Université Lumière Lyon 2

MSc Economics & Finance
Master Thesis

Market-Consistent Option Pricing: A Weighted Monte Carlo Approach applied to Black-Scholes-Merton and Heston Models

Baris KARADAS

Under the supervision of Bertrand Tavin

27 August, 2025

Contents

1 Introduction

In modern derivative markets, a fundamental requirement for any pricing model is market consistency, the ability to align prices with those observed on the market. However, aligning model outputs with market data is challenging, particularly in the presence of market features such as stochastic volatility, asset price jumps, path-dependent payoffs, and so on. To tackle these complexities, practitioners often rely on Monte Carlo simulation as a flexible numerical technique. Monte Carlo methods are widely used in derivative pricing and risk analysis due to their ability to handle path-dependent payoffs and complex stochastic dynamics for which no closed-form solutions exist. As Glasserman (*Glasserman, 2003*) notes, Monte Carlo simulation has become "an essential tool in the pricing of derivative securities and in risk management", enabling valuation under different assumptions.

The Black-Scholes-Merton (*Black & Scholes, 1973*) model provides a simple, closed-form solution to price standard European options. It assumes constant volatility over time and across strike price. This assumption makes the model easy to use, but it does not match what we actually see in markets. In reality, implied volatility (the number traders plug into the BSM formula to match market prices) varies across strikes and maturities. This creates patterns such as volatility smile, volatility skew, where implied volatilities for out-of-the-money and in-the-money are different than at-the-money ones. The BSM cannot explain this because of constant volatility assumption. To better fit the market data, advanced models let volatility change over time or depend on other factors. One well-known example is the Heston Model (*Heston, 1993*), where volatility is stochastic and tends to revert to a long-term average. Other extensions include local volatility models, where volatility is a function of both asset price and time ($\sigma_t = \sigma(S_t, t)$), jump-diffusion models, which allow for sudden 'jumps' in the asset price or its volatility. Indeed, these models fit better what we have on the market but they also come with more parameters, which need to be calibrated.

Calibrating a stochastic volatility model means finding a set of model parameters that produce theoretical option prices as close as possible to market prices across strikes and maturities. This is essentially an inverse problem typically solved by numerical optimization. Efficient techniques have been developed for this purpose, for instance, Carr & Madan's (*Carr & Madan, 1999*) Fast Fourier Transform (FFT) method allows quick computation of option prices from the Heston model's characteristic function, enabling calibration of Heston's parameters by matching model prices to an entire smile in seconds. However, even the best-fitting model may not match all observed quotes.

To solve these discrepancies, one approach is to adjust the model's output after simulation. This is the idea behind the Weighted Monte Carlo (WMC) method introduced by Avellaneda et al. (*Avellaneda et al., 2001*). WMC modifies the simulation results so it reproduces the market prices of selected benchmark instruments, like vanilla options. First, a Monte Carlo simulation is run to generate a large number of possible price paths under the assumptions of the initial model (BSM, Heston, etc..). In standard Monte Carlo, options values are computed by taking the equally-weighted average of all simulated payoffs. In contrast, WMC assigns new weights to paths so the weighted average of simulated payoffs matches the market prices which alters the risk-neutral measure implied by the simulation. The method reshapes the model's output distribution to align with the real market. Note that because there are usually more simulated paths than

calibration constraints, there are infinitely many ways to choose the weights.

To resolve this non-uniqueness, we use an entropic version of Weighted Monte Carlo. Rather than selecting arbitrary weights, the entropic version selects the solution that makes the smallest possible change to the original model's distribution. In practice, this is realised by minimising the relative entropy also known as the Kullback-Leibler (KL) divergence (*Avellaneda et al., 2001*) between the new weighted probability distribution and the original one. This entropy-based criterion ensures that the adjustment to the simulated probability measure is as conservative as possible (one that deviates the least). The results is a new calibrated risk-neutral probability measure that reproduces the market prices without changing the model dynamics. When the market prices used for calibration are consistent (i.e., free from arbitrage), this constrained optimisation problem has a unique and well-defined solution (*Avellaneda et al., 2001*). In that case, the WMC procedure yields a single, market-consistent measure by construction.

Research Question. How effectively can the Weighted Monte Carlo (WMC) method, via entropic re-weighting of simulation paths, reduce pricing error and estimator variance to match observed market quotes when the underlying Black-Scholes- Merton and Heston models are miscalibrated?

1.1 Literature Overview

The challenge of making option pricing models consistent with observed market prices has been a key issue in quantitative finance since Black-Scholes-Merton (*Black & Scholes, 1973*). While BSM was a big step forward, it does not include real-world features such as stochastic volatility. To address this gap, the literature quickly evolved. Stochastic volatility models, most notably the Heston model (*Heston, 1993*), making it possible to fit the market implied volatility surface more accurately than BSM model. Further developments, such as jump-diffusion and local volatility models, expanded the range of available tools. Yet, even these models often fails to fit all observed market prices across different strikes and maturities.

Much research focuses on model calibration, adjusting the model's parameters so the results match market data. Many calibration techniques have been developed such as least-squares optimisation, Fast Fourier Transform (FFT) method of Carr and Madan (*Carr & Madan, 1999*). While these techniques can reduce discrepancies between model and market prices, they also introduce new challenges, such as instability, over-fitting, or high computational cost.

As summarized by Glasserman (*Glasserman, 2003*), Monte Carlo methods offer unmatched flexibility for pricing complex and path-dependent derivatives, but their accuracy is sensitive to the quality of model calibration, and they often suffer from high variance. To overcome these issues, a different branch of research has emerged. Rather than increasing model complexity, researchers have explored adjusting the risk-neutral probability measure itself. The Weighted Monte Carlo (WMC) method introduced by Avellaneda et al. (*Avellaneda et al., 2001*) is an example. Instead of modifying the model, WMC recalibrates the outputs of a Monte Carlo simulation by assigning new weights to the simulated paths, so the weighted prices match observed market prices.

This approach is based on concepts from information theory, especially, the idea of minimising entropy. It has led to more research, for example, Gudmundsson & Vyncke

(*Gudmundsson & Vyncke, 2021*) study how reliable and practical the WMC method is. Even though WMC is promising in theory, several important questions remain. For instance, how well does WMC perform in real situations compared to standard calibration methods, especially when the basic model is not well-calibrated or is too simple? What are the trade-off when it comes to pricing accuracy, reducing variance, and computational cost? Most existing studies have focused mainly on theory or on limited tests with just one model or very simple market setups.

1.2 Thesis Objectives and Structure

This thesis aims to investigate the pricing performance of standard Monte Carlo and Weighted Monte Carlo (WMC), using the Black-Scholes and Heston models. The goal is to see how well each method works in realistic market situations. By doing this, we want to expose the strengths and limits of WMC. Our empirical analysis will focus on S&P 500 options data (August 2025), chosen for its liquidity and data availability. We keep option with a maturity < 1 year and select strike to reduce our dataset to 361 options.

The rest of this thesis is organised into six sections. **Section 2** explains the basics of the Monte Carlo simulation, how it works, how it is used to price options, and its main limitations. **Section 3** introduces the Weighted Monte Carlo (WMC) method, including the idea of entropy minimisation, how simulation path weights are calculated, and how the method can be used to match market prices. **Section 4** applies these methods to the Black-Scholes-Merton, showing how WMC can recalibrate a simple model to match market prices, and discussing improvements in pricing accuracy and reduction of variance. **Section 5** focuses on the Heston model, covering how it can be calibrated using Monte Carlo, and discussing practical challenges in simulation. **Section 6** extends the WMC method to the Heston model, exploring how re-weighting affects market consistency and the impact on pricing. **Section 7** compares all the approaches used, analyses their advantages and disadvantages, and summarises the key findings, while suggesting directions for further research.

2 Monte Carlo Methodology

2.1 Monte Carlo Simulation in Derivatives Pricing

Monte Carlo simulation is a numerical technique for computing expectations by random sampling. In derivative pricing, the fundamental no-arbitrage principle says that the fair price of a financial derivative is the expected present value of its future payoff under a risk-neutral probability measure. If X is the random payoff at maturity T, and r is the risk-free rate, the price at time 0 is (*Glasserman, 2003*):

$$P_0 = \mathbb{E}^{\mathbb{Q}}\left[e^{-rT}X\right] \tag{2.1}$$

where $\mathbb{Q}$ indicates the risk-neutral measure. Monte Carlo methods approximate this expectation by simulating many random scenarios instead of solving analytically. Monte Carlo is particularly useful for path-dependent payoffs or situations involving several sources of uncertainty (*Glasserman, 2003*).

To use Monte Carlo for pricing, you specify a model for the underlying asset under the risk-neutral measure. Then, the simulation generates a large number of random sample paths and uses these to estimate the expected payoff. By simulating many independent scenarios and applying the law of large numbers, the estimator improves as more samples are used. We suppose we want to compute an expectation $I = \mathbb{E}\left[f(Y)\right]$ for some random variable Y and $f(Y)$ is the discounted payoff as a function of that path, then the Monte Carlo estimator with N simulated scenarios is:

$$\hat{I}_N = \frac{1}{N}\sum_{i=1}^{N} f(Y_i) \tag{2.2}$$

where $Y_1, Y_2, \ldots, Y_N$ are independent samples drawn from the distribution of Y. By construction, $\hat{I}_N$ is an unbiased estimator of I (since $\mathbb{E}\left[\hat{I}_N\right] = I$), and it is also consistent, converging to the true value I as $N \to \infty$ by the law of large numbers (*Glasserman, 2003*). In option pricing, each Y_i is a simulated path, and $f(Y_i)$ is the present value of the payoff in that scenario.

For example, consider a European call option with strike price K and maturity T. Its payoff is $(S(T) - K)^+$ (where $(x)^+ = \max\{x, 0\}$), and the present value of that payoff is $e^{-rT}(S(T) - K)^+$. Simulating N independent price paths for the underlying asset, each ending at $S_i(T)$, gives the Monte Carlo estimator:

$$\hat{C}_N = \frac{1}{N}\sum_{i=1}^{N} e^{-rT}\left(S_i(T) - K\right)^+ \tag{2.3}$$

where $S_i(T)$ is the terminal asset price on path i. Then, $\mathbb{E}\left[\hat{C}_N\right] = \mathbb{E}\left[e^{-rT}(S(T) - K)^+\right]$, so if the model is correct, the Monte Carlo average will converge to the Black–Scholes–Merton price as N grows large.

Because Monte Carlo pricing is based on random sampling, it naturally gives an estimate of the uncertainty or error in the price. Thanks to the central limit theorem, for large N, the estimator $\hat{I}_N$ is approximately normal around the true value I. The standard error (standard deviation) of the estimator declines as $1/\sqrt{N}$. In fact, one can show that $\sqrt{N}(\hat{I}_N - I)$ converges in distribution to $N(0, \sigma^2)$, where $\sigma^2 = \mathrm{Var}[f(Y)]$ is the variance of

the payoff (*Glasserman, 2003*). This means that to reduce the statistical error by a factor of 2, one must roughly quadruple the number of simulation trials. The Monte Carlo error can be estimated from the sample, if $\hat{\sigma}$ is the sample standard deviation of the discounted payoff values $\{f(Y_i)\}_{i=1}^{N}$, then an approximate 95% confidence interval for the true price is $\hat{I}_N \pm 1.96\,\hat{\sigma}/\sqrt{N}$ (*Glasserman, 2003*). In practice, analysts often report such intervals or error bars with their Monte Carlo estimates. These statistical measures reinforce that Monte Carlo gives a numerical approximation of the price not an exact value but, with enough computation, the approximation can be made arbitrarily accurate in theory.

2.2 Efficiency and Limitations of Monte Carlo

Monte Carlo, as we said, is a flexible method, but it also comes with limitations. One primary drawback is the slow convergence of the estimator. The estimator's error decays as $\mathcal{O}(1/\sqrt{N})$ (*Glasserman, 2003*), implying that reducing the error by a factor of 10 requires 100 times more simulations. In practice, achieving high accuracy may necessitate millions of paths, which is computationally expensive.

Another limitation is statistical variance. Because results rely on random sampling, prices fluctuate between simulation runs. This is especially problematic for payoffs with high variance, e.g., deep out-of-the-money options, where most paths yield zero and only a few contribute significantly, leading to slow convergence. In such cases, variance reduction techniques, such as control variates, importance sampling, antithetic variates (*Glasserman, 2003*) can reduce the number of required paths, though some residual error always remains. These fluctuations may be challenging in real-time application. For instance, MC-based Greeks or risk metrics may display noise from one day to the next, complicating hedging (*Glasserman, 2003*). Ensuring that the simulation error remains within acceptable bounds is crucial.

A further challenge is time discretisation. Since most models are continuous, paths must be simulated in discrete time. Large time steps Δt introduce discretisation bias (*Glasserman, 2003*), while smaller ones increase computational load. Advanced methods, like higher-order schemes and exact simulation can reduce this bias but at the cost of complexity.

In summary, Monte Carlo pricing must navigate between 3 key sources of error, model error (poor model specification), statistical error (finite sampling), discretisation error (approximating continuous dynamics).

2.3 Monte Carlo, Model Calibration, and Market Consistency

It is important to understand that Monte Carlo (MC) is a numerical method, not a model itself. It computes expectations under a given model, but the accuracy of those prices depends on the model's accuracy. Even with millions of simulations, if the model is miscalibrated or not coherent, the computed prices may deviate from market prices. For this reason, calibration is essential. This process adjusts model parameters to align estimated prices with observed prices for a set of liquid instruments, vanilla options. For instance, the BSM model can be calibrated by setting its volatility to match the price of a single option, giving the implied volatility. Nonetheless, when implied volatilities vary across strikes/maturities (volatility surface), one volatility parameter is insufficient to capture market behaviour. Models like Heston (*Heston, 1993*), which include stochastic

volatility, or local/jump models, introduce additional parameters to fit a broader range of options, at the cost of complexity.

Calibration in these models involves solving an inverse problem where we adjust parameters to match observed quotes. Since this requires repeated evaluations of the pricing function, MC is rarely used during calibration, due to its high computational cost. Instead, analytical techniques (like Fourier inversion or FFT methods in Heston) are preferred, enabling rapid estimation of prices during optimisation.

Even so, perfect calibration is rarely possible. Models may capture the general shape of the implied volatility surface, but some residual differences between theoretical prices and observed prices often remain. To address this, we can use the Weighted Monte Carlo (WMC), introduced by Avellaneda et al. (*Avellaneda et al., 2001*). The next section explores this method in detail.

3 Weighted Monte Carlo Methodology

As Glasserman (*Glasserman, 2003*) notes, while Monte Carlo (MC) offers great flexibility, it is "sensitive to calibration quality". In practice, models are calibrated to fit market prices of vanilla options before being used in simulations, but perfect calibration is rarely possible. This leads to small but persistent errors in pricing. One way to reduce these errors is to adjust the simulation results directly, without changing the underlying model. This is the idea behind the Weighted Monte Carlo (*Avellaneda et al., 2001*). Recall that WMC assigns different weights to the paths to match selected market prices. This allows us to keep a certain model while improving the accuracy of the results (*Avellaneda et al., 2001*).

3.1 Entropy Minimisation Framework

The need of entropy minimisation is intuitive, without it, the re-weighting problem admits infinitely many solutions (*Avellaneda et al., 2001*), as there are many ways to assign weights that match observed prices. To address this, the Weighted Monte Carlo (WMC) approach is formalised as a constrained optimisation problem based on relative entropy (*Avellaneda et al., 2001*). The key idea is to adjust the probability distribution of simulation outcomes as little as possible in order to exactly fit selected market prices.

Let $u = (u_1, \ldots, u_\nu)$ denote the original risk-neutral under which the Monte Carlo paths are generated, and let $p = (p_1, \ldots, p_\nu)$ denote the adjusted distribution after reweighing. WMC seeks p that matches the prices of N benchmark instruments (the calibration set) while remaining as "close" as possible to u. Closeness is measured by the Kullback-Leibler divergence (relative entropy) $D_{\mathrm{KL}}(p \,\|\, u)$:

$$D(p \,\|\, u) = \sum_{i=1}^{n} p_i \log\left(\frac{p_i}{u_i}\right) \tag{3.1}$$

(*Avellaneda et al., 2001*) where the sum runs over all sample paths $i = 1, \ldots, \nu$ in the Monte Carlo Simulation. This quantity is always non-negative, and equals zero only when $p = u$. By minimising this divergence while enforcing the calibration constraints, WMC finds the adjusted distribution that is as close as possible to the original model, while matching market data. This principle leads to a well-defined optimisation problem.

Calibration Constraints. Suppose we have a set of N benchmark instruments, indexed by $j = 1, \ldots, N$, with observed market prices C_j. Each instrument j has a discounted payoff vector, denoted by g_{ij} for $i = 1, \ldots, \nu$. In standard Monte Carlo, all paths are equally weighted ($u_i = 1/\nu$), so the price estimate for instrument j is $\frac{1}{\nu}\sum_{i=1}^{\nu} g_{ij}$.

WMC instead assigns a weight p_i to each path i (with $p_i \geq 0$ and $\sum_{i=1}^{v} p_i = 1$) and chooses these weights such that the weighted average of each payoff matches the market price. In other words, p is chosen to satisfy the N linear pricing constraints.

$$\sum_{i=1}^{v} p_i \, g_{ij} = C_j, \qquad j = 1, \ldots, N \tag{3.2}$$

which ensure that under the new weighted measure p, the expected payoff of each calibration instrument j equals its market price. Writing $p(\text{path } i) = p_i$, Equation (5) is simply $\mathbb{E}_P[g_j] = C_j$ for each j. Typically, we have $\nu > N$ (far more simulation paths

than constraints), so there are infinitely many choices of $\{p_i\}$ that would satisfy these equations. This indeterminacy is resolved by the entropy-minimization criterion. Among all solutions $p = (p_1, \ldots, p_v)$ that satisfy the constraints, we select the one that minimises $D_{\mathrm{KL}}(p \,\|\, u)$.

In practice, the original distribution u is implicitly defined by the MC simulation. If the simulation is an unbiased sample from the risk-neutral measure, then each simulated path initially has equal weight, so $u_i = 1/v$. More generally, if the simulation uses variance reduction or biased sampling, then each path may have a different initial weight u_i, still satisfying $\sum_{i=1}^{v} u_i = 1$. We can now formally state the WMC optimisation problem and find the optimal weights p_i by solving:

$$\begin{aligned} &\text{Minimise}_{\{p_i\}} \quad D(p \,\|\, u) = \log v + \sum_{i=1}^{\nu} p_i \log p_i \\ &\text{subject to} \qquad \sum_{i=1}^{\nu} p_i = 1, \\ &\qquad\qquad \sum_{i=1}^{\nu} p_i \, g_{ij} = C_j, \quad j = 1, \ldots, N \end{aligned} \tag{3.3}$$

This is a convex optimisation problem. The objective (relative entropy) is convex in $\{p_i\}$, and the constraints are linear. Standard results from convex optimisation (via the method of Lagrange multipliers or the Kuhn–Tucker conditions) guarantee a unique solution for p, provided the constraints are consistent (*Avellaneda et al., 2001*). In particular, Avellaneda et al. show that if the target prices $\{C_j\}$ are arbitrage-free (i.e., internally consistent), then the entropy-minimisation has a unique solution for p (*Avellaneda et al., 2001*). Intuitively, an arbitrage-free set of calibration prices ensures that the linear constraints does not contradict each other, so there is a well-defined "closest" probability measure that fits them.

Solution via exponential tilting. We now derive the form of the optimal weights that solve the above problem. Consider the Lagrangian $\mathcal{L}(p, \lambda, \zeta)$, introducing Lagrange multipliers $\lambda_1, \ldots, \lambda_N$ for the N pricing constraints and a multiplier ζ for the normalization constraint $\sum_{i=1}^{\nu} p_i = 1$:

$$\mathcal{L}(p, \lambda, \zeta) = D(p \,\|\, u) - \sum_{j=1}^{N} \lambda_j \left(\sum_{i=1}^{\nu} p_i g_{ij} - C_j \right) - \zeta \left(\sum_{i=1}^{\nu} p_i - 1 \right) \tag{3.4}$$

Setting the derivative $\partial \mathcal{L} / \partial p_i$ to zero yields the first-order condition:

$$\frac{\partial \mathcal{L}}{\partial p_i} = \log p_i + 1 - \sum_{j=1}^{N} \lambda_j g_{ij} - \zeta = 0 \tag{3.5}$$

Solving for p_i gives:

$$p_i^* = \frac{\exp\left(\sum_{j=1}^{N} \lambda_j g_{ij} \right)}{Z(\lambda)} \tag{3.6}$$

where

$$Z(\lambda) = \sum_{i=1}^{\nu} u_i \exp\left(\sum_{j=1}^{N} \lambda_j g_{ij}\right) \tag{3.7}$$

and $u_i = 1/\nu$ for uniform sampling.

The optimal weights p_i^* can be interpreted as the Radon-Nikodym derivative $\frac{dP^*}{dP_0}$ evaluated at each path i. This means each path's new probability under the re-weighted measure P^* is its original probability times an exponential correction factor, ensuring consistency with market prices. The resulting measure P^* is continuous with respect to P_0 and is the "closest" probability measure to P_0 in the sense of Kullback–Leibler divergence.

Dual Formulation. The optimal probability weights p_i^* are determined by solving the dual optimisation problem, which consists in minimising the convex function.

$$W(\lambda) = \log(Z(\lambda)) - \sum_{j=1}^{N} \lambda_j C_j = \log\left\{\frac{1}{\nu}\sum_{i=1}^{\nu} \exp\left(\sum_{j=1}^{N} g_{ij}\lambda_j\right)\right\} - \sum_{j=1}^{N} \lambda_j C_j. \tag{3.8}$$

where C_j are the observed market prices of the calibration instruments. Minimising $W(\lambda)$ yields the unique set of multipliers that ensure the WMC prices fit the market exactly (see, e.g., Avellaneda et al., 2001). To match market prices, we obtain the optimal multipliers λ_j^* by minimising $W(\lambda)$, i.e, by solving.

$$\frac{\partial W}{\partial \lambda_k} = 0 \tag{3.9}$$

which leads to the following system of equations:

$$\frac{\sum_{i=1}^{\nu} g_{ik} \exp\left(\sum_{j=1}^{N} g_{ij}\lambda_j\right)}{\sum_{i=1}^{\nu} \exp\left(\sum_{j=1}^{N} g_{ij}\lambda_j\right)} = C_k, \qquad k = 1, \ldots, N \tag{3.10}$$

At the optimum, the expected value of each benchmark payoff under the new measure matches the observed market price, ensuring consistency. Once the optimal λ^* is found, the corresponding weights are given by

$$p_i^* = \frac{1}{Z(\lambda^*)} \exp\left(\sum_{j=1}^{N} g_{ij}\lambda_j^*\right) \tag{3.11}$$

This establishes the link between the dual formulation and the moment-matching system for the weights. In practice, the dual objective $W(\lambda)$ is a convex function of the multipliers λ, and the optimal solution can be efficiently found using gradient-based optimization algorithms, such as L-BFGS, as implemented in Avellaneda et al. (*Avellaneda et al., 2001*). Once the optimal multipliers λ^* are found, the corresponding weights p_i^* are computed by the exponential tilting formula (*Avellaneda et al., 2001*). This algorithm ensures that the WMC simulation matches the market prices while minimally distorting the original distribution.

3.2 Practical Implementation

The Weighted Monte Carlo (WMC) can be summarized in four main steps. First, you generate N asset price paths using a model (e.g., Black-Scholes or Heston), assigning equal initial weights to each path. Second, for every simulated path, payoffs are computed for each calibration instrument (e.g., set of vanilla options), ensuring all payoffs are discounted to present value if necessary. Third, the optimal weights for each path are determined by solving a constrained entropy-minimisation problem, which adjusts the path probabilities to match observed market price. Finally, the re-weighted paths are used to price other derivatives under the new, market-consistent probability measure (*Avellaneda et al., 2001*).

3.3 Summary and Comparison with Importance Sampling

Although both Weighted Monte Carlo (WMC) and importance sampling use non-uniform path weights, their objectives and implementation are different. Importance sampling is a variance-reduction technique applied before simulation, where we alter the sampling distribution to focus on "important" scenarios, then compensate with likelihood ratio weights to maintain unbiasedness. In contrast, WMC modifies path weights after simulation, it corrects the model's bias rather than reducing variance. For this reason, WMC should be viewed as a recalibration technique applied after the simulation, not as a variance-optimisation tool like importance sampling.

Overall, the WMC method provides a practical and robust way to improve market consistency in Monte Carlo simulation, and it enables models to fit observed market data. By using convex optimisation and entropy minimisation, WMC gives market-consistent pricing and helps reduce estimator variance. The following chapters will illustrate WMC's effectiveness in practice for the Black-Scholes and Heston models.

4 Weighted Monte Carlo under Black-Scholes-Merton

4.1 Assumptions and Mathematical Formulation

The Black-Scholes-Merton (BSM) model gives an analytical way to price European options. It is based on several assumptions, no arbitrage and no transaction costs, no dividends, constant risk-free rate, constant volatility, log-normal price behaviour. Under the rik-neutral measure $\mathbb{Q}$ (i.e., after adjusting for risk preferences), the evolution of the underlying asset price S_t is given by the stochastic equation (*Black & Scholes, 1973*):

$$dS_t = r\, S_t\, dt + \sigma\, S_t\, dW_t \tag{4.1}$$

where dW_t is a standard Wiener process (Brownian motion). The solution to this stochastic differential equation under the risk-neutral measure is (*Black & Scholes, 1973*):

$$S_T = S_0 \exp\left((r - \tfrac{1}{2}\sigma^2)T + \sigma\sqrt{T}\, Z\right), \quad Z \sim \mathcal{N}(0,1) \tag{4.2}$$

where S_T is the price of the underlying at maturity, S_0 the initial price of the underlying, σ the constant volatility, r the constant risk-free rate, T the maturity of the option, and Z a standard normal random variable. Under the assumptions of Black-Scholes, we obtain a closed-form solution for pricing European-style options. For instance, the price of a European call option C with strike K and maturity T is given by:

$$C = S_0\, N(d_1) - K\, e^{-rT}\, N(d_2) \tag{4.3}$$

where

$$d_1 = \frac{\log(S_0/K) + (r + \tfrac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}, \qquad d_2 = d_1 - \sigma\sqrt{T} \tag{4.4}$$

In this formula, $N(\cdot)$ denotes the cumulative distribution function of the standard normal law. The term $S_0 N(d_1)$ represents the expectation under $\mathbb{Q}$ of the payoff S_T conditional on the option finishing in the money, while $Ke^{-rT}N(d_2)$ is the present value of the strike weighted by the risk-neutral probability of exercise. The BSM model thus provides a closed-form solution for the theoretical price of a standard European option. The same pricing result can be expressed using the risk-neutral valuation formula. In this framework, the call price is given as the discounted expected payoff under the risk-neutral probability measure $\mathbb{Q}$ (*Glasserman, 2003*):

$$C = e^{-rT}\, \mathbb{E}^{\mathbb{Q}}\left[(S_T - K)^+\right] \tag{4.5}$$

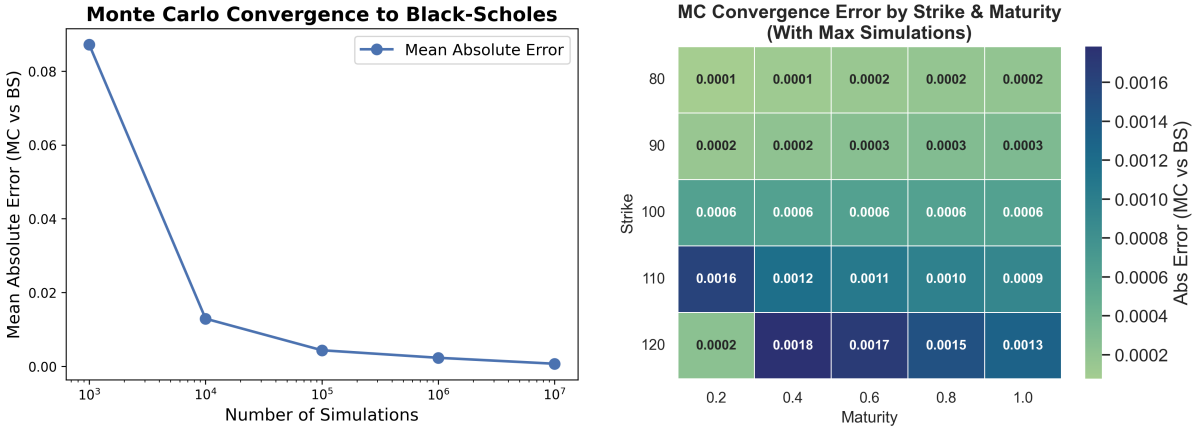

Figure 1: Monte Carlo Convergence to Black-Scholes

4.2 Monte Carlo Simulation under BSM Model

Using N independent paths, the Monte Carlo approximation for the price of a European call option with strike K and maturity T is (*Glasserman, 2003*):

$$\hat{C}^{\text{MC}} = \frac{1}{N} \sum_{i=1}^{N} e^{-rT} \max(S_T^{(i)} - K, 0) \tag{4.6}$$

To calibrate the Monte Carlo simulation to observed option prices, we use a least-squares error method to compare simulated prices with market data. Also, we include a penalty term to avoid extreme values of the calibrated parameter σ. This results in a regularised objective function of the form:

$$\text{Loss}(\sigma) = \sum_{j=1}^{m} \left(\hat{C}_j^{\text{MC}}(\sigma) - C_j^{\text{market}} \right)^2 + \lambda \cdot \sigma^2 \tag{4.7}$$

This is equivalent to applying an L^2 penalty (Tikhonov regularisation), where $\lambda > 0$ is a hyper-parameter that controls the strength of the regularisation. We then solve:

$$\sigma^\star = \arg\min_{\sigma} \text{Loss}(\sigma) \tag{4.8}$$

This procedure ensures that the calibrated volatility $\sigma^\star$ fits the market prices well while avoiding values that are too large or unrealistic.

4.2.1 Calibration to Market Surface

As we have shown (Figure 1.) that the Monte Carlo simulation converges to Black-Scholes (*Black & Scholes, 1973*) closed-form solution, we use the closed-form (Eq. 4.3) to calibrate the volatility parameter by fitting to observed option prices across various strike prices K, different maturity T. Using 78 options from our dataset, we obtain a calibrated volatility of $\sigma^\star \approx 0.147$. Then to assess the quality of the Black-Scholes calibration, we report the mean absolute pricing errors across different dimensions (maturity, strike, and surface) on 361 options.

Metric Maturity	Mean Eror (abs)
Mean (Filtered for maturity < 0.3, Price = 143.4487)	14.1701
Mean (Filtered for maturity >= 0.3 and < 0.6, Price = 255.5377)	19.6949
Mean (Filtered for maturity >= 0.6, Price = 358.4788)	23.5197

Figure 2: BS - Mean absolute error by maturity

Figure 2. shows the mean absolute error by option maturity. Errors are highest for long-dated options, with a mean error of 23.52, and decrease for shorter maturities but still remain significant. We have to keep in mind the initial price of the option, since a 1 euro difference matters much more for 10 euro than for a 100 euro option.

Metric Strike	Mean Eror (abs)
Mean (Filtered for strike < 100, Price = 357.6859)	19.3907
Mean (Filtered for strike >= 100 and < 105, Price = 212.3817)	12.3856
Mean (Filtered for strike >= 105, Price = 75.9592)	24.5741

Figure 3: BS - Mean absolute error by strike

Figure 3. shows that the model fits best for strikes below 105, with the lowest mean absolute error at 12.39 for strikes between 100 and 105. However, the error jumps to 24.57 for deep out-of-the-money options (strike above 105).

Metric Surface	Mean Eror (abs)
Mean (Filtered for price < 81.70$, 20% Quantile)	21.3337
Mean (Filtered for price > 81.70$, 20% Quantile)	17.3396
Mean (Filtered for price > 214.65$, 50% Quantile)	17.5303
Mean (Filtered for price > 370.98$, 80% Quantile)	27.7857
Mean	18.1739
Median	17.7032

Figure 4: BS - Mean absolute error by surface

Figure 4. summarises the error distribution over the entire surface, using price quantiles. Errors are particularly high in the lowest price segment, reaching 21.33, and even higher for the most expensive options at 27.79. The mean error is 18.18, while the median is 17.70, which points to a distribution slightly skewed by large errors, especially for options that are far out-of-the-money.

While the Black-Scholes (BS) provides a simple way to price option pricing, it has several limitations when applied to real market data. The key reason that it fails to fit the volatility surface, is the constant volatility assumption across all strikes and maturities. In practice, market exhibit volatility smile and volatility skew. This inflexibility leads to systematic pricing errors. Note that, the BS model is not used for its pricing accuracy, it is used as benchmark to extract market-implied volatility. Its main value lies in providing a simple, standardised reference for volatility.

4.3 Weighted Monte Carlo Simulation under BSM Model

We have discussed the foundations and mathematical formulation of the Weighted Monte Carlo (WMC) approach, we now apply this methodology to the Black-Scholes-Merton (BSM) model (*Avellaneda et al., 2001*). We simulated 50000 paths of Eq.(4.1). We use the average implied volatility from our dataset as $\sigma_{IV} = 0.1416$. The gradient tolerance in the L-BFGS-B routine was set to be 10^{-12} and we used equal weights $w_i = 10^{-5}$ for least-squares approximation. The algorithm started with $\lambda_i = 0$, $i = 1, \ldots, 36$ (with 6 forward) converges after approximately 125 iterations of the L-BFGS-B routine. The entire calibration takes about 3 seconds on a standard PC. Then we use the calibrated WMC to price our dataset (*Avellaneda et al., 2001; Gudmundsson & Vyncke, 2021*).

Metric Maturity	Mean Eror (abs)
Mean (Filtered for maturity < 0.3, Price = 143.4487)	4.6443
Mean (Filtered for maturity >= 0.3 and < 0.6, Price = 255.5377)	4.9590
Mean (Filtered for maturity >= 0.6, Price = 358.4788)	7.2572

Figure 5: WMC-BS - Mean absolute error by maturity

Figure 5. shows that with the WMC-BS approach, the mean absolute error drops across all maturities. For short maturities, the error falls to 4.64, and even for longer maturities, it stays relatively low at 7.25. This highlights an improvement over the standard Black-Scholes model.

Metric Strike	Mean Eror (abs)
Mean (Filtered for strike < 100, Price = 357.6859)	8.8014
Mean (Filtered for strike >= 100 and < 105, Price = 212.3817)	4.0792
Mean (Filtered for strike >= 105, Price = 75.9592)	2.1464

Figure 6: WMC-BS - Mean absolute error by strike

Figure 6. By comparing across strikes, we see a substantial improvement in the pricing errors. For strikes below 100, the error drops from 19.39 to 8.80. For strikes between 100 and 105, it decreases from 12.39 to 4.08. For deep out-of-the-money options, the error falls sharply from 24.57 to 2.14 with WMC-BS.

Metric Surface	Mean Eror (abs)
Mean (Filtered for price < 81.70$, 20% Quantile)	2.3496
Mean (Filtered for price > 81.70$, 20% Quantile)	6.0888
Mean (Filtered for price > 214.65$, 50% Quantile)	7.8715
Mean (Filtered for price > 370.98$, 80% Quantile)	9.1073
Mean	5.3103
Median	2.7789

Figure 7: WMC-BS - Mean absolute error by surface

Figure 7. The calibrated WMC-BS model reduced considerably across the entire surface. The mean error drops to 5.31 and the median to 2.78, compared to 18.18 and 17.70. WMC procedure can significantly enhances pricing accuracy while using a simple model (BSM). However, the simplicity of the model comes at a cost.

The statistical summary (Figure 8) and Figure 9 show the impact of WMC calibration on the distribution of path weights. While the mean probability per path remains at the uniform value (0.00002 for 50000 paths), the high kurtosis (145.13) and skewness (7.33) indicate that a small number of paths receive disproportionately large weights. This is also visible in the plot, where many spikes correspond to outlier paths. The KL divergence (relative entropy) between the calibrated risk-neutral measure and the prior measure is relatively high at $D(p \,\|\, u) = 0.38$, which corresponds to an $\alpha = 1 -$

$\frac{D(p\,\|\,u)}{\log \nu} = 0.9649$ and "effective number of paths" of $N = \nu e^{-D(p\,\|\,u)} = 34186$ (68.37% of the original sample). In other words, although 50000 scenarios were simulated, after WMC calibration, only around 34000 of them carry significant probability weight. The effective number of paths quantifies the true diversity of scenarios that contribute to the calibrated price. A lower effective number suggests that the calibration relies heavily on a small subset of scenarios, which can make the results less robust and more sensitive to individual paths. Contrary, when the effective number stays close to the total simulated paths, the calibration preserves more of the original diversity, leading to a more stable and reliable measure.

That is a fundamental limitation of the BSM model. "Twisting" the probability measure can lead to an unstable model. This highlights the need for sophisticated models, such as those incorporating stochastic volatility. Stochastic volatility models capture the non-constant implied volatility surface observed in markets much better.

Statistic	Value
Mean	2e-05
Standard Error	9.69825e-08
Median	1.41753e-05
Mode	1.66453e-08
Standard Deviation	2.16859e-05
Sample Variance	4.7028e-10
Kurtosis	145.133
Skewness	7.32916
Range	0.000800237
Minimum	1.66453e-08
Maximum	0.000800253

Figure 8: WMC-BS - Descriptive statistics for calibrated probabilities $(p_1, \ldots, p_{50000})$

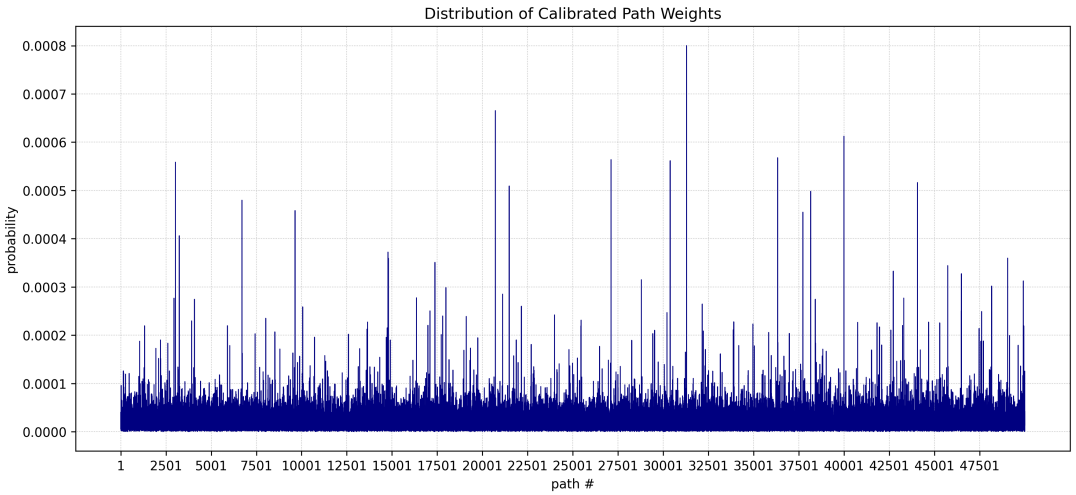

Figure 9: WMC-BS - Distribution of calibrated Path Weights

5 The Heston Model

5.1 Closed Form Formula

The Heston model is a widely used extension of Black-Scholes that allow stochastic volatility, reflecting real market features (*Heston, 1993*). The model admits a closed form solution for European options. This allows us to compute theoretical prices much faster than by simulating every possible path, which is valuable when calibrating to the market data. The dynamics of the underlying price and its variance, under risk-neutral measure, are given by (*Heston, 1993*):

$$\begin{cases} dS_t = rS_t\,dt + \sqrt{V_t}\,S_t\,dW_t^{(1)} \\ dV_t = \kappa(\theta - V_t)\,dt + \sigma\sqrt{V_t}\,dW_t^{(2)} \end{cases} \tag{5.1}$$

with

$$\mathrm{Cov}(dW_t^{(1)}, dW_t^{(2)}) = \rho\,dt \tag{5.2}$$

Here, S_t is the asset price at time t, V_t is the instantaneous variance (i.e., the square of volatility), r is the risk-free interest rate, κ is the rate at which variance reverts to its long-term mean, θ is the long-term average variance, σ is the volatility of variance (often called "vol of vol"), and ρ is the correlation between asset price and its variance. This framework allows volatility to fluctuate randomly over time, to exhibit clustering (periods of high vol and low vol), and to mean-revert. These features are consistent with empirical evidence from financial markets (*Heston, 1993*). Under Heston model, the closed form of an European call option at time 0 with strike K and maturity T, which relies on the Fourier inversion of the characteristic function, is given by (*Heston, 1993*):

$$C(S_0, K, v_0, \tau) = S_0 P_1 - K e^{-rT} P_2 \tag{5.3}$$

Where P_1 and P_2 are defined as:

$$P_j = \frac{1}{2} + \frac{1}{\pi}\int_0^{+\infty} \mathrm{Re}\left[\frac{e^{-i\phi \ln K} f_j(X, K, v_0, \tau; \phi)}{i\phi}\right] d\phi \quad j \in \{1, 2\} \tag{5.4}$$

Here, the characteristic functions, given by:

$$f_j(X, K, v_0, \tau; \phi) = \exp\left(C(\tau; \phi) + D(\tau; \phi)v_0 + i\phi X\right) \tag{5.5}$$

where the parameters of characteristic function can are:

$$C(\tau; \phi) = i\phi r\tau + \frac{a}{\sigma^2}\left([b_j - i\phi\rho\sigma + d]\tau - 2\ln\frac{1 - ge^{d\tau}}{1 - g}\right) \tag{5.6}$$

$$D(\tau; \phi) = \frac{b_j - i\phi\rho\sigma + d}{\sigma^2}\left[\frac{1 - e^{d\tau}}{1 - ge^{d\tau}}\right] \tag{5.7}$$

$$g = \frac{b_j - i\phi\rho\sigma + d}{b_j - i\phi\rho\sigma - d} \tag{5.8}$$

$$d = \sqrt{(i\phi\rho\sigma - b_j)^2 - \sigma^2(2i\phi u_j - \phi^2)} \tag{5.9}$$

$$u_1 = \frac{1}{2} \qquad u_2 = -\frac{1}{2} \qquad a = \kappa\theta \tag{5.10}$$

$$b_1 = \kappa + \lambda - \rho\sigma \qquad b_2 = \kappa + \lambda \tag{5.11}$$

These formulas allow the option to be computed by evaluating the characteristic function and performing a single numerical integration for each P_1 and P_2, which is more efficient than path-by-path simulations. However, this approach still requires a separate numerical integration for each option. In practice, calibrating the model to the volatility surface, often involving hundreds of option quotes, can be computationally intensive. In real markets, rapid pricing is crucial. This leads to use more efficient technique, such as the Fast Fourier Transform (FFT). Nonetheless, I use the standard numerical integration approach for each option, which, while less efficient, is more straightforward to implement and sufficient for demonstrating the Weighted Monte Carlo methodology.

5.2 Calibration of the Heston Closed Form Model

For better results, we have to use some realistic initial parameters. We take the average implied volatility from our dataset as $\sigma_{IV} = 0.1416$, so $V_0 = 0.02$ (0.1416^2). Similarly, we use a θ near that number. Also, in the equity market, we historically observed negative correlation between volatility and prices, so we set a negative $\rho = -0.5$.

Initial Parameters	
Mean reversion κ	1.5
Correlation between Brownian Motions ρ	-0.5
Volatility of the volatility σ	0.3
Long-term mean of variance θ	0.02
Variance initial value V_0	0.02

For calibrating the Heston model, we use the closed-form pricing formula as given in Eq. (5.3), with characteristic functions defined by Eqs. (5.5)–(5.11). The calibrating uses the same data as section 3 (78 options). We use, again a L-BFGS-B algorithm with a gradient tolerance set to be 10^{-5}. The model converges after 29 iterations and takes about approximately 250 seconds to calibrate.

Calibrated Parameters	
Mean reversion κ	2.1329
Correlation between Brownian Motions ρ	-0.2038
Volatility of the volatility σ	0.1146
Long-term mean of variance θ	0.0173
Variance initial value V_0	0.0203

With the calibrated parameters, we price the entire dataset of 361 options in order to assess the quality of Heston calibration. We report the mean absolute pricing errors across different dimensions (maturity, strike, and surface). In addition, thanks to the stochastic volatility feature of the Heston model, the volatility is no longer constant. Therefore, it is useful to compare the implied volatility surface generated by the model to that observed in the market.

Metric Maturity	Mean Eror (abs)
Mean (Filtered for maturity < 0.3, Price = 143.4487)	6.7269
Mean (Filtered for maturity >= 0.3 and < 0.6, Price = 255.5377)	6.2575
Mean (Filtered for maturity >= 0.6, Price = 358.4788)	6.2137

Figure 10: HestonCF - Mean absolute error by maturity

Figure 10. shows that the Heston model delivers stable mean absolute errors across all maturities. While its accuracy for intermediate and long maturities is comparable to WMC-BS, the error for short maturities remains slightly higher.

Metric Strike	Mean Eror (abs)
Mean (Filtered for strike < 100, Price = 357.6859)	9.2830
Mean (Filtered for strike >= 100 and < 105, Price = 212.3817)	5.6573
Mean (Filtered for strike >= 105, Price = 75.9592)	3.5780

Figure 11: HestonCF - Mean absolute error by strike

Figure 11. Calibrated Heston model achieves lower errors for at-the-money and out-of-the-money, and similar for in-the-money strikes compared to the standard BS model. However, the errors are still higher than those obtained with WMC-BS across strike.

Metric Surface	Mean Eror (abs)
Mean (Filtered for price < 81.70$, 20% Quantile)	5.1794
Mean (Filtered for price > 81.70$, 20% Quantile)	6.7500
Mean (Filtered for price > 214.65$, 50% Quantile)	8.2757
Mean (Filtered for price > 370.98$, 80% Quantile)	9.3834
Mean	6.4459
Median	5.1133

Figure 12: HestonCF - Mean absolute error by surface

Figure 12. shows that the Heston model keeps mean absolute errors low across most of the surface, with a mean error of 6.45 and a median of 5.11. However, while this is a clear improvement over standard Black-Scholes, it still does not reach the lower error levels achieved by WMC-BS.

Compared to the Black-Scholes model, the Heston model brings a clear improvement in pricing accuracy, mainly by capturing the volatility smile and skew, which are absent in BSM. However, while Heston delivers similar performance to WMC-BS for at-the-money and most out-of-the-money options, it still falls short of the exact fit achieved by WMC-BS. The WMC-BS, by construction, enforces exact fit to market prices through optimal re-weighting of simulated paths, even if it means to distort the probability measure. In contrast, even though Heston offers a more robust model dynamic, the model is limited and cannot perfectly match all observed prices.

This motivates the application of Weighted Monte Carlo method to the Heston model (WMC-Heston). WMC-Heston is the combination of a robust model dynamics with

stochastic volatility and the market-fitting power of WMC, aiming to achieve the best pricing accuracy while minimising the KL divergence between the re-weighted and prior probability measure. But first, we compare the implied volatility surface generated by the Heston model to that observed in the market.

5.3 Implied Volatility - Heston

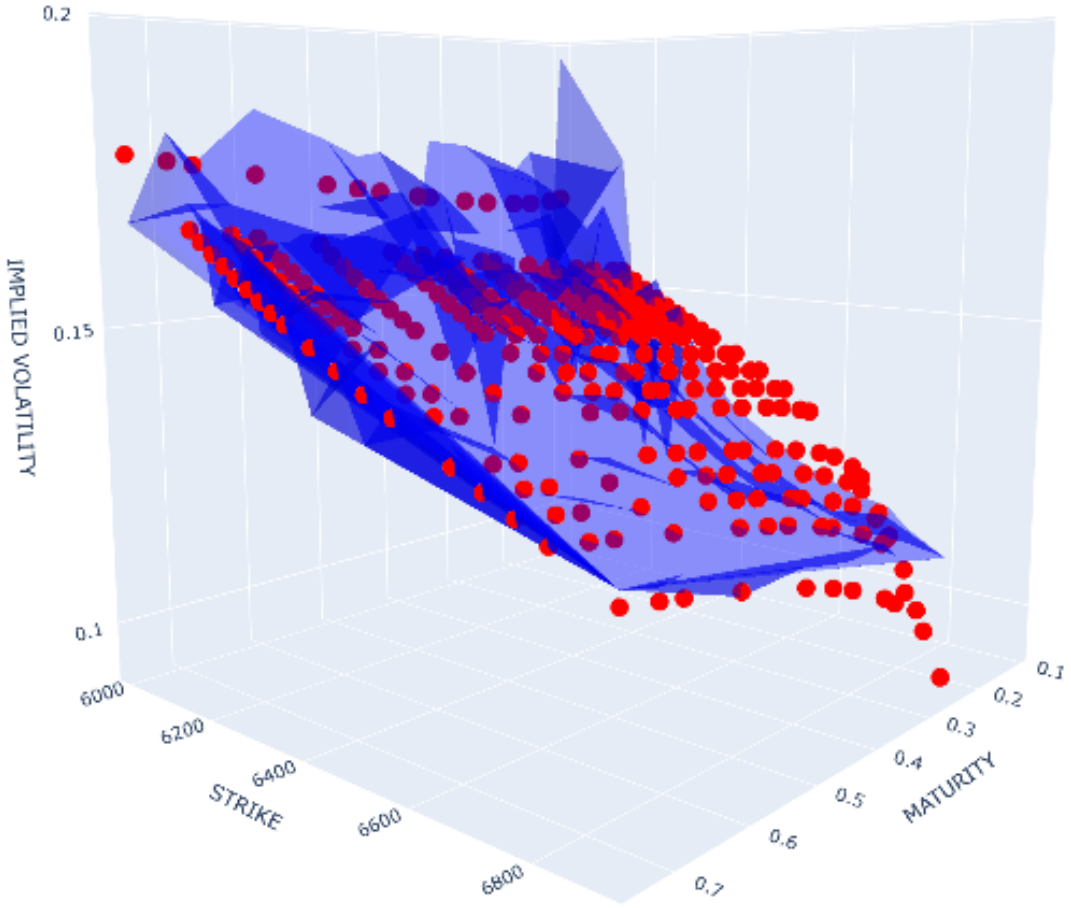

Figure 13: HestonCF - Comparison between Heston IV (Markers) and Market IV (Mesh)

Figure 13. illustrates the strengths and limits of the Heston model in replicating the market implied volatility surface. The model provides a good fit for at-the-money options and across most of the maturity-strike grid, especially compared to Black-Scholes. However, significant discrepancies remain at very short maturities and for deep out-of-the-money strikes, where the market IV surface exhibits abrupt changes or local features, such as skews, smile that the Heston model cannot capture. These limitations are due to the smoothness and parametric nature of the Heston surface. With a fixed set of parameters, the model cannot reproduce all the complexities and irregularities present in real market data. Nevertheless, this structural constraint is also an advantage against over-fitting, ensuring a robust and interpretable model. The visual gap observed between model and market IV in certain regions motivates the introduction of WMC-Heston. This approach combines the flexibility of the Weighted Monte Carlo approach with the stochastic volatility dynamics of Heston, it becomes possible to achieve a closer fit to the entire market surface while maintaining a sound probabilistic framework.

6 Weighted Monte Carlo under Heston

The goal here is to push pricing accuracy even further, but without twisting the underlying probability distribution too much. First, we focus at the simplest scenario, a single maturity. In this setup, the WMC-Heston method can perfectly match market prices for all strikes, as the number of constraints is manageable and the model's flexibility is sufficient. Then, we move to a general case, where strikes and maturities multiply.

6.1 WMC-Heston model for one maturity

Figures 14 and 15 show that the WMC-Heston approach can achieve an almost perfect fit for a single maturity. By optimally adjusting the path weights, the model is able to match observed market prices across all strikes, with negligible pricing errors. However, it is important to remember. This "perfect fit" is possible because the system is still relatively simple. The number of conditions (strikes and prices) is limited, so the optimization stays stable. As we will see in the next section, when the calibration involves many more constraints. There no longer exists a unique solution that fits every price exactly. The "perfect fit" property does not automatically exist to higher dimensions.

WMC_Heston	Market Price	Error (abs)
636.2256	636.2	0.0256
488.9051	488.88	0.0251
295.6415	295.62	0.0215
206.7228	206.71	0.0128
129.174	129.17	0.004
6490.4813	6490.4547	0.0266

Figure 14: WMC-Heston - Absolute error by strike for one maturity

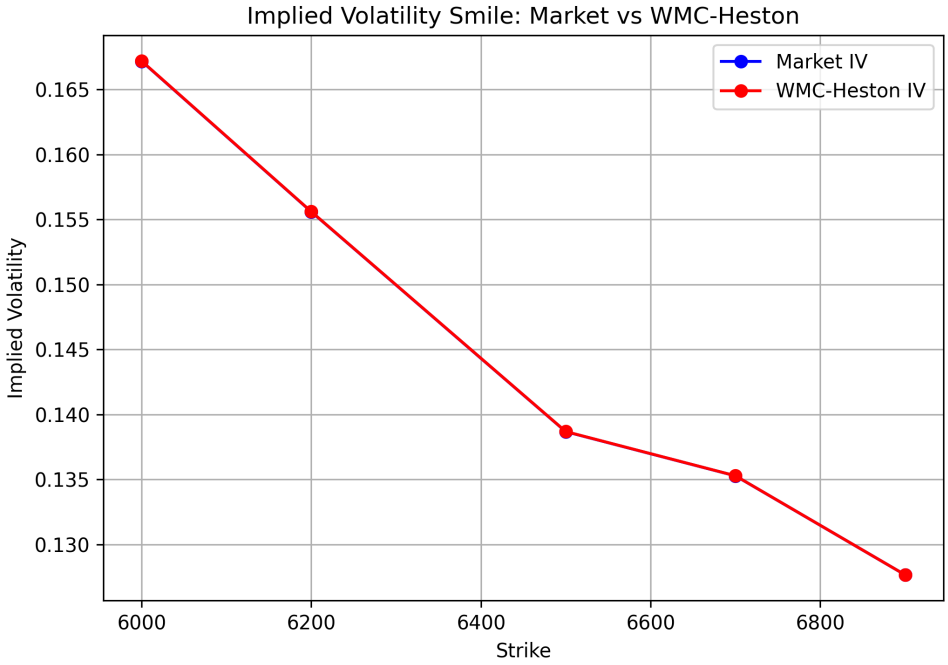

Figure 15: WMC-Heston - WMC-Heston IV against Market IV for one maturity

6.2 WMC-Heston model for market surface

We simulated 50,000 paths of the Heston stochastic volatility model, as defined in Eq. (5.1). We use the same simulation and optimization setup as described previously for the WMC-BS case, now applied to the Heston model. WMC-Heston calibration converges in about 100 iterations and takes roughly 3 seconds on a standard PC. We then use the calibrated WMC to price the full dataset (361 options).

Metric Maturity	Mean Eror (abs)
Mean (Filtered for maturity < 0.3, Price = 143.4487)	3.3016
Mean (Filtered for maturity >= 0.3 and < 0.6, Price = 255.5377)	4.8428
Mean (Filtered for maturity >= 0.6, Price = 358.4788)	7.3273

Figure 16: WMC-Heston - Mean absolute error by maturity

Figure 16. shows that WMC-Heston achieves low mean absolute errors across all maturities, with 3.30 for short maturities and 7.33 for the longest. Compared to the WMC-BS model, WMC-Heston performs slightly better for short maturities.

Metric Strike	Mean Eror (abs)
Mean (Filtered for strike < 100, Price = 357.6859)	8.2710
Mean (Filtered for strike >= 100 and < 105, Price = 212.3817)	3.4760
Mean (Filtered for strike >= 105, Price = 75.9592)	1.4785

Figure 17: WMC-Heston - Mean absolute error by strike

Figure 17. WMC-Heston delivers low mean absolute errors across strike. The error is 8.27 for strikes below 100, 3.48 for strikes between 100 and 105, and just 1.48 for strikes above 105. Compared to WMC-BS, WMC-Heston achieves similar or even better accuracy for deep out-of-the-money options.

Metric Surface	Mean Eror (abs)
Mean (Filtered for price < 81.70$, 20% Quantile)	0.8335
Mean (Filtered for price > 81.70$, 20% Quantile)	5.7189
Mean (Filtered for price > 214.65$, 50% Quantile)	7.7176
Mean (Filtered for price > 370.98$, 80% Quantile)	8.9573
Mean	4.7169
Median	2.0224

Figure 18: WMC-Heston - Mean absolute error by surface

Figure 18. shows that WMC-Heston achieves the lowest mean and median absolute errors across the entire volatility surface, with a mean of 4.72 and a median of 2.02. These results are better than those obtained with both WMC-BS and standard Heston, confirming that WMC-Heston provides the most accurate fit to observed market data across all strikes and maturities in this study.

Figure 19. shows that the path weight distribution in WMC-Heston is much more balanced than in WMC-BS, where a few paths carried disproportionately large weights. With WMC-Heston, the re-weighting is more moderate, as confirmed by the higher effective number of paths about 44869, or nearly 90% of the total sample. The lower KL divergence of 0.108 also highlights that the calibration requires less distortion of the original probability measure compared to WMC-BS. Figure 20. visually confirms this stability, showing a smoother distribution of path weights. This demonstrates that combining WMC with Heston allows for a close market fit while preserving much of the diversity of the original simulation.

Overall, WMC-Heston approach succeeds in closely matching market prices, while still preserving realistic dynamics and avoiding over-fitting. By minimizing the KL divergence from the original Heston simulation, it achieves high pricing accuracy with a stable and interpretable probability measure.

Statistic	Value
Mean	2e-05
Standard Error	4.38437e-08
Median	1.91026e-05
Mode	8.19925e-07
Standard Deviation	9.80374e-06
Sample Variance	9.61133e-11
Kurtosis	11.7994
Skewness	1.87144
Range	0.000161094
Minimum	8.19925e-07
Maximum	0.000161914

Figure 19: WMC-Heston - Descriptive statistics for calibrated weights $(p_1, \ldots, p_{50000})$

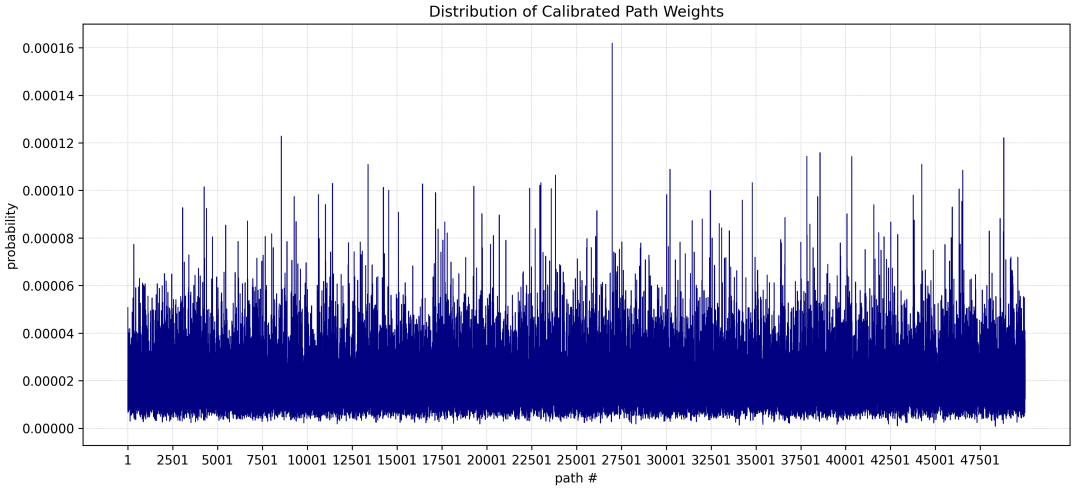

Figure 20: WMC-Heston - Distribution of calibrated Path Weights

6.3 Implied Volatility - WMC Heston

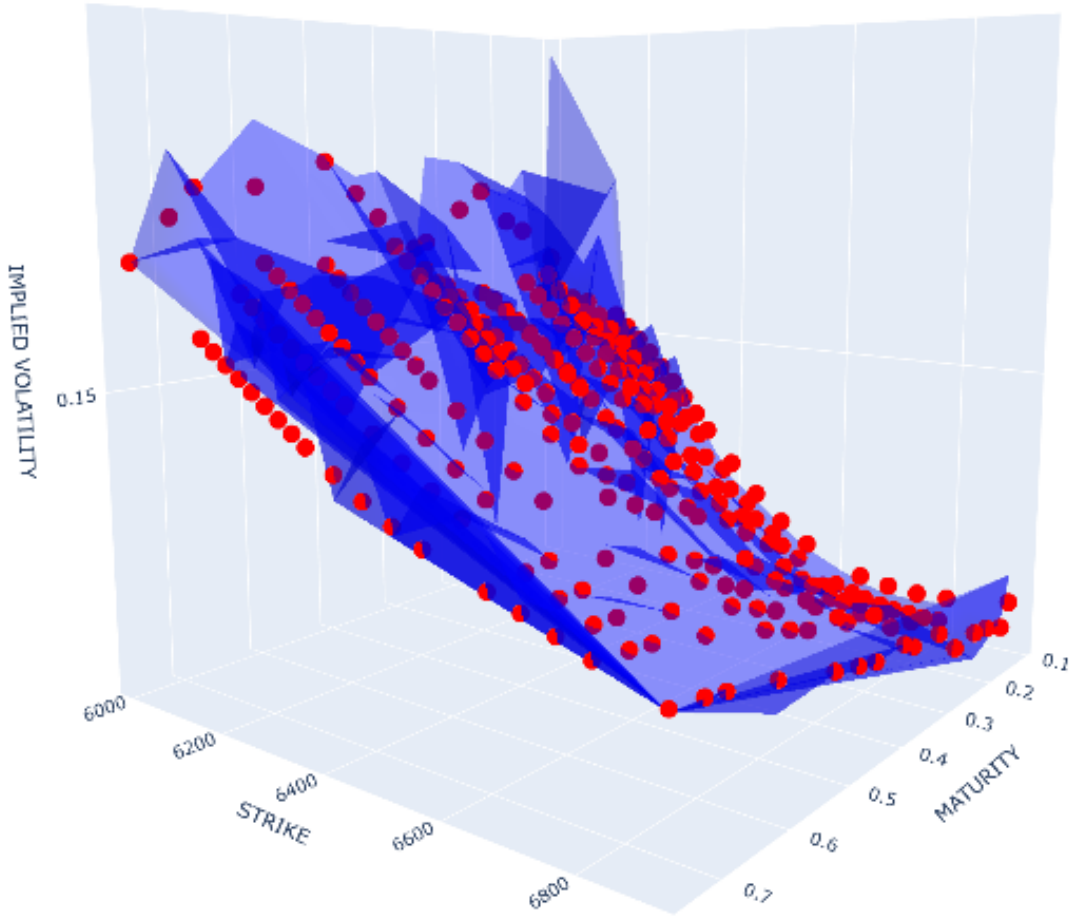

Figure 21: WMC-Heston - Comparison between WMC-Heston IV (Markers) and Market IV (Mesh)

Figure 21. gives a picture of the implied volatility surface produced by WMC-Heston. It shows how WMC-Heston matches the market implied volatility surface almost point by point across all strikes and maturities. The red markers represent market-implied volatilities, while the blue mesh shows the model-implied values after WMC calibration. Unlike the smooth and regular surface produced by the closed-form Heston model, WMC-Heston can capture both the broad features and the local distortions seen in real data. This flexibility allows us to fit even abrupt changes or irregularities in the observed market.

WMC-Heston, thanks to its flexible calibration, it can fit a much larger set of market prices. In Figure 21., you can see the blue mesh hugging the bumps and twists of the market IV (the red markers), all while keeping the overall structure consistent with Heston's stochastic volatility. Yet, this flexibility also comes with new challenges. The number of constraints increases, the calibration may face issues of stability or over-fitting, and the solution is not always unique. Still, the improvement in local accuracy is obvious. Compared to closed-form Heston, the WMC-Heston model is better at capturing both the overall shape of the surface and the smaller variations and irregularities seen in market (*Avellaneda et al., 2001; Gudmundsson & Vyncke, 2021*).

Looking at both approaches side by side shows the progress made. The trade-off that come with switching from parametric models to weighted Monte Carlo methods in the Heston framework (*Avellaneda et al., 2001*).

7 Conclusion

Model	Type	Mean Error	Median Error	Speed
Black-Scholes	Closed-form	18.174	17.703	**Very fast**
Heston	Semi-closed	6.446	5.113	Slow
WMC-BS	Simulation	5.31	2.779	Fast
WMC-Heston	Simulation	**4.717**	**2.022**	Fast

While weighted Monte Carlo (WMC) is effective at matching market prices when calibrating to a single maturity, things quickly get more complicated as the number of constraints grows (*Avellaneda et al., 2001*). With just a few strikes or maturities, you have enough flexibility to achieve a perfect calibration, mean and median errors drop to nearly zero, highlighting the strength of the entropy-minimizing approach. However, as soon as the number of constraints grows, the calibration problem becomes much more challenging. This is precisely what I encountered, while case 1 yields almost perfect matches, case 2 inevitably leads to small but non-zero errors (*Avellaneda et al., 2001; Gudmundsson & Vyncke, 2021*).

This limitation is not unique to my implementation, it's a fundamental property of the entropy minimisation calibration approach as discussed in the literature like Avellaneda et al. When the market surface becomes too complex or has too many constraints, it is hard to achieve a perfect fit. There's always a trade-off between matching market prices exactly and keeping the solution robust and stable. That has been said, this challenge interesting directions for future work. For example, while I already use regularisation and penalty terms to keep the calibration stable and avoid over-fitting, it would be interesting to explore new types of regularisation, or even look for closed-form formulas that could match market prices exactly in some cases. Also, machine learning is another promising methods like neural networks or generative models could make it easier to fit the full complexity or real implied volatility surfaces, with more flexibility than standard approaches.

References

[1] Avellaneda, M., Buff, R., Friedman, C., Grandchamp, N., Kruk, L., & Newman, J. (2001). *Weighted Monte Carlo: A New Technique for Calibrating Asset-Pricing Models.* International Journal of Theoretical and Applied Finance, 4(1), 91–119.

[2] Paul Glasserman. *Monte Carlo Methods in Financial Engineering.* Springer Science+Business Media, New York, 2003. Applications of Mathematics, Vol. 53. ISBN 978-1-4419-1822-2.

[3] Heston, S. L. (1993). *A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options.* The Review of Financial Studies, 6(2), 327–343.

[4] Gudmundsson, H., & Vyncke, D. (2021). A Generalized Weighted Monte Carlo Calibration Method for Derivative Pricing. *Mathematics*, 9(7), 739.

[5] Caleb Migosi. *Build the Heston Model from Scratch in Python (Part II).* Medium, 2022. `https://calebmigosi.medium.com/build-the-heston-model-from-scratch-in-python-part-ii-5971b9971cbe` (accessed June 15, 2025).

[6] Black, F., & Scholes, M. (1973). *The Pricing of Options and Corporate Liabilities.* Journal of Political Economy, 81(3), 637–654.

[7] Merton, R. C. (1973). *Theory of Rational Option Pricing.* The Bell Journal of Economics and Management Science, 4(1), 141–183.

[8] TheQuantPy. *Heston Model Calibration in the "Real" World with Python – S&P500 Index Options.* GitHub, 2022. `https://github.com/TheQuantPy/youtube-tutorials/blob/main/2022/001%20Jan-Mar/2022-03-25%20Heston%20Model%20Calibration%20in%20the%20_Real_%20World%20with%20Python%20-%20S_P500%20Index%20Options.ipynb` (accessed June 20, 2025)

Appendix

A Main Code Blocks

A.1 Weighted Monte Carlo (entropy minimisation)

```
class WMC(object):
    '''WMV(g,C)

        g is a N x V matrix of derivatives payouts
        C is a 1 x N matrix of derivative prices

        N = number of instruments
        V = number of paths

        Once initiated call .solve to calculate optimal lambdas
        Resulting probabilities are stored in variable p
    '''

    def __init__(self, g, C):
        self.g = np.array(g)   # N_paths x N_options
        self.C = np.array(C)   # N_options
        self.v = self.g.shape[0]
        self.last = None

    def recalc(self, lambda_):
        if self.last != tuple(lambda_.tolist()):
            lambda_ = np.matrix(lambda_)
            self.egl = np.exp(self.g * lambda_.T)
            self.Z = np.sum(self.egl)
            self.p = (self.egl / self.Z).T
            self.w = float(np.log(self.Z) - self.C * lambda_.T + self.e/2 * (lambda_*lambda_.T))
            self.fPrime = np.array((self.p * self.g - self.C + self.e * lambda_))[0]
            self.last = tuple(lambda_.tolist())

    def _objective(self, lambda_):
        self.recalc(lambda_)
        return self.w

    def _fPrime(self, lambda_):
        self.recalc(lambda_)
        return self.fPrime

    def solve(self, fPrime=True, e=1e-5, method="L-BFGS-B", bounds=None, options=None):
        ''' fPrime = True/False (use gradiant)
            e      = "weight" in the least squares implementation
            disp   = Print optimization convergence
        '''
        self.e=e
        x0 = np.zeros_like(self.C)

        if fPrime:
            result = minimize(self._objective, x0, jac=self._fPrime, method=method, bounds=bounds, options=options)
        else:
            result = minimize(self._objective, x0, method=method, bounds=bounds, options=options)

        self.opt_result = result
        self.lambda_ = np.array(result.x)  # Store lambdas

        return self.p
```

A.2 Black-Scholes and Monte Carlo Pricing

```python
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)
    call_price = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
    return call_price

def monte_carlo_price(S0, K, T, r, sigma, N):
    Z = np.random.randn(N)
    ST = S0 * np.exp((r - 0.5 * sigma**2) * T + sigma * np.sqrt(T) * Z)
    payoff = np.maximum(ST - K, 0)
    return np.exp(-r * T) * np.mean(payoff)
```

A.3 Loss Function and BSM Calibration

```python
def loss_function(x, lam=0.1):
    losses = []
    sigma = x

    for maturity, strike, price in zip(T, strikes, market_prices):
        mc_prices = black_scholes_call(S0, strike, maturity, r, sigma)
        losses.append(np.sum((mc_prices - price)**2) + lam * sigma**2)  #lam to penalize extreme value of sigma

    return sum(losses)

result = minimize(
    loss_function,
    x0=np.array([0.2]),
    method='L-BFGS-B',
    bounds=[(1e-2, 0.5)],
    options={'maxiter': 1e6, 'ftol': 1e-6})

result
```

A.4 Monte Carlo Path Generation

```python
def monte_carlo_path(S0, r, sigma, N_paths, N_steps):
    Z = np.random.randn(N_paths, N_steps+1)
    paths = (r - 0.5 * sigma**2) * dt + sigma * np.sqrt(dt) * Z
    return np.array(S0*np.exp(np.cumsum(paths,axis=1)))
```

A.5 Heston Model Simulation

```python
def simulate_heston(S0, V0, r, kappa, theta, sigma_v, rho, N, N_steps):
    dt = 1/252 # Amplitude of step
    epsilon = 1e-10
    S = np.zeros((N, N_steps + 1))
    V = np.zeros((N, N_steps + 1))
    S[:, 0] = S0
    V[:, 0] = V0
    for t in range(1, N_steps + 1):
        Z1 = np.random.randn(N)
        Z2 = rho * Z1 + np.sqrt(1 - rho**2) * np.random.randn(N)
        V_prev = np.maximum(V[:, t - 1], 0)
        V[:, t] = V_prev + kappa * (theta - V_prev) * dt + sigma_v * np.sqrt(V_prev + epsilon) * np.sqrt(dt) * Z2
        V[:, t] = np.maximum(V[:, t], 0)
        S[:, t] = S[:, t - 1] * np.exp((r - 0.5 * V_prev) * dt + np.sqrt(V_prev + epsilon) * np.sqrt(dt) * Z1)
    return S
```

A.6 Weighted Monte Carlo Optimisation

```
w = WMC(g_normalize, C_normalize)
w.solve(options={'maxiter': 1e6, 'ftol': 1e-12}, e=1e-5, fPrime=True)
w.opt_result
```

A.7 Heston Characteristic Function

```
def heston_charfunc(phi, S0, v0, kappa, theta, sigma, rho, lambd, tau, r):

    a = kappa*theta
    b = kappa+lambd

    rspi = rho*sigma*phi*1j
    d = np.sqrt( (rho*sigma*phi*1j - b)**2 + (phi*1j+phi**2)*sigma**2 )

    g = (b-rspi+d)/(b-rspi-d)
    exp1 = np.exp(r*phi*1j*tau)
    term2 = S0**(phi*1j) * ( (1-g*np.exp(d*tau))/(1-g) )**(-2*a/sigma**2)

    max_exponent = 700
    exp_value = a * tau * (b - rspi + d) / sigma**2 + v0 * (b - rspi + d) * ((1 - np.exp(d * tau)) / (1 - g * np.exp(d * tau))) / sigma**2
    exp_value = np.clip(exp_value, -max_exponent, max_exponent)
    exp2 = np.exp(exp_value)

    return exp1*term2*exp2
```

A.8 Heston Pricing Functions

```
def integrand(phi, S0, K, v0, kappa, theta, sigma, rho, lambd, tau, r):

    args = (S0, v0, kappa, theta, sigma, rho, lambd, tau, r)
    numerator = np.exp(r*tau)*heston_charfunc(phi-1j,*args) - K*heston_charfunc(phi,*args)
    denominator = 1j*phi*K**(1j*phi)

    return numerator/denominator

def heston_price_rec(S0, K, v0, kappa, theta, sigma, rho, lambd, tau, r):

    args = (S0, v0, kappa, theta, sigma, rho, lambd, tau, r)
    P, umax, N = 0, 100, 10000
    dphi=umax/N

    for i in range(1,N):
        phi = dphi * (2*i + 1)/2
        numerator = np.exp(r*tau)*heston_charfunc(phi-1j,*args) - K * heston_charfunc(phi,*args)
        denominator = 1j*phi*K**(1j*phi)
        P += dphi * numerator/denominator

    return np.real((S0 - K*np.exp(-r*tau))/2 + P/np.pi)

def heston_price(S0, K, v0, kappa, theta, sigma, rho, lambd, tau, r):

    args = (S0, K, v0, kappa, theta, sigma, rho, lambd, tau, r)
    real_integral, err = np.real(quad(integrand, 0, 100, args=args))

    return (S0 - K*np.exp(-r*tau))/2 + real_integral/np.pi
```

A.9 Heston Calibration with L-BFGS-B

```
params = {
    "v0": {"x0": 0.021, "lbub": [1e-4, 0.5]},
    "kappa": {"x0": 1.5, "lbub": [1e-3, 5]},
    "theta": {"x0": 0.02, "lbub": [1e-4, 0.5]},
    "sigma": {"x0": 0.3, "lbub": [1e-2, 1]},
    "rho": {"x0": -0.5, "lbub": [-1, 1]},
    "lambd": {"x0": -0.5, "lbub": [-1, 1]},
    }

def SqErr(x):
    v0, kappa, theta, sigma, rho, lambd = x
    err = np.sum((market_price - heston_price_rec(S0, K, v0, kappa, theta, sigma, rho, lambd, maturity, r))**2 / len(market_price))
    return err + 0


x0 = [param["x0"] for key, param in params.items()]
bnds = [param["lbub"] for key, param in params.items()]

start = time.time()
result = minimize(SqErr, x0, method='L-BFGS-B', options={'maxiter': 1e6, 'ftol': 1e-5, 'disp':True}, bounds=bnds)
end = time.time()

total_time = end - start

result
```

A.10 Black-Scholes Implied Volatility (Newton Method)

```
def bs_price_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 * norm.cdf(d1) - K * np.exp(-r*T) * norm.cdf(d2)

def bs_vega(S, K, T, r, sigma):
    d1 = (np.log(S/K) + (r + 0.5*sigma**2)*T) / (sigma * np.sqrt(T))
    return S * norm.pdf(d1) * np.sqrt(T)

def implied_vol_newton(price, S, K, T, r, tol=1e-8, max_iterations=100):
    sigma = 0.2 # Initial guess
    for i in range(max_iterations):
        price_bs = bs_price_call(S, K, T, r, sigma)
        vega = bs_vega(S, K, T, r, sigma)
        price_diff = price_bs - price
        if abs(price_diff) < tol:
            return sigma
        sigma -= price_diff / vega
        sigma = max(sigma, 1e-8) # Security to avoid 0
    return sigma #If no convergence, return the last value
```

B Implementation Details and Data Access

B.1 Imports

Listing 1: Imports

```
import numpy as np

from scipy.optimize import minimize
from scipy.stats import kurtosis, skew, mode, norm
from scipy.integrate import quad

np.random.seed(42)
```

B.2 Data Sources

All market data used in this study, including underlying asset prices and vanilla option quotes, were obtained from **Yahoo Finance**. The dataset corresponds to S&P 500 index European call options, observed on 18 July 2025, across a range of maturities and strikes.

Data extraction was performed via Python's *yfinance* library. Underlying, `^GSPC` (S&P 500 index spot prices). Risk-free rate, U.S. Treasury yields of matching maturities, as available via the same source or assumed constant when missing. Option quotes, midpoint between bid and ask when both available, last traded price otherwise.

B.3 Programming Environment

The numerical implementation was carried out in **Python 3.11.9**, running under `Visual Basic Code` on a **Windows 11** system with an `AMD Ryzen 7 5800X3D` and 32 GB RAM.

B.4 Computational Performance

All calibrations and simulations were executed on the above hardware without parallelisation. The computational effort for each main task was as follows. Monte Carlo simulation $\approx$ 2–4 seconds. Weighted Monte Carlo calibration (BSM model) $\approx$ 3–5 seconds. Heston closed-form calibration $\approx$ 240 seconds. Weighted Monte Carlo under Heston $\approx$ 4–6 seconds.