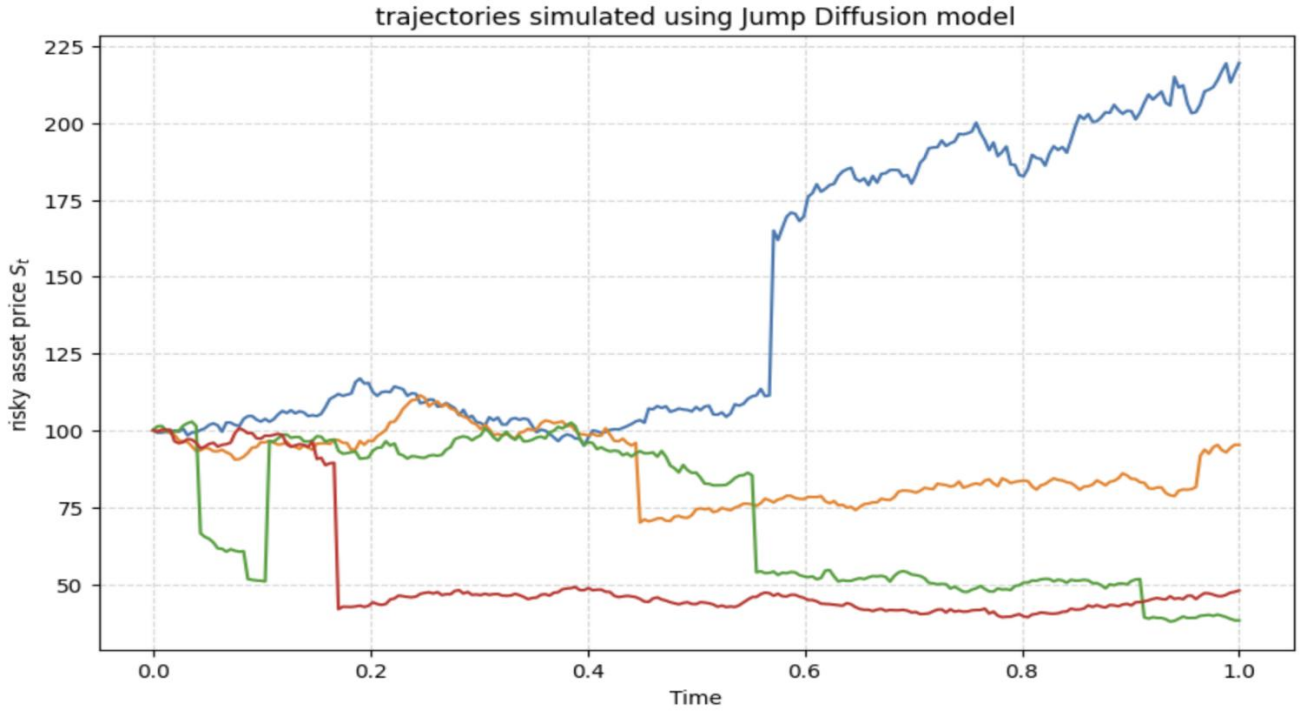
$$S(T) = S(0) * e^{\left(\mu - \frac{1}{2}\sigma^2\right) * T + \sigma W(T)} * e^{\sum_{j=0}^{N(T)} \ln(Y_j)}$$

And, because of the property of the exponential $e^{a+b} = e^a * e^b$, and because $e^{\ln(Y_j)} = Y_j$,

$$\boldsymbol{S(T) = S(0) * e^{\left(\mu - \frac{1}{2}\sigma^2\right) * T + \sigma W(T)} \left(\prod_{j=0}^{N(T)} Y_j\right)}$$

For each future time t, we can find the expression of the jump-diffusive asset $S(t)$ using this expression. We see that the main difference between the "classic" solution within the Black and Scholes framework is the product term, which sums up the impact of the – random – jumps. The two different sources of randomness (the "continuous" one, $W(t)$, and the "discontinuous" one, Y_j) are completely independent from one another – otherwise we could try to predict the occurrence of the jumps, which is the opposite of the interpretation of *random jumps*. An example of trajectories for $S(t)$ simulated using this framework is the shown in *Figure 3*.

Figure 3



Assuming an initial price of $S(0) = 100$ and T = 1, we can clearly see the sudden jumps in each simulated trajectory for the asset price.

2.1.2 Comparison of First and Second moment with the Black-Scholes asset

Now that we have defined the equation for the price of the jump diffusive asset at any time t, it is time to dwell more on the first two moments for this variable: namely, the **expected value** and the **variance** of $S(t)$ under the physical probability measure $\mathbb{P}$. The computations for these measures are strongly simplified by the notion that the three main sources of randomness - $W(t), N(t)$ and Y_j – are independent. In fact, using the expectations operator we can see that:

$$E^{\mathbb{P}}[S(t)] = E^{\mathbb{P}}[S(0)e^{\mu t - \frac{1}{2}\sigma^2 t + \sigma W(t)} * \prod_{j=0}^{N(T)} Y_j \ ]$$

Because of $\sigma W(t) \sim N(0, \sigma^2 t)$, and $E^{\mathbb{P}}\left[e^{\sigma W(t)}\right] = e^{\frac{1}{2}\sigma^2}$ - given by the expected value of a log-normal variable, we can rewrite the expectation and simplify to reach:

$$E^{\mathbb{P}}[S(t)] = S(0)e^{\mu t}E^{\mathbb{P}}[\prod_{j=0}^{N(T)} Y_j \ ]$$

The first part of this equation is precisely the case of the asset following a Geometric Brownian motion, as in the Black Scholes model; the presence of the jumps has an impact on the expected value of the risky asset through the sum of the random jumps. So, the term to look at now is the $E^{\mathbb{P}}[\prod_{j=0}^{N(T)} Y_j \ ]$.
We can rewrite this using the well-known Law of Total Probability

$$E^{\mathbb{P}}[\prod_{j=0}^{N(T)} Y_j \ ] \ = \sum_{n \in N} E^{\mathbb{P}}\left(\prod_{j=0}^{N(T)=n} Y_j \ \middle| N(T) = n\right) * \mathbb{P}[N(T) = n]$$

Exploiting the fact that Y_j and $N(T)$ are independent, and knowing that $N(T)$ is distributed according to a homogenous Poisson distribution – whose probabilities for each value of n are known, and that the sum of expectations is equal to the expectation of the sum - $\sum E[X] = E[\sum X]$ - we can rewrite this as:

$$E^{\mathbb{P}}[\prod_{j=0}^{N(T)} Y_j \ ] \ = \ \sum_{n \in N} (\mu_Y)^n * \frac{(\lambda t)^n}{n!} * e^{-\lambda t}$$

And this expression results in:

$$E^{\mathbb{P}}[\prod_{j=0}^{N(T)} Y_j \ ] \ = e^{\lambda t(\mu_Y - 1)}$$

Finally, the first moment of the jump-diffusive asset can be expressed as:

$$E^{\mathbb{P}}[S(t)] = S(0)e^{\mu t}e^{\lambda t(\mu_Y - 1)}$$

So, the jump-diffusive asset has a slightly different mean compared to the traditional Black-Scholes asset, and this difference is determined by the instantaneous intensity of the Poisson process governing the number of jumps λ and by the mean of the specified variable governing the size of the single jump μ_Y. As λ is a non-negative parameter, it all depends on μ_Y: if $\mu_Y > 0$, jump diffusive asset will be on average higher than BS asset; if $\mu_Y < 0$, the jump diffusive asset will be on average lower than BS asset. In fact, the implied drift rate for a BS asset to have the same expected value as a jump-diffusive asset for a time t would be
$\mu^* = \mu + \lambda(\mu_Y - 1)$.

Intuitively, the jump-diffusive asset has a higher variance than the standard BS asset. As the jumps are unforecastable, and massive in their impact, it is understandable that the asset subject to these jumps will have an equal or higher variance than the standard, jumps-free asset. To show it more rigorously, we know that the variance of the asset is equal to:

$$Var^{\mathbb{P}}[S(t)] = E^{\mathbb{P}}[S^2(t)] - E^{\mathbb{P}}[S(t)]^2$$

Considering the case of a BS asset with the specified drift rate equal to μ^*, in such a way that $E^{\mathbb{P}}[S^*(t)] = E^{\mathbb{P}}[S(t)]$, where $S^*(t)$ is the BS asset, we can see that claiming that $Var^{\mathbb{P}}[S(t)] > Var^{\mathbb{P}}[S^*(t)]$ is the same as claiming that $E^{\mathbb{P}}[S^2(t)] > E^{\mathbb{P}}[S^{*2}(t)]$, since the first moments of the two assets are matched by construction – we chose the drift rate μ^* for the BS asset.
Since the expression for the second moment can be reduced to:

$$E^{\mathbb{P}}[S^2(t)] = S^2(0)e^{(2\mu+\sigma^2)t} * e^{\lambda t\left(E^{\mathbb{P}}\left[Y_j^2\right]-1\right)}$$

We can solve that the expression above where we compare the two second moments can be simplified into the following inequality – by working out the algebra of the two second moments: one for the jump-diffusive, one for the BS asset.

$$E^{\mathbb{P}}\left[\left(Y_j - 1\right)^2\right] > 0$$

The expression above is always true, except for one interesting exception: when $Y_j = 1$.
If we have that $Y_j = 1$, the jump has no impact on the risky asset, as $S(\tau_j) = S(\tau_j^-)Y_j$. Since Y_j is a random variable by itself, we must talk in probability terms: if and only if $\mathbb{P}[Y_j \neq 1] > 0$, then the risk-diffusive asset will have a greater variance than the BS one. If $\mathbb{P}[Y_j = 1] = 1, \forall j: 1 \dots N(t)$, then the variance of the jump-diffusive and of the BS one will be equal, as the jump will leave the jump-diffusive asset unaffected.

2.1.3. No-arbitrage analysis

In this section, we look at the market, within the continuous time framework, generated by the jump-diffusive asset $S(t)$ and $B(t)$. Referring to the first and second fundamental theorems of asset pricing, we will show that this market is free of arbitrages but incomplete.

(First Fundamental Theorem of Asset Pricing): A financial market with finite time horizon T and price processes of one risky asset and one risk-less asset is free of any arbitrage opportunities if and only if we can find an ***Equivalent Martingale Measure (EMM)*** *such that:*

- *The EMM, also denoted as $\mathbb{Q}$ to claim it is a risk-neutral measure, is equivalent to $\mathbb{P}$.*
- *The discounted price process of the risky asset is, under this risk-neutral probability, a martingale.*

(Second Fundamental Theorem of Asset Pricing): A financial market with finite time horizon T and price processes of one risky asset and one risk-less asset is complete, meaning that any contingent claim X(T) can be replicated with a trading strategy of the two securities if and only if we can find a unique Equivalent Martingale Measure for the specified market

Hence, in order to carry out our *No-Arbitrage analysis,* we must look for a way to get an equivalent martingale measure $\mathbb{Q}$ under which the price process of the discounted jump-diffusive asset is a martingale and verify whether we can find an infinite number of measures of probabilities that allow this or only one. In the former case, the market would be arbitrage-free but incomplete; in the latter, it would be both arbitrage-free and complete.

A martingale is defined as a stochastic process whose conditional expected value of the infinitesimal change is equal to 0, i.e. a process that stays, on average, constant over time. More rigorously, we can define a martingale $M(t)$ under the risk neutral probability $\mathbb{Q}$, where $\mathcal{F}_t$ indicates the set of information contained at time t*:*

$$E^{\mathbb{Q}}\left(dM(t)|\mathcal{F}_t\right) = 0$$

We are interested in modelling the discounted price process of the risky asset $\tilde{S}(t) = e^{rt}S(t)$. Using the Ito's lemma presented in the section above, taking as our target function $f(t,x) = e^{-rt}x$, we can easily derive the stochastic differential equation for the discounted price, and this is equal to:

$$d\tilde{S}(t) = (\mu - r)dt + \sigma dW^{\mathbb{P}}(t) + \Delta J(t)$$

On this occasion, we write $W^{\mathbb{P}}(t)$ to mark that this is a standard Brownian motion under the physical probability $\mathbb{P}$, and so it is not the one suitable with the no-arbitrage analysis we are trying to carry out, as we need a Brownian motion under the new, risk-neutral, probability measure $\mathbb{Q}$. The same logic is applied to the counting process $N(t)$. In fact, also this process is a Poisson process under the physical probability, and we need to find the risk-neutral counterparty for both $N^{\mathbb{P}}(t)$ and $W^{\mathbb{P}}(t)$. To "shift" the probability measure we use another fundamental theorem, called the **Girsanov's Theorem**:

(Girsanov's Theorem): Given parameters θ and $\varphi > 0$, there exists a probability measure $\mathbb{Q}$ equivalent to $\mathbb{P}$ ($\mathbb{Q} \sim \mathbb{P}$) such that under $\mathbb{Q}$:

- $W^{Q}(t) = W^{\mathbb{P}}(t) + \theta dt$ *is a Standard Brownian Motion*
- $N^{\mathbb{Q}}(t)$ *is a Poisson process with intensity* $\lambda^{\mathbb{Q}} = \varphi\lambda^{\mathbb{P}}$

With all processes being independent under the $\mathbb{Q}$ probability measure.

Thus, Girsanov's theorem is the tool that we use to find the risk neutral "counterparts" for both the Brownian Motion and the Poisson process that are suitable for our analysis. The search for an Equivalent Martingale Measure $\mathbb{Q}$ is basically reduced to the search of the parameters θ and $\varphi > 0$ that would allow us to specify and define the two processes under the $\mathbb{Q}$ measure. If we can find such parameters, we can apply Girsanov's theorem and verify the existence of a probability measure under which $\tilde{S}(t)$ is indeed a martingale.

By construction, we will find the two parameters that guarantee us that the last condition is verified. By imposing the martingale requirement to $\tilde{S}(t)$, we will see if and how many parameters exist that automatically satisfy the first fundamental theorem of asset pricing. Our goal would be thus to find:

$$\theta \; and \; \varphi > 0 \; s.t. E^{\mathbb{Q}}\left((\mu - r)dt + \sigma\big(dW^{\mathbb{Q}}(t) - \theta dt\big) + \Delta J(t)\middle|\mathcal{F}_t\right) = 0$$

Given that $E^{\mathbb{Q}}(\Delta J(t)|\mathcal{F}_t) = (\mu_Y - 1)\lambda^{\mathbb{Q}}dt$, and that $E_t^{\mathbb{Q}}\left[dW^{\mathbb{Q}}(t)\right] = 0$, we can rewrite the expression above as:

$$(\mu - r)dt + \sigma\,(-\theta dt) + (\mu_Y - 1)\lambda * \varphi dt = 0$$

To solve this equation means to find the two parameters under which the discounted price of the jump-diffusive asset is a martingale. This is in turn equal to:

$$\mu - r - \sigma\theta + ((\mu_Y - 1)\lambda * \varphi = 0$$

By isolating the parameter θ on the left-hand side of the equation we can find that:

$$\boldsymbol{\theta} = \frac{\mu + (\mu_Y - 1)\boldsymbol{\varphi}\lambda - r}{\sigma}$$

Thus θ being a function of φ. This means that for any possible parameter chosen $\varphi > 0$ we can find a unique parameter θ, and, by using those two parameters, $\tilde{S}(t)$ would be a martingale by construction. We have infinite possible vectors of parameters (θ, φ) that make our discounted price a martingale, so we have infinite EMM, and the second fundamental theorem does not apply: Market is free of arbitrage but incomplete.
It is important to stress that the pair of parameters chosen for our EMM will then be the parameters used to get the risk-neutral counterparties for $W^{\mathbb{Q}}(t)$ and $N^{\mathbb{Q}}(t)$, according to the formulas provided within the Girsanov's Theorem.

2.2. *Hawkes Processes*

The novelty of this paper lies in the usage of a different counting process compared to the classical case of a Poisson distribution for $N(t)$.

2.2.1. Stationarity and Kernel of the non-homogenous counting process

The process, as we have already highlighted in the previous section, shows a constant, homogenous intensity parameter λ. This parameter is invariant in time and is not impacted by the occurrence of a jump. This represents the main difference with the other counting process we are going to describe now: The Hawkes Process.
Initially used to model the occurrence of seismic events, a Hawkes process is a counting process which can accurately model the occurrence of jumps for the jump-diffusive asset. The main feature of these processes is that the intensity parameter is not constant but, on the contrary, is a function of time $\lambda(t)$. This is not to be seen as a *deterministic* function of time, since the parameter is driven by a *stochastic* nature. Moreover, this parameter presents a self-exciting nature: the intensity parameter is positively impacted by the occurrence of past jumps by a function named *kernel* according to the expression below:

$$\lambda(t) = \lambda(0) + \sum_{m=0}^{M} \varphi(t - t_m)$$

Where M is the total number of events occurred before time t, and t_m indicates the time of each event. The formula expressed above is the specific case of a univariate Hawkes process – the one that will be of interest for us, as we are going to model only one asset, but a more general formula can be found for a Multivariate Hawkes process, where $\boldsymbol{N(t)} = \{N^i(t)\}_{i=1}^{D}$, to jointly model D different counting processes.

$$\boldsymbol{\lambda}(t) = \boldsymbol{\mu} + \boldsymbol{\Phi} * d\boldsymbol{N}(t)^{7}$$

Where $\boldsymbol{\Phi(t)} = \{\varphi^{ij}(t)\}_{i,j=1}^{D}$ is a matrix-valued kernel that requires three conditions to be met:

- $\varphi^{ij}(t) \geq 0, \forall 1 \leq i, j \leq D$; meaning that all the kernels, for each process included in the multivariate distribution, must be non-negative.
- If $t < 0, \varphi^{ij}(t) = 0, \forall 1 \leq i, j \leq D$.
- Each component $\varphi^{ij}(t)$ belongs to the space of L^1-integrable functions.

A more direct way of visualizing the multivariate process is using the following equation:

$$\lambda_i(t) = \lambda(0)_i + \sum_{j=1}^{D} \sum_{k:t_{j,k}<t} \varphi_{ij}(t - t_{j,k}), \forall i: 1, \dots, D^{8}$$

The notion of kernel introduced is particularly important as it represents the implicit "memory" of the counting process itself; if a jump occurs, the intensity parameter will increase making it more likely to witness other jumps in the immediate future. This increase is due to the kernel, as it cannot be negative, but its effects tend to decay over time. The amplitude and the subsequent speed of decay depend on the specification of the kernel; in other words, it depends on the functional form we set for the kernel $\varphi(t)$.
The main assumption we will impose and will rely on is the stationarity of the counting process. Stationarity is ensured by looking at one important measure in linear algebra: the spectral radius of a matrix. The Spectral radius is the biggest eigenvalue for a given matrix *A*. In our case, our matrix $\boldsymbol{\Phi(t)}$ is a matrix of functions $\{\varphi^{ij}(t)\}_{i,j=1}^{D}$. So, we indicate the L^1-norm of a function as $||\varphi|| = \int |\varphi(t)|dt$, and we will write the matrix containing the norms of

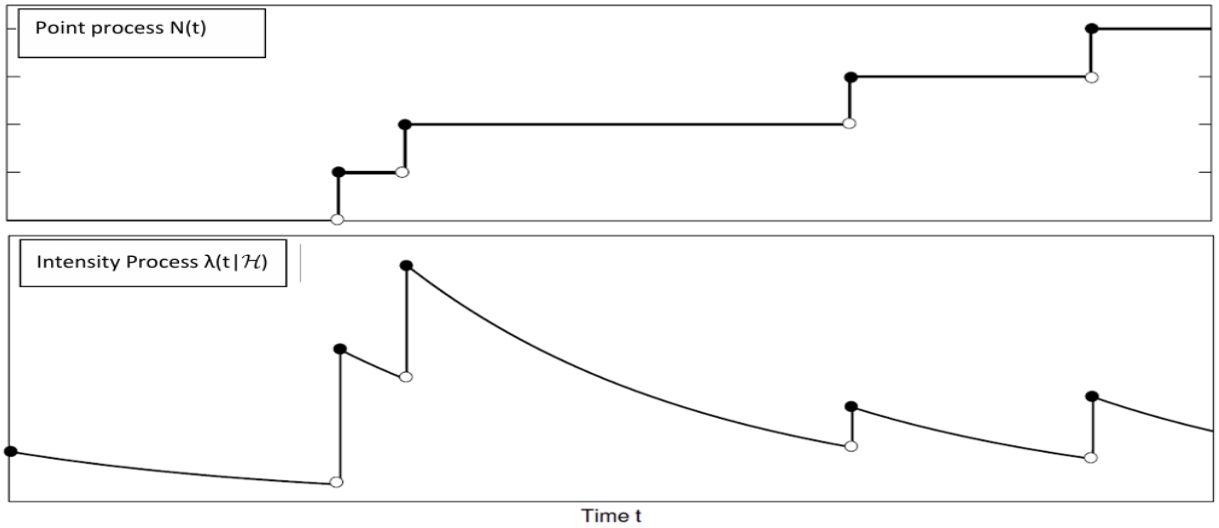
[7] Hawkes Processes in finance, Bacry et al. (2015)
[8] tick.hawkes — tick 0.6.0 documentation

each function as $||\boldsymbol{\Phi}(\boldsymbol{t})|| = \left\{||\varphi^{ij}||\right\}_{i,j=1}^{D}$. The spectral radius will thus be the biggest eigenvalue of this matrix of norms.

(Stationarity) The process N(t) has asymptotically stationary increments and $\lambda(t)$ is asymptotically stationary if the matrix $||\boldsymbol{\Phi}(\boldsymbol{t})||$, where $\boldsymbol{\Phi}(\boldsymbol{t})$ is the Kernel, has spectral radius smaller than 1 - Stability Condition.

This property insures that, cumulatively, there is not "too much memory" allowing the process to still be stationary and not containing a unit root.

Figure 4[9]



In *Figure 4* we can see an example of a simulated Hawkes Process. The upper graph shows the counting process itself, whereas the lower graph shows the modelling for the stochastic intensity parameter $\lambda(t)$. We clearly see that whenever jumps occur - $\Delta N(t) = 1$, a jump in the stochastic intensity occurs as well, triggered by the kernel of the expression. Slowly, the positive effect on the intensity parameter decays over time, and it jumps back up as soon as another jump is witnessed. It is worth noting that in this example the exogenous parameter λ_0 is set to be a deterministic function of time, whereas in our case we simply have taken a constant starting value.

2.2.2. Exponential and Power-Law kernel

The two main functional forms for the kernel $\varphi(t)$ are exponential kernel and power-law kernel**.** The first specification consists of using an exponential function to model our kernel, and, thanks to the time-decaying property of the exponential distribution, it is among the most common within the finance and mathematics world. The "reaction" function of this kernel – in the univariate case - is the following[10]:

$$\varphi(t) = \alpha e^{-\beta t}$$

The first parameter α in the function measures the initial intensity of the jump, the overall strength of the interaction, whereas the second parameter β measures the decaying effect over time of past events. The following graphs show this functional relationship in function of time with different parameters:

[9] Simulation, Estimation and Application of Hawkes Processes, Kataryna Obral, June 2016, pag. 15

[10] A Brief Introduction to Hawkes Processes | by Hey Amit | We Talk Data | Medium

Figure 5

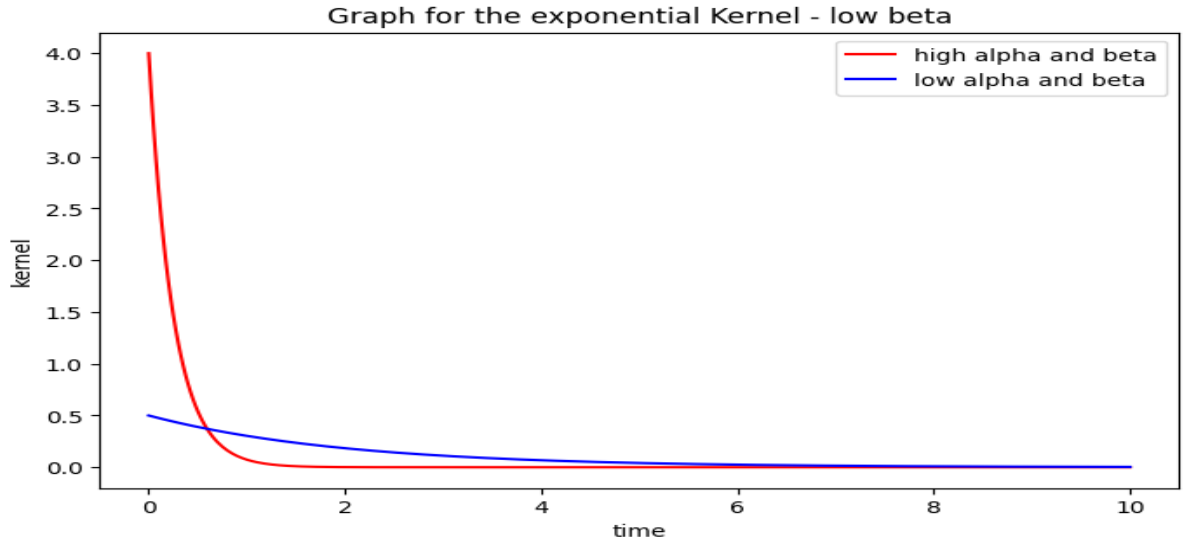


Figure 5 shows two different kernel functions. The red one presents high parameters and has, thus, a very strong but ephemeral impact: it does not have a long-lasting memory, indicating that, in the occurrence of a jump at time 0, the subsequent increase in the intensity parameter will not last long. The blue one presents low parameters, in both alpha and beta. The impact is visibly limited in size, but it is much more long-lasting compared to the previous case – driven by the parameter β.

The exponential specification is very much used also for another interesting and useful property: under this specification, the Hawkes process is in fact *Markovian* and thus useful for simulation and estimation. In fact, the intensity $\lambda(t)$ can be further recast in Markovian form as a *stochastic differential equation:*

$$d\lambda(t) = \beta(\lambda_0 - \lambda(t))dt + \alpha dN(t)^{11}$$

This representation is very useful as it shows that the infinitesimal difference in time t is not relying on the whole history of the process, which is indeed the Markovian property. The first component of the *SDE* $-\beta\lambda(t)dt$ refers to the time decaying feature of the process, whereas the second one $\alpha dN(t)$ is related to the sudden jump in the intensity parameter in the occurrence of a jump ($\Delta N(t) = 1$). This equation for the stochastic intensity parameter is only viable under the exponential kernel.

The second specification often used to model this process is the power-law kernel. In this specification, the kernel has the following form:

$$\varphi(t) = \frac{\alpha\beta}{(1+\beta t)^{1+\gamma}}$$

This specification allows for the intensity parameter to have more memory since each jump that occurred has a more long-lasting effect on the stochastic parameter compared to the exponential kernel. In this case, the intensity is not a Markovian variable and the intensity parameter $\lambda(t)$ depends on the whole history of $N(t)$, and not just the last realization.

[11] Hawkes Processes in finance, Bacry et al. (2015)

2.3. Monte Carlo Technique

2.3.1. General Idea and Pricing of European Call

Monte Carlo is a powerful probabilistic technique to carry out **simulations** that we will exploit to carry out the pricing of lookback options. According to the traditional mathematical finance definition, the price of an option can be written as a conditional expectation of the future payoff and, thus, as an integral.
For instance, the price of a call option at time 0 can be written as:

$$C(0) = E_0^{\mathbb{Q}}[e^{-rT} * max\{S_T - K; 0\}]$$

Because the only random variable contained in the expression is S_T, and because this is a continuous random variable, we know that the first moment of a random variable can be expressed as an integral:

$$E[g(X)] = \int_{-\infty}^{\infty} g(x) f_X(x) dx$$

The price of the call can, then, be expressed as the following integral:

$$C(0) = e^{-rT} \int_K^{\infty} (s - K) * f_{S(T)}(s) ds$$

In other words, the price of the call is the infinitesimal summation of the future payoff of the call - $(s - K)$ – weighted by the probability of having that value $S(T) = s$ - $f_{S(T)}(s)$ is the probability distribution function of the $S(T)$ – for each infinitesimal difference in the value of the underlying asset - ds. Taking the integral from the strike price K to infinity implied that we are only considering the cases where the call is in the money, rightfully.

Monte Carlo technique is a simple way to estimate this unknown quantity by *discretizing* the integral shown above. Instead of taking *infinitesimal* differences in *s,* we **simulate** *N* different values for $S(T)$. From this simulated prices of the underlying, we compute the implied payoff for the European call – which will be positive only if the call is in the money, is $S(T) > K$. Finally, we compute the price of the call for that specific simulation by discounting the payoff back to time 0, and we carry the same analysis N times. The Monte Carlo estimate for the price of the call would be the average of all these N simulated prices. Analytically, the MC price can be represented in the following expression:

$$\widehat{C(0)} = \frac{1}{N} * \sum_{i=1}^{N} e^{-rT} (S_i(T) - K)^+$$

Where $S_i(T)$ is the simulated stock price at time T in the i-th simulation.
The next important point would lie in how to accurately simulate $S(T)$. In order to do, we need to have specified a mathematical framework to model the stochastic dynamic of the underlying within $[0, T]$. In the most common case, a Geometric Brownian motion for the asset will be assumed.
As we only need to model the final value of the underlying – because it is a plain-vanilla, European call, we will exploit the notion that:

$$S(T) = S_0 e^{\left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma W(t)}$$

Since $W(t) \sim N(0, t)$, we can generalize and say that $W(t) \sim t^{0.5} Z,\ Z \sim N(0,1)$. Thus, it is easy to simulate different value for $S(T)$; we simply must simulate different values for the standard normal and then plug the simulated values into the expression below:

$$S_i(T) = S_0 e^{\left(\left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma t^{0.5} * \mathbf{Z_i}\right)}$$

It is much easier to simulate values for a normal standard. Simulate a value from a *Uniform* distribution $U[0:1]$, and use the Inverse cumulative function for the standard normal $F^{-1}(u)$ to get the relative quantile for that simulated level of probability. This is the so-called *Inverse Function Method,* and it is feasible because the *CDF* $F(x)$ of the standard Normal is invertible – it does not have constant traits on some interval.

Figure 6

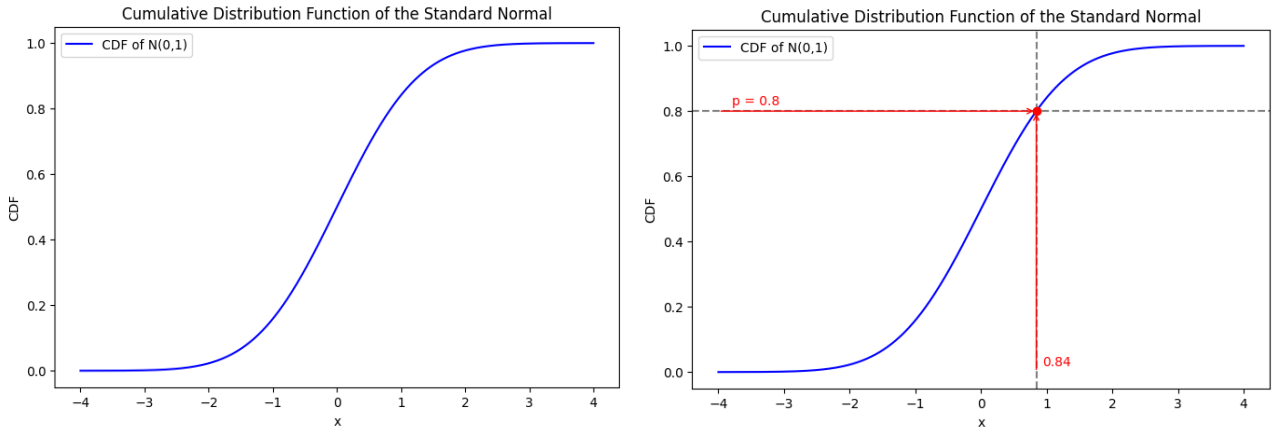


Figure 6 shows the Inverse Function Method to simulate values for $N(0,1)$ from the *Cumulative distribution function* of the variable.

2.3.2. Euler Discretization

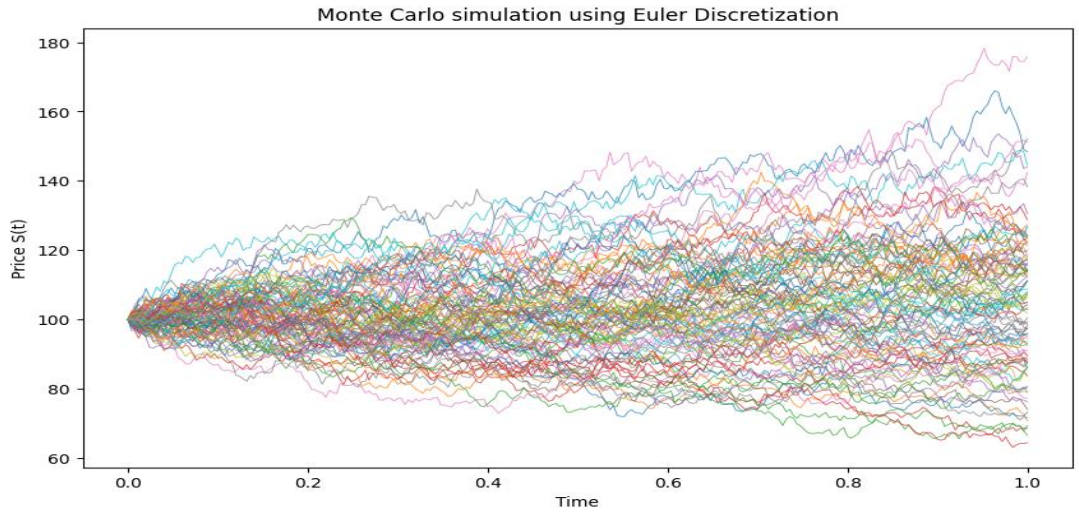
In this "simple" exercise, we are only interested in simulating the prices of the asset at maturity of the call option $S(T)$. Because this is plain vanilla, the payoff only depends on this variable. In the later section we would be dealing with the pricing of lookback options, which is an exotic product whose payoff depends on the *whole path* of the stock, and not just the price at one point in time. We will thus use the well-known **Euler discretization** to simulate the whole path:

$$S(t + \Delta t) = S(t) e^{\left(\mu - \frac{1}{2}\sigma^2\right)\Delta t + \sigma(\Delta t)^{0.5} Z_{i\,12}}$$

In this exercise we would need to specify both the number of simulations N and the number of time steps for each simulation M. Visually, we would thus end up with a matrix of simulated values.

[12] Numerical Methods II, prof. Mike Giles, Oxford University Mathematical Institute

Figure 7



In *Figure 7* we can see an application of the Euler discretization to carry out 100 simulations of the stock price, including 250 time-steps between today and a year from today.
In the pricing of the lookback call, we will use these simulated paths to compute the maximum and the minimum of the asset's price within the life of the option, which will be crucial in computing the payoff of the fixed / floating lookback option. The payoff of the floating lookback call is:

$$S(T) - \min_{0 \le t \le T} S(t)$$

Whereas the payoff of the fixed lookback call is:

$$\max(\max_{0 \le t \le T} S(t) - K, 0)$$

2.3.3. Statistical Properties of Monte Carlo estimator

Monte Carlo produces an **Unbiased** and **Consistent** estimator of the true variable we are trying to estimate by discretization and simulation. Unbiased means that the estimator is, on average, equal to the true parameter:

$$E\left[\widehat{C(0)}\right] = C(0)$$

Consistency is another statistical property relying on the strong law of large numbers. In fact, it basically claims that the estimator converges to the true value as the sample size increases to infinity:

$$\widehat{C(0)} \rightarrow_{n \to \infty} C(0)$$

These two properties justify the usage of the Monte Carlo estimator, guaranteeing that the prices obtained have statistical meaning and are thus reliable and useable.
Another useful property of the Monte Carlo estimator is that, thanks to the existence of the Central limit theorem, we can build an interval of confidence around the point estimate of the price of the option – we can use the quantiles of a standard normal or of a T student. The formula for the interval is the usual:

$$[\widehat{C(0)} \pm \frac{\sigma_f}{n^{0.5}} * z_{\frac{\delta}{2}}]$$

Where the first term of the radius $\frac{\sigma_f}{n^{0.5}}$ is the standard error of our estimator, and the second $z_{\frac{\delta}{2}}$ is the quantile of the standard normal for the desired significance level δ. σ_{f_f} is the standard deviation of the estimator, which is naturally

unknown and must be thus replaced with the sample standard deviation $\widehat{\sigma_n}$.
This measures the volatility of the different simulated call prices among the *vector* of simulated call prices by the Monte carlo estimator, and can be thus computed as:

$$\widehat{\sigma_n} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(C_i - \widehat{C_0}\right)^2}$$

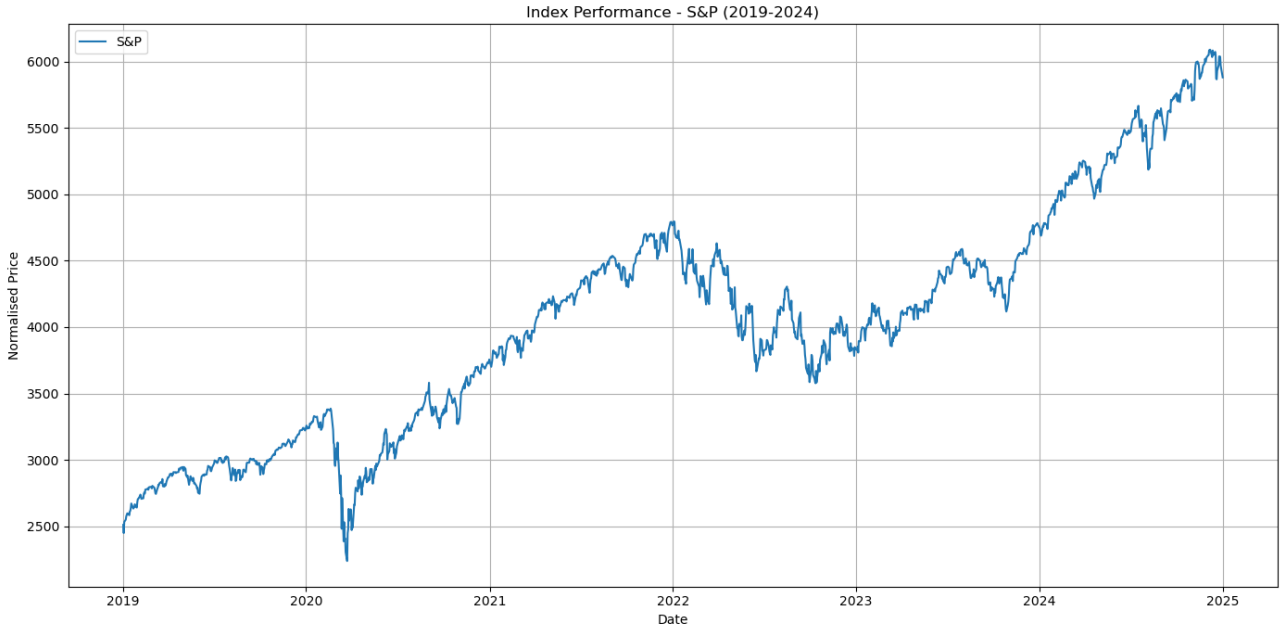
This parameter measures how dispersed the individual simulated call prices are and thus how variable our estimator is.

3. Numerical Application

In this section we analyze the algorithms to carry out pricing of specific path-dependent options using the augmented Jump diffusion model as framework for the risky asset.

First, we need to choose a specific equity underlying for the options that we are going to consider. Due to availability of data and meaningfulness for the analysis, we are going to consider options on the S&P 500 index, whose performance in the last years is shown in *Figure 8.*

Figure 8



As was stated in section 1, the price path is subject to two different sources of randomness: the exposure to $W(t)$ and to $J(t)$. The first exposure determines variation in the price dynamic in a *continuous* fashion, whereas the second one introduces big variations within the form of *discontinuous jumps.* In practice, it can be troublesome to distinguish these sources: a daily return of $+5\%$ can either indicate a sudden jump in the price or an increase in the diffusion coefficient which makes the path much more volatile. In statistics, many tests have been developed to test whether a jump has occurred on a certain date or not, but they require access to *high-frequency-data* to model the Realized Variance and build the test statistic. For instance, the Barndorff-Nielsen and Shephard (2006) test proposes a test statistic with the realized variance to assess the null hypothesis H_0 that there is no jump at a given time t[13]. Since having access to High-Frequency data would be troublesome and outside of the scope of this thesis, we will carry out a much simpler rule of thumb to assess the presence of jumps.

For each stock, we compute the first differences in prices $\Delta P_t = P_t - P_{t-1}$ in absolute terms for each stock. We then compute the average and standard deviation of these first differences $\mu_{\Delta P}$ and $\sigma_{\Delta P}$. We then build a z_score to which is the series of first differences normalized for the two moments computed:

$$z_i = \frac{\Delta P_i - \mu_{\Delta P}}{\sigma_{\Delta P}}$$

And we will consider as a *jump-day* all the days in which $|z_i| > 3$.

We can analyze how many jumps in the index have occurred in the same time period in *Figure 9.*

[13] R: Barndorff-Nielsen and Shephard (2006) tests for the presence...

Figure 9



In total there have been a total of 25 jumps in this period, and we see enough evidence of clustering as jumps tend to be close to each other – in the early 2020, during COVID pandemic, we see many downward jumps clustered.

3.1. Calibration of plain-vanilla European call closed-end formula to market prices

Before dealing with the pricing using a stochastic intensity parameter for the jump component, we will develop an initial framework to find the input parameters for the traditional Merton model: once we have these input parameters, we will extend the model to include a varying and stochastic intensity parameter λ. The procedure of finding the best input parameters for our pricing model is called Calibration: we look for parameters that make out theoretical prices as close as possible to market prices, so that our pricing process will be as realistic as possible. Within the Merton model, we mainly have four input parameters that need to be calibrated:

- σ: the diffusion coefficient for the jump-diffusive asset
- λ: the intensity parameter for the jump component
- a: the parameter taken for the mean of the jump size Y_j
- b: the second parameter taken for the variance of the jump size Y_j

Initially, we will need to find the market data for plain vanilla call prices on SPX across a range of strikes and maturities. Then, for each collected market price, we will compute the corresponding theoretical call prices using the closed formula within the Merton model. Finally, we will calibrate the pricing model by applying optimization algorithm to minimize a chosen loss function measuring the discrepancy between market and theoretical prices.
All the cross-sectional prices that are going to be used for the calibration have been observed at the same date at the end of our sample of prices, so at date $31/12/2024$.
Market prices for calls for the underlying were taken from *WRDS-OptionMetrics*[14], and we downloaded the mid prices for all the options with all available strikes. We selected European options with many different maturities clustered into three labels: *short* maturity, ranging from 1 day to 30 days' time to maturity, *medium* maturity, from 70 to 100 days, and *long* maturity, from 180 to 260 days. This resulted in approximately wide cross-data on option market prices on SPX, to have a sound calibration of parameters under the risk neutral probability. Overall, we have a panel of European call option market prices, only at date $31/12/2024$, of 5689 prices, of which 590 from the long maturity cluster, 1147 from the medium one and 3952 from the short one. In the last cluster there were some options with time to maturity $T = 0$ which have been removed

[14] *WRDS, OptionMetrics, Option Prices*

The risk-free rates r used as the drift parameter in the risk neutral world has been extracted from the American bonds' yield curve. Since the maturities for our clusters are around 1,3 and 6 months, we get the correct yields directly from the curve at $31/12/2024$ [15]: the yields are thus equal to $r_{1M} = 4.872\%$, $r_{3M} = 4.876\%$ and $r_{6M} = 4.866\%$. The initial index value at the same date is: $S0 = 5881.63\$$.

Now, let us dwell into the closed-end formula to find the corresponding theoretical prices of European calls found above. As shown in the previous section, we will use the Merton framework, where:

$$S(T) = S(0) * e^{\left(\mu - \frac{1}{2}\sigma^2\right)*T + \sigma W(T)} \left(\prod_{j=0}^{N(T)} Y_j\right)$$

Under the Physical probability measure $\mathbb{P}$. Naturally, we are now interested in the price dynamics under the risk neutral $\mathbb{Q}$ measure – as this is the common framework in option pricing, where the discounted price $\widetilde{S_t}$ is a martingale. So, we need the expression for $S(T)$ under the $\mathbb{Q}$ probability measure. As we have previously seen, there are infinite possible risk-neutral measures (*EMMs)* we can use, since there are infinite vectors of (θ, φ) that kills the drift of $\widetilde{S_t}$ under $\mathbb{Q}$ probability measure. We select the specific case of $\varphi = 1$, meaning that the intensity parameter of the jump is the same in both probability measures $\mathbb{P}$ and $\mathbb{Q}$. This implicitly means that investors do not require any premium for the jump risk, they are themselves risk neutral with respect to jump risk. The *stochastic differential equation* for the asset under $\mathbb{Q}$ will thus be:

$$\frac{dS(t)}{S(t)} = (r - (\mu_Y - 1)\lambda^Q)dt + \sigma dW^Q(t) + \Delta J(t)$$

Briefly, the term $(\mu_Y - 1)\lambda^Q dt$ has been added within the drift term because, under $\mathbb{Q}$, we need to ensure that no arbitrage condition holds. That term represents the potential extra-return that is achievable under $\mathbb{Q}$ thanks to the presence of jump, since μ_Y is the expected value and λ is the intensity parameter of these – since we assume $\varphi = 1$, the two intensity parameters are equal. In fact, $E_{t^-}^Q[\Delta J(t)] = (\mu_Y - 1)\lambda^Q dt$, where $(\mu_Y - 1)$ in the conditional mean of the size of the jump in an infinitesimal time interval and $\lambda^Q dt$ is the probability of occurrence of a jump in the same infinitesimal time interval.

Following the same procedure followed in section "Counting process and *stochastic differential equation",* thus exploiting the Ito's lemma to get the *SDE* for the logarithmic price under $\mathbb{Q}$, we can come up with the following expression:

$$S(T) = S(0)e^{\left(r - (\mu_Y - 1)\lambda^Q - \frac{1}{2}\sigma^2\right)T + \sigma W^Q(T)} \prod_{j=1}^{N(T)} Y_j$$

Under Merton's assumption, $Y_j \sim LogNormal(a, b^2)$, so $Y_j \sim e^{X_j}$ where $X_j \sim N(a, b^2)$. The parameters a and b are, respectively, the mean and the standard deviation of the natural logarithm of the lognormal variable Y_j, as $X_j = \ln(Y_j)$. The price of a European call option can be derived in the following way.

We can rewrite the expression above as follows:

$$S(T) = S(0)e^{\left(r - \frac{1}{2}\sigma^2\right)T + \sigma W^Q(T) + \sum_{j=1}^{N(T)} X_j - (\mu_Y - 1)\lambda T}$$

We now write the same expression of the stock price conditioning on a precise number of jumps occurred in the period $[0; T]$ n. So, we will rewrite the expression of the stock price at time T knowing that in the period there have been n jumps – This way, we will treat $N(T)$ as a fixed value and not as a randon variable.

$$S(T)|_{N(T)=n} = S(0)e^{\left(r - \frac{1}{2}\sigma^2\right)T + \sigma W^Q(T) + \sum_{j=1}^{n} X_j - (\mu_Y - 1)\lambda T}$$

[15] *WRDS, OptionMetrics* Zero Coupon Yield Curve

We notice that the second $\sigma W^Q(T)$ and third term $\sum_{j=1}^{n} X_j$ are both Normal variables. In fact, $\sigma W^Q(T) \sim N(0, \sigma^2 T)$and $\sum_{j=1}^{n} X_j \sim N(na, nb^2)$.
Since the sum of two Normal Distributions is still a Normal distribution, we have that $\sigma W^Q(T) + \sum_{j=1}^{n} X_i \sim N(na, nb^2 + \sigma^2 T)$. Rewriting the variance of this new normal as $\sigma_n^2 T$ and using the Standardization rule for the Normal distribution - $\frac{N_i - \mu}{\sigma} = Z_i$, where $Z_i \sim N(0,1)$, we can rewrite the expression above as:

$$S(T)|_{N(T)=n} = S(0)e^{\left(r - \frac{1}{2}\sigma^2\right)T + na + \sigma_n\sqrt{T}Z - (\mu_Y - 1)T\lambda}$$

Further simplifying, and setting $r_n = r - (\mu_Y - 1)\lambda + \frac{nlog(\mu_Y)}{T}$, we can rewrite this as:

$$S(T)|_{N(T)=n} = S(0)e^{(r_n - \frac{1}{2}\sigma^2)T + \sigma_n\sqrt{T}Z}$$

Conditioning on $N(T) = n$, we can thus see the price of the jump-diffusive asset at time T is indeed the same, under the $\mathbb{Q}$ probability measure, as the price of a standard Black Scholes asset with drift r_n and diffusion σ_n. These parameters, as shown above, are indeed adjusted versions of the original drift and diffusion parameters to account for the number (n) and size of jumps occurred in the period.
The price of the European Call will thus be equal to:

$$C(0) = E^Q[(S(T) - K)^+ e^{-rT}] =$$

$$\sum_{n \in N} E^Q\left[\left(S(0)e^{\left(r_n - \frac{1}{2}\sigma^2\right)T + \sigma_n\sqrt{T}Z} - K\right)^+ e^{-rT} \middle| N(T) = n\right] * \frac{e^{-\lambda^Q T}(\lambda^Q T)^n}{n!}$$

Which can be interpreted as an average of all the expected payoffs of the call for each possible realization of number of jumps n. The second term $\frac{e^{-\lambda^Q T}(\lambda^Q T)^n}{n!}$ is in fact the probability of having exactly n jumps, $\mathbb{Q}(N(T) = n)$, so we are taking the expected payoff from the call at time T $\forall n$: 1 ..., weighted by the risk neutral probability of occurrence of those n number of jumps. One last adjustment is needed before getting to the final closed-end formula. In the expression above, the drift rate r_n is different from the discount rate r used within the expected payoff formula. To price that expected payoff with the Black Scholes Formula, we need to make sure that there two rates coincide. Thus, by taking e^{-rT} out of the expectation operator, and by both multiplying and dividing by $e^{r_n T}$, we can rewrite the expression as:

$$\sum_{n \in N} E^Q[\left(S(0)e^{\left(r_n - \frac{1}{2}\sigma^2\right)T + \sigma_n\sqrt{T}Z} - K\right)^+ e^{-r_n T}] * \frac{e^{-\lambda' T}(\lambda' T)^n}{n!}$$

Where $\lambda' = \lambda^{\mathbb{Q}} * \mu_Y$ is a slightly adjusted intensity parameter. This adjustment has been carried out to make sure that the discount and drift rates are the same. The mathematics of this is outside the scope of this paper.
Finally, we can use the Black Scholes formula to price those conditional payoffs, as, thanks to the specification of r_n and σ_n, we are back within the traditional Geometric Brownian Motion dynamic for the underlying asset.
So, the theoretical price of a plain-vanilla European call using a jump diffusive asset as in the Merton model is an *average* of all the Black-Scholes call prices for each different value of n weighted by the risk neutral probability of having precisely those n jumps.

$$C(0) = \sum_{n \in N} bsc(S(0), r_n, \sigma_n) * e^{-\lambda' T}\frac{\lambda' T^n}{n!}$$

Now that we have the closed-end formula, we set up a Python algorithm to find the corresponding theoretical prices for all the calls for which we gathered the market prices, to see how the model fits the cross-data and to calibrate the main parameters of the model: σ, a, b, λ. The algorithm used is the following:

```
#These are the formulas to compute the closed price
def black_scholes_call(S, K, T, r, sigma):
    d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    return S * stats.norm.cdf(d1) - K * np.exp(-r * T) * stats.norm.cdf(d2)

def merton_jump_call_price(S, K, T, r, sigma, lamb, a, b, N=50):

    k = np.exp(a + 0.5 * b**2) - 1  # mu_y - 1
    lamb_tilde = lamb * (1 + k)     # adjusted intensity parameter

    price = 0
    for n in range(N):
        poisson_weight = np.exp(-lamb_tilde * T) * (lamb_tilde *T)**n /
math.factorial(n)
        sigma_n = np.sqrt(sigma**2 + (n * b**2) / T)
        r_n = r - lamb * k + n * np.log(1 + k) / T
        price += poisson_weight * black_scholes_call(S, K, T, r_n, sigma_n)

    return price
```

With this algorithm we will compute the price of a European plain vanilla call using the Merton closed-end formula. Now that we have our theorical prices, we need to compare them with market prices and define a loss function which we will try to minimize. This loss function will be a measure of the discrepancy between the two prices. Our first candidate for loss function is the following:

$$Loss_\theta = \sum_{j}^{Short} \left(lnC^i_{Merton,\vartheta} - lnC^i_{Mkt}\right)^2 + \sum_{i}^{Medium} \left(lnC^i_{Merton,\theta} - lnC^i_{Mkt}\right)^2 + \sum_{i}^{Long} \left(lnC^i_{Merton,\theta} - lnC^i_{Mkt}\right)^2$$

Where ϑ is the vector containing all four parameters we are calibrating $\vartheta = (\sigma, \lambda, a, b)$. The algorithm to compute this loss function is described in the following section.

```
def loss_function_log_scaled(params, calib_data, S0, N=50, eps=1e-6):
    sigma, lamb, a, b = params
    loss = 0.0
    for d in calib_data:
        C_model = merton_jump_call_price(
            S=S0,
            K=d["K"],       # array di strike
            T=d["T"],       # maturity fissa
            r=d["r"],
            sigma=sigma,
            lamb=lamb,
            a=a,
            b=b,
            N=N
        )
        diff = np.log(C_model + eps) - np.log(d["C"] + eps)
        loss += np.dot(diff, diff)

    return float(loss)
```

Our goal is now to set up an optimization problem to find the optimal parameters ϑ^* so that the loss function is minimized; in other words, we wish to find the parameters of input ϑ^* such that the discrepancy between theoretical

prices produced by the pricing model and the market prices is minimized, so that in later uses these parameters will be used. In mathematical terms, our problem is:

$$\underset{\vartheta}{\operatorname{argmin}}\ Loss_{\vartheta}$$

Where $Loss_{\vartheta}$ is essentially a function defined on four variables $f: \mathbb{R}^4 \to \mathbb{R},\ f(a, b, \lambda, \sigma)$. We are trying to find the minimum in a static optimization problem where the target function is defined in four dimensions. Naturally, there are some constraints that must be met by our optimum: $b > 0, \sigma > 0\ and\ \lambda \geq 0$. Note that whereas b and σ must be positive, as they both refer to standard deviations, the intensity parameter for the jump must be non-negative: it is plausible to have $\lambda = 0$, in which case we would simply go back to the traditional Black-Scholes model. This calibration procedure has also a clear econometric meaning: it is essentially a least-square estimation, in which we want to find the parameters such that the sum of squared residuals is minimized. The only difference from a classical linear regression model is that this model is clearly nonlinear (since the Merton pricing formula is nonlinear), and we operate under a restricted parameter space, with economically meaningful constraints imposed on the admissible values of the parameters.: hence, we can consider these estimators as *RNLS* (*Restricted nonlinear least squares).*

$$C_i^{MKT} = C_i(\sigma, \lambda, a, b) + \varepsilon_i$$

We assume $E[\varepsilon \mid C_i(\sigma, \lambda, a, b)] = 0$ so that, under suitable regularity conditions, this estimator is consistent.

Thus, our constrained optimization model has the following form:

$$\underset{(a|b|\lambda|\sigma)}{\operatorname{argmin}} \sum_{j}^{Short} \left(lnC^i_{Merton,(a,b,\lambda,\sigma)} - lnC^i_{Mkt}\right)^2 + \sum_{i}^{Medium} \left(lnC^i_{Merton(a,b,\lambda,\sigma)} - lnC^i_{Mkt}\right)^2 + \sum_{i}^{Long} \left(lnC^i_{Merton(a,b,\lambda,\sigma)} - lnC^i_{Mkt}\right)^2$$

s.t

$$a \in \mathbb{R}$$

$$b > 0$$

$$\sigma > 0$$

$$\lambda \geq 0$$

3.1.1. L-BFGS-B and Gradient Descent algorithms

To carry out this minimization problem, we have tried two different algorithms, with slightly different features. The first one used is **L-BFGS-B**[16]**.** This second-order, quasi-Newton algorithm tries to find the optimum by using both the gradient and the Hessian matric of the target function. The main idea of this class of optimization algorithm is that starting from an initial point of the multivariate function $f(\boldsymbol{x})$, they use the Taylor expansion of second order to approximate a quadratic function from that point, and they minimize that function g. From the new point they repeat this procedure until they reach the local minimum. For each iteration, an estimation of the Hessian matrix used for the Taylor expansion is necessary:

$$g(x) = f(q) + (x - q)^T \nabla f(g) + \frac{1}{2}(x - q)^T B(x - q)$$[17]

Where q represents the point of the previous iteration, $\nabla f(g)$ the gradient of the target function in the point of the previous iteration and B the estimated Hessian matrix. This is the common setting for any second-order minimization algorithm, where they use both the first derivative (Gradient) and the second derivative (Hessian matrix) in the

[16] Una breve introduzione all'algoritmo di ottimizzazione BFGS

[17] *Understanding scipy.minimize part 1: The BFGS algorithm*

process. What is specific about L-BFGS-B is the way in which this algorithm updates the Hessian matrix. Let us call B' the Hessian matrix of the previous iteration. The matrix B for the new iteration is obtained under the following criterion:

$$B\ s.t.\min_{B} ||W(B^{-1} - B'^{-1})W||_F \quad [18]$$

Where $||.||_F$ is the Frobenius norm for a matrix. The matrix W in inserted to guarantee that B is positive definite, so that our minimization problem does not become a maximization one, and we take the inverse of the Hessian for numerical efficiency.

This algorithm yields a minimum in a rather short period of time and proposes the following optimal parameters:

a^*	*-0.369*
b^*	*0.346*
σ^*	*0.106*
λ^*	*0.029*

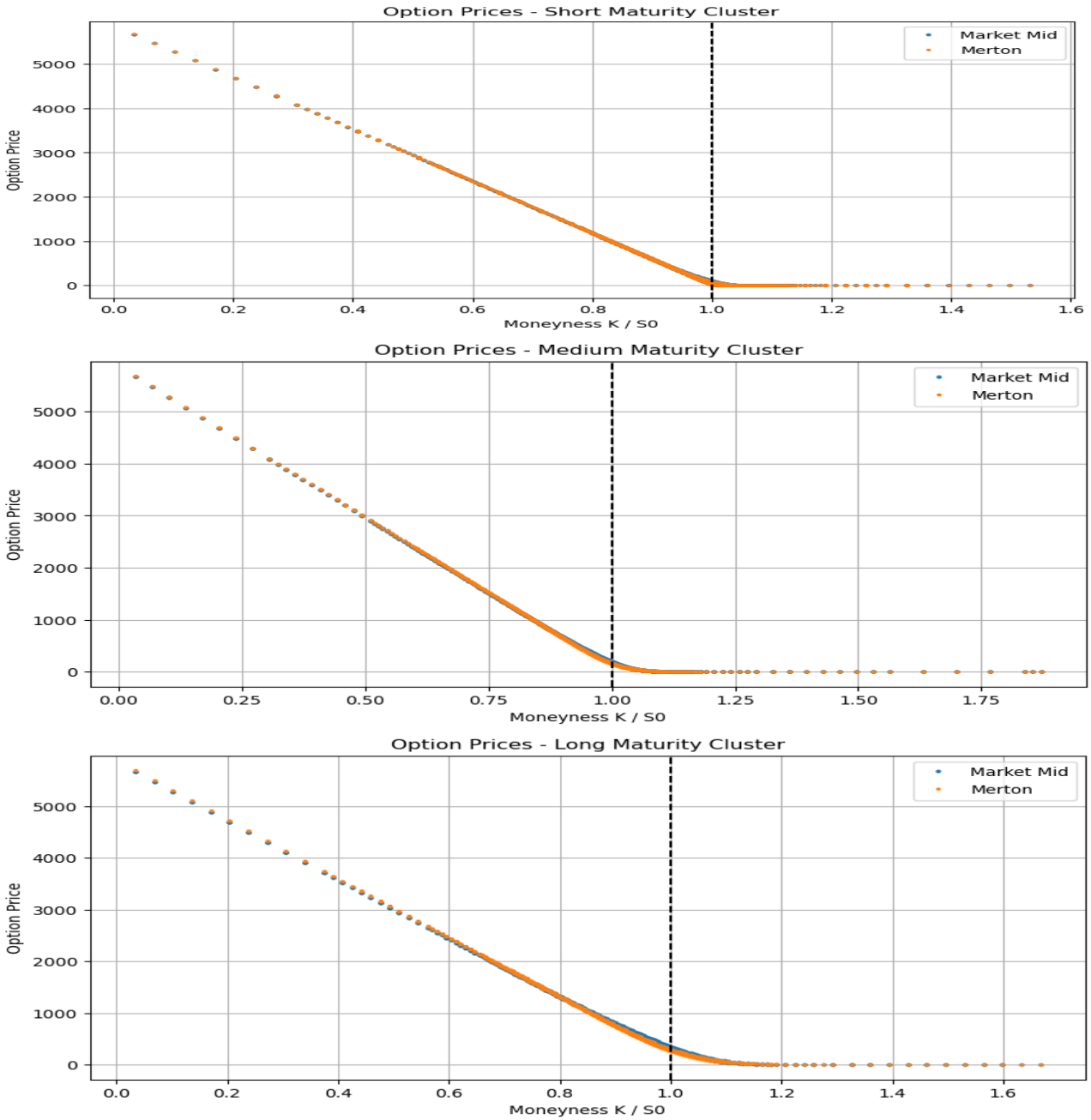
The Python code to carry out the algorithm is presented below: note that some constraints in values have been artificially imposed on the parameters, to reduce the computing time and complexity of the algorithm. Nevertheless, these numerical constraints can be relaxed.

```
def calibra_merton(df, S0):
    # Initial guess per i parametri: sigma, lambda, a, b
    initial_guess = [0.25, 1.0, -0.1, 0.2]
    # Bounds: (min, max)
    bounds = [
        (0.01, 1.0),     # sigma > 0
        (0.0, 5.0),      # lambda ≥ 0
        (-1.0, 1.0),     # a: mean of log-jump (it can be negative)
        (0.01, 1.0)      # b > 0
    ]
    # Minimization of Loss function
    result = minimize(
        loss_function,
        x0=initial_guess,
        args=(df, S0),
        bounds=bounds,
        method='L-BFGS-B',
        options={'disp': True, 'maxiter': 500}
    )
    # results
    if result.success:
        sigma_opt, lambda_opt, a_opt, b_opt = result.x
        print("☑ Solution found")
        print(f"sigma:   {sigma_opt:.4f}")
        print(f"lambda:  {lambda_opt:.4f}")
        print(f"a:       {a_opt:.4f}")
        print(f"b:       {b_opt:.4f}")
        return result.x
    else:
        print("✗ Solution not found")
        print(result.message)
        return None
```

The resulting theoretical prices compared with market prices are graphed in *Figure 10.*

[18] *Understanding scipy.minimize part 1: The BFGS algorithm*

Figure 10



We see that theoretical prices are well calibrated for most options, especially for those in the short maturity bucket. For the medium and long buckets, we see that the model prices are a bit below the actual market prices for *ATM* option but are in line with *OTM* and *ITM* options.

The parameters we found are interesting. We see a moderately high exposure to the continuous source of randomness, the Brownian motion, captured my parameter $\sigma = 10.6\%$. On the contrary, the parameter λ indicates a very weak exposure to jumps / discontinuous price changes. A parameter $\lambda = 0.03$ indicates an average 0.03 jumps per year, or differently put, an average of 3 jumps every 100 years. The option data on the $31/12/2024$ hence indicate a very low parameter for jumps. For comparison with other results in the literature, it is essential to keep in mind that this parameter has been calibrated on the market data on one single date, not on more complex panel data of options. The parameters governing the log-normal jump size a, b, as will be further analyzed in the next chapter, indicate an average downward jump of the underlying. This is reasonable given the many and massive negative price variations occurred during the COVID crisis.

3.2. Pricing European Calls using Merton Model

Having obtained the optimal calibrated parameters, we must now use these coefficients to simulate paths for the stock price according to Merton's stochastic differential equation. Using the Euler discretization method with the calibrated parameters as inputs it is indeed possible to obtain a matrix containing M simulated paths for the stock price with jumps. Since Merton's *stochastic differential equation* to describe the continuous dynamics of the asset price is the following:

$$\frac{dS(t)}{S(t^-)} = \mu dt + \sigma dW(t) + \Delta J(t)$$

We can shift to the risk neutral probability measure $\mathbb{Q}$ selecting one of the infinite possible *Equivalent martingale measures.* We will select the one where $\varphi = 1$ – indicating no jump risk premium, since $\lambda^{\mathbb{Q}} = \lambda^{\mathbb{P}}$ - and $\vartheta = \frac{\mu + (\mu_Y - 1)\lambda - r}{\sigma}$. With these two parameters we are guaranteed that the discounted stock price is a martingale process, as they *kill* the drift of the discounted price under the *EMM*, and we can rewrite the *stochastic differential equation* by applying the Girsanov's Theorem as:

$$\frac{dS(t)}{S(t^-)} = rdt + \sigma\left(dW^{\mathbb{Q}}(t) - \vartheta dt\right) + \Delta J(t)$$

Since $\sigma\vartheta = \mu + (\mu_Y - 1)\lambda - r$, we can rewrite this expression as:

$$\frac{dS(t)}{S(t^-)} = (r - (\mu_Y - 1)\lambda)dt + \sigma dW^{\mathbb{Q}}(t) + \Delta J(t)$$

By discretizing the infinitesimal differences, we can express this as:

$$\frac{\Delta S(t)}{S(t)} = (r - (\mu_Y - 1)\lambda)\Delta t + \sigma\sqrt{\Delta t} * Z + \Delta J(t)$$

The important component in this expression we need to look at for numerical analysis is $\boldsymbol{\Delta J(t)}$**.** The discrete variation of the compound Poisson process can take on two different values:

$$\Delta J(t) = \begin{cases} Y_j - 1 & \Delta N(t) = 1 \\ 0 & \Delta N(t) = 0 \end{cases}$$

Meaning that if a jump occurs at time t then the impact in size to the asset price is measured by the other lognormal random variable $Y_j \sim LogN(a, b^2)$.

Using the solution provided in the previous sections for the price $S(t)$ governed by the Merton's dynamics we can write the discretization Euler algorithm used to simulate M paths for the stock:

$$\begin{cases} S(t) = S(t - \Delta t)e^{\left(r - (\mu_Y - 1)\lambda - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t} * Z} & if\ \Delta N(t) = 0 \\ S(t) = S(t - \Delta t)e^{\left(r - (\mu_Y - 1)\lambda - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t} * Z} * (Y_j * \Delta N(t)) & if\ \Delta N(t) = 1 \end{cases}$$

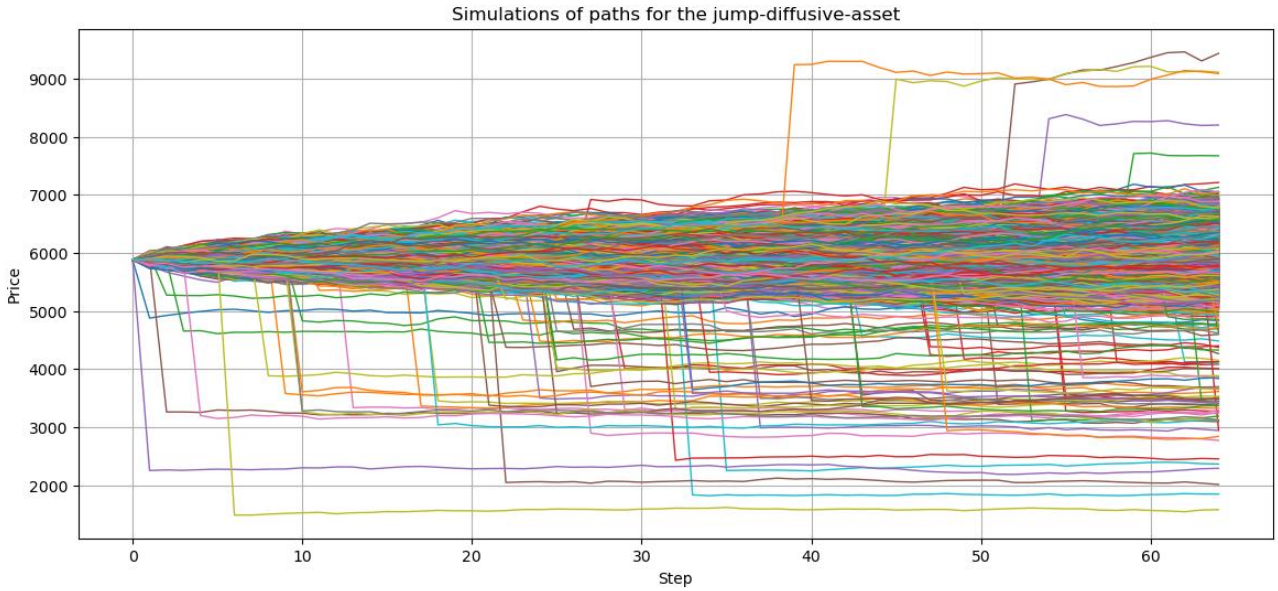
In fact, the last term is included to consider the potential jump occurred in Δt where the intensity parameter of the Poisson is $\lambda * \Delta t$. If a jump occurs in t then this will be reflected in the stock price at time t by the last multiplication term, and the size of this jump is determined by the realization of the lognormal variable Y_j.

By setting the proper Python function, we can create a matrix $A \in \mathbb{R}^{MxN}$ where the M rows are the number of simulated paths and the N columns are the discretized points in time in $[0, T]$ such that $\Delta t = T/N$.

3.2.1 Generation of matrix of simulated paths

Using the optimal calibrated parameters obtained by the minimization problem we obtain the following 10,000 simulated paths for the 64 trading days between the 31^{st} of December and 31^{st} of March (maturity of the option) and we plot it in *Figure 12*. So, we will consider the pricing of the call option maturing in March, for a time to maturity parameter $T = 0.247$. We used the prices also from other maturities for calibration purposes and now we focus on the option maturing the 31^{st} of March. Hence, we simulate the daily stock price under the risk neutral measure until that date.

Figure 12

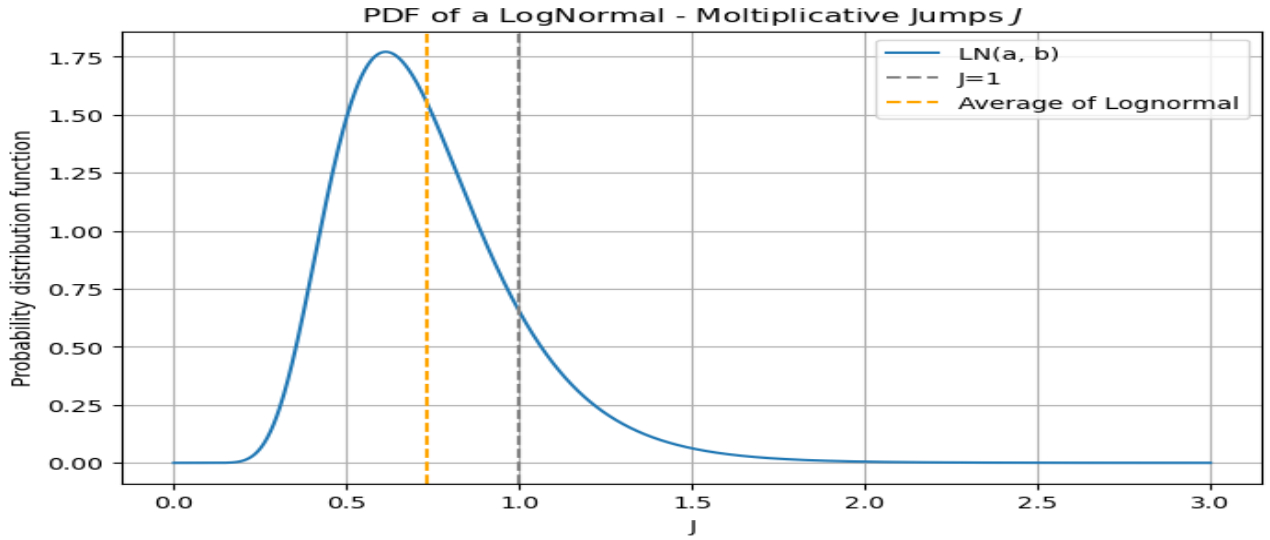


In this plot we can see that jumps are relatively rare and there is a much higher *continuous* variability: most of the variations of the stock price within the time framework are due to the *continuous* risk exposure to the Brownian motion rather than volatility induced by the compound Poisson process and hence by sudden jumps. This is due to the calibrated parameter $\lambda = 0.03$ which is indeed a very low coefficient compared to the volatility parameter $\sigma = 0.106$.

Given the properties of the lognormal distribution, we can also study the size jumps simulated for SPX. The expected value for the size of the jump is indeed $E[Y_j] = e^{a-\frac{1}{2}b^2} = e^{-0.369-\frac{1}{2}0.346^2} = 0.651$. On average, the jump occurred in the stock price is negative will have a 0.65x downward impact, as $S(\tau_i) = S(\tau_i^-) * Y_{\tau_i}$ and $E[S(\tau_i)|\mathcal{F}_{\tau_i^-}] = S(\tau_i^-) * \mu_{Y_{\tau_i}}$. Taking the initial price $S0 = 5881.63$ as a reference, the average impact of the jump would bring the price up to $5881.63 * 0.651 = \mathbf{3828.94}$.

We can also compute the probability of having a downward jump, which is easily achievable by taking the integral from $-\infty$ to 1 of the *probability distribution function* of the lognormal distribution Y. In *Figure 13* we show the PDF of this distribution highlighting with a grey vertical bar the right extreme of the integral to compute $\Pr[Y < 1] = \int_{-\infty}^{1} f_Y(j)dj$, where $f_Y(j)$ is the PDF of the lognormal.

Figure 13

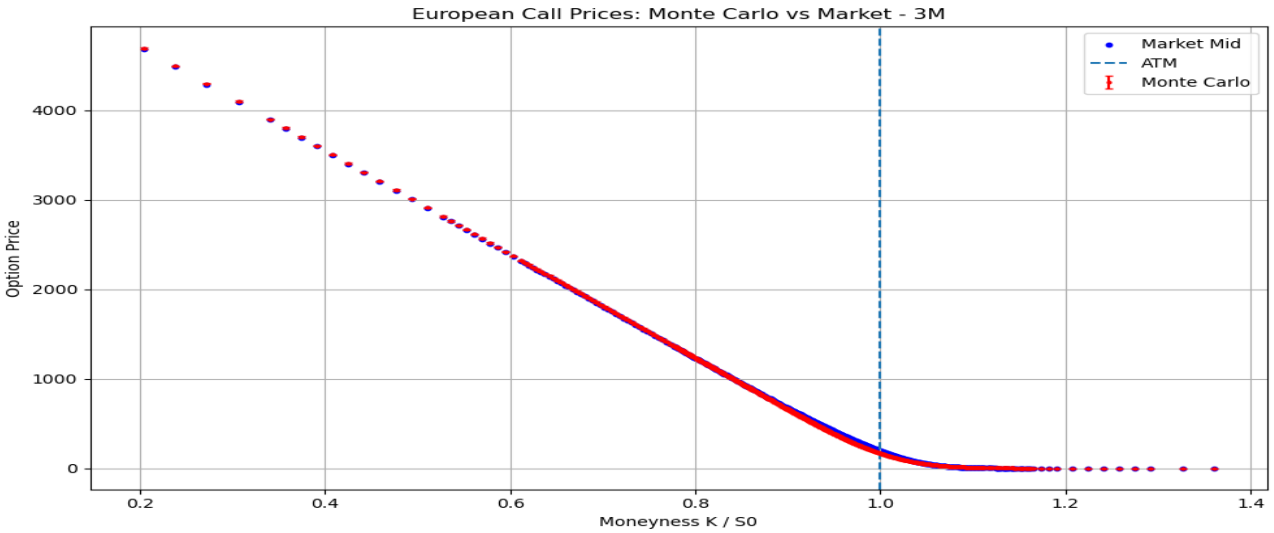


We see graphically that the integral would be big, and it is indeed equal to $0.857 \sim 85.7\%$.
We know have the matrix of simulated prices for the underlying. To compute the Monte Carlo estimator of the European plain vanilla call we need only the last column of the simulated prices since the payoff of this derivative is not path dependent. As explained in the previous sections, the formula for this consistent and unbiased estimator for the call price is the following:

$$C_{MC} = \frac{1}{M}\sum_{i=1}^{M}\left(S(T)^i - K\right)^+ e^{-rT}$$

Where $S(T)^i$ are all the M rows of the last column contained in the matrix of the simulations. We will also compute the standard error of this estimator and build an interval of confidence around the point estimate.
In *Figure 14* we can see the prices computed using Monte Carlo, including an interval of confidence, for the call option on SPX for different values of moneyness, plotted against the market prices for each call option. We see that the outcome of this numerical exercise is coherent with the theoretical prices for the European calls computed in the previous section: whereas the algorithm prices the ITM and OTM call correctly, it has a slight underpricing for the ATM calls. The radius of the confidence interval is very small thanks to the big number of simulations computed ($M = 10{,}000$).

Figure 14



3.3. Pricing Fixed Lookback Calls using Merton Model

3.3.1. Pure Monte Carlo estimator

We will now dig deeper into the pricing of the Fixed Lookback call using the Merton Model we have been developing in the previous sections. Initially, we will keep the intensity parameter constant as originally postulated by Merton, and only subsequently will we carry out the numerical exercise assuming that the counting process is a Hawkes process to allow a stochastic intensity parameter. The Fixed lookback call option has a path-dependent payoff of the following form:

$$payoff_T = \max\{(\max_t S(t) - K); 0\}$$

The main difference between the fixed and the floating lookback lies in the strike: in the fixed lookback, the strike is, indeed, fixed, and the payoff is given by the difference between the maximum price observed in the specific time horizon and the pre-specified strike K. The pricing of this product, rather than the floating counterpart, seems more logical so we can compare the call prices with the European calls for a given strike.
To carry out the pricing, we still use the same matrix $A \in \mathbb{R}^{MxN}$ which contains all the simulated prices. In contrast with the previous exercise, we now need to look at the whole matrix and not just the last column, as every simulated price matters to form the final payoff. The logic of the following algorithm is the following: for every row $i = 1 \dots M$ we save the maximum price achieved in that simulation and compute the payoff of the fixed lookback in accordance with the expression above. As usual, we then discount the payoff and take the average. This is the Python code to compute the pure Monte Carlo price and build an interval of confidence.

```
def monte_carlo_lookback_fixed(paths, K, r, T, alpha=0.95):
    S_max = np.max(paths, axis=1)
    payoffs = np.maximum(S_max - K, 0)
    discounted = np.exp(-r * T) * payoffs

    M = len(discounted)
    price = np.mean(discounted)
    se = np.std(discounted, ddof=1) / np.sqrt(M)

    from scipy.stats import t
    t_quantile = t.ppf(1 - (1 - alpha)/2, M-1)
    ci_lower = price - t_quantile * se
    ci_upper = price + t_quantile * se
    return price, se, (ci_lower, ci_upper), payoffs
```

The price for an At the Money ($K = 5881.63$) fixed Lookback call option on SPX with maturity in March 2025 is the following:

Price	270.54 $
Standard Error	2.19
Lower Bound (95%)	266.25 $
Upper Bound (95%)	274.84 $

We see that the price of the Lookback is significantly higher than the one for ATM European call ($C(0) = 169.37$ $) and this is perfectly logical: the payoff of a fixed lookback is greater or equal than the payoff of a European call, since it does not take the final price of the stock at time T but rather it takes the maximum in the whole period $[0; T]$. Intuitively, the payoff of the lookback cannot be smaller than the one of the European call. Hence, since the price of an option is a conditional expected value of the payoff, weighted by the probabilities of each outcome, the higher the payoff the higher the price.

The only issue we can find in this procedure is the radius of the confidence interval; it is not extremely wide, but still it can be improved. To do so, we introduce a **Variance reduction technique** used to reduce the uncertainty regarding a Monte Carlo estimator for the same number of simulations carried out. The three main techniques we can use are the following: Control Variate, Antithetic Variate and Quasi Monte Carlo. For this analysis we will carry out the first technique for the following reasons: Antithetic Variate technique is helpful because for a one simulation of Z you save both Z and $-Z$, but it may be problematic because of the asymmetry of the stock distribution because of the presence of jumps (we have argued that jumps are asymmetric as the expected impact is 1.2x). Quasi-Monte Carlo technique is also very helpful as it allows the matching of the first moment by changing the weights given to each simulation but is very burdensome when dealing with path-dependent options like the one we are pricing.

3.3.2. Control Variate estimator to reduce variance

The logic of the Control Variate variance reduction technique is the following. Given our target variable Y, we wish to estimate, by Monte Carlo, the unknown expected value $E[Y]$. We can exploit the information we have about another variable X, whose $E[X]$ is instead known, to reduce the error in the estimation of the *unknown* quantity Y. The Control Variate Estimator has the following structure:

$$Y_i(b) = Y_i - b(X_i - E[X])$$ [19]

$$\widehat{Y_n}(b) = \frac{1}{n}\sum_{i=1}^{n} Y_i(b)$$

Instead of taking the values of Y_i, we apply a simple correction and take $Y_i(b)$ to make the estimator instead. This estimator will have the benefit of having a reduced variance and standard error, whilst still preserving the properties of unbiasedness and consistency. The last two properties heavily rely on the choice of the variable X: this estimator must be unbiased and consistent as well.

$$E\left[\widehat{Y_n}(b)\right] = E[\frac{1}{n}\sum_{i=1}^{n} Y_i - b(X_i - E[X])] = E[Y] - b(E[X] - E[X]) = E[Y]$$

This expression only holds if $E\left[\widehat{X_n}\right] = E[X]$, when the average of the sample mean of X is truly the true parameter $E[X]$. If a bias is present in the X variable, it will be kept in the Control Variate Monte Carlo estimator as well. The same logic can be used to assess the consistency of the Control Variate MC estimator, and in this case the property is preserved only if $\lim_{n\to\infty} \widehat{X_n} = E[X]$.

So, the estimator is unbiased, and we can prove that the variance of this new estimator is lower than the one of the *pure* Monte Carlo as long as the two variables X and Y have a correlation different from zero, $|\rho_{XY}| \neq 0$. In the estimator formula we see the usage of a coefficient b to link the variable X and Y, and we need to establish a criterion, a rule to set the optimal parameter b^*. The most immediate and natural logic would be to pick the coefficient b^* such that the variance of each replication $Y_i(b)$ is minimized:

$$\min_b Var(Y_i(b))$$

The variance of the individual replication can be rewritten as follows:

$$Var(Y_i(b)) = Var(Y_i - b(X_i - E[X])) = Var(Y_i) + b^2 Var(X_i - E[X]) - 2bCov(Y_i; X_i - E[X])$$

Hence, the variance of the single replication of control variate estimator is a function of b; in particular, we can easily see that it is an upward parabola. Hence, to find the optimal parameter b^* we can take the first derivative of the variance function and set it equal to 0.

[19] Simulation efficiency and an Introduction to Variance Reduction Methods, 2017, Martin Haugh, Columbia University

$$\frac{\partial Var(Y_i(b))}{\partial b} = 0$$

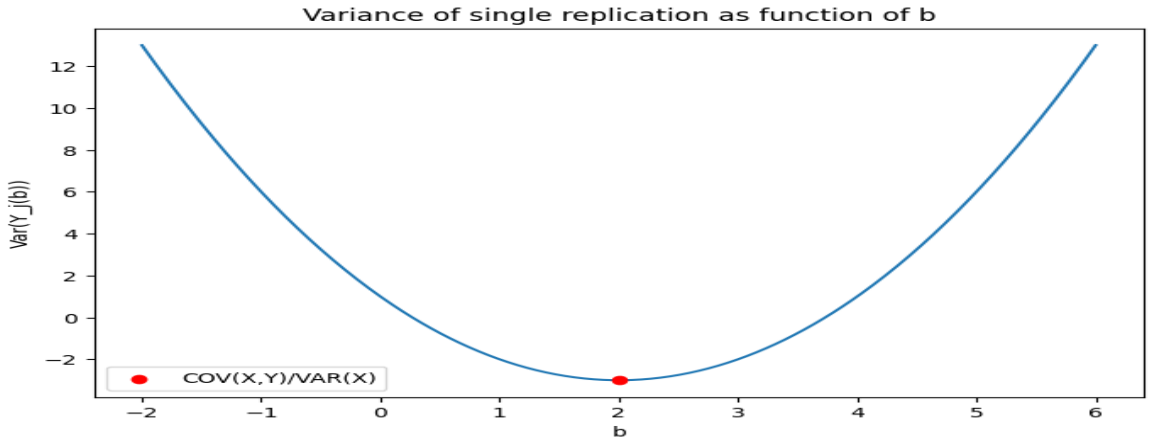
$$\frac{\partial(b^2\sigma_X^2 - 2b\sigma_X\sigma_Y\rho_{XY} + \sigma_Y^2)}{\partial b} = 0$$

$$2b\sigma_X^2 - 2\sigma_X\sigma_Y\rho_{XY} = 0$$

$$b^* = \frac{2\sigma_X\sigma_Y\rho_{XY}}{2\sigma_X^2} = \frac{Cov(X,Y)}{Var(X)}$$

This outcome can be visualized in *Figure 15*

Figure 15



The optimal coefficient b^* is thus the *OLS* coefficient between the series of Y_i taken as the target variable and X_i as the independent variable. To get this coefficient it will be enough to run a simple univariate regression between these two variables including the intercept.

The control variate estimator has variance $Var\left[\widehat{Y_n}(b^*)\right] = Var\left[\frac{1}{n}\sum_{i=1}^{n} Y_i(b^*)\right] = \frac{1}{n^2} nVar\left(Y_i(b^*)\right) = \frac{Var(Y_i(b^*))}{n}$, where $Var(Y_i(b^*))$ is the variance of the individual, single simulation according to the Control Variate Estimator. Using the same properties of the variance operator, we can come up that the variance of the pure MC estimator is $\frac{Var(Y_i)}{n}$, where $Var(Y_i)$ is, again, the variance of the individual simulation. Hence, to compare the variances of the two estimators we need to assess whether and when:

$$Var\left(Y_i(b^*)\right) < Var(Y_i)$$

If this inequality is met, then we have an actual advantage by using the control variate rather than the pure Monte Carlo. By using the decomposition of $Var\left(Y_i(b^*)\right)$ already used above we can further rewrite this as $\sigma_Y^2(1 - \rho_{XY}^2)$. Thus,

$$\sigma_Y^2(1 - \rho_{XY}^2) < \sigma_Y^2$$

If and only if:

$$|\rho_{XY}| > 0$$

To conclude, we can say that it is reasonable to use the control variate technique when we use a variable which is correlated, either positively or negatively, with the target variable.
Now, we can look for a control variate for the pricing of the fixed lookback within the Merton Model. Again, this variable must be both correlated with our target and must also have a known expected value $E[X]$. In our case, we

selected as X variable the price of the European plain vanilla call options using the Merton jump-diffusive asset. The choice is reasonable because the two prices are extremely highly correlated ($\rho_{XY} = 0.909$) and because, as we have shown in the previous sections, the expected value of the price of the European calls is indeed known and given by the closed end formula:

$$C_0 = \sum_{n \in N} bsc(S(0), r_n, \sigma_n) * e^{-\lambda' T} \frac{\lambda' T^n}{n!}$$

Again, we will deal with an ATM option, whose strike is $K = 5881$. The optimal b^* computed by the linear regression between X and Y is close to unity, equal to 1.06. Nevertheless, we will not be using this coefficient for our estimator to avoid adding bias in the estimate.

Since the true regression parameter β_{OLS} is not known, we will use the sample counterpart of the same coefficient $\widehat{\beta_{OLS_n}} = \frac{\widehat{Cov(X,Y)}}{\widehat{Var(X)}}$. From traditional econometric theory, we know from the Gauss-Markov theorem that, under the *OLS Assumptions*, this estimator is the *B.L.U.E* - Best linear unbiased estimator – and is also consistent with the true population parameter. Despite all these nice statistical properties, we find that using $\hat{Y}_n(\hat{b}_n)$ rather than $\hat{Y}_n(b^*)$ introduces a positive bias in the estimate for $E[Y]$. In fact:

$$E\left[\hat{Y}_n(\hat{b}_n)\right] - E[Y] = E[\hat{Y}_n - \hat{b}_n(\hat{X}_n - E[X])] - E[Y] = -E[\hat{b}_n(\hat{X}_n - E[X])] + E[Y] - E[Y]$$

If $\hat{b}_n$ and $\hat{X}_n$ were independent, then the bias would be zero. Because $E[X * Y] = E[X] * E[Y]$ if the two variables are independent, we could indeed *break* the expectations operator in $E\left[\hat{b}_n\right] * E\left[\hat{X}_n - E[X]\right] = 0$, as the second term is, by unbiasedness of the control variate, equal to 0. Because we estimate $\hat{b}_n$ from the whole sample, we use the same data to both compute $\hat{b}_n$ and $\hat{X}_n$, making them not independent from each other and inducing a bias. To avoid this bias, we simply need to come up with an algorithm that allows $\hat{b}_n$ and $\hat{X}_n$ to be independent from each other. A procedure to do so would be to divide the dataset into two sub-datasets composed by n_1 and n_2 datapoints, respectively, and use the two datasets for the two different purposes. We would use the first n_1 observations of X and Y to compute the coefficient $\hat{b}_{n_1}$, and the remaining n_2 simulations of X and Y to compute the estimator:

$$Y_i(\hat{b}_{n_1}) = Y_i - \hat{b}_{n_1}(X_i - E[X])$$

$$\hat{Y}_{n_2}(\hat{b}_{n_1}) = \frac{1}{n_2} \sum_{i=1}^{n_2} Y_i(\hat{b}_{n_1})$$

By using a portion of the dataset only to compute the $\hat{b}_{n_1}$ and the rest to compute the estimate guarantees that $\hat{b}_{n_1}$ and $\hat{X}_n$ are independent, since they are computed on different datapoints, different simulations. By using this procedure, we avoid adding bias in our estimate. The following Python code is the algorithm used to price the *At-The-Money* Fixed Lookback call using this procedure, taking the prices of the European plain vanilla as control variate to reduce the variance of the estimator.

```
def stima_control_variate_unbiased(payoffs_Y, payoffs_X, call_theo, r, T, alpha=0.95,
split_ratio=0.3):
    M = len(payoffs_Y)
    n1 = int(M * split_ratio)
    # Split dataset
    Y1, Y2 = payoffs_Y[:n1], payoffs_Y[n1:]
    X1, X2 = payoffs_X[:n1], payoffs_X[n1:]
    # linear regression
    X1_reg = sm.add_constant(X1)
    model = sm.OLS(Y1, X1_reg).fit()
    beta_hat = model.params[1]
    # formula
    Y2_star = Y2 - beta_hat * (X2 - call_theo)
    # Discount the payoffs
    discounted = Y2_star
    N = len(discounted)

    # compute price, standard error and interval of confidene
    price = np.mean(discounted)
    se = np.std(discounted, ddof=1) / np.sqrt(N)
    t_quantile = t.ppf(1 - (1 - alpha)/2, N - 1)
    ci_lower = price - t_quantile * se
    ci_upper = price + t_quantile * se
    print(f"🎯 Price lookback (Control Variate Unbiased): {price:.4f}")
    print(f"📊 Standard Error: {se:.6f}")
    print(f"📉 CI {int(alpha*100)}%: ({ci_lower:.4f}, {ci_upper:.4f})")

    return Y2_star, price, se, (ci_lower, ci_upper)
```

By using this algorithm, we come up with the following values:

Price	273.56 $
Standard Error	1.11
Lower Bound (95%)	271.39 $
Upper Bound (95%)	275.73 $

As expected, the point estimate of the control variate estimator is very close to the estimate already obtained with the pure Monte Carlo technique, but the standard error of this estimate is much smaller than before: the interval of confidence is much narrower, indicating more precision and confidence in the price provided. Again, the price of the At-the-money Lookback is higher than the corresponding price for the At-the-money European Call, and, as explained above, this makes perfect sense.

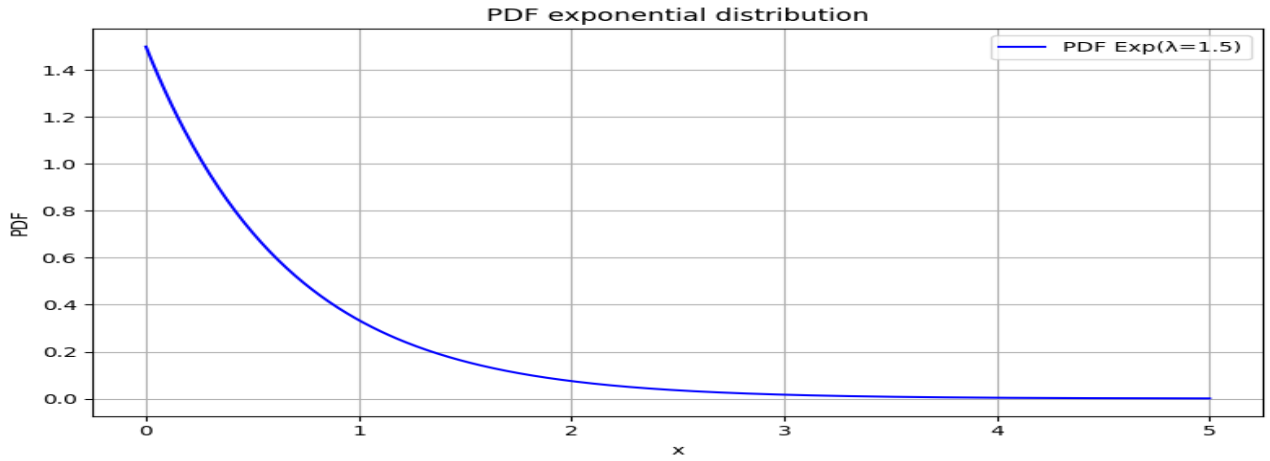
3.4. Extension to a non-homogenous, stochastic intensity parameter λ

After having developed a solid procedure to price path-dependent, lookback options using the risk-neutral framework proposed by Merton, we now extend our model to relax the assumption of constant intensity parameter. As we have stressed in the previous sections, the classical framework sees the counting process $N(t)$ governed by a Poisson distribution with parameter $\lambda > 0$: the *Probability distribution function* of the number of jumps occurred within time t can be expressed as:

$$\Pr[N(t) = n] = \frac{e^{-\lambda t}(\lambda t)^n}{n!}$$

Where the waiting periods between jumps $\tau_j - \tau_{j-1}$ are independent random variables distributed according to an exponential distribution with parameter λ. The important feature of this process is that each event occurs independently from each other, and there is no statistical relationship between different events (jumps) at different times. This distribution is decaying overtime, according to the following graph:

Figure 16



While this is a natural and intuitive assumption, it does not match the real dynamics of stock prices. The occurrence of a jump is often followed, empirically, by other jumps. A reasonable explanation for this phenomenon is information asymmetry among investors. As presented by Chen, Clements and Urquhart[20], there is substantial clustering in price and volatility jumps in high frequency, intraday data, which lasts between 2.5 and 6 hours on average. As investors are differently informed about firm's financial status or, in general, macro-finance news, there may be a mistiming of actions between investors, causing multiple jumps at different times after a certain news is released. If a company were to publish negative quarterly results, the most aware and informed investor would act immediately, causing an immediate price variation. Other investors may come behind, inducing a subsequent jump within the next hours or even days. Statistical tests to verify the need for an introduction of self-exciting behavior across jumps is beyond the purpose of the paper. Many tests have been carried out in other papers, as in the one cited above, proving the existence of self-exciting behavior. Our goal is to assess how the pricing of exotic, path-dependent options changes when we relax the assumption of independence between jumps in a Merton framework. We will thus use a univariate Hawkes process – since we are not dealing with rainbow options, otherwise we would have needed a multivariate process, as counting process $N(t)$ for our model. We will choose the exponential specification for the kernel of the stochastic intensity parameter λ, as it easily managed by the two parameters α and β. Our intensity parameter will thus have the following specification:

$$\lambda(t) = \lambda(0) + \sum_{m=0}^{M} \varphi(t - t_m)$$
$$with \qquad \varphi(t - t_m) = \alpha e^{-\beta(t-t_i)}$$

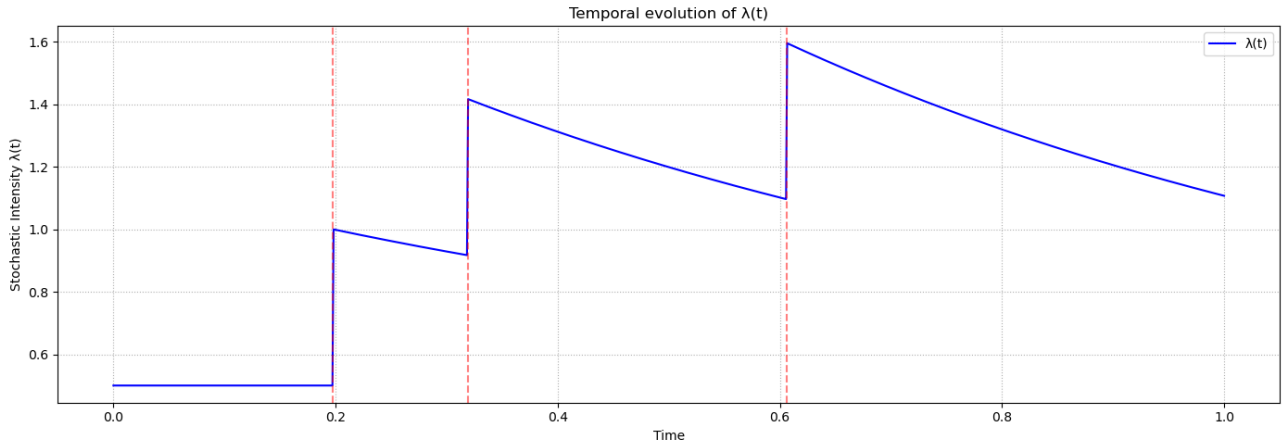
[20] Modelling variance and price jump clustering using the Marked Hawkes Process; Chen, Clements and Urquhart, Journal of Financial Econometrics, 2023

Where M is the total number of events occurred before time t, and t_m indicates the time of each event. The algorithm to simulate a path of this stochastic intensity parameter is the following:

```
def simulate_hawkes(mu, alpha, beta, T, dt=0.01):
    t = 0
    jumps = []
    lambda_t = mu
    history = []
    while t < T:
        u = np.random.uniform()
        s = -np.log(u) / lambda_t
        t += s
        lambda_bar = mu + np.sum(alpha * np.exp(-beta * (t - np.array(history))))
        d = np.random.uniform()
        if d <= lambda_bar / lambda_t:
            if t <= T:
                history.append(t)
                jumps.append(t)
            lambda_t = lambda_bar
        else:
            lambda_t = lambda_bar
    return np.array(jumps)
```

And an example of a simulated path for $\lambda(t)$ is in *Figure 17*

Figure 17



The red vertical lines represent the occurrence of a jump in the price (meaning that $\Delta N(t_j) = 1$ in those periods), adding up to a total of 3 jumps, $N(1) = 3$. Other than having a jump in the price, we see that we also have a jump in the intensity parameter itself: thanks to the specification of the kernel, the parameter λ has a self-exciting memory of past jumps, meaning that if a jump occurs at a given time, the probability of observing another jump in the following periods increases. The parameters α and β control the dynamics of this process: α indicates the strength of the jump in the intensity parameter in the occurrence of a jump induced by the intensity parameter at the previous date ($\Delta N(t_j) = 1$); high values for α indicate high jumps. Parameter β indicates the strength of the decay once the jump has occurred. The function of λ as a function of time must be non-increasing in the non-jump periods to maintain the stationarity condition – otherwise the probability of jumps would keep on increasing. High values for β indicate that the decay is slower and takes more time.

3.4.1. Ogata's thinning algorithm

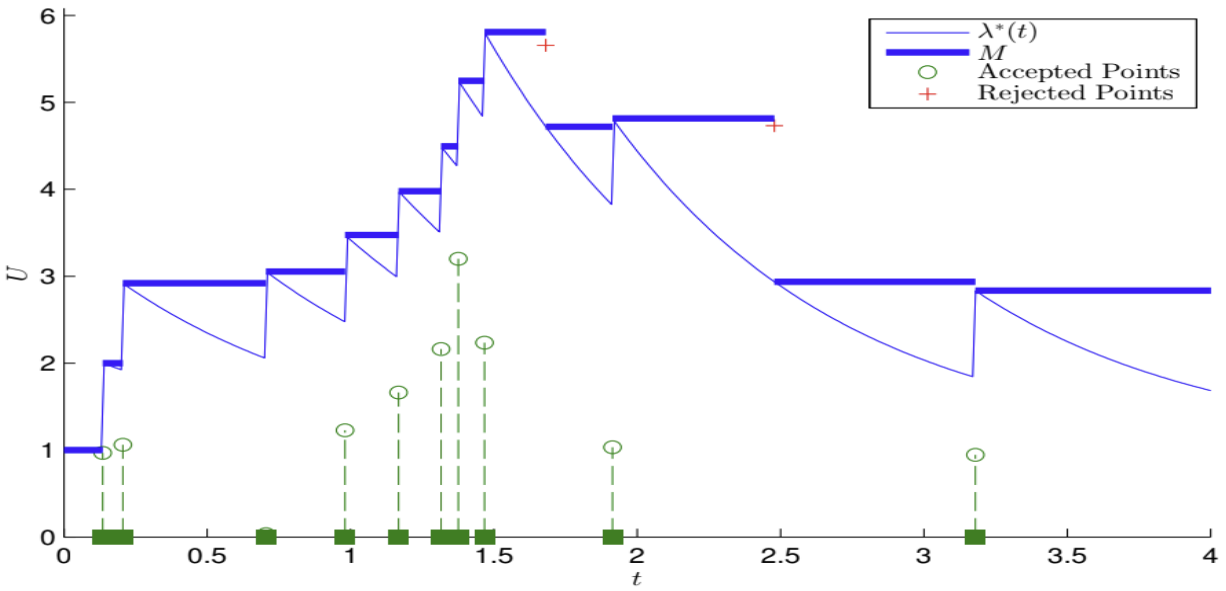
We must now get deeper in how we were able to simulate and generate this process for the stochastic intensity parameter. A well-known algorithm used is called the Ogata's Algorithm[21], introduced in 1981 as a thinning algorithm, and used in the previous code provided. The logic of this algorithm is to use a "*faster*" *homogenous* Poisson process, by imposing a higher intensity parameter lambda, and to later remove candidate jumps that are not aligned, in probabilistic terms, with the actual intensity parameter governed by the Hawkes Process. We will thus set an upper bound M, which moves over time as well in occurrence of jumps, and we will use this parameter to simulate the value $E = \tau_{i+1} - \tau_i$, that represents the waiting period between two jumps and can be sampled from an exponential distribution with intensity parameter M. The precise steps of the algorithm are the following[22]:

1. We start from initial time t, we set a small value $\varepsilon = 10^{-5}$, and we set our initial upper bound M to be equal to $M = \lambda(t + \varepsilon)$. This value will be kept constant until the next candidate time τ_i for a jump.
2. We sample the next candidate point $E = \tau_{i+1} - \tau_i \sim Exp(M)$ and we set the new time index equal to $t = t + E$. All the periods in between these two periods will not be considered explicitly by the algorithm. The important thing to keep in mind is that the M will remain constant in the interval, whilst our actual intensity parameter $\lambda(t)$ will slowly decay because of the presence of the coefficient β.
3. The new time t is our next candidate point to have a jump. We must now remember that this point has been generated using the *homogenous intensity parameter M,* which is indeed higher than the actual intensity parameter we are trying to simulate. Thus, we need a criterion to decide whether this same candidate jump time *would have been generated by the lower intensity parameter λ(t) eventually*. The algorithm thus proceeds in generating a random point U^* from a Uniform distribution $U[0:M]$: the candidate point will be kept as a jump time only if this randomly generated value U^* is lower than the actual intensity parameter at the candidate time $\lambda(t)$. If $U^* < \lambda(t)$, then we accept the candidate point as a true jump, and we will have a jump there. Otherwise, the candidate point is rejected and there will be no jump at that time.
4. Regardless of that, by any candidate jump point t we will update our new maximum bound $M = \lambda(t + \varepsilon)$. Now the procedure starts over, generating a random value for the exponential and updating the time index accordingly.

The final output of this algorithm is to generate a simulated path for λ overtime and also, accordingly, a realization of the Hawkes process $N(t)$. The jumps counted by the $N(t)$ must, always, coincide with the jumps *implied* by the path of λ: whenever we see a jump in the intensity parameter, it means that an actual jump for the prices process has occurred, and this is counted by $N(t)$.
Figure 19 helps with the visualization of the whole process:

[21] Hawkes processes: Simulation, Estimation and Validation; Patrick Laub, 2014; The University of Queensland, Australia
[22] Hawkes processes: Simulation, Estimation and Validation; Patrick Laub, 2014; The University of Queensland, Australia

Figure 19[23]



(b) Hawkes process

In the graph, the blue thick horizontal line indicates the upper bound M explained above; we see that it gets updated any time we draw a candidate jump point. The blue thin line represents the intensity parameter $\lambda(t)$; we see that it gets updated upward only in proximity of green void circles, whereas it keeps decaying in proximity of the red crosses. This is because some candidates points are accepted as jump times – with the green void circle, as we see that the sampled value from the Uniform distribution is smaller than intensity parameter, $U(t^*) < \lambda(t^*)$, where t^* is the accepted jump time; On the contrary, we can spot also two candidates points that have been rejected – with the red cross, because we see visually that the sampled Uniform lies above the respective intensity parameter, $U(t^{**}) > \lambda(t^{**})$, where t^{**} is the rejected jump time. The occurrence of a jump triggers in fact a spike in the intensity parameter, while the rejected points do not have any Impact on the parameter. Nevertheless, the latter have an impact on the M because, as stressed in the previous description, the upper bound gets updated iteratively.

3.4.2. Simulation under the $\mathbb{P}$ probability measure

Before getting into the pricing algorithms and simulations, it is crucial to find optimal parameters α^* and β^* of the Hawkes process to later be used in our Monte Carlo simulations. The estimate of these two parameters is troublesome: we set up a loss function comparing the actual path of SPX index value and the paths simulated in the $\mathbb{P}$ probability measure in the same period. The loss is the following:

$$\hat{L}(\alpha,\beta) = \frac{1}{T}\sum_{t=1}^{T}(lnS_t^{MC}(\alpha,\beta) - lnS_t^{MKT})^2$$

Where $T = 31/12/2024$ and $t = 1$ is the beginning of our sample $01/01/2019$. So, we are trying to calibrate these two parameters under the physical probability measure with the historical path of the SPX index. Instead of relying on the single simulated path by our model, we want to use an average path, so for each time t in our discretization, we will simulate more draws and then take the average value as our Monte Carlo estimate:

$$S_t^{MC}(\alpha,\beta) = \frac{1}{M}\sum_{j=1}^{M} S_t^j(\alpha,\beta)$$

[23] Hawkes processes: Simulation, Estimation and Validation; Patrick Laub, 2014; The University of Queensland, Australia

Where M is the number of simulations.

We will create an algorithm to simulate the path for SPX in the historical time frame observed $[01-01-2019; 31-12-2024]$ by discretizing the *stochastic differential equation* proposed by Merton – using the Euler's discretization scheme presented in the introduction. We will allow the intensity parameter to vary stochastically following the dynamics of a Hawkes process with exponential kernel, and we will find the parameters α^* and β^* that minimize our defined loss function.

$$(\alpha^*, \beta^*) = argmin\ \hat{L}(\alpha, \beta)$$

The issue with this path matching is that the loss function is very noisy. In fact, since the MC estimator included in the definition of this function is, indeed, stochastic, the loss function itself will also keep this stochastic feature. For a fixed couple of parameters (α, β) we will have different values of loss function depending on the seed that the algorithm is working on. This is clearly problematic when optimizing since, when you find parameters to be the minima, you do not whether they truly minimize the function or the loss function value is low because of noise.

A crucial notion that needs to be dealt with in this section is the probability measure we will employ to carry out this simulation. As we have stressed in previous sections, the risk-neutral measure $\mathbb{Q}$ is employed in option pricing because it guarantees that the discounted stock price is a martingale and no arbitrages are introduced in the market formed by the risky security and the risk-free asset. Under this probabilistic measure, the coefficients introduced in Girsanov's Theorem (θ, φ) are set so that the drift of the discounted price process is equal to 0, to reflect the fact that investors are assumed to be risk neutral and thus require no risk premium for their investments. In the framework we will develop now, there is no need for this assumption as we are not trying to build an arbitrage-free market to correctly price options, rather we want to define a stochastic dynamic for the stock price to accurately replicate the path that the stock followed within the real world, where investors are far from neutral to risk. Thus, it is justified to develop our framework within the physical probability measure $\mathbb{P}$, where the SDE by Merton is:

$$\frac{dS(t)}{S(t^-)} = \mu dt + \sigma dW(t) + \Delta J(t)$$

Where μ is the mean return of the stock price (annualized). Using Ito's lemma to solve this differential equation – as shown in the previous sections, we can rewrite the discretized expression of the evolution of $S(t)$ as:

$$\begin{cases} S(t) = S(t-\Delta t)e^{\left(\mu-\frac{1}{2}\sigma^2\right)\Delta t+\sigma\sqrt{\Delta t}*Z} & if\ \Delta N(t) = 0 \\ S(t) = S(t-\Delta t)e^{\left(\mu-\frac{1}{2}\sigma^2\right)\Delta t+\sigma\sqrt{\Delta t}*Z} * (Y_j * \Delta N(t)) & if\ \Delta N(t) = 1 \end{cases}$$

Where $N(t)$ is the Hawkes process. The code to simulate a path in this Merton-Hawkes model is the following:

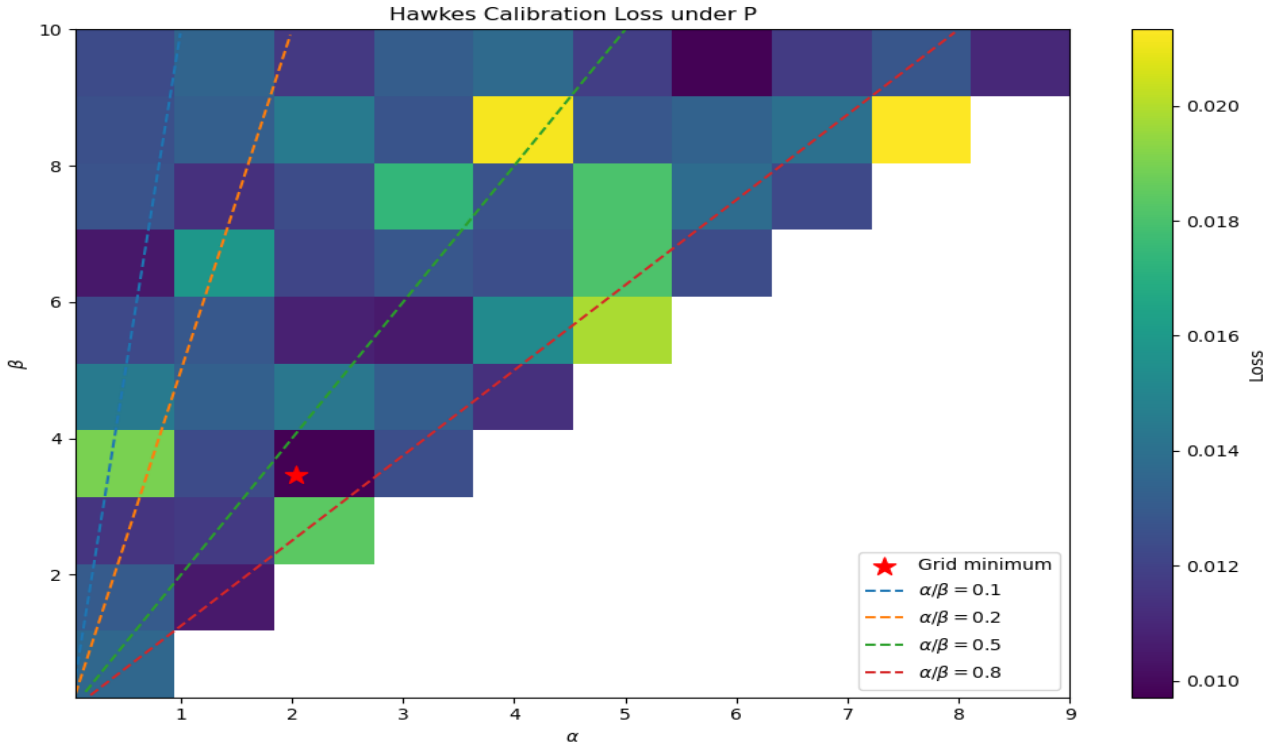
```
def simulate_hawkes_series(mu, alpha, beta, trading_days, dt, rng=None):
    if rng is None:
        rng = np.random.default_rng()
    T = len(trading_days)
    lambda_series = np.zeros(T)
    lambda_t = mu
    jump_times = []
    decay = np.exp(-beta * dt)
    for t in range(T):
        lambda_series[t] = lambda_t
        jump = rng.random() < lambda_t * dt
        if jump:
            jump_times.append(t)
        lambda_t = mu + (lambda_t - mu) * decay + alpha * int(jump)
    return lambda_series, np.array(jump_times)

def simulate_jump_diffusion_hawkes(S0, avg_ret, sigma, a, b, mu, alpha, beta, trading_days, dt):
    lambda_series, jump_times = simulate_hawkes_series(mu, alpha, beta, trading_days, dt)
    N = len(trading_days)
    S = np.zeros(N)
    S[0] = S0
    k = np.exp(a + 0.5 * b**2) - 1
    for t in range(1, N):
        Z = np.random.normal()
        if q_probab_indicator:
            drift = (r - lambda_series[t-1] * k - 0.5 * sigma**2) * dt
        else:
            drift = (avg_ret - 0.5 * sigma**2) *dt
        jump = np.random.lognormal(a, b) if t in jump_times else 1.0
        S[t] = S[t-1] * np.exp(drift + sigma * np.sqrt(dt) * Z) * jump
    return S, lambda_series, jump_times
```

We will thus try to find the optimal parameters α^* and β^* by making simulations with different pairs of values to see which pair of parameters gives us the most realistic sets of simulations. The first procedure to find this couple of parameters is through a heatmap. We discretize the parameter space $\theta_\alpha \times \theta_\beta$ and for a discrete grid of values of both parameters we compute the loss function in each point of this bidimensional grid. We take as the minimum the point with the lowest loss function. Because of high computational cost, few points on the grid were chosen: $10\ x\ 10 = 100$ points, and 100 Monte Carlo simulations were carried out for each evaluation of the loss function. This approach is also very useful to make sure that we can identify both α and β, and not only their ratio $\frac{\alpha}{\beta}$. A common problem in these econometric settings is when two different parameters vectors result in the same reduced-form model, which is known as wrongful identification of the model. If we see a common trend along the same fictitious line on this space, like $\frac{\alpha}{\beta} = k, for\ some\ k$, it means that different couples of parameters with common ratio provide the same loss function value, indicating that the model can correctly identify only the branching ratio of the parameters, and not those individually.

The heatmap of this procedure is reported in *Figure 20.*

Figure 20



We observe no clear trend along those candidate lines; hence we can conclude that we can identify those two parameters. The white boxes are points of the grid where stationarity of Hawkes is not met. We find the global minimum of the loss function to be:

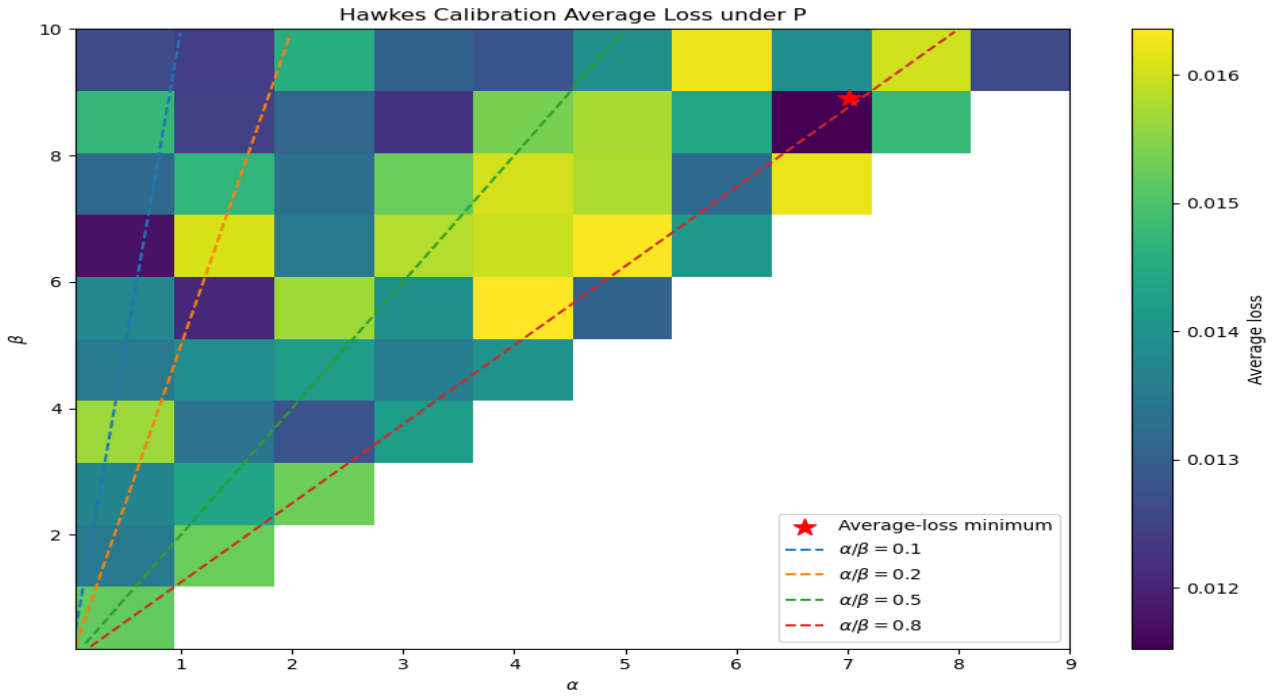
α^*	2.0389
β^*	3.4667
$Loss^*$	0.0097
α^*/β^*	0.5881

Thus, this is the couple of parameters we would find by minimizing the loss. The persistent issue is that the loss function we just minimized is stochastic and keeps changing for different simulations keeping these optimal parameters (α^*, β^*) constant, and we have not kept this feature into account in this minimization. A more sound and stable approach is to include the fact that this loss function is stochastic for different seeds and hence try to minimize not the single loss function, but the average loss function per different seeds. In this way we can reduce the noise in the parameter estimation, even if a significant portion of noise is inherent to this estimation problem. The loss function is now equal to:

$$\widehat{L^*}(\alpha,\beta) = \frac{1}{S}\sum_{s=1}^{S}\frac{1}{T}\sum_{t=1}^{T}(lnS_{t,s}^{MC}(\alpha,\beta) - lnS_t^{MKT})^2$$

Where S is the number of seeds used and $lnS_{t,s}^{MC}(\alpha,\beta)$ is still the Monte Carlo estimator for a given time t and a given seed s. We now apply the same logic as above to construct a new heatmap for the discretized parameter space and for each point in this grid we compute this modified version of average loss $\widehat{L^*}(\alpha,\beta)$ and the corresponding standard deviation. We then take as the minimum of the loss function the couple of parameters (α,β) resulting in the lowest average loss value. This heatmap is presented in *Figure 21*.

Figure 21



We find the global minimum of thiss loss function to be:

α^*	7.0111
β^*	8.9111
$Avg.Loss^*$	0.0115
$StDev.Loss^*$	0.0014
α^*/β^*	0.7868

We see that the parameters are slightly different in the two procedures, and we select the latter couple of parameters estimated as the procedure is more solid. Nevertheless, this estimation is noisy by definition and, despite the efforts to reduce the noise, a significant portion of it is still present.

Using the vector of parameters $(\alpha^*, \beta^*) = (7.0111, 8.9111)$ produces simulations can be conducted to the empirical observed path for SPX. Two examples of paths are *Figure 22* and *Figure 23*.

Figure 22

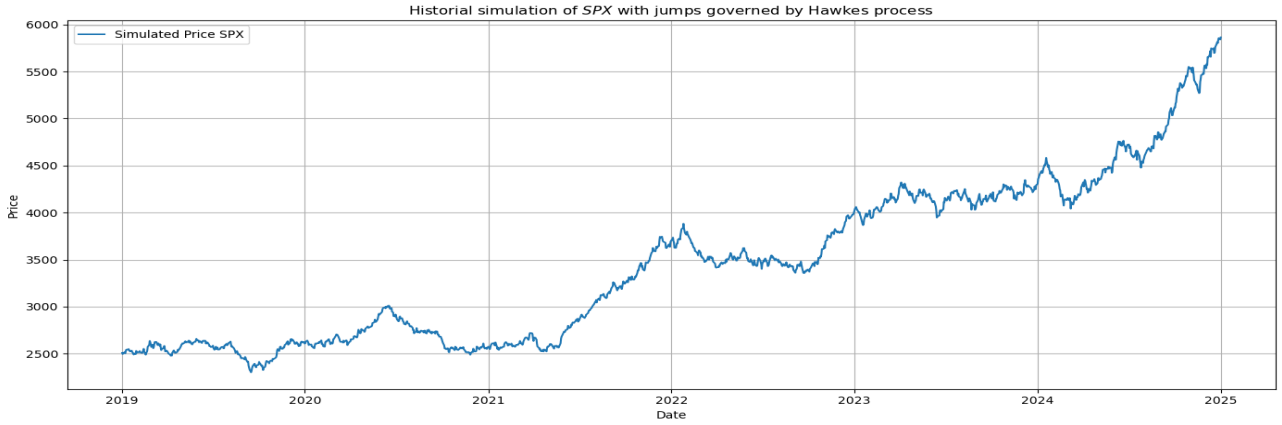
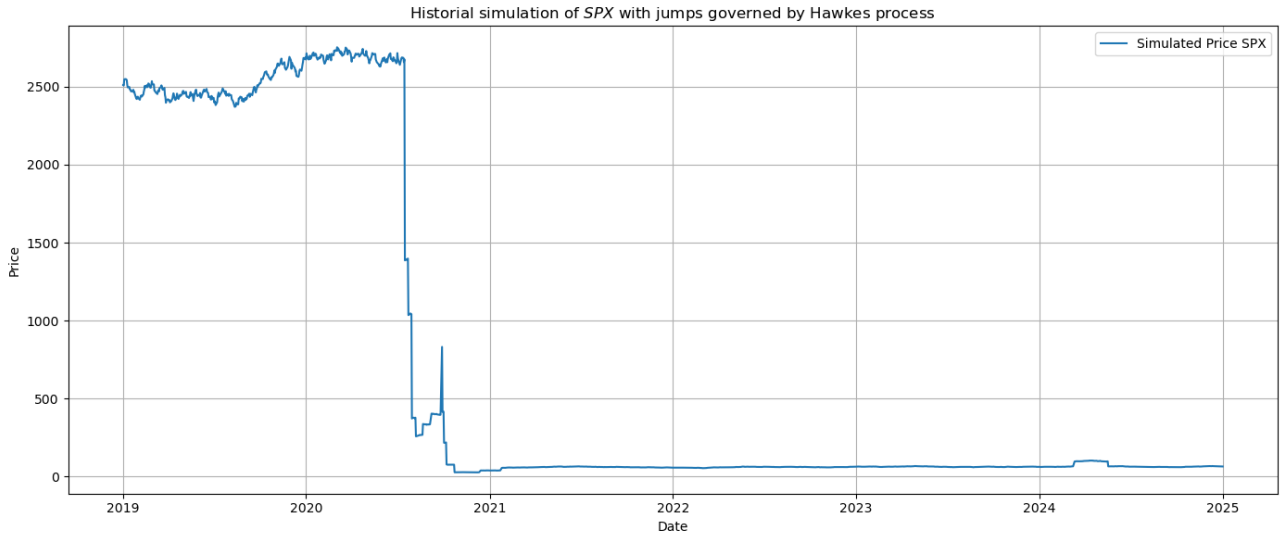


Figure 23



The parameters of α and β have been chosen in such a way to guarantee the stationarity of the process of $\lambda(t)$: $\frac{\alpha}{\beta} < 1$[24].

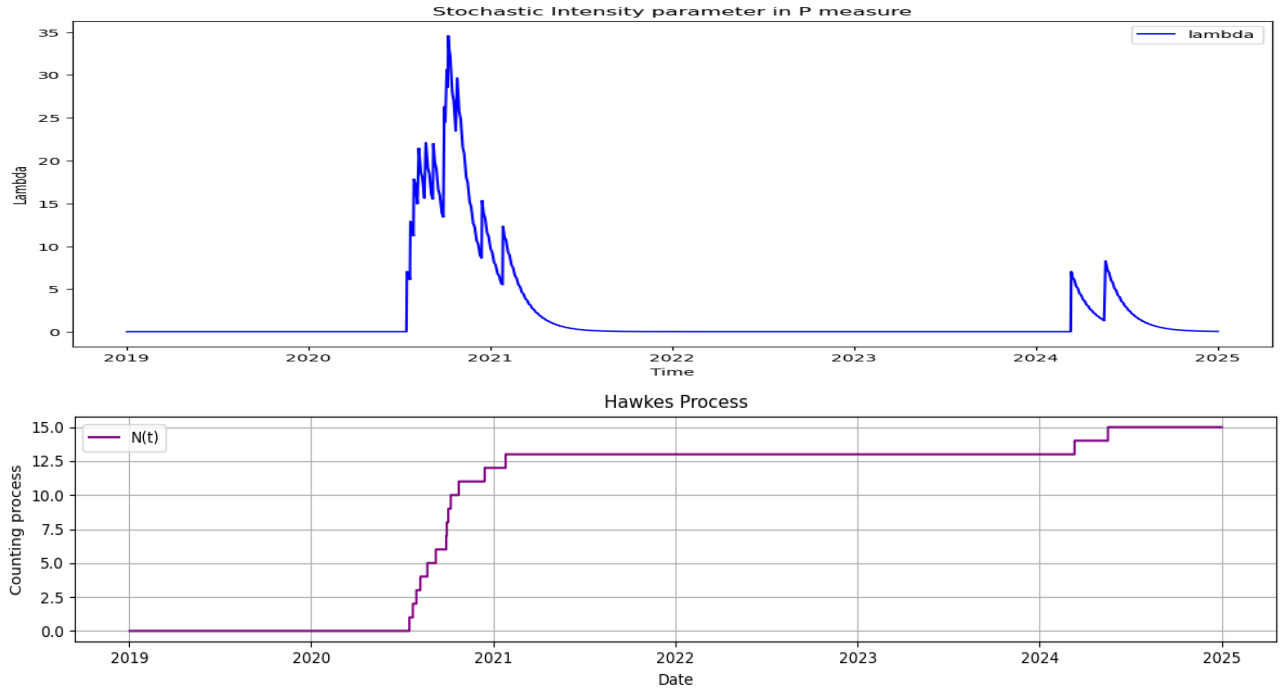
All the optimal parameters already computed in previous sections for $(\sigma^*, \lambda_0^*, a^*, b^*)$ were kept for this simulation. The optimal lambda parameter employed in the analysis within classical Merton framework has been taken as the *initial* parameter intensity λ_0, so that, in case of no jumps, the stochastic intensity parameter is constant and equal to the initial homogenous intensity parameter.

A very important point to make is that whereas the first four parameters $(\sigma^*, \lambda_0^*, a^*, b^*)$ have been estimated directly from a cross section of option prices and, hence, are measured under the risk neutral $\mathbb{Q}$ probability, the parameters α and β have been estimated from historical prices of the underlying and hence incorporate risk premia and are evaluated under the physical $\mathbb{P}$ probability measure. For simplicity, we assume that these parameters are the same in the risk neutral measure, so that $\alpha^{\mathbb{P}} = \alpha^{\mathbb{Q}}$ and $\beta^{\mathbb{P}} = \beta^{\mathbb{Q}}$ and are then usable in the scope of option pricing and are already identified. This assumption is simplistic and reflects a deeper meaning on the jump dynamics. By imposing that the parameters are the same in the two probabilistic worlds we are basically assuming that the market does not require an additional risk premium coming from the contagion mechanism of jumps, so that the clustering of jumps is not driven by risk aversion and does not require any premium in parameters. This is completely different from claiming that there is no jump risk premium, as this assessment only regards the dynamics of self-excitement of jumps and not jumps themselves – it would be *jump-clustering risk.* So, the market price of jump risk is not zero and embedded in the risk neutral jump parameter, whereas the price of clustering risk is indeed zero.

To present a complete overview of the algorithm, plots for the stochastic intensity parameter and Hawkes process inherent in the simulated path in *Figure 24* are present ed below:

[24] Bacry, Mastromatteo, Muzy (2015), Hawkes processes in finance. Market Microstructure and liquidity.

Figure 24



These are the paths of the stochastic intensity and the corresponding counting process for the last of the two simulations presented above. We see that the jumps are pretty much clustered together, thanks to the high value of α, which make the intensity parameter jump after a first jump has been witnessed. The baseline value of intensity is very low, hence indicating a very low probability of observing jumps: in many simulations, in fact, there were not any. But once one jump is observed, then it is much more likely to observe further ones, as shown in the Hawkes process above. Apart from the last cluster of jumps we see that jumps are mainly downward, and this was expected by the parameters estimated from option prices governing the lognormal density of jump size.

We see that the number of jumps presented in $N(t)$ is coherent the one simulated by the algorithm. Hence, we will employ these parameters for our Hawkes process used in risk-neutral simulations to price our Lookback option:

$\boldsymbol{\alpha^*}$	**7.011**
$\boldsymbol{\beta^*}$	**8.911**

The "optimal" parameters found describe very little dynamics of the stochastic intensity parameter. As $\alpha^*$is very high, the impact of the occurrence of a jump to the intensity parameter will be strong. The high value of β^* indicates low persistence of the impact: once a jump occurs, it tends to decay quickly. The sum of these two components makes us think that the data may be calling for a Hawkes process to model the counting process, and this can also be explained by the already very low value estimated for $\lambda(0) = 0.03$ by our optimization algorithm. We employ this as our *background* intensity, which is a very high starting point.

3.4.3. Pricing of Lookback Options using the counting process with stochastic intensity

Having found the right parameters to specify an exponential Hawkes process, we now engage in the final purpose of our analysis: the pricing of lookback options using a Merton model with stochastic intensity parameter. We will price lookback fixed *at-the-*money option with a maturity of 90 days from the reference date $31/12/2024$. To do so, we will again employ Monte Carlo estimator based on M simulations of the option's underlying asset.

The first intuitive difference from the simulation run in the previous section is that we must now switch back to the risk neutral probability measure $\mathbb{Q}$. The no-arbitrage condition is essential to accurately price options; hence we will simulate from the discretized stochastic differential equation that describes the dynamics of the stock price S under the risk-neutral dynamics. The EMM used ensures the property that the discounted price $\tilde{S}_t$ is a martingale in this probability measure, meaning that the stock, in this framework, must have a drift equal to 0. To avoid arbitrage, we must employ a *pure jump martingale process under* $\mathbb{Q}$ in the stochastic differential equation, in the form of $CJ_t = J_t - (\mu_Y - 1)\lambda^{\mathbb{Q}}t$. The only source of uncertainty in this peculiar process is the jump exposure, but by subtracting the expected value of the jumps themselves we ensure that the drift of this process is equal to 0 under $\mathbb{Q}$ - thus implying the martingale property.

In fact, $E^{\mathbb{Q}}[\Delta J_t|\mathcal{F}_t] = E^{\mathbb{Q}}\big[(Y_j - 1)\big] * Q[occurence\ of\ a\ jump\ in\ dt] = (\mu_Y - 1) * \lambda^{\mathbb{Q}}dt$

Where $(\mu_Y - 1)$ indicates the average impact of a jump, given that a jump occurs in dt and $\lambda^{\mathbb{Q}}dt$ indicates the probability of jump occurring in dt, as shown in previous sections. Subtracting this measure from the jump process makes sure that the drift is *corrected* to account for the presence of jumps and still to ensure the no arbitrage condition. Finally

$$\frac{dS(t)}{S(t^-)} = rdt + \sigma dW^{\mathbb{Q}}(t) + \Delta CJ(t)$$

$$\frac{dS(t)}{S(t^-)} = rdt + \sigma dW^{\mathbb{Q}}(t) + \big(\Delta J(t) - (\mu_Y - 1)\lambda^{\mathbb{Q}}dt\big)$$

$$\frac{dS(t)}{S(t^-)} = \big(r - (\mu_Y - 1)\lambda^{\mathbb{Q}}\big)dt + \sigma dW^{\mathbb{Q}}(t) + \Delta J(t)$$

As previously indicated by page 20, the solutions, for each time t, of this stochastic differential equation can be found using Ito's lemma and are equal to:

$$S(T) = S(0)e^{\left(r-(\mu_y-1)\lambda_t-\frac{1}{2}\sigma^2\right)T+\sigma W^{Q}(T)}\prod_{j=1}^{N(T)} Y_j$$

Note that to derive this solution we have chosen a specific EMM. Out of the infinite possible solutions of (θ, φ) we have chosen the vector with $\varphi = 1$. Since we are trying to introduce a stochastic intensity parameter, we will have a vector of parameters with different intensities in different times.
The main novelty with respect to the Merton model lies in this: in each cycle of the *for* loop, we must update the intensity parameter and take, at each time t, the parameter of that period according to the Hawkes process. If a jump occurs at time τ_i, we know that λ_{τ_i} will also experience a jump driven by the parameter α. So, the intensity parameter for that cycle will be higher and thus the reducing impact on the drift of S stronger: when a jump occurs, the time deterministic component will push the stock price down, with a subsequent gradual recovery to lower values of λ – driven by parameter β, assuming no further jumps push the intensity parameter even higher. In this new framework, the number of jumps $N(T)$ we should witness in every simulation should be higher compared to the Merton's case, as we have introduced a self-exciting behavior of jumps while keeping the initial intensity parameter fixed.

Using the following code, and the function defined above – having changed the code to get the drift under the $\mathbb{Q}$ probability $\left(r-(\mu_Y-1)\lambda^{\mathbb{Q}}\right)$ rather than μ under the $\mathbb{P}$ probability, we can generate 10,000 simulations of the stock price.

```
S_full = np.zeros((M, N_dates))

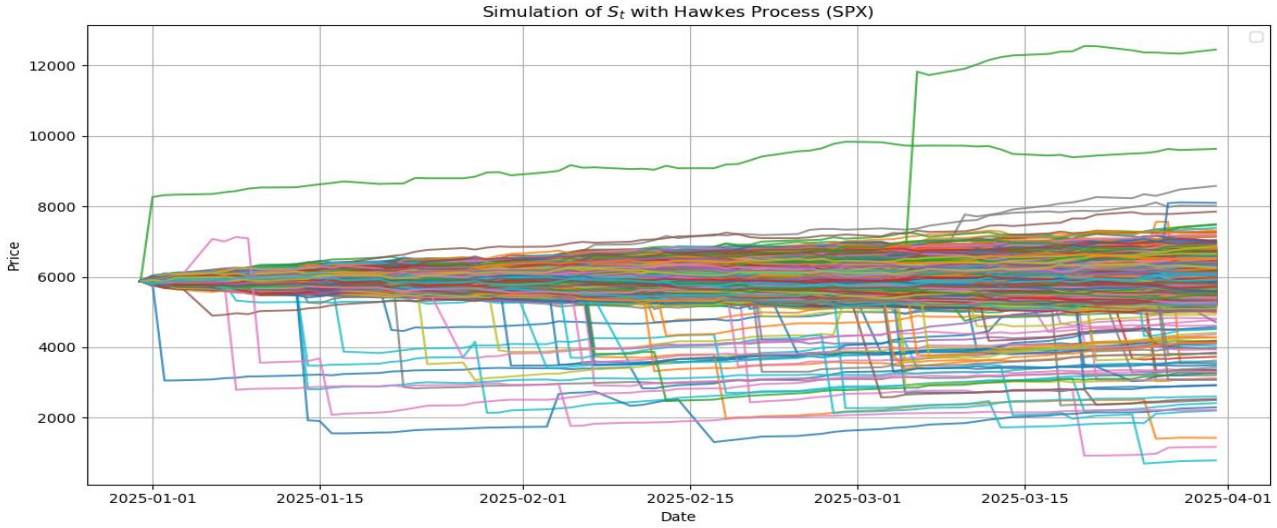

rng = np.random.default_rng(12345)


Z_all = rng.normal(size=(M, N_steps))
U_jump_all = rng.uniform(size=(M, N_dates))
Z_jump_size_all = rng.normal(size=(M, N_steps))


for m in range(M):
    S, lambda_series, jump_times = simulate_jump_diffusion_hawkes(S0,r,sigma,a,b,mu,
                          alpha, beta, trading_days, dt,
                          Z_all[m],
                          U_jump_all[m],
                          Z_jump_size_all[m]
    )
    S_full[m, :] = S
```

It is very important to stress that for each of the 10,000 simulations carried out we call the defined function to get a vector of simulated lambdas. Since lambda is stochastic, to get a specific path of $\lambda(t)$ one must run a simulation. If we used the same vector of simulated values for all the 10,000, the paths for S would not be independent because they would all be using the same Hawkes process, with the same number of jumps per period, and the resulting Monte Carlo estimator would be biased. Hence, it is crucial to stress that we simulated both the jump times and the lambda vector for each of the M simulations, so that each has its own vector of stochastic intensity parameters. The simulated matrix is shown in *Figure 25*

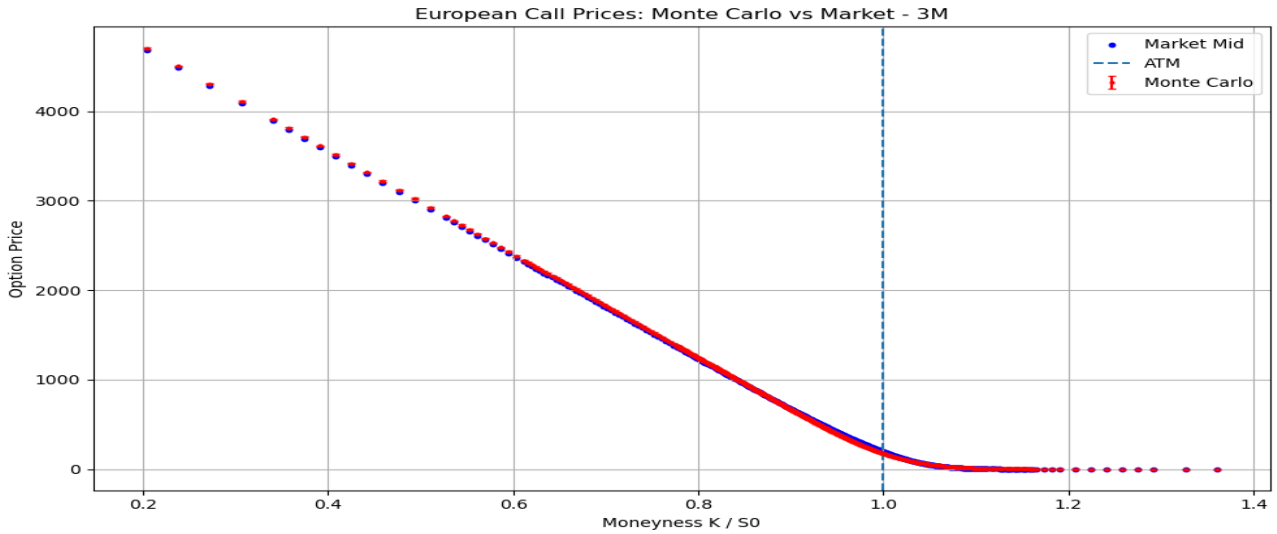
Figure 25



As in the previous simulation matrix, we have taken 3 months maturity. So, we are simulating the stock price for all the trading days between the 31^{st} of December and 31^{st} of March.

For starters, we use this Pandas Data frame of M simulations to run initial Monte Carlo simulations to price European calls for different values of spot moneyness K/S in this new mathematical framework. The code is the same as the one used in previous sections and will not be reported here. The plot is displayed in *Figure 26:*

Figure 26



We see that Monte Carlo estimator works well in fitting the market prices. The price for the ATM call is equal to 174 $. As in the previous section, we now compute the price of the Fixed lookback option ATM. The price for a fixed lookback option ATM on SPX at date 31^{st} of December 2025 is:

Price	283.99 $
Standard Error	2.39
Lower Bound (95%)	279.31 $
Upper Bound (95%)	288.67 $

We first note that, as the parameters α and β are indicative of a big difference in the stochastic intensity parameter path, the price of the lookback in this new framework is higher than the price of the same product within the Merton model: The change represents a 3.3% increase in the derivative's price $(\frac{279}{270} - 1)$.

To reduce the variance of the Monte Carlo estimator, we also run the same code defined above to carry out a control variate estimator to price the lookback. Again, the OLS coefficient beta has been computed in such a way to guarantee that β_{OLS} and $\widehat{X_n}$ are independent and that unbiasedness of the estimator remains untouched. As above, we take as control variate the price of the European Call option under the traditional Merton framework. We do know the closed formula for the theoretical call price, so it is useable as control variate. Hence, we simulate the $N = 10{,}000$ simulated payoffs of European ATM call in the traditional Merton model, whose expected value $E[X]$ is known, and the same number of simulated payoffs of lookback ATM option in the Merton-Hawkes model. The correlation between these two variables (from the N point data) is very close to zero, specifically $\rho = -0.0082$, suggesting that the control variate estimator is no good in reducing the variance, and the standard error of the estimator. The issue is the mathematical pricing model we are taking as reference for the two variables: the traditional Merton and Merton-Hawkes model yield different simulated matrices of stock price paths, and hence these payoffs can present a correlation equal to zero.

The results of the CV estimator are the following:

Price	284.27 $
Standard Error	2.86
Lower Bound (95%)	278.64 $
Upper Bound (95%)	289.89 $

We see that the standard error of the control variate is even higher than the traditional counterpart, which looks counterintuitive and suspicious. This issue can be explained by the fact that we are only using 70% of our simulated MC payoffs in the control variate estimator, resulting in an increase of the standard error. By splitting the sample to estimate the control variate coefficient β independently from the pricing sample, we avoid introducing estimation bias at the cost of reducing the effective Monte Carlo sample size from 10,000 to 7,000 observations. By using the whole sample for the MC estimation – setting the parameter split_ratio = 0, we find a value of standard error almost equal to the one found with the traditional estimator, showing that the main issue for the lack of variance reduction is the almost-zero correlation between the target variable and the control variate.

We have carried out this analysis for the control variate estimator selecting the instrument and the right procedures to make sure that this estimator is unbiased. Possibly, other procedures may be available that relax this requirement while maximizing the statistical properties of the estimator in *MSE* (Mean Square error) terms. For instance, with other control variates it may be possible to reduce the standard error of the estimator whilst adding bias. We chose this procedure to guarantee that the latter estimator would be theoretically unbiased with respect to the true, theoretical price.
A more natural choice of control variate estimator would be the European call option under the Merton-Hawkes model rather than in the traditional Merton one. Since we do not know the theoretical price of the call in this augmented model, we can still use a Monte Carlo estimate of $E[X]$ with many simulations so to reduce the noise around this estimate. This is because the correlation parameter between the payoffs of the Lookback and the payoffs of European call in the same Merton-Hawkes model is much higher: $\rho_{XY} = 0.91$ which justifies the usage of this new control variate. The proposed price and standard error of the lookback with this new control variate estimator are the following:

Price	278.63$
Standard Error	1.04
Lower Bound (95%)	276.6 $
Upper Bound (95%)	280.66 $

Hence, we see that the standard error has indeed reduced now at the cost of adding bias in the point estimate as we are using a Monte Carlo proxy for $E[X]$ and not the true value. Finally, we have many choices between different estimators to provide a good estimate for the option price. A clear trade-off between Unbiasedness and Efficiency is at work here, and one must decide in which direction to go. A good, comprehensive approach would be to choose the estimator that minimizes the *MSE*.

$$MSE(\theta) = Bias(\theta)^2 + Var(\theta)$$[25]

The issue is that we do not know the true price $E[Y]$, where Y is the discounted payoff of the lookback. Without this, we cannot compute the Bias of the estimator θ. A comprehensive procedure on how to estimate the *MSE* is outside of the scope of this thesis. Hence, we will not use a Control Variate in this section and will rely on the pure Monte Carlo estimator.

In this pricing exercise we see that the price of the Fixed lookback is higher than the European call and this, as explained earlier, makes perfect sense. Furthermore, we also notice that the price of the Lookback from the control variate within the stochastic intensity parameter model is higher than the price computed previously with the homogenous intensity parameter. This is perfectly coherent with the framework used since the intensity parameter, thanks to the addition of the kernel to include some memory, is greater than or equal to the homogenous intensity parameter used in the Merton framework.

[25] Errore quadratico medio - Wikipedia

As we have set the optimal homogenous parameter lambda as $\lambda(0)$ in the formula

$$\lambda(t) = \lambda(0) + \sum_{m=0}^{M} \varphi(t - t_m)$$

and $\lambda(0)$ is the same homogenous parameter used earlier, it is intuitive that $\lambda(t) > \lambda(0)$, $\forall t$. Since $\alpha, \beta \in \mathbb{R}^+$, the occurrence of a jump increases the parameter until this effect decays to 0 and $\lambda(t) = \lambda(0)$. The positive difference in prices between the different mathematical models used is to be conducted to the self-exciting nature of jumps introduced by the Hawkes process.

However, it is true that in the no-arbitrage framework imposed by the risk-neutral measure $\mathbb{Q}$ a higher parameter lambda has a negative effect on the drift of the simulated paths, but numerically this is more than compensated by the higher frequency of jumps. This increases the simulated payoff for the fixed lookback (as the $\max\{S_t\}$ increases), and, thus, the no-arbitrage price.

3.4.4. Sensitivity analysis

Now that we have come up with a pricing model and both a punctual and interval estimate of the price for this exotic derivative, it is crucial now to carry out a brief analysis on the sensitivities of this option price. The Sensitivities are called *greeks* and represent the partial derivative of the model option price with respect to the different input variables used in the model. The main ones are:

$$\Delta = \frac{\delta C}{\delta S}$$

$$\gamma = \frac{\delta^2 C}{\delta S^2}$$

$$\upsilon = \frac{\delta C}{\delta \sigma}$$

Delta, the first partial derivative, measures the infinitesimal change in the option price because of an infinitesimal change of the underlying stock price. These measures are crucial in risk management or portfolio management view, as they allow dynamic management of the PnL of an options portfolio. Usually, traders set their portfolio to be *delta neutral* by a procedure called *delta hedging*. Essentially, the goal of the trader is to have a portfolio of options on a given underlying asset in such a way he is hedged against any fluctuation of the underlying price, because the market value of his portfolio is neutral with respect to these risk factors. For a trader to efficiently hedge his portfolio, he must have at least an estimate of the delta of the options he has in his portfolio here lies the importance of numerically computing the sensitivities of a given exotic option. The trader, with his pricing model, must estimate these sensitivities to accurately measure the risks and exposures embedded in each product, so that he dynamically manages and hedges his portfolio.

The main technique used to estimate a Greek is the Finite Difference Approximation, according to the following expression:

$$\frac{\partial C}{\partial S} = \frac{\widehat{C_n}(S^* + h) - \widehat{C_n}(S^* - h)}{2h} \; as \; h \to 0^+$$

Where $\widehat{C_n}$ is the Monte Carlo estimator for the option using n simulations. It is basically a procedure to discretize the infinitesimal variation and estimate the partial derivative of the function $C(S, \sigma, T-t, r)$ with respect to S in the point $S = S^*$, by computing the slope of the secant line between $\hat{C}(S^* + h)$ and $\hat{C}(S^* - h)$. This technique is called **Central-Difference Estimator (CDE)** and its logic is to estimate the derivative in a symmetric way around the point S^* in which you want to compute the derivative. Another valid alternative is the **Forward-Difference Estimator (FDE)**, with the following expression:

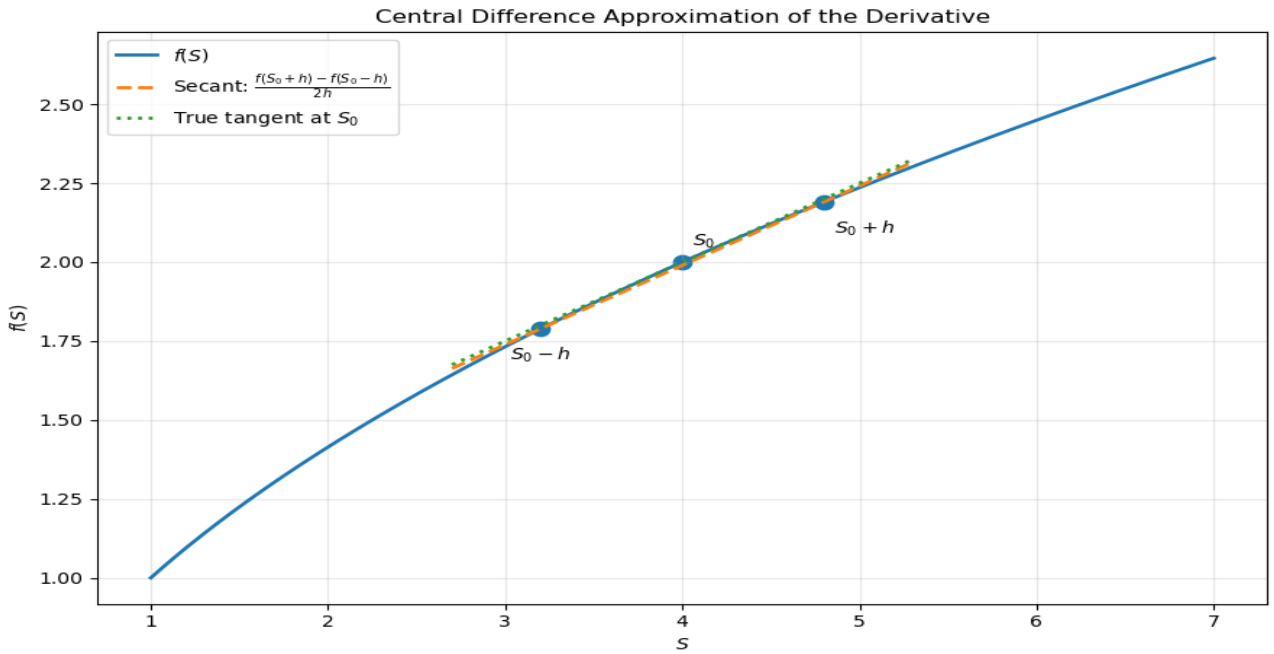
$$\frac{\partial C}{\partial S} = \frac{\widehat{C_n}(S^* + h) - \widehat{C_n}(S^*)}{h} \; as \; h \to 0^+$$

The logic of this estimator is very similar to the previous one, with the difference that the derivative is not estimated symmetrically around the point S^* but, rather, it only takes one step in one direction and computed the derivative as the slope of the secant line between the function in that step and S^* itself. Note that the FDE is essentially an estimation of the derivative of a function in a point, and it tries to discretize the classic definition of a derivative:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

In our case we are dealing with a partial derivative, so we need to consider $f(x)$ as a multivariate function defined in more variables and h as a discrete variation in one of the variables over which the function is defined. The graph in *Figure 27* summarizes the logic of this algorithm to estimate the derivative in a point $S0$:

Figure 27



Both estimators are unfortunately biased and non-consistent, so the estimates produced will not be extremely reliable statistically. Nevertheless, we will use the CDE estimator as the bias of this estimator is positive but grows linearly in h, whereas the bias of FDE grows quadratically in h.

$$Bias\left[\widehat{\Delta_{CDE}}(n,h)\right] = constant * h$$

$$Bias\left[\widehat{\Delta_{FDE}}(n,h)\right] = constant * h^2$$

An important decision to make before estimating numerically these partial derivatives is the length of h. This is a very important choice as it represents a trade-off between Bias of the estimator and variance of the estimator. The variances of the two estimators are:

$$Var\left[\widehat{\Delta_{CDE}}(n,h)\right] = Var\left[\frac{\widehat{C_n}(S^*+h) - \widehat{C_n}(S^*-h)}{2h}\right] = \frac{1}{4}h^{-2}Var[\widehat{C_n}(S^*+h) - \widehat{C_n}(S^*-h)]$$

$$Var\left[\widehat{\Delta_{FDE}}(n,h)\right] = Var\left[\frac{\widehat{C_n}(S^*+h) - \widehat{C_n}(S^*)}{h}\right] = h^{-2}Var[\widehat{C_n}(S^*+h) - \widehat{C_n}(S^*)]$$

So, we see analytically that increasing h is negative for the bias, since we are discretizing for a higher interval, but it is positive for the variance, as now the estimate is less noisy. Choosing the optimal h^* is a result of a trade-off between the variance and the bias of the estimator, so it is to be kept in mind that the smallest h is not always the optimal. We should choose to have a low value for h and a high value for n so that the increased number of simulations can help reduce the variance of $\widehat{\Delta_{CDE}}(n,h)$.

Numerically, we define two variables by taking both an upward and downward variation of 10% the initial price. The initial price is set to $S0 = 5881.63$ and we take a deviation in both directions to approximate the derivative of the pricing function in that point: $S_{up} = S0 * (1 + 0.1) = 6469.8$, $S_{down} = S0 * (1 - 0.1) = 5293.48$.

We then want to compute the price for the lookback fixed option for these two levels of underlying stock price, $\widehat{C_n}(S^*+h)$ and $\widehat{C_n}(S^*-h)$. Essentially, for both levels of initial stock price we simulate the whole matrix of simulated paths using the same function defined above. The essential requisite for this simulation technique is that all other inputs (including $K, \sigma, T-t, r$) must be kept the same.
We are conditioning the derivative only on the underlying stock price, making it, in fact, a partial derivative. We then compute the two option prices and compute the ratio to be the Delta

Finally, we take the difference between the two option prices, and we divide this value by $2h$ using the formula presented above to compute a delta of $\widehat{\Delta_{CDE}}(10{,}000, h = 10\%) = 0.046$. The value estimated is reasonable if we look at the context of the derivative: we are evaluating the $\widehat{\Delta_{CDE}}$ of the exotic option at the beginning of his life and has time to maturity $T = 0.24$. Given that the payoff is computed as the difference between the maximum price achieved during the life of the option and the contractual strike, it is intuitive that an infinitesimal change in the underlying will not have a huge impact on the price of the option, given that 3 months are still left to maturity. From this framework we can also compute the gamma of this product, which refers to the second derivative of the option price with respect to underlying price $\frac{\delta^2 C}{\delta S^2}$.

We thus want to estimate the second derivative of a function in each point, which would be included in the main diagonal of the Hessian matrix for a multivariate function (like the function for the option price). The purpose would be to analyze the curvature of our function in the point, to assess whether it goes in the same direction of the $\widehat{\Delta_{CDE}}$ or if it goes in the opposite direction. A positive gamma means a positive curvature and convexity of the function: this is positive for the PnL of our portfolio as, intuitively, the Δ increases when S increases and it decreases when S decreases. The Δ, being the first derivative, only captures the linear, first order impact of a variation of S, and γ helps us in having a bigger picture.
The numerical approximation by finite differences of the second derivative can be computed as follows:

$$\gamma = \frac{(\widehat{C_n}(S^* + h))' - (\widehat{C_n}(S^*))'}{h} \; as \; h \to 0$$

Where $(\widehat{C_n}(S^* + h))'$ represents the first derivative of $\widehat{C_n}(S^* + h)$ with respect to S. In fact, the second derivative can be seen as the derivative of the first derivative. Then:

$$\gamma = \frac{\frac{\widehat{C_n}(S^* + h) - \widehat{C_n}(S^*)}{h} - \frac{\widehat{C_n}(S^*) - \widehat{C_n}(S^* - h)}{h}}{h} = \frac{\widehat{C_n}(S^* + h) + \widehat{C_n}(S^* - h) - 2\widehat{C_n}(S^*)}{h^2}$$ [26]

And it was equal to $\gamma = 0.0002$. This indicates a very weak curvature of the option price with respect to the underlying, implying that the delta is non-responding for whatever change in the underlying price.
The last important sensitivity that needs to be computed is Vega $\upsilon = \frac{\delta C}{\delta \sigma}$, the sensitivity of the option price with respect to the volatility parameter. The procedure to compute this partial derivative is the same as in the case for the Δ. The only difference is that we now keep S fixed and the infinitesimal variation h refers to the volatility parameter only:

$$\upsilon = \frac{\Delta C}{\Delta \sigma} = \frac{\widehat{C_n}(\sigma^* + h) - \widehat{C_n}(\sigma^* - h)}{2h}$$

We employ a similar algorithm as before by setting a change of 10% to the initial parameter σ^*.
And we compute a $\upsilon = 21.8$. Although this seems like a high value for this parameter, this reflect the change in price for the lookback option for a $\Delta\sigma = 1$, which is equal to a variation of 100% of volatility. Hence, we need to divide this estimate by 100 to have a realistic risk measure: when volatility changes of 1% ($from\ 10\%\ to\ 11\%$), the price of the lookback changes of $\frac{21.8}{100} = 0.22$ \$. Now, we conclude that also the Vega of this derivative is low and has a negligible impact.
To sum up, these are the sensitivities estimated:

Δ	**0.046**
γ	**0.0002**
υ/100	**0.22**

[26] Differenze finite: introduzione al metodo numerico e primi esempi

3.4.6. Results

In this concluding section we summarize all our results in terms of prices and greeks for the exotic derivative we have dealt with during the paper. We have priced a floating lookback exotic derivative in two different mathematical frameworks: the original Merton model with constant intensity parameter and an augmented Merton model with stochastic intensity parameter whose counting process is governed by a Hawkes process. Since the novelty is the pricing of the latter exotic derivative, we will report prices and greeks for this product for different sets of parameters All prices reported consist of the point estimate using the Control-Variate Monte Carlo estimator.

We will report the quantities for different pairs of parameters:

	Params	***Params_2***	***Params_3***	***Params_4***	***Params_5***
Price	**283.99**	**283.31**	**285.13**	**284.31**	**283.71**
Lower-bound	279.31	278.63	280.4	279.59	279.05
Upper-bound	288.67	287.98	289.85	289.04	288.36
Delta	**0.046**	**0.048**	**0.048**	**0.048**	**0.048**
Lower-bound	0.041	0.043	0.043	0.043	0.043
Upper-bound	0.051	0.054	0.054	0.054	0.054
Gamma	**0.000002**	**0.000009**	**-0.000007**	**0**	**-0.00001**
Lower-bound	-0.00028	-0.00032	-0.000039	-0.0000033	-0.00004
Upper-bound	0.00033	0.00034	0.000027	0.0000033	0.00002
Vega/100	**0.22**	**0.25**	**0.26**	**0.25**	**0.25**
Lower-bound	-298.4	-292.02	-314.49	-311.98	-296.7
Upper-bound	298.92	292.53	315.00	312.47	297.20

The values reported in the first column refer to the parameter highlighted in the other parts of the work, so $\lambda_0 = 0.03, \alpha = 7.011, \beta = 8.911, \sigma = 0.106, a = -0.369, b = 0.346$.

λ_0	0.03
α	7.011
β	8.911
σ	0.106
a	-0.369
b	0.346

We are interested in assessing how prices and greeks would change for a variation in some relevant parameters. For instance, an important parameter of our pricing model is *baseline intensity* λ_0. Since this value has a relevant impact on the whole stochastic process of $\lambda(t)$, we recompute all the measures above for $\lambda_0^* = \lambda_0 - 0.003 = 0.03 - 0.003 = 0.027$ and $\lambda_0^{**} = \lambda_0 + 0.003 = 0.03 + 0.003 = 0.033$, while keeping all other parameters constant. The second column (***Params_2)*** contains all the metrics when baseline intensity is λ_0^*, and the third column (***Params_3)*** is filled with value for λ_0^{**}.

The price of the fixed lookback is higher when the baseline intensity parameter is higher, and it is lower when the respective baseline parameter is lower. Although one may expect a higher jump intensity to reduce the option value due to the negative average jump size, this effect is offset under the risk-neutral dynamics by the jump compensator in the drift. Moreover, since the payoff depends on the running maximum, changes in jump intensity also affect path dispersion. In our numerical results, the price differences across intensity scenarios remain within the Monte Carlo confidence intervals, suggesting that the sensitivity to the baseline intensity is not statistically significant.

We now want to test how price and Greeks would be by changing the parameters governing the Hawkes process: α and β. The fourth column (***Params_4***) includes all the relevant metrics computed by changing parameters in the following way: $\alpha^* = \alpha + 0.7 = 7.011 + 0.7 = 7.711$ and $\beta^* = \beta - 0.89 = 8.911 - 0.89 = 8.021$.

These adjustments are aimed in assessing how price and greeks would change if the jumps on the underlying had a stronger, upward impact on the intensity parameter. A higher α^* implies a bigger jump in the intensity in occurrence of a jump, and a lower β^* indicates a lower decay of the intensity. The fifth column (***Params_5***) includes all metrics with $\alpha^{**} = \alpha - 0.7 = 7.011 - 0.7 = 6.311$ and $\beta^{**} = \beta + 0.89 = 8.911 + 0.89 = 9.802$. The logic is now assessing the relevant metrics if the jumps on the underlying had a weaker impact on the intensity parameter.

Note that the values for the fourth and fifth columns are computed using the original baseline intensity parameter λ_0. Overall, we could argue that variation in parameters of the Hawkes process do not imply a significant change in any of the computation.

4. Volatility Skew fitting

In this last chapter of this thesis, we carry out an analysis of the implied volatility skew of this mathematical model to price options. It is well known by practitioners and by academics that in the equity options market the shape of the implied volatility as function of moneyness is asymmetric and negatively sloped, creating the so called "volatility skew".[27]
The implied volatility of *out-of-the-money puts* (OTM puts) is indeed higher than the one of *out-of-the-money calls* (OTM calls). A possible explanation for this asymmetry in prices can be found in the market-crash fear of investors; in fact, an OTM put represents some sort of insurance agains market crashes, and this could be a possible explanation for higher demand for these products, resulting in a higher price and hence higher volatility.[28]

After the 1987 crash this skew has been particularly clear and has consolidated in quantitative finance literature. A way to assess the practical performance and the features of a specific pricing model is to verify to what extent the model can fit the empirical, observed volatility skew in the options market. The failure of the classic Black Scholes model to replicate the volatility skew has in fact pushed practitioners to find other *ad-hoc* pricing models that can capture this empirical feature. The broad class of models proposed in this sense are the stochastic volatility models, which include the famous Heston model (1993)[29]:

$$dS(t) = \mu S(t)dt + \sqrt{V(t)}S(t)dz_1(t)$$

$$dV(t) = k\big(\theta - V(t)\big)dt + \sigma\sqrt{V(t)}dz_2(t)$$

Where $V(t)$ is the variance at time t, θ is the long run ergodic variance, k in parameter measuring the strength of the mean-reverting drift and σ is the volatility-of-volatility parameter. $z_1(t)$ and $z_2(t)$ are two Brownian motion which evolve stochastically over time with correlation ρ.

The model we proposed in this thesis is utterly different, as it does not present a stochastic volatility, rather it presents a stochastic intensity parameter of the jump component:

$$dS(t) = \mu S(t^-)dt + \sigma S(t^-)dz_1(t) + S(t^-)dJ(t)$$

$$d\lambda(t) = \beta(\lambda_0 - \lambda(t))dt + \alpha dN(t)$$

We are interested in seeing how this different model can replicate the observed market volatility skew from the SPX options market. First, we need to get the observed volatility skew for SPX in a given period.
For this analysis, we choose to analyze the period from $02-01-2019$ until $31-12-2020$, so to analyze how the implied volatility skew has evolved during the Covid crisis and to assess to what extent our model has been able to capture that. We download the data from *OptionMetrics* from the *Volatility surface* section, where it reports for each day in the specified trading period the level of implied volatility of these options for many different levels of moneyness.

The moneyness in this database is measured with a different metric from the ratio between the strike and spot K/S used before. Now it used the Black Scholes Delta Δ_{BS} to indicate the moneyness of the options. The reasoning is the following:

[27] Jim Gatheral, "The volatility surface, a practitioner's guide"
[28] David Bates, "Post –'87 Crash Fears in S&P 500 futures options", National Bureau of Economic research (1997),

[29] Steven Heston, "A Closed-Form solution for options with stochastic volatility with applications to bond and currency options", The Review of Financial Studies (1993)

In the Black Scholes model, the Δ_{BS} is the partial derivative of the call price with respect to the spot price and is then equal to $\Delta_{BS} = N(d_1) = N\left(\frac{\ln\left(\frac{S}{K}\right)+\left(r-\frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}}\right)$[30].

Fixing T and σ, there is a clear relationship between $\ln\left(\frac{S}{K}\right)$ and Δ_{BS}. Furthermore, because of the basic properties of the Call options, we know that the delta is always included in $[0; 1]$, and an option with $\Delta_{BS} = 0.5$ is generally considered *at-the-money*. Using the same logic a Call option with $\Delta_{BS} = 0.25$ is considered an *out-of-the-money* call option, indicating that the spot price is below the strike, and a Call option with $\Delta_{BS} = 0.75$ is considered an *in-the-money* call option. This is a standard metric to measure moneyness by practitioners in the industry.

Notice that a $\Delta_{BS} = 0.5$ does not imply that the spot price is equal to the strike, as we can see from the Black Scholes formula:

$$N(d_1) = 0.5$$

$$d_1 = 0$$

Since the $N(\cdot)$ is the CDF of a Standard normal symmetric around 0.

$$\frac{\ln\left(\frac{S}{K}\right) + \left(r - \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}} = 0$$

$$\ln\left(\frac{S}{K}\right) = -\left(r - \frac{1}{2}\sigma^2\right)T$$

$$\ln\left(\frac{K}{S}\right) = \left(r - \frac{1}{2}\sigma^2\right)T$$

$$K = Se^{\left(r-\frac{1}{2}\sigma^2\right)T}$$

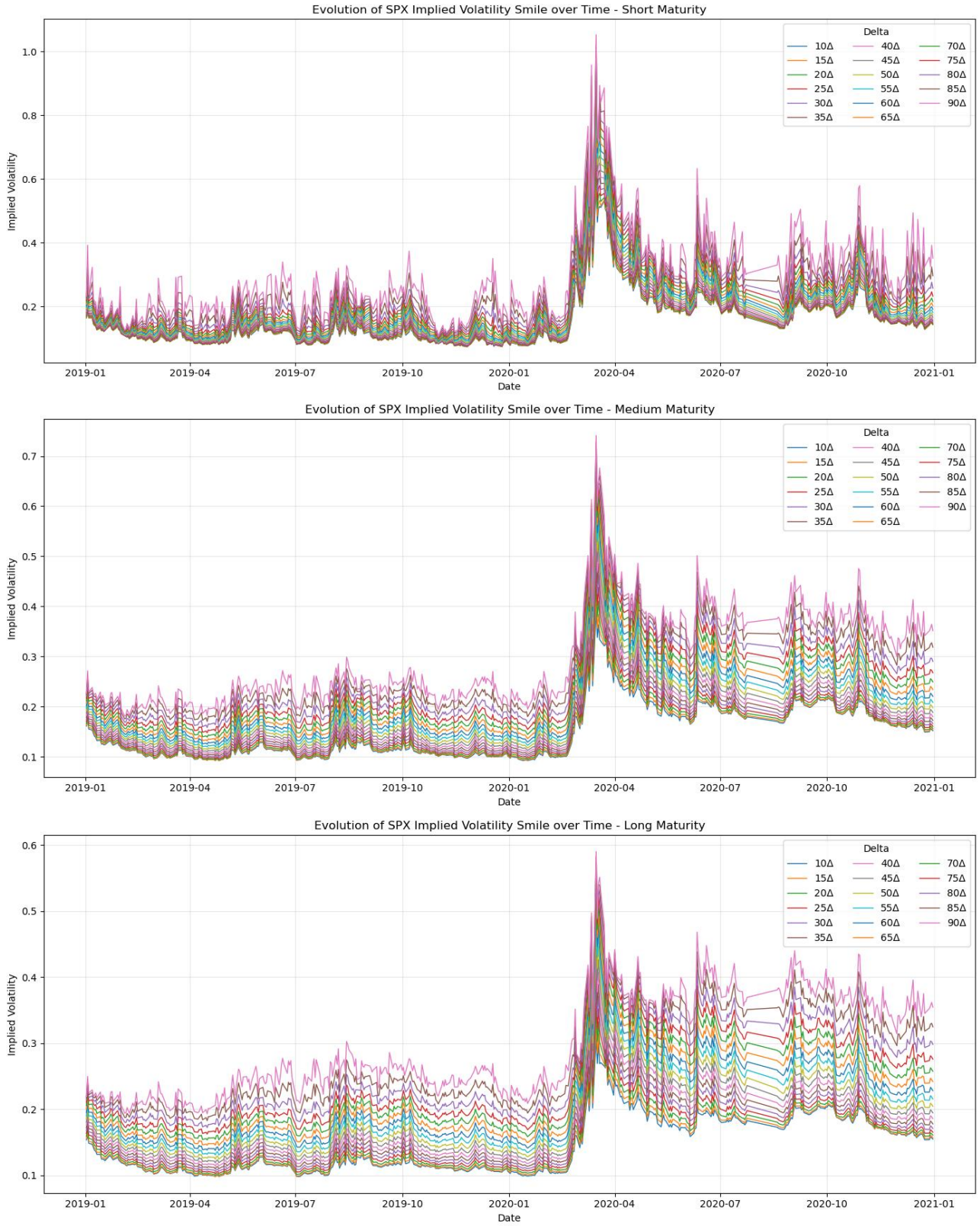
Which is usually a bit above the spot price. In most cases, this is the procedure we will use to go from a certain level of moneyness in delta to the corresponding strike, which will be helpful in the next steps when computing the implied volatilities of the pricing model for different levels of delta.

After Downloading the options data for the same 3 different maturity buckets as before – *Short* (1-30 days), *Medium* (70-100 days) and *Long* (130 – 300 days), we create a Pivot table to present each level of implied volatilities as function of time and moneyness.

The result is three different dataframes (one for each maturity bucket) containing implied volatilities for 488 days between $02-01-2019$ and $31-12-2020$ and for 17 different levels of moneyness measured by delta values ranging from 10 to 90, with a step of 5 units. The total panel of implied volatility has 8,296 data for each maturity cluster. The plots for the empirical skew over time for the three maturities are presented in *Figure 28.*

[30] Properties of European Options, Lorenzo Naranjo (2023), 13. The Black-Scholes Model — Options, Futures and Derivatives Securities

Figure 28



From these graphs, we can see that options with shorter maturities typically have higher implied volatilities and that all volatilities, for all maturities witnessed massive spikes during the covid panic crisis in April of 2020. Implied volatility for short maturities reached extremely high values even above 100%, especially for deep ITM calls (which would correspond to deep OTM puts, the crash insurance product).

To use our pricing model to fit this market skew we first need to calibrate the model. In fact, many parameters are fed to the model to price options using Monte Carlo algorithms. Despite a calibration on option data from the full

sample would be ideal, it is computationally unfeasible and hence we will calibrate the model using the cross section of option prices for a given day, as we did for the pricing of lookback in the previous section. Hence, we download option data for the $02-01-2019$ and we calibrate the Hawkes model with the same loss function as we did in the previous section. The optimal parameters to fit option market prices in that given date are:

σ	0.15469
a	-0.16250
b	0.26967
λ	0.02974
α	7.0111
β	8.9111

Given these results from the calibration, some clarifications are due. First, we use the same parameters for the Hawkes process as in the previous analysis. Despite the incoherency in this assumption, the loss function would be as noisy as before, so we do not believe this changes the analysis as much. Secondly, the calibration relies on the idea of minimizing the squared error difference between the actual options prices and the theoretical prices for that specific date. As in our cross section we have option prices from the three maturity buckets, we need to compute the theoretical price of call options for three maturities that "represent" those buckets. The loss function to be minimized is indeed:

$$Loss_\theta = \sum_{j}^{Short} \left(lnC^i_{Merton,\vartheta} - lnC^i_{Mkt}\right)^2 + \sum_{i}^{Medium} \left(lnC^i_{Merton,\theta} - lnC^i_{Mkt}\right)^2 + \sum_{i}^{Long} \left(lnC^i_{Merton,\theta} - lnC^i_{Mkt}\right)^2$$

We choose the following maturities for the corresponding bucket: $T_{short} = 14\ days$, $T_{medium} = 77\ days$ and $T_{long} = 259\ days$. We have chosen these specific maturities so to have the exact risk-free rates for those maturities from the American zero-coupon curve from *OptionMetrics.*

On the date $02-01-2019$, the risk-free rates from the zero-coupon curve for those three maturities were in fact $r_{14d} = 0.024744$, $r_{77d} = 0.02722$ and $r_{259d} = 0.02729$. we used these rates to compute the theoretical call prices, and they are automatically under the risk neutral $\mathbb{Q}$ measure. For the parameters of the Hawkes, again, we assume no risk premium for the clustering of jumps so that the parameters under the $\mathbb{P}$ and $\mathbb{Q}$ are the same. We also give the initial spot price at the date which is equal to $S0 = 2510.03$

With these calibrated parameters, the aim is now to find the implied volatility surface coming from out Hawkes model. This task is more demanding computationally, because for each maturity T, for each t of our sample period we need the price a European call option on SPX for different level of moneyness. From these theoretical prices, we then invert the Black Scholes formula to find the correct implied volatility parameter from that specific price. The computational issue that we face is that for each time t, keeping T fixed, we have a different risk-free rate r_t from t to $t+T$ and a different index value S_t. These are relevant because they are key parameter in the Monte Carlo procedure we will use to price each of these options. Thus, when creating the Data frame containing all the option prices (we consider that rows will be all the different periods t, while the columns will have different levels of moneyness expressed in Δ_{BS}) we must consider that in each time we must extract different historical data both from the risk-free rate and from the spot price of the underlying. Before giving out the code and results, two further points are due.

First, for some dates t between $02-01-2019$ and $31-12-2020$ the zero-coupon curve may not have a risk-free rate for those precise three maturities we selected for the three buckets. Hence, for each day we will carry out a linear interpolation of the curve and then select the right rate[31]. This linear interpolation works as follows. If you have the risk-free rates for 11 days maturity and 15 days maturity you can find the rate for the 14 days computing the slope of the secant between the two closest points of the curve for which you know the values:

[31] Come interpolare i tassi di interesse, EtfFin

$$r(14) = r(11) + \frac{14-11}{15-11}(r(15) - r(11))$$

Second, you need to compute for each value of moneyness Δ_{BS} the corresponding value of the strike K, as you are going to need it to price the European call options.

$$N(d_1) = \Delta_{BS}$$

$$d_1 = N^{-1}(\Delta_{BS})$$

$$\frac{\ln\left(\frac{S}{K}\right) + \left(r - \frac{1}{2}\sigma^2\right)T}{\sigma\sqrt{T}} = N^{-1}(\Delta_{BS})$$

$$\ln\left(\frac{S}{K}\right) = N^{-1}(\Delta_{BS})\sigma\sqrt{T} - \left(r - \frac{1}{2}\sigma^2\right)T$$

$$K = S * e^{\left(r - \frac{1}{2}\sigma^2\right)T - N^{-1}(\Delta_{BS})\sigma\sqrt{T}}$$

Which, for a fixed $S_t, r_t, \sigma^2 \; and \; T$, is a mapping from Δ_{BS} to K

Using these codes, we can get the data frames full of theoretical options prices for that day for a specific level of moneyness. The goal is now to use the inverted Black Scholes formula to find the implied volatility parameter from these prices. This problem can be written as:

$$\sigma_{t,K}^{impl} s.t. \;\; C_{BS}\left(S_t, r_t, T, K, \sigma^{impl}\right) = C_{t,K}^{Hawkes}$$

$$\sigma_{t,K}^{impl} s.t. \;\; C_{BS}\left(S_t, r_t, T, K, \sigma^{impl}\right) - C_{t,K}^{Hawkes} = 0$$

$$\sigma_{t,K}^{impl} s.t. \;\; f\left(\sigma_{t,K}^{impl}\right) = 0$$

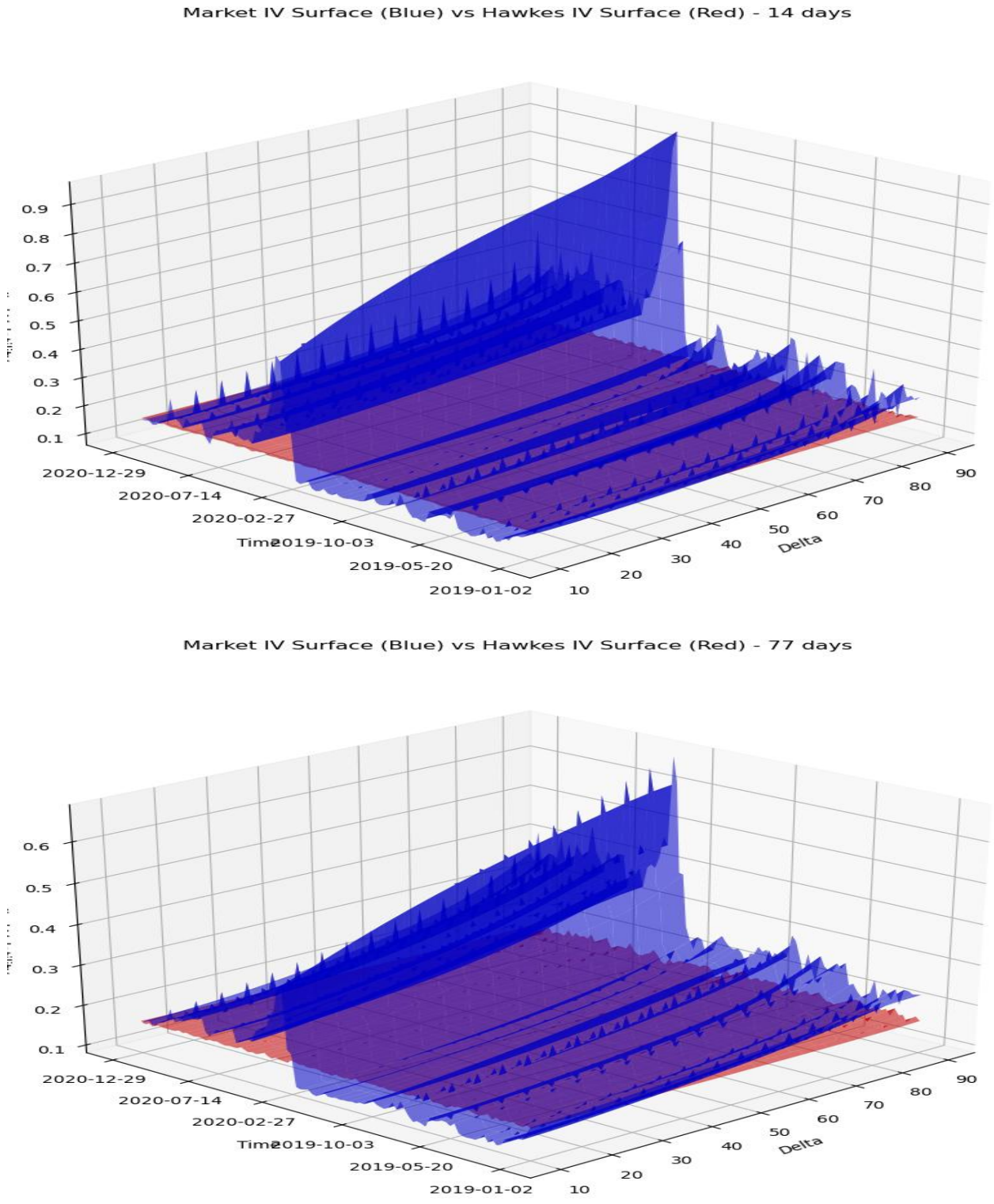
So, it can be solved numerically by finding the value of parameter $\sigma_{t,K}^{impl}$ such that that function $f(\sigma) = 0$, so we aim to find a zero of this function. The most used algorithms to solve this problem are the Newton Rapson algorithm[32], which relies on the first derivative of this function, and the Brent algorithm[33], which instead does not rely on that quantity. Despite the Vega being relatively easy to compute, it can be close to zero and hence problematic to deal with. That is why carry out a Brent algorithm to find the solution from the SciPy library in Python.

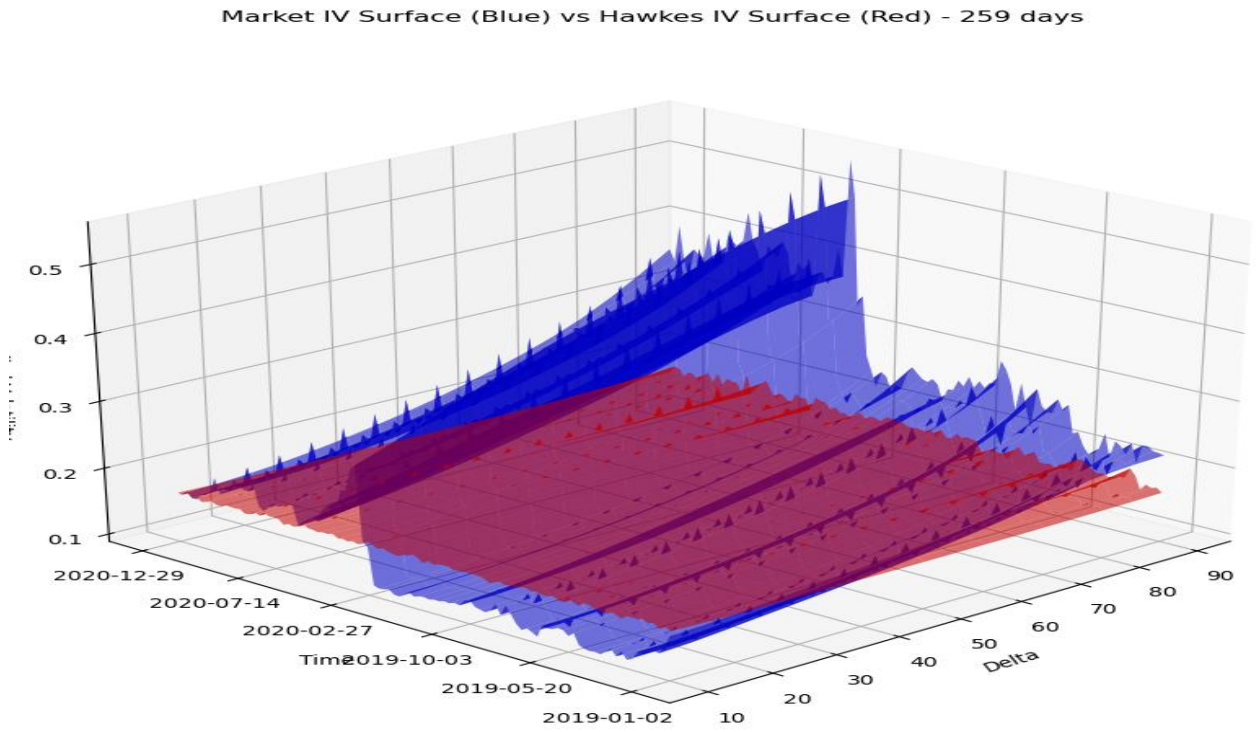
By using this algorithm to find the implied volatility $\sigma_{t,K}^{impl}$ for each t and K, and for each of the three maturities, we can compute the skew implied by the Hawkes model and finally compare it with market skew over time. The graphs that compare these skews over time are in *Figure 29*

[32] Tribhuvan Bisen, Quant Insider, (2025), Implied Volatility: Formulation, Computation, and Robust Numerical Methods

[33] Wikipedia, Brent's Method for numerical optimization, Metodo di Brent - Wikipedia

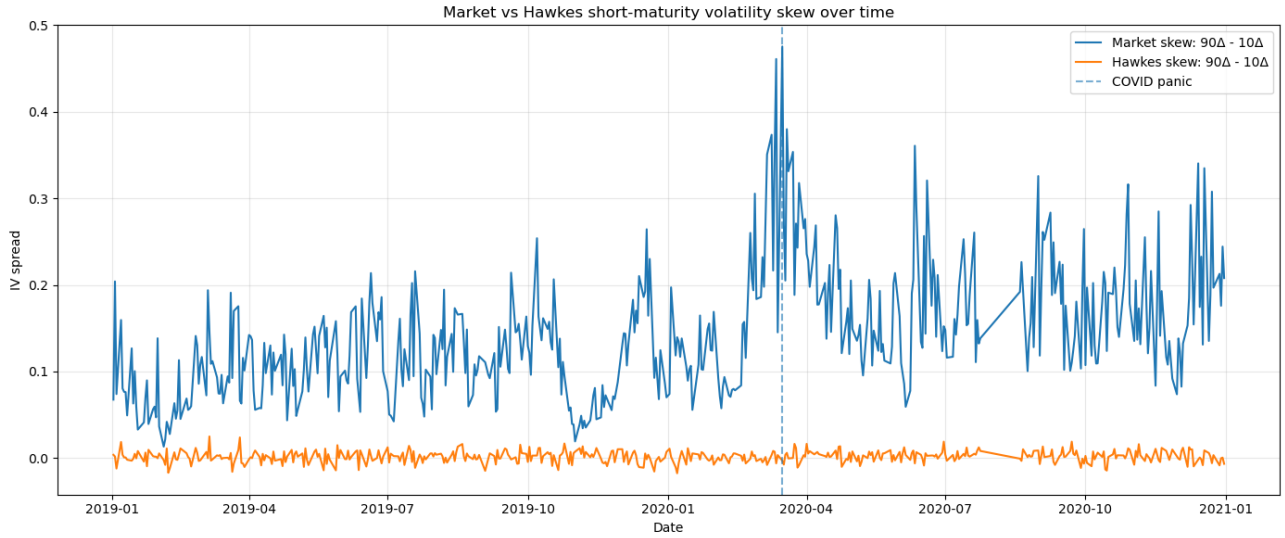
Figure 29


Market IV Surface (Blue) vs Hawkes IV Surface (Red) - 14 days
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
2020-12-29
2020-07-14
2020-02-27
2019-10-03
2019-05-20
2019-01-02
Time
10
20
30
40
50
60
70
80
90
Delta
Market IV Surface (Blue) vs Hawkes IV Surface (Red) - 77 days
0.6
0.5
0.4
0.3
0.2
0.1
2020-12-29
2020-07-14
2020-02-27
2019-10-03
2019-05-20
2019-01-02
Time
10
20
30
40
50
60
70
80
90
Delta



From these graphs, the Hawkes pricing model cannot correctly replicate the volatility skew of the SPX options. Despite the skew being not constant as in the Black Scholes model, the curves are too flat to correctly mimic the observed skews of the SPX. Another useful graph for visualizing this difference is in *Figure 30*, where for each day, only for the short maturity option, we compute the skew as the difference between the implied volatilities at level of 10Δ and 90Δ and we can see that the skew is much stronger in the observed market data rather than in the one implied by our pricing model, confirming that the model is not able to incorporate the higher implied volatility for OTM puts.

Figure 30



This finding is indeed reasonable and can be explained using different arguments. First, we noted before that the intensity parameter for the jump counting process is very low, hence indicating a weak impact on pricing, becoming then comparable to the Black Scholes model. With an extremely low intensity parameter, our model is much like the standard BS model, as the novelty that we add in the stochastic process of the asset has by construction low impact – in fact, the implied skew over time oscillates around zero, advocating in favor of a "almost-flat" implied volatility skew.

The main reason why this weak fit is not surprising is the complete absence in our model of the leverage effect[34]. It has been established in the literature that the main driver of the implied volatility skew is in fact this effect, that can be modelled by stochastic volatility models. In the Heston model seen above, the leverage effect can be easily included by finding $\rho < 0$, implying that there is a negative correlation between the asset price return and the stochastic volatility parameter – which is indeed the leverage effect. In our model we do not let the volatility parameter vary over time, hence we are completely disregarding this feature that could potentially help in reproducing the skew. Alternatives may thus be to extend the model to allow for a stochastic volatility parameter as well and see how this augmented model performs in fitting compared to the benchmark Heston model. In this way the potential impact on fitting of self-exciting jumps could be isolated and assessed. The model as was studied in this thesis, without a stochastic volatility, is then by structure unable to fit the volatility skew, as it focuses uniquely on jumps.

Last, let us also keep in mind that the calibration has been performed using option data at one date, due to computational complexity. A better-calibrated model, perhaps using option prices panel from the whole sample, may perform better in fitting the curve at extreme dates like those during the Covid panic.

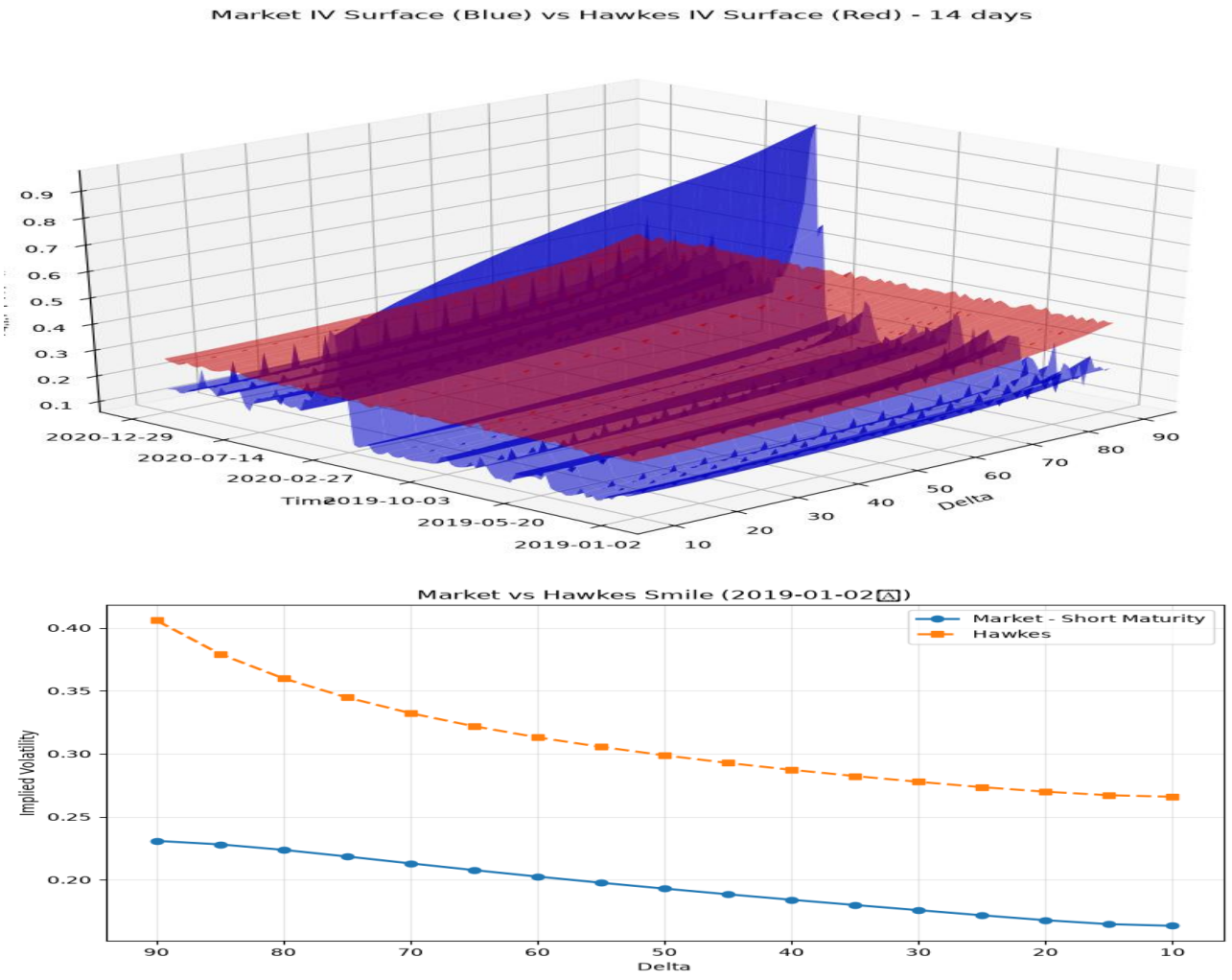
[34] Peter Carr and Liuren Wu, "Leverage Effect, Volatility Feedback and Self-exciting market disruptions", Journal of Financial and quantitative analysis (2019).

A last analysis we can carry out in the regard of volatility smile fitting is to see whether changing the parameters of the model can in any way impact the fitting. In particular, the parameter λ is peculiar as it measures the exposure to the jump risk factor; hence, we can try to assess how the fitting changes for different values of this parameter. The first test we will carry out is by using the following parameters' values:

σ^2	0.2043
λ	0.8297
a	-0.5625
b	0.4697
α	7.5111
β	9.4111

So, we will increase the λ parameter (baseline value for the intensity) of 0.8. The graphs for this specific vector parameters are the following:

Figure 31

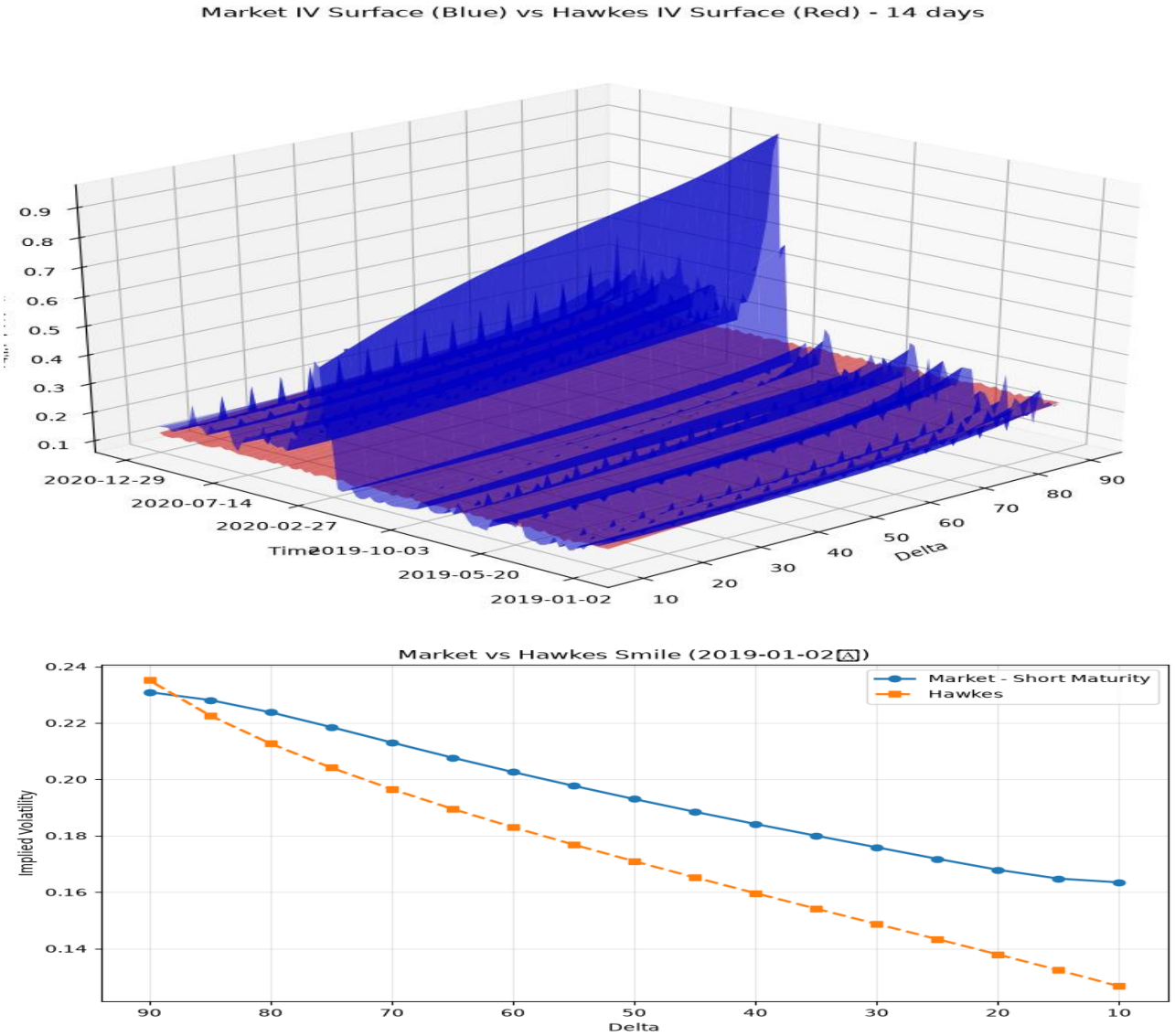


We see now the average value around the skew oscillates is much higher, thanks to the greater exposure to jump risk. We can also see a slightly higher negative slope in the implied volatility skew for a single day, but the fitting remains quite poor due to the low variability of the model skew through time. For higher maturities we see that the model surface gets higher values and lies entirely above the market volatility surface. Hence, we can try to lower the variance parameter to pull down the model surface. A second set of parameters to carry out this test is:

σ^2	0.0543
λ	0.8297
a	-0.5625
b	0.4697
α	7.5111
β	9.4111

The volatility fitting graphs for this vector of parameters are shown in *Figure 32*:

Figure 32



We can see that the average value of implied volatility produced by our Hawkes model is lower compared to the previous case, hence confirming that parameter σ^2 has a direct positive impact on the surface. The fitting on a normal day like in $2019-01-02$ is apparently better, but we can see from the bigger graph that the model still fails in particularly volatile days as those around the Covid panic. For higher maturities the model surface is still high (especially for longer maturities), indicating the same pattern as in the previous test.

5. Conclusions

In this thesis, a pricing model for fixed lookback options was developed based on an augmented Merton jump-diffusion framework, enriched with a stochastic intensity governed by a Hawkes Process. The intention was to augment the original model proposed by Merton in 1973 by introducing a self-exciting behavior in jumps which has been studied and observed empirically.

Information asymmetry among investors has been signaled as a possible cause of this clustering of jumps, but nevertheless it is plausible that more jumps are triggered by the occurrence of a single jump. Because of high levels of macro uncertainty, by the beginning of April the world has witnessed large fluctuations in the major global stock indices. We can consider the historical daily $+9.5\%$ of the $S\&P500$ registered on the $9^{th}$of April a jump, as the size of this increase is so massive in a short period of time that, in mathematical terms, it is hardly driven by the continuous risk exposure of a Geometric Brownian Motion. This jump does not stand alone, as in the previous days the same index also had extreme daily fluctuations, -6.65% on the 3^{rd} of April and -6% on the $4^{th}$of the same month. These values are well below the average daily return of the index and levels of standard deviation. So, we aimed at quantifying how strong this self-exciting behavior really is in a relatively "*jumpy*" index like S&P 500 (SPX) by computing the parameters of the exponential kernel of the Hawkes process and we also aimed at quantifying how much of a difference this introduction could trigger on a path-dependent exotic option.

The results, in this sense, indicate a clear difference in this pricing. The parameters for Hawkes process α and β imply a self-excited stochastic process, which parts from its background intensity parameter. The high value found for α implies a strong upward force in case of a price jump, whereas the high value β implies a strong decaying motion after a price jump. This would imply that there is a quite significant need to use Hawkes as a counting process and introduce stochastic features to the intensity parameter. Naturally, there is very much noise in the estimate of these parameters, and more sophisticated (perhaps Bayesian) methods to estimation may be a room of improvement for this analysis.

Nevertheless, the difference in the prices computed by the two mathematical models is significant. The traditional Merton model prices the *At-the-money* fixed lookback at 270 \$, whereas the Merton model with this stochastic intensity parameter prices the same product at 284 \$, implying a difference of 5%. This difference is not huge, but still statistically relevant as each point estimate does lie outside the other's estimator interval of confidence, implying a significant difference.

To complete the analysis also the Greeks computed with the augmented Merton framework has been computed by finite difference methods. Despite being gamma neutral, this exotic derivative priced using Hawkes process is delta positive, implying an upward slope of the option pricing function with respect to the underlying price space. It is also Vega neutral, implying that a change in the volatility parameter would not affect the price of the option.

In conclusion, we would argue that the clustering of jumps in the American index empirically observed in the data for SPX between $01-01-2020$ and $08-01-2025$ is relevant for option pricing purposes and supports the need for incorporating a Hawkes process as the jump intensity component within the Merton model. Potential improvements for the analysis include a deeper search for a global minimum for the loss function minimization problem and a solid and robust statistical procedure to calibrate parameters α and β. Despite this pricing model not being able to fit the observed market skew of SPX, it remains interesting for pricing purposes because of this observed and significant differences in the prices for the lookback compared to the benchmark Merton model.

Bibliography

Option prices when Underlying stocks returns are discontinuous; Robert C. Merton; Journal of Financial Economics; January 1976
Topics in Quantitative Finance, lecture notes for the graduate course Applied numerical Finance; Anna Battauz; Bocconi University; 2024-2025, 83 – 99
Hawkes Processes: Simulation, Estimation, and Validation; Patrick Laub, supervised by Phil Pollet; The University of Queensland, Australia; 2014; 9-22
Implementing Derivatives Models; Les Clewlow and Chris Strickland; 1998
Option pricing and hedging in jump diffusion models; Yu Zhou; Uppsala Universitet; 2010
Hawkes Processes in finance; Emmanuel Bacry, Iacopo Mastromatteo, Jean-Francois Muzy; Ecole Polytechnique, Université de Corse; 2015
Control Variates Techniques for Monte Carlo simulation; Roberto Zvechman; Operations Research Department; 2003
Exponential Kernels with Latency in Hawkes Processes: Application in Finance; Marcos Costa Santos Carreira; January 2021
Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-constrained Optimization; Ciyou Zhu, Richard Byrd, Jorge Nocedal; Northwestern University, University of Colorado at Boulder; 1-3
Numerical Methods II; Mike Giles; Oxford University Mathematical Institute
Monte Carlo – 1st Class; Anna Battauz; Bocconi University
Variance Reduction Techniques; Anna Battauz; Bocconi University
Estimating the sensitivities; Anna Battauz; Bocconi University
Modelling price and variance jump clustering using the Marked Hawkes Process; Chen, J., Clements; Oxford University Press; Journal of Financial Econometrics; 2023
Simulation, Estimation and applications of Hawkes Processes; Katarzyna Obral; University of Minnesota; June, 2016; 3-29
Simulation Efficiency and an Introduction to Variance Reduction methods; Martin Haugh; Colombia University; 2017
Acheroui, I., Bacry, E., Gaïffas, S., & Poulenard, A. (2020). tick.hawkes — tick 0.6.0 documentation. École Polytechnique. Retrieved from tick.hawkes — tick 0.6.0 documentation
Amit, H. (2020, August 26). A brief introduction to Hawkes processes. Medium. Retrieved from A Brief Introduction to Hawkes Processes | by Hey Amit | We Talk Data | Medium
Chen, J. (2023). Monte Carlo simulation: What it is, how it works, history, 4 key steps. Investopedia. Retrieved from Monte Carlo Simulation: What It Is, How It Works, History, 4 Key Steps
Borsa Italiana. (n.d.). Opzioni su azioni. Retrieved April 8, 2025, from https://www.borsaitaliana.it
TradingView. (n.d.). Curva dei rendimenti dei titoli di Stato dell'Italia: IT10Y. Retrieved April 8, 2025, from https://it.tradingview.com/symbols/TVC-IT10Y/yield-curve/
AI Post. (2023). Ottimizzazione locale e globale: trovare la soluzione migliore in grandi set di dati. Retrieved from https://blog-ai.it/ottimizzazione-locale-globale
HashDork. (2023). Cos'è l'algoritmo di ottimizzazione: una guida completa. Retrieved from Cos'è l'algoritmo di ottimizzazione: una guida completa
HashDork. (2023). Esempi di algoritmo di discesa del gradiente. Retrieved from bing.com/ck/a?!&&p=676c876358d5963008eb14d617bd7c0c6b7c3d9abeaebeb773a52a0240be34a1JmltdHM9MTc0OTUxMzYwMA&ptn=3&ver=2&hsh=4&fclid=324cb3bb-1ea9-6480-3428-a60a1f746526&psq=Esempi+di+algoritmo+di+discesa+del+gradiente+-+HashDork&u=a1aHR0cHM6Ly93d3cuaGFzaGRvcmsuY29tL2l0L2VzZW1waW8tZGktYWxnb3JpdG1vLWRpLWRpc2Nlc2EtZGVsLWdyYWRpZW50ZS8&ntb=1
HashDork. (2023). Una breve introduzione all'algoritmo di ottimizzazione BFGS. Retrieved from https://it.python-3.com/?p=4397
AI Post. (2023). BFGS in poche parole: un'introduzione ai metodi quasi-Newton. Retrieved from https://ichi.pro/it/bfgs-in-poche-parole-un-introduzione-ai-metodi-quasi-newton-37043576956164
Bing. (n.d.). Video. Retrieved 2025, from Understanding scipy.minimize part 1: The BFGS algorithm
Barndorff-Nielsen, O. E., & Shephard, N. (2006). Econometrics of testing for jumps in financial economics.

Journal of Econometrics, 131(1–2), 377–398.
Numerical Lab. (2023). Differenze finite: introduzione al metodo numerico e primi esempi. Retrieved from Differenze finite: introduzione al metodo numerico e primi esempi
WRDS, OptionMetrics, Option Prices
WRDS, OptionMetrics, Volatility Surface
Wikipedia, MSE, Errore quadratico medio - Wikipedia
Tribhuvan Bisen, Quant Insider, (2025), Implied Volatility: Formulation, Computation, and Robust Numerical Methods
Wikipedia, Brent's Method for numerical optimization, Metodo di Brent - Wikipedia
Properties of European Options, Lorenzo Naranjo (2023), 7. Properties of European Options — Options, Futures and Derivatives Securities
Properties of European Options, Lorenzo Naranjo (2023), 13. The Black-Scholes Model — Options, Futures and Derivatives Securities
David Bates, "Post –'87 Crash Fears in S&P 500 futures options", National Bureau of Economic research (1997),
Jim Gatheral, "The volatility Surface", A Practitioner's guide, John Wiley & Sons, Inc.
Peter Carr and Liuren Wu, "Leverage Effect, Volatility Feedback and Self-exciting market disruptions", Journal of Financial and quantitative analysis (2019).
Ciliberti, Bouchad, Potters, "Smile dynamics – a theory of the implied leverage effect" (2008)

AI Disclosure Statement

I hereby declare and confirm that the work in this dissertation is completely my own and I guarantee its authenticity. No portion of the work referred to in the dissertation has been previously published, in any format.

Generative artificial intelligence tools (AI) have not been used in the writing of this work. In the preparation of the thesis, regarding the coding with Python, ChatGBT-5.6 Sol. by OpenAI has been used. Mostly for complicated portions of code, the AI tool has been efficient in reducing computational time in algorithms. Despite being later modified and adjusted by the author, codes provided by AI have represented an efficient building block from where to start.

The main risk with this procedure is inefficient codes that may stall the working flow, or codes that lack precision and disconnected from theoretical results. Human supervision has been embraced to avoid this issue, by checking every line of the codes provided to ensure coherency and soundness.

In the author's perspective, AI enable to efficiently reduce time in writing codes whilst keeping a good structure and quality in the output. AI helped in making the work better, but did not, in any way, did the work alone without the author's supervision.