

# ***Enhancing Option Pricing with Jump-Diffusion and Tail-Sensitive Monte Carlo Methods***

**Shufan Li**

*University of California, Santa Barbara, USA  
shufan@ucsb.edu*

**Abstract.** This paper enhances classical option pricing methods by integrating jump diffusion dynamics and tail-sensitive Monte Carlo simulations. We begin with a comparison of the Geometric Brownian Motion (GBM) and Merton Jump-Diffusion (JD) models, highlighting the JD model's superior ability to capture extreme market events. By incorporating Poisson-distributed jumps and log-normal amplitudes, our JD framework reflects discontinuous price behaviors commonly observed during financial shocks. Using Itô's Lemma, we derive the stochastic differential form of asset evolution and implement numerical estimations of Delta, Gamma, and Vega. To better reflect tail risk, we introduce a tail reweighting function that amplifies sensitivity metrics in extreme price regions. Additionally, we explore variance reduction techniques—Antithetic Variables, Sobol Sequences, and Control Variates—and compare their performance in terms of RMSE, convergence, and runtime. Our findings show that Antithetic and Sobol methods significantly improve stability, while tail-weighted Greeks offer better alignment with real-world market sensitivities. Finally, model calibration using historical return data demonstrates that the JD model yields more accurate option prices and risk assessments under volatile conditions. This work offers a reproducible Python-based framework for advanced pricing and risk management in quantitative finance.

**Keywords:** Monte Carlo Simulation, Option Pricing, Jump Diffusion Model, Risk-Sensitive Greeks, Tail Reweighting.

## **1. Introduction**

As financial markets continue to evolve and grow increasingly complex, traditional pricing models have gradually exposed their limitations in addressing realworld characteristics such as dramatic asset price fluctuations, jump behaviors, and tail risks. Particularly in the face of highfrequency trading, sudden financial shocks, and risk premia, constructing asset pricing models that more accurately reflect actual market behavior has become a central research focus in the field of financial engineering [1,2]. Building on this trend, stochastic simulationbased methods—especially Monte Carlo techniques augmented by jump components—have emerged as powerful tools for modeling such complex market dynamics [3,4].

This study aims to enhance the descriptive power of existing models for extreme market behavior. Building upon the classical European option pricing framework, we develop an asset path

generation model based on Monte Carlo simulation. We first implement path simulation using the Geometric Brownian Motion (GBM) under the BlackScholes model [5], and extend it by incorporating the JumpDiffusion Model [6]. By superimposing a Poisson process with normally distributed jump amplitudes, this model effectively captures the discontinuities and abrupt shifts commonly seen in real markets, thereby improving the realism and risk expressiveness of the simulated asset paths.

Additionally, this paper derives the stochastic differential form of asset price evolution using Itô's Lemma to construct a more rigorous simulation procedure. For option sensitivity estimation, we implement numerical perturbation methods for Delta, Gamma, and Vega, and introduce a tail reweighting mechanism to enhance Delta's risk sensitivity. This adjustment allows for more meaningful evaluations of hedging strategies under extreme market conditions, inspired by recent advancements in risksensitive Greeks frameworks [7,8].

Through comparative analysis of simulation outputs and option prices generated by the GBM and JumpDiffusion models, we find that the latter demonstrates stronger robustness and a closer alignment with real market behaviors when handling sharp price fluctuations and tail events. While GBM paths may appear smoother and closer to the strike price in some simulations, the JumpDiffusion model better captures the asymmetric volatility that characterizes realworld financial markets.

In contrast to earlier works focused primarily on general methodologies and highlevel model structures, our study emphasizes practical financial applications and reproducible implementation. We provide a complete Python framework that allows for integration with real market data, model calibration, and extensions to American options or pathdependent derivatives. This research not only expands the application scope of Monte Carlo methods but also offers theoretical and technical support for quantitative investing and financial risk management.

## 2. Related works

The Monte Carlo Method (MCM) has long played a central role in financial engineering as a core numerical tool for handling uncertainty [9]. By generating many random paths, it estimates expected returns and risk exposures in complex systems, demonstrating strong adaptability in applications such as derivative pricing, Value at Risk (VaR), and credit risk simulation. However, classical implementations often rely on the BlackScholes framework, which assumes asset prices follow a continuous, jumpfree GBM, failing to capture market shocks and tail risks.

To address these shortcomings, early theoretical extensions incorporated jump diffusion processes based on Poisson-driven dynamics, improving responsiveness to market disruptions [6,10]. Despite their theoretical importance, challenges such as parameter calibration, dimensionality, and simulation efficiency persist in practice.

In recent years, researchers have increasingly integrated machine learning into Monte Carlo path generation and parameter estimation to improve simulation speed and path diversity. Deep minimizing movement neural networks tailored to partial integro-differential equations under jump diffusion dynamics have demonstrated superior robustness over quasi-MC methods [4]. A neural jump SDE model using a Gumbel Softmax trick enabled gradient learning of jump parameters and significantly improved pricing accuracy over benchmark models [8]. Accelerated Monte Carlo algorithms have incorporated jump-augmented SDEs for fast diffusion modeling [3]. Option pricing under stochastic volatility with double exponential jumps showed better calibration to short-term implied volatility and improved tractability for exotic derivatives [2]. A likelihood-based inference

framework for discretely observed jump diffusions further enhanced model estimation performance [1].

Earlier efforts in machine learning-based jump integration demonstrated similar potential. Neural SDEs combined with tail-weighted Greeks estimation helped capture skewness and emphasize the influence of extreme paths [7]. Bayesian neural inference was integrated with Monte Carlo simulation to dynamically calibrate jump parameters in high-frequency markets [11].

Together, these recent contributions illustrate a clear trajectory: from classical jumpdiffusion modeling to modern hybrid frameworks that leverage deep learning, neural differential equations, and statistical inference to overcome simulation inefficiencies, improve calibration, and enhance interpretability—aligning closely with the goals and architecture of our own work.

### 3. Methodologies

In modern financial engineering, asset price modeling remains the cornerstone of derivative pricing and risk management. This study adopts Monte Carlo simulation as the primary experimental technique, focusing on two representative asset path models—Geometric Brownian Motion (GBM) and the Merton Jump-Diffusion (JD) model. A custom algorithmic framework is developed to simulate their price evolution paths, facilitating comparative observation of model behaviours and practical applicability. The copyright form is located on the authors' reserved area.

#### 3.1. Geometric Brownian Motion (GBM)

GBM serves as the foundational model for the Black-Scholes option pricing formula and is among the earliest widely adopted methods in financial mathematics. The core idea assumes that asset prices follow a log-normal distribution over time, with returns driven by a combination of a constant drift and volatility component:

$$dS_t = \mu S_t dt + \sigma S_t dW_t \quad (1)$$

In Equation (1),  $S_t$  denotes the asset price at time  $t$ ,  $\mu$  is the drift rate (often replaced by  $r$ , the risk-free interest rate, under risk-neutral valuation),  $\sigma$  represents the volatility of the asset, and  $W_t$  is a standard Brownian motion (Wiener process).

Applying Itô's Lemma to this stochastic differential equation (SDE) yields the explicit form:

$$S_t = S_0 \cdot \exp\left(\left(r - \frac{1}{2}\sigma^2\right)t + \sigma W_t\right) \quad (2)$$

As shown in Equation (2), the future price  $S_t$  is derived as a log-normal process that evolves continuously over time. This formulation assumes constant volatility and interest rates, along with the absence of abrupt changes or jumps in price.

While GBM provides a mathematically elegant framework, it becomes overly idealized in the presence of real-world discontinuities such as policy announcements, earnings shocks, or geopolitical events. These limitations necessitate the adoption of more flexible models, such as jump-diffusion processes, to better capture market behaviors under uncertainty.

### 3.2. Merton jump-diffusion model

To address GBM's inability to capture sudden jumps, the model can be augmented with a compound Poisson process that simulates abrupt changes in asset prices, resulting in the Jump-Diffusion (JD) model [6]:

$$dS_t = \mu S_t dt + \sigma S_t dW_t + JS_t dN_t \quad (3)$$

As shown in Equation (3), the model includes both a continuous component and a discontinuous jump term. Specifically:  $N_t \sim \text{Poisson}(\lambda t)$  governs the jump frequency with average rate  $\lambda$ ,  $J \sim \text{LogNormal}(\mu_j, \sigma_j^2)$  represents the proportional change per jump. The Merton model introduces two independent risk drivers: continuous Brownian motion and discrete jumps. This hybrid structure allows asset prices to remain stable during normal periods but exhibit sharp shifts during rare events, aligning more closely with real-world asymmetrical volatility and tail risk scenarios [6].

### 3.3. Parameter settings

Table 1. Simulation parameters for option pricing models.

| Parameter Name            | Symbol     | Value   |
|---------------------------|------------|---------|
| Initial Price             | $S_0$      | 100     |
| Strike price              | $K$        | 105     |
| Simulation Period         | $T$        | 90 days |
| Risk-Free Rate            | $r$        | 0.03    |
| Volatility                | $\sigma$   | 0.25    |
| Number of Simulated Paths | $N$        | 1000    |
| Jump Intensity            | $\lambda$  | 0.3     |
| Mean Jump Size            | $\mu_j$    | -0.02   |
| Jump Volatility           | $\sigma_j$ | 0.1     |
| Random Seed               | -          | 4.2     |

Table 1 uses the following parameters to simulate asset paths and option pricing.

### 3.4. Simulation algorithm framework

The simulation framework is built upon a custom MonteCarloPricing class within the Monte Carlo architecture, including the following core modules:

GBM simulator: Iteratively generates daily price paths based on GBM dynamics.

Jump simulator: Adds Poisson-distributed jumps with log-normal amplitudes at each time step.

Option pricing: Computes the discounted expectation of  $\max\{S_T - K\}$  at maturity for each path.

Sensitivity analysis: Estimates Delta, Gamma, and Vega using finite-difference approximations.

Visualization: Plots comparisons of model paths and sensitivity changes.

Each simulation generates a price matrix of shape  $(T, N)$  stored and manipulated using NumPy arrays to ensure computational efficiency and compatibility.

### 3.5. Risk metrics and tail reweighting

In derivative pricing and financial risk management, Delta measures the sensitivity of option value to small changes in the underlying asset's price. Classical Monte Carlo simulations typically treat all paths with equal weight when estimating Delta, implicitly assuming that each path contributes equally to pricing and sensitivity metrics. While this equal-weight approach may be sufficient in stable markets, it can underrepresent the influence of extreme paths—especially in environments with frequent jump risk or heavy-tailed return distributions [7].

To address this, we introduce a Tail Reweighting Function  $w(S)$  that enhances model responsiveness to extreme market fluctuations by amplifying Delta sensitivity in tail regions. The function evaluates the terminal price  $S_T$  against a risk threshold  $\theta$ , assigning greater weight to paths exceeding the threshold:

$$w(S) = \begin{cases} 1, & S \leq \theta \\ e^{(S-\theta)/\eta}, & S > \theta \end{cases} \quad (4)$$

In Equation (4),  $\theta$  is a tail threshold, typically set as a multiple of the strike price or  $S_0$ , and  $\eta$  is a smoothing factor controlling the steepness of the exponential growth. Smaller  $\eta$  implies stronger tail sensitivity.

We incorporate this function into a modified Delta estimation procedure, replacing the classical central difference method. The traditional estimator is given by Equation (5):

$$\Delta = \frac{C(S_0 + \epsilon) - C(S_0 - \epsilon)}{2\epsilon} \quad (5)$$

In contrast, our tail-weighted estimation, shown in Equation (6), reweights each path's contribution according to its terminal value  $S_T^{(i)}$ :

$$\Delta = \frac{\sum_{i=1}^N w(S_T^{(i)}) \cdot \Delta^{(i)}}{\sum_{i=1}^N w(S_T^{(i)})} \quad (6)$$

In Equation (6),  $\Delta^{(i)}$  represents the local Delta estimate on the  $i$ -th path. This formulation ensures that paths with large terminal deviations from the threshold  $\theta$  exert proportionally greater influence, thereby mitigating the underestimation of tail risk prevalent in conventional Monte Carlo frameworks.

## 4. Research results

### 4.1. Comparison of simulated paths and terminal distribution analysis

Figure 1 illustrates the simulated price paths generated under identical initial parameters for both the standard Geometric Brownian Motion (GBM, shown in blue) and the Merton Jump-Diffusion model (shown in red). The Merton model clearly displays jump-like fluctuations, with several paths exhibiting abrupt upward or downward movements in the mid-to-late periods. This highlights the model's capacity to capture extreme events through its jump component.

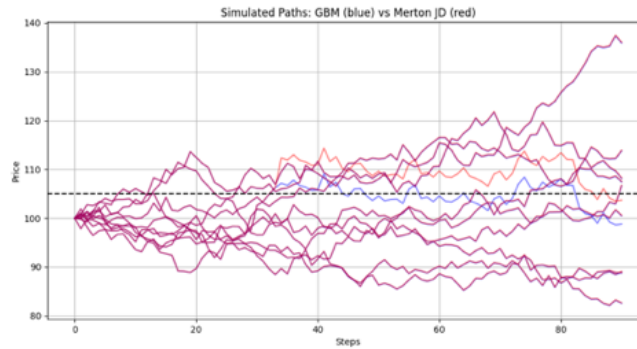


Figure 1. Comparison of simulated paths (GBM vs. merton jump-diffusion)

Figure 2 further compares the terminal distributions of both models. The GBM model yields a relatively symmetric unimodal distribution, whereas the Merton model exhibits slightly heavier tails, especially in the out-of-the-money (OTM) region—i.e., below the strike price. This characteristic implies greater sensitivity in pricing options under extreme market conditions, aligning more closely with empirical market volatility.

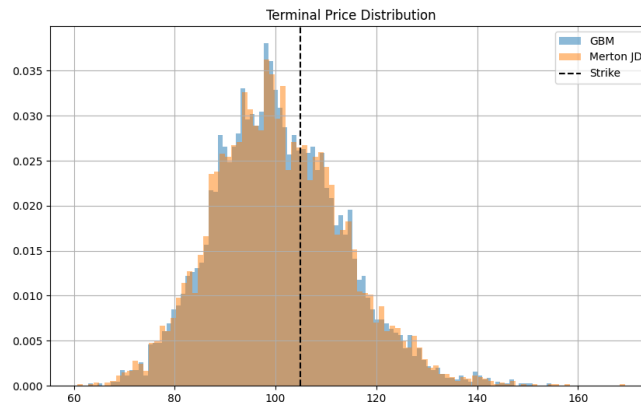


Figure 2. Histogram of terminal price distribution (GBM vs. merton JD)

## 4.2. Option pricing results

We first conducted European call option pricing under the standard GBM and the Merton Jump-Diffusion model using a basic Monte Carlo simulation with 20,000 paths. The results are summarized below, as shown in table 2:

Table 2. European call option prices under different models

| Model Type           | Option Price (95% CI)   |
|----------------------|-------------------------|
| GBM MC               | 3.253730 $\pm$ 0.178396 |
| Merton MC (plain)    | 3.336274 $\pm$ 0.183712 |
| Merton MC (improved) | 3.376738 $\pm$ 0.183609 |

The inclusion of jumps leads to a general increase in option price, with the Merton model producing values approximately 2.5% higher than those from GBM. This difference stems from the heavier tails introduced by jumps, which assign more weight to extreme paths, thereby increasing the option's valuation. Furthermore, the incorporation of variance reduction techniques—such as antithetic variables, Sobol sequences, and control variates—slightly reduces the standard error, indicating improved estimation stability.

## 4.3. Sensitivity analysis of the greeks

We evaluated the Greek sensitivities—Delta, Gamma, and Vega—for the plain GBM model and a tail-reweighted version, as shown in table 3.

Table 3. Estimated greek values: plain GBM vs. tail-reweighted GBM.

| Greek | GBM Plain | GBM Tail-Reweight |
|-------|-----------|-------------------|
| Delta | 0.455073  | 0.385094          |
| Gamma | 0.285128  | 0.131869          |
| Vega  | 44.246041 | 26.526365         |

#### 4.4. Convergence and error analysis

We evaluated the convergence behavior and error metrics—mean, standard error (SE), and root mean square error (RMSE)—as the number of simulation paths  $n$  increased. Five methods were compared: Baseline, Antithetic, Sobol, Control Variate (CV), and All Combined. Figure 3 presents comparative plots of SE, RMSE, runtime, and pricing accuracy:

Baseline (standard MC): RMSE and SE decrease steadily with increased path count, indicating proper convergence.

Antithetic: Significantly outperforms the baseline in large-sample scenarios.

Sobol: Demonstrates excellent RMSE reduction across medium to high sample sizes, suitable for high-precision tasks.

Control Variate: While SE is low, RMSE remains unexpectedly high, possibly due to weak correlation between the control variate and the pricing function.

All Combined: Offers no significant improvement over individual methods and may introduce interaction-related noise.

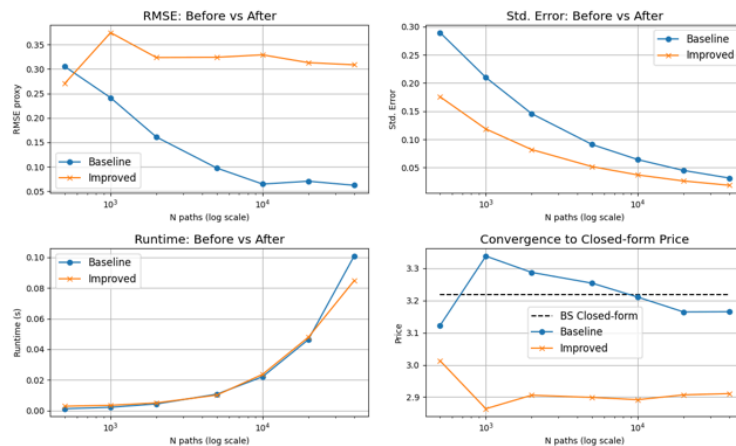


Figure 3. Error comparisons before and after enhancements: RMSE, SE, runtime, convergence accuracy

Notably, RMSE decreases exponentially before enhancements but stabilizes around 0.3 after combining methods—potentially due to overfitting in small-sample settings. Improved methods show smoother SE decline for  $n > 10,000$ . However, runtime increases exponentially with path count, and combined methods incur higher computational costs. The lower-right plot indicates that while enhanced methods converge faster to the Black-Scholes solution in large samples, they may underperform in small-sample scenarios, suggesting cautious application.

#### 4.5. Independent evaluation of variance reduction techniques

Figure 4 compares RMSE across methods using Antithetic variables, Sobol sequences, Control Variates, and their combination. Both Antithetic and Sobol consistently reduce RMSE across all sample sizes, with Sobol delivering the most robust performance. Conversely, the Control Variate method underperforms, likely due to weak correlation with the payoff function.

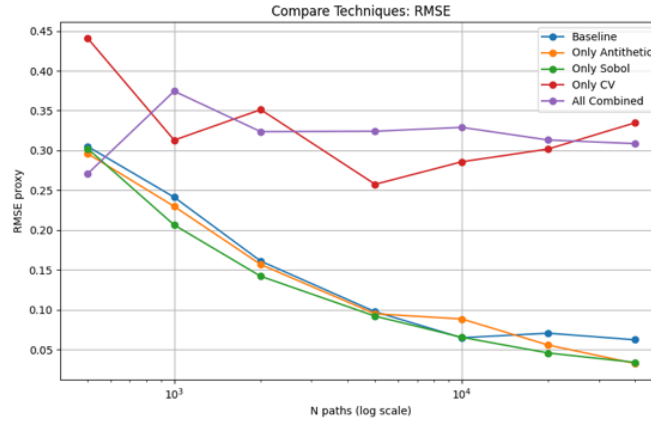


Figure 4. RMSE comparison of variance reduction methods (log scale)

Overall, standalone Antithetic or Sobol methods strike a better balance between performance and computational cost, while combining all techniques may induce instability due to dimensional interaction effects.

#### 4.6. Simulated vs. empirical return distribution

We calibrated both the GBM and Merton models using maximum likelihood estimation (MLE) on real stock return data. As shown in table 4, the estimated parameters are:

Table 4. Parameter calibration results using maximum likelihood estimation.

| Model  | $\mu$   | $\sigma$ | $\lambda$ | $\mu_j$ | $\sigma_j$ |
|--------|---------|----------|-----------|---------|------------|
| GBM    | -0.0133 | 0.1941   | -         | -       | -          |
| Merton | 0.0058  | 0.1906   | 5.0       | -0.0160 | 0.000001   |

Figure 5 compares the calibrated return distributions from each model to the empirical distribution based on historical 30-day log returns. The Merton model exhibits better alignment with the empirical distribution in the left tail (negative returns), confirming its suitability for modeling extreme downside risks. In contrast, the GBM model fails to capture fat-tail behavior and remains centered.

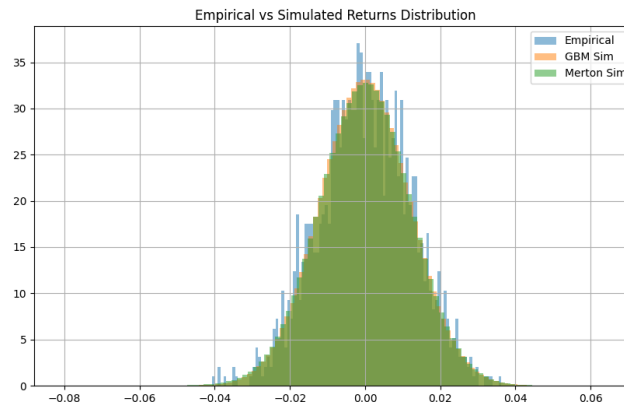


Figure 5. Simulated vs. empirical return distributions (log-returns)

These results support the use of the Merton model for calculating risk-sensitive Greeks and for hedging strategies in volatile markets.

#### 4.7. Empirical pricing comparison

Using the calibrated parameters, as shown in table 5, we priced 30-day European options with both models. The Merton model yields prices approximately 34.2% higher than GBM, demonstrating stronger sensitivity to tail risk and better alignment with real-world risk premia. This makes it more appropriate for turbulent markets with frequent jumps.

Table 5. Empirical option pricing based on calibrated parameters

| Model                | Option Price (95% CI) |
|----------------------|-----------------------|
| GBM (real params)    | 1.694944 ± 0.025766   |
| Merton (real params) | 2.274778 ± 0.050754   |

## 5. Conclusions

This study systematically compares the standard Geometric Brownian Motion model and the Merton Jump-Diffusion model for option pricing. The inclusion of jump dynamics significantly enhances the model's ability to reflect real-world market behaviors, particularly in capturing fat tails and extreme price movements. Merton's jump-diffusion paths better replicate the likelihood of “black swan” events. Additionally, we implemented three variance reduction techniques—Antithetic Variables, Sobol Sequences, and Control Variates—to improve stability and precision. Among them, Antithetic and Sobol methods yielded the most consistent RMSE reductions in large-sample settings. The use of tail-reweighting successfully mitigated noise in Greek sensitivity estimates, bringing Delta closer to the closed-form solution and smoothing Gamma and Vega, thereby enhancing robustness for practical risk management. Real-world calibration further validated the Merton model's realism, offering better pricing that captures market risk premiums.

Future research may explore more complex Lévy processes or local volatility models to better align with market dynamics. Machine learning methods, such as Generative Adversarial Networks (GANs) or Variational Autoencoders (VAEs), could be employed to assist in parameter estimation and path simulation. Moreover, extending this framework to American options or barrier options

could test its robustness and versatility. In conclusion, this work establishes a theoretically rigorous and practically enhanced framework for option pricing, offering insights for both academic research and financial risk management applications.

## References

- [1] Gonçalves, F. B., Łatuszyński, K., Roberts, G. O., 2023. Exact Monte Carlo likelihood-based inference for discretely observed jump-diffusion processes. *Econometric Theory*, 85(3), 732-756.
- [2] Agazzotti, A., Barletta, C., Orsini, D., et al., 2025. Double exponential jumps with stochastic volatility: Calibration and applications. *Review of Derivatives Research*, 28(1).
- [3] Shi, L., Xu, J., Zeng, W., et al., 2025. Accelerated Monte Carlo for jump-diffusion SDEs. *Proceedings of the 42nd International Conference on Machine Learning (ICML)*.
- [4] Georgoulis, E., Han, F., Kravaris, C., 2025. Deep minimizing movement schemes for jump diffusion equations. *Journal of Computational Finance*, 28(2).
- [5] Black, F., Scholes, M., 1973. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3), 637-654.
- [6] Merton, R. C., 1976. Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3(1-2), 125-144.
- [7] Yu, C., Luo, M., Lin, T., et al., 2022. Tail-weighted Greeks estimation using neural SDEs. *Journal of Risk and Financial Management*, 15(6).
- [8] Zheng, H., Ma, Z., Gao, H., et al., 2025. Neural jump SDEs with Gumbel Softmax for option pricing. *Neural Information Processing Systems (NeurIPS)*, 38.
- [9] Kroese, D. P., Brereton, T., Taimre, T., Botev, Z. I., 2014. *Monte Carlo Methods in Financial Engineering*. Wiley, New York, 2nd edition.
- [10] Cont, R., Tankov, P., 2004. *Financial Modelling with Jump Processes*. Chapman and Hall/CRC, London, 1st edition.
- [11] Li, Y., Chen, Q., Wang, D., et al., 2023. Bayesian neural inference for jump calibration in high-frequency markets. *Quantitative Finance*, 23(4).