



Is Cross-Index Generalization Achievable via Neural Calibration of Parametric Option Pricing Models?

Cranfield School of Management

Msc Finance

Academic Year: 2024 - 2025

September 2025

Contents

Abstract	iii
1 Introduction	iv
2 Literature Review	v
2.1 Introduction to Option	v
2.2 Implied Volatility and Reconsideration of log normal distribution of returns	v
2.3 Calibration: A two Step Process	vi
2.4 Back Propagation: Adjusting inputs parameters to match outputs	vii
2.5 Solving models equations	ix
2.6 Two Principal Approaches to Model Calibration Using Neural Networks . .	x
2.7 Calibration Generalization	xi
3 Option Pricing Models and Global Optimizer	xii
3.1 Option Pricing	xii
3.2 Geometric Brownian Motion	xii
3.3 Heston Model (1993)	xiv
3.4 Bates Model (1991)	xv
3.5 Global Optimizer : Differential Evolution Algorithm	xvii
4 Method	xix
4.1 Two Step Approach	xix
4.1.1 The Forward Pass: Learning the Pricing Function	xx
4.1.2 The Backward Pass: Calibration of Model Parameters	xxii
4.2 Forward Pass	xxiii
4.2.1 Dataset Construction	xxiii
4.2.2 Generation of Model parameters and Option Properties	xxiv
4.2.3 Monte Carlo setting	xxvi
4.2.4 Neural Network Training	xxvii
4.3 Backward Pass	xxix
4.3.1 Index Data	xxix
4.3.2 Backward pass on Index Data and Error Prediction	xxx
5 Results	xxxvi
5.1 Same Index Calibration Performance	xxxvi
5.1.1 Predicting S&P using S&P	xxxvi
5.1.2 Predicting Nasdaq using Nasdaq	xxxviii
5.2 Cross Index Calibration Performance	xl
5.2.1 Predicting Nasdaq using S&P	xli
5.2.2 Predicting S&P using Nasdaq	xliii
5.3 Discussions	xlvi
6 Conclusion	xlviii
Appendix	xl ix
Ethical approval letter	xl ix
Bibliography	xl ix

List of Figures

3.1	Brownian Motion Example ($r=0.15$, $\sigma=0.20$, $T=1$, $S_0=50$)	xiii
3.2	Price Path using Heston Model	xv
3.3	Volatility Path using Heston Model	xv
3.4	Price Path using Bates Model	xvii
3.5	Volatility Path using Bates Model	xvii
4.1	Two Step Approach	xix
4.2	Forward Pass	xxii
4.3	Backward Pass	xxiii
4.4	Average relative error to BS prediction ($N=50$)	xxvii
4.5	Heston Training Loss Evolution ($e=10$, MSE)	xxviii
4.6	Bates Training Loss Evolution ($e=10$, MSE))	xxviii
4.7	Invesco QQQ Trust Series 1 performance in comparison with NASDAQ-100 Index)	xxix
4.8	SPDR S&P 500 ETF Trust (SPY) performance in comparison with S&P 500 Index (SPX)	xxx
5.1	Candlestick Plot of Losses by Time Horizon using Bates model (SPY→SPY)xxxvi	
5.2	IQR Curve by Time Horizon for Bates model (SPY→SPY)	xxxvii
5.3	Candlestick Plot of Losses by Time Horizon using Heston model (SPY→SPY)xxxvii	
5.4	IQR Curve by Time Horizon for Heston model (SPY→SPY)	xxxviii
5.5	Candlestick Plot of Losses by Time Horizon using Bates model (QQQ→QQQ)xxxviii	
5.6	IQR Curve by Time Horizon for Bates model (QQQ→QQQ)	xxxix
5.7	Candlestick Plot of Losses by Time Horizon using Heston model (QQQ→QQQ)	xxxix
5.8	IQR Curve by Time Horizon for Heston model (QQQ→QQQ)	xl
5.9	Candlestick Plot of Losses by Time Horizon using Bates model (SPY→QQQ) xli	
5.10	IQR Curve by Time Horizon for Bates model (SPY→QQQ)	xli
5.11	Candlestick Plot of Losses by Time Horizon using Heston model (SPY→QQQ) xlii	
5.12	IQR Curve by Time Horizon for Heston model (SPY→QQQ)	xlii
5.13	Candlestick Plot of Losses by Time Horizon using Bates model (QQQ→SPY)xliv	
5.14	IQR Curve by Time Horizon for Bates model (QQQ→SPY)	xliv
5.15	Candlestick Plot of Losses by Time Horizon using Heston model (QQQ→SPY) xlv	
5.16	IQR Curve by Time Horizon for Heston model (QQQ→SPY)	xlvi

List of Tables

4.1	Sampling range for the Heston model parameters	xxiv
4.2	Sampling range for the Bates model parameters	xxv
4.3	Descriptive Statistics for the generated dataset (count=160000, Bates model)	xxvi
4.4	Descriptive Statistics for the generated dataset (count=160000, Heston model)	xxvi
4.5	Neural network training configuration	xxvii
4.6	Performance metrics of the trained neural network (Heston)	xxvii
4.7	Performance metrics of the trained neural network (Bates)	xxviii
4.8	Performance as of June 30, 2025 of Invesco QQQ Trust Series 1 (in %)	xxix
4.9	Total Return (As of 06/30/2025)	xxx
4.10	RMSE dataset architecture matrix Ω_k^* across forecast windows W_k	xxxi
4.11	Descriptive Statistics of SPY dataset	xxxii
4.12	Descriptive Statistics of QQQ dataset	xxxii
4.13	Differential Evolution algorithm settings	xxxii
4.14	Architecture of the calibrated parameters dataset	xxxiii
4.15	Average relative errors over 30 weeks of calibration	xxxiii
4.16	Prediction Error Dataset Architecture for one model	xxxv
5.1	Comparison of Heston and Bates models for predicting QQQ using parameters calibrated on SPY (<i>Mann-Whitney test</i>)	xliii
5.2	Comparison of Heston and Bates models for predicting QQQ using parameters calibrated on SPY (<i>t-test</i>)	xliii
5.3	Comparison of Heston and Bates models for predicting SPY using parameters calibrated on QQQ (<i>Mann-Whitney test</i>)	xlvi
5.4	Comparison of Heston and Bates models for predicting SPY using parameters calibrated on QQQ (<i>t-test</i>)	xlvi

Academic Integrity Declaration

I declare that:

- the thesis submitted has been written by me alone.
- the thesis submitted has not been previously submitted to this university or any other.
- that all content, including primary and/or secondary data, is true to the best of my knowledge.
- that all quotations and references have been duly acknowledged according to the requirements of academic research.

I understand that to knowingly submit work in violation of the above statement will be considered by examiners as academic misconduct.

Abstract

Calibration of option pricing models has been widely studied, with research focusing both on developing accurate objective function optimizers and on methods for solving option pricing models efficiently. A large part of the calibration process consists not only of the parameters optimization but also of repeated option price calculations. To accelerate this process, I replace traditional pricing methods with a pre-trained neural network, aiming to reduce computation time while maintaining a high level of accuracy. The network is trained to approximate the pricing functions of the Bates (1991) and Heston (1993) models. I then investigate same index calibration performance and then study whether parameters calibrated on one equity index can be transferred to another (“cross-index generalization”) and still produce efficient predictions.

I consider two U.S. equity indices, the S&P 500 and the Nasdaq 100 and follow a two-step process: the Forward Pass where the neural network is trained on synthetic data, which enables coverage of a wide range of parameters, enhancing generalization capacity of the neural network ; and the Backward Pass which combine the trained network with a Differential Evolution algorithm to calibrate model parameters on real market data.

After calibrating each model on its respective index, results demonstrate good performance, especially for the first 10 to 15 weeks, with an expected growing predictive errors with time horizon. To extend the study, I analyze the distribution of prediction errors across different horizons ahead of the calibration week in order to detect model instabilities. The findings reveal that models are particularly unstable when predicting the Nasdaq option chain with parameters calibrated on itself for horizons between approximately the 15th and 25th weeks. Similarly, I detect instabilities when predicting the S&P 500 option chain with parameters calibrated on the same index between the 12th and 26th weeks. Following this study, I test cross-index predictions by using parameters calibrated on one index to predict the option chain of the other one. Results indicate no significant difference between cross-index and same-index predictions errors when S&P 500-calibrated parameters are used to predict Nasdaq options over horizons 13 to 27 weeks after calibration. However, this potential transferability contradicts the evidence of instability observed for same-index predictions within the same period, casting doubt on the existence of true cross-index generalization. Conversely, when Nasdaq-calibrated parameters are used to predict S&P 500 options over horizons 45 to 50 weeks, no significant difference is observed. However, same-index performance with these time horizons did not show substantial instability, leaving the possibility of cross-index generalization plausible under specific conditions.

Although, in some cases cross index test results did indicate significant difference between same index and cross index predictions errors for specific time horizons, cross index predictions errors remain relatively small in the short term horizons (especially when calibrating on the S&P 500 to predict the Nasdaq), therefore making the cross index calibration valid and useful depending on the level of accuracy required.

1. Introduction

Option pricing models play a key role in modern finance, providing the foundation for derivative valuation, risk management, and trading strategies. Since the work of Black and Scholes (1973), several extensions have been proposed to address empirical limitations, such as the volatility smile and skew, by incorporating stochastic volatility Heston (1993), jumps Bates (1991), and other complex market dynamics (e.g. Bates (1996a, c), Scott (1997)). While these advanced models improve accuracy, they also require frequent calibration to reflect market conditions. However, calibration is costly in both computation time and complexity, as it involves both repeated model predictions and parameter optimization.

Recent advances employing machine learning techniques have introduced neural networks as a potential solution for accelerating calibration. Hernandez (2017) has been a pioneer in this domain by proposing a Neural Network that is able to map model input parameters to implied volatility. By learning the pricing function, a neural network can replace traditional numerical methods, allowing faster and potentially more accurate calibration. However, he finds that this method results in a lack of control over the mapping function, between implied volatility surfaces and the model. Therefore his work has been extensively investigated by Bayer et al. (2019) and Horvath et al. (2019) who significantly enhanced the method by developing a two-step process rather than a single step one. Nonetheless, existing literature has focused on evaluating such methods on synthetic or single-market data, leaving the question of accuracy on real market data unanswered and especially the feasibility of a cross-market generalization unexplored.

This thesis investigate the calibration performance on both considered index and test whether parameters calibrated on one financial index can be effectively applied to another, a concept referred here as cross-index generalization. Grzelak et al. (2022) propose a multivariate local volatility model that enables the simultaneous calibration of option prices on both individual stocks and the corresponding index. They demonstrate good results emphasizing on the synergies between the assets to collaborate in enhancing the pricing accuracy. From this works born the idea of cross-index generalization. Therefore, using the Bates (1991) and Heston (1993) models, I implement a two-step calibration framework, closely following the contributions of Liu et al. (2019a) and Caleiras (2024). The Forward Pass trains a neural network on synthetic option data to approximate the model's pricing function, while the Backward Pass uses the trained network combined with a Differential Evolution optimizer to calibrate parameters to real market option chains.

In summary, this thesis applies the proposed two-step calibration framework to two major U.S. indices, the S&P 500 and the Nasdaq 100, over a two-year period. The results demonstrate that while same-index calibrations achieve good performance overall, the distribution of prediction errors reveals localized instabilities, particularly for mid-term horizons. These instabilities impact the interpretation of cross-index transferability, as the potential success of predicting Nasdaq options using S&P 500-calibrated parameters between the 13th and 27th weeks could be explained by unstable same-index performance for the same time horizons. Conversely, transferring Nasdaq-calibrated parameters to the S&P 500 shows plausibility in the 45th–50th week range without such instability, suggesting that cross-index generalization may hold under specific conditions. Together, these findings provide both evidence of the potential and limitations of cross-index generalization, leaving a nuanced discussion in the remainder of this thesis.

2. Literature Review

2.1 Introduction to Option

An option is a financial contract that gives the right to sell or buy a specified quantity of an asset at a specific time and at a pre-determined price. We denote two types of options, “American” type and “European” type. The fundamental difference between these two types of option is that “American” option can be exercised at any time before maturity whereas an “European” can be exercised only at maturity. Pricing option has been subject to many years of research and the most significant steps has been made by Black & Scholes (1973), where they proposed an analytical formula to price European options. To derive the analytic formula, they have make several assumptions listed below:

- short sales are allowed
- there are no transactions costs
- the market operates continuously
- the risk-free interest rate is constant
- there is no riskless profit opportunity from trading the underlying asset and the option (no arbitrage)
- the price of the underlying asset follows a geometric Brownian motion with constant volatility
- the stock price is continuous
- the stock pays no dividends
- the option can only be exercised at the terminal date of the contract (European option)

Afterwards, other model emerged to relax some assumptions of the original Black and Scholes model, in attempt to get closer to the true nature of option market. To name but a few, Heston (1993) developed a model involving stochastic volatility and Dupire (1994) with a local volatility model. While the previous model improve modeling, other also includes jumps into their model, consequently allowing stock price to be discontinuous as presented by Merton (1976) and Bates (1991). A stochastic interest rates model has been developed by Amin and Jarrow (1992). Finally, later the stochastic-volatility jump-diffusion models of Bates (1996a, c), and Scott (1997) has been developed combining all the broad features.

2.2 Implied Volatility and Reconsideration of log normal distribution of returns

As a motivation of the emergence of new option pricing models, some assumptions of the Black & Scholes (1973) model have been considered to be unrealistic and not reflecting the true nature of option market. An example of criticism received is reflected is the so called « smile » of implied volatility. Consequently, this gave birth to the study of the implied volatility surface and its dynamic that has been at heart of much research especially by Cont & Da Fonseca (2002), justifying the use of the Greek Vega for hedging

purposes. Indeed, when engineering back Black and Scholes formula using market prices, I also find that “implied volatility” is not the same for options with different moneyness. This phenomenon indicates that volatility can’t be the same over the life of the contract, hence the idea of stochastic volatility models.

To extend the idea of implied volatility bias in Black and Scholes, the model is also criticised for its assumptions of normal distribution of returns. Indeed, the smile of implied volatility also suggests that investors believe that higher returns are more likely to happen than predicted by the model. Indeed, having higher implied volatility for far OTM (Out of the money) options indicates that such options are more likely to become exercisable than forecasted by the model, which explain a consequent increase in price for those options. As indicated by Hull & White (1998), daily changes exceed 3 standard deviations on 1.30% of days. The lognormal model predicts that this should happen on only 0.27% of days. Daily changes exceed 4, 5, and 6 standard deviations on 0.49%, 0.24%, and 0.13% of days, respectively. The lognormal model predicts that we should hardly ever observe this happening. Therefore, this provides evidence to support the existence of heavy tails and the volatility smile used by traders.

Moreover, following the 1987 stock market crash, the empirical shape of the implied volatility curve for equity index options particularly those on the S&P 500 have seen a significant transformation. Contrary to the flat volatility surface predicted by the Black-Scholes model, a pronounced skew emerged, implied volatilities became notably higher for out-of-the-money (OTM) put options. Rubinstein (1994) attributes this asymmetry to what he terms "crashophobia" a persistent fear among investors of another severe market downturn. This fear drives demand for protective puts, leading to their overpricing relative to model-based expectations. The result is a volatility "smirk" or skew that reflects the market’s demand for downside insurance, rather than an inherent inefficiency.

In conclusion, although the Black-Scholes model offers a fundamental theoretical foundation for option pricing, it frequently produces discrepancies when confronted with real market data. Specifically, it fails to account for key empirical features such as the volatility surface. As previously discussed, and thoroughly investigated by Bakshi, Cao, and Chen (1997), models that incorporate stochastic volatility and jump components provide a more realistic representation of market behaviour. More precisely, they distinguish short- and long-term options and demonstrate that adding the random-jump feature to the stochastic volatility model can explain pricing short-term options while modelling stochastic interest rates can improve the fit of long-term options. The authors finally conclude that when building a model with stochastic volatility and stochastic interest with the jumps feature, the errors is related to contract-specific or market-conditions-related biases. Consequently the holy grail has not been found yet but closely approached. Indeed, these advanced frameworks, by better capturing the dynamics of the volatility surface, offer significant improvements in replicating observed option price patterns.

2.3 Calibration: A two Step Process

Although these models improved option pricing modeling it does not come without a cost. These models require parameters that need to be calibrated on market data. This process is broadly thought as an inverse problem were based on output data we would like to find implied input parameters. However, it can also be thought as a loop process.

Indeed, calibration is an iterative task where both outputs prediction process and inputs adjustments matter. Consequently, calibration involve two phases, two types of methods. The first one is the method used to predict the price from the model and the second is the one that adjust the inputs to fit the outputs. The first phase can be really complex as models often do not provide an analytical formula as indicated by Fournié, Lasry & Touzi (1997) and consequently require external methods to be solved.

To make matters worse, speed, robustness and fitting are also at heart of this topic since calibration must fill all these fields. Indeed, financial markets are dynamic and are constantly evolving, which means that the parameters that provide the best fit today may not be the best tomorrow. Consequently, models require frequent, almost daily re-calibration, emphasizing on the importance of developing fast, accurate and robust calibration method. Calibration is an important field of research during the last decades ranging from various type of methodologies, each of them arising with their advantages and drawbacks. This task is vital and must be conducted carefully because once calibrated models are used to determine the fair price of over the counter (OTC) exotic financial contracts or for hedging and risk management purposes. At this stage, further discussion will focus on the two phases of the calibration process as described above.

2.4 Back Propagation: Adjusting inputs parameters to match outputs

Firstly, the process of adjusting model inputs to match observed option prices often results in high-dimensional optimization problems due to the large number of parameters to calibrate. Indeed, a high-dimensional problem refers to a scenario where the number of variables (parameters to be calibrated) is large, making the search space complex and computationally intensive. In the context of option pricing models, each parameter (e.g., volatility, mean reversion speed, jump intensity) adds a new dimension to the optimization problem, significantly increasing its complexity. Such problems frequently require the use of heuristic techniques, a class of solution methods that aim to find good solutions when traditional analytical or gradient-based methods are ineffective. These techniques, such as differential evolution, simulated annealing, and particle swarm optimization, rely on iterative processes and randomness to explore the solution space and are particularly useful when the problem is non-convex and contains many local minimas. Indeed, the calibration of option pricing models is widely recognized as a non-convex optimization challenge which means that the objective function, which measures the error between model-predicted prices and observed market prices, may have multiple local minima. As demonstrated by Gilli and Schumann (2012), when calibrating the Heston model, varying two parameters while keeping others fixed revealed an objective function with several local minima. However, a local minimum is a point in the parameter space where the objective function (e.g., the calibration error) is lower than at nearby points, but not necessarily the lowest overall. In contrast, a global minimum is the point where the objective function reaches its absolute lowest value across the entire parameter space. Hence, finding a local minimum does not ensure to obtain the best possible fit between the model prediction and the observed market data. Therefore, it justifies the use of heuristics techniques to find the best possible fit between market and model.

Consequently, this task necessitates the use of heuristics techniques such as global optimization algorithms, which are designed to escape local minima and better approximate the true global optimum, thus improving calibration robustness and accuracy. Regarding

recent studies about global optimization algorithms, we denote some algorithms that stand out from the crowd. Indeed, Differential evolution presented by Storn, R. and Price, K. (1997), describe a methodology that converges faster and with more certainty than many other global optimizations process. This method is based on a population of candidate solutions (vectors) that evolves over time through mutation, crossover, and selection. Differential Evolution optimizer maintain a population of candidate solutions and create new candidate solutions by combining existing ones with each other (mutation) and then keep whichever candidate solution has the best fit with the current optimization problem. In opposition with a local optimization solver, we can notice that Differential Evolution method does not require an accurate initial guess to converge, as candidates are meant to mutate and evolve over the process.

Another widely used global optimizer is the Particle Swarm optimizer inspired by the work of Reynolds (1987). Indeed, that model was originally developed to simulate the movement of a flock of birds in the late 80s that Kennedy, J. and Eberhart, R. (1995) and Wang, D., Tan, D., and Liu, L. (2018) extended to build the Particle Swarm Optimizer. As its name suggests, this method is based on swarm intelligence, simulates a group of individuals (called particles) that move through the search space to find the optimum of a given objective function. By using very simple displacement rules (in the space of solutions), the particles can gradually converge towards a global minimum.

Another popular methodology is based on Bayes framework and famously known as Bayesian methodologies, are commonly used, as decribed by Kennedy, M. C. and O'Hagan, A. (2001) and Tohme, T., Vanslette, K., and Youcef-Toumi, K. (2020). This process is based on two key characteristics. First, it provides probabilistic predictions that incorporate all sources of uncertainty, including residual uncertainty in the estimated parameters. Second, it introduces a mechanism to account for model inadequacy, by explicitly filling the gap between observed financial data and the model's predictions. In other words, these methods can address the structural issues of the model in comparison to reality.

Finally Simulated Annealing, is another empirical optimisation method (metaheuristic) inspired by a process, annealing, used in metallurgy. In this process, slow cooling and reheating (annealing) cycles are alternated to minimise the energy of the material. The method comes from the observation that the natural cooling of certain metals does not allow the atoms to be placed in the most solid configuration. The most stable (more solid) configuration is reached by controlling the cooling and slowing it down with external heat or insulation. From those observations, the idea has been applied to optimisation process as presented by Kirkpatrick, S. (1984). By modifying a given state of the system, we obtain another state. Either this state improves the criterion that we are trying to optimise (we then say that we have reduced the system's energy) or it degrades it. If we accept a state that improves the criterion, we tend to look for the optimum close to the initial state. Accepting a "bad" state then allows us to explore a larger part of the state space and tends to avoid getting trapped too quickly in the search for a local optimum.

Among all of these global parameters, the preference between one and another sits on the task we want to perform, as some algorithm will be faster or more accurate for specific tasks. Indeed, each algorithm has its own philosophy of research, thus its application and accuracy depend upon the task. However, researchers agree that Particle Swarm and Differential Evolution are often the best ones among various type of objective function. As indicated by Arıcı, F., & Kaya, E. (2019) Particle Swarm and Differential Evolution shows promising results. Moreover, for a 10-dimension problem, Differential Evolution produce

slightly better results than Particle Swarm. Although it has been shown that Particle Swarm performance decrease with the number of dimensions, this algorithm remains accurate for a reasonable number of dimensions. However, in our case the dimension is below 10, thus the complexity of Differential Evolution Algorithm and Particle Swarm will be fully sufficient to perform our tasks.

2.5 Solving models equations

During the phase of solving the equations of a model to determine the predicted quantities, such as option values, various computational techniques have been extensively studied. As mentioned earlier, many option pricing models do not have closed form analytical solutions. This underlying difficulty requires the use of advanced numerical methods to obtain reliable estimates. As a result, the implementation of these models becomes more complex and requires careful methodological development to ensure both accuracy and efficiency.

Monte Carlo simulation, which uses repeated random sampling to approximate expected values, is one of the most common numerical approaches. This method is particularly well suited to high-dimensional problems and to models with trajectory-dependent characteristics. Despite its simplicity, Monte Carlo simulation is often computationally demanding as it requires large number of path simulations, particularly when high accuracy is expected.

Consequently, new methodologies appeared, aiming to reduce computing time while enhancing accuracy. However, these improvements often come at the cost, making these models more complex but more efficient. A well-known example is the application of the Fast Fourier Transform (FFT), originally developed by Cooley and Tukey (1965) for use in signal processing and telecommunications. This method was later adapted to financial engineering by Carr and Madan (1999), who demonstrated how the FFT could be applied to efficiently price European-style options by using the characteristic function of the underlying asset's log-price. Building on this foundation, Fang and Oosterlee (2009) introduced the COS method (Fourier-Cosine Series Expansions), a refined Fourier-based approach that employs cosine series expansions to further improve the accuracy and computational efficiency of option pricing, particularly under models with known characteristic functions. Collectively, these contributions underscore a significant evolution in numerical methods for derivative pricing, emphasizing the utility of transform-based techniques in handling complex financial models.

In the recent years, deep learning methods, particularly neural networks, have shown promising results in addressing the challenges posed by the option pricing problem. Indeed, in many applications, the speed with which calibration can be performed influences the usability of a model. Using Neural Networks can be more flexible and accurate than traditional mathematical models. Hutchinson et al. (1994) simulate Black-Scholes option prices and show that neural networks can recover the Black-Scholes formula from a two-year training set of daily options prices, and that the resulting network formula can be used successfully to both price and delta-hedge options out-of-sample. Similarly, Anders et al. (1998) conduct a comparative analysis between observed call option prices and those generated by the Black-Scholes model and several neural network architectures. They conclude that neural networks were increasingly better than the Black-Scholes model in terms of pricing accuracy.

From these observations, Neural Networks became a really promising methods in the financial modelling area. Hence, this method has been adapted for calibration purposes. In this approach, the neural network first learns to map the model, i.e. predict the outputs given the inputs. Consequently, it allows to heavily reduce the calibration process as it significantly accelerates option pricing. Indeed, as explained above, some models do not provide an analytical formula and prices prediction are often time consuming. Hence, the use of Neural Networks arose to solve this issue and improve calibration process.

Neural networks have been increasingly applied to the calibration of option pricing models. Recently, Hernandez (2017) applied a neural network to calibrate the single-factor Hull-White model. Advancing further, several studies have addressed the calibration of stochastic volatility models using neural networks, including Dimitroff et al. (2018), McGhee (2018) , and Liu et al. (2019a). Furthermore Itkin (2019) analyzed calibration methodologies using Neural Networks, identifying key limitations and proposing improvements that enhance both the performance and accuracy of the calibration process.

2.6 Two Principal Approaches to Model Calibration Using Neural Networks

Among the methodologies employed to calibrate models using Neural Networks we distinguish two emerging ones. Regardless the structure of the Neural Networks, the key difference between the two approaches sits on the number of steps involved in the calibration process. Hernandez (2017) trained its Neural Networks such that the model can map implied volatility surfaces to model inputs in a single step, taking market data as inputs and returning calibrated model parameters. However, he observed a significant decrease in calibration accuracy while testing the model on out-of-sample data. Although this method looks “straightforward” it comes with some drawbacks especially arising from the lack of control over the mapping function between implied volatility surfaces and model input, suggesting a not fully satisfactory generalisation of the learned map. Hernandez (2017) explained that retraining the Neural Networks every 2 or 3 months would help to fix this issue and allow the process to remain in acceptable bounds.

As a result, a second approach emerged, targeting to counter react the lack of control over the pricing function, by simply splitting the calibration process in two separate phases. This two-step approach has been employed by Bayer et al. (2019) and Horvath et al. (2019), who developed a Neural Network to approximate pricing map that is then supplemented by a traditional model calibration algorithm. In the first phase, called the forward pass, a neural network is trained offline to approximate the pricing map which associates the parameters of a stochastic model with corresponding option prices or implied volatilities. This step is performed only once as it allows the neural network to act as a simplified substitute for the pricing function, which is typically very computationally heavy. In the second phase, called the backward pass the calibration is performed online by adjusting the model parameters (Neural Networks inputs) such that the neural network predictions match the market data. This two-step process allows a reduction in computing time during the calibration stage, while maintaining a high level of accuracy. Benefits of this two-step approach comes from the efficiency of the pricing functions, improving training speed and flexibility while enabling the use of artificial data for training, as indicated by Bayer et al. (2019). Supported by Liu et al. (2019a), they highlighted that a two-step process significantly improve calibration accuracy in comparison with the one step process. They conclude that using Neural Networks combined with Differential Evolution

as a global optimizer was an efficient combination to calibrate option pricing models.

As described by Hernandez (2017), calibration methods using Neural Networks involve either Convolutional Neural Networks (CNN) or Feedforward Neural Network (FNN). However, the use of CNNs does not provide much improvement over simpler FNNs. Nevertheless, Recurrent Neural Networks, particularly Long Short-Term Memory networks (LSTM) show interesting results in terms of time series forecasting, pattern recognition, and risk assessment, which has motivated Caleiras (2024) to work with this type of Neural Network architecture. Indeed, by setting up a two-step calibration process, using a LSTM Neural Networks combined with the Differential Evolution algorithm, she demonstrated that calibration accuracy can be improved in comparison with FNNs.

2.7 Calibration Generalization

As we have seen previously, Neural Networks showed promising results in terms of calibration and especially concerning the accuracy and speed of the process. To extend this new approach of calibrating option pricing model using Neural Networks, I test the capacity of the calibrated parameters to be generalized to other underlying assets. Precisely, whether it is possible to calibrate an option pricing model on an index option chain and still have accurate results when predicting another index option chain with the same calibrated parameters. Grzelak et al. (2022) in their paper, propose a multivariate local volatility model that enables the simultaneous calibration of option prices on both individual stocks and the corresponding index. They have demonstrated that this joint calibration framework significantly improves pricing accuracy compared to separate calibrations, particularly when the underlying assets show strong dependencies. Moreover, their model guarantees the absence of arbitrage opportunities between options on the index and on individual stocks, providing a robust and economically solid foundation. This work shows that dependencies between markets can work together to improve calibration of option model. From one market to another, here basket option of an index and options on the same index, Grzelak et al. (2022) showed that synergies between these two markets helped to improve model performance. Consequently, authors have demonstrated that a joint calibration significantly improves the pricing accuracy and then that a potential cross-asset generalization is achievable.

The robustness of parameters calibrated using neural networks on real market data remain underexplored in the literature. Indeed, test of calibration using Neural Networks have been run on artificial data which casts doubt on its effectiveness on real market data. Moreover, it remains unclear whether these calibrated parameters can be generalized to predict option chains on different assets. Although neural network-based calibration methods have shown accuracy on artificial out-of-sample data, their effectiveness on actual market data and their ability to generalize across different indices has yet to be demonstrated.

3. Option Pricing Models and Global Optimizer

3.1 Option Pricing

An option is a financial contract, where the buyer has the right to buy (or sell) an asset at a fixed price and at a determined future date. We denote this predetermined future date T , which is called maturity. T simply tells you how much time the holder has left before being able to exercise the option. Indeed, for an European type option, it can only be exercised at maturity rather than for an American type option where the exercisability can be claimed at any time during the life of the contract. Moreover, a call is an option that gives the right to the holder to buy the asset and a put option gives the right to the holder to sell the asset. For simplicity we will provide a brief idea on how to price an option, using an European call option. From the previous explanations we can deduce that the payoff at maturity of a "call" option is $P_{call} = \max(S_T - K; 0)$, where S_T is the price of the underlying asset at time T (maturity date). The difficulty of pricing an option lies in getting a close estimate of what would be the value of the underlying asset S_t , at maturity. This topic will subject to intensive research, and lead to the development of multiple models such as the Heston model (1991) or the Bates model (1991). A famous method to express the price of an option is to work under the risk-neutral probability. Under the risk-neutral probability, the price of an option is calculated as the discounted expected value of its future payoff that is, its value at maturity. Risk-neutral probability allows risky assets to grow on average at the risk-free rate, which simplifies the calculation by avoiding the need to model risk premium. Thus, the option's price today is equal to the expected future payoff, discounted at the risk-free rate.

The pricing formula of an option under the risk-neutral probability is then:

$$C_0 = e^{-rT} \mathbb{E}^Q[\max(S_T - K, 0)]$$

Explanation of the terms:

- C_0 : the price of the option today (at time 0).
- e^{-rT} : the discount factor at the risk-free rate r over the period T .
- $\mathbb{E}^Q[\cdot]$: the expectation taken under the risk-neutral measure Q .
- S_T : the price of the underlying asset at maturity T .
- K : the strike price of the option.

3.2 Geometric Brownian Motion

Before starting to present the two models, let's start with the foundation, which is the Geometric Brownian Motion. As presented previously, the challenge here is to get a precise idea of the future price of the underlying asset. Consequently, a first idea is to model the asset price as a stochastic process, and more specifically using a Geometric Brownian motion. The idea is to discretize time into small fractions (increments). For instance, to

discretize time T to N fractions, each fraction's length will be T/N . At each steps, the asset price move according to the following formula (under risk-neutral probability) :

$$dS_t = rS_t dt + \sigma S_t dW_t \quad (3.1)$$

Where:

- dS_t : infinitesimal change in the asset price at time t
- r : risk-free interest rate
- S_t : asset price at time t
- dt : small increment of time
- σ : volatility of the asset (measures the size of fluctuations)
- dW_t : infinitesimal increment of a standard Brownian motion under the risk-neutral measure Q , modeling the randomness of the asset price path

Figure 3.1 shows an example of what a price path may look like using a Geometric Brownian Motion :

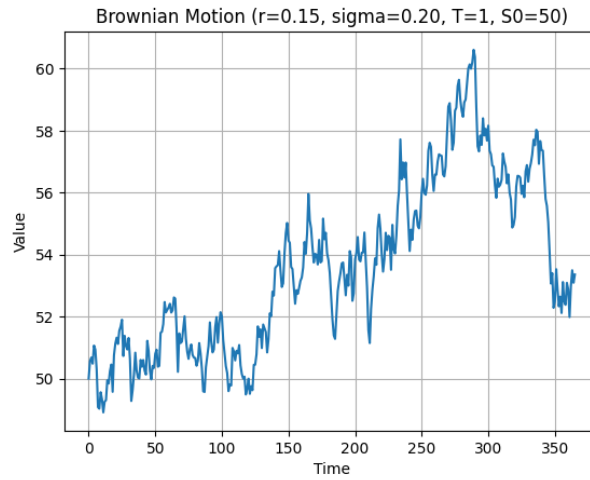


Figure 3.1: Brownian Motion Example ($r=0.15$, $\sigma=0.20$, $T=1$, $S_0=50$)

Equation ?? constitutes the foundation for the famous Black and Scholes formula. However the Black and Scholes model present some limitations that lies into modeling the volatility surface. Indeed, the volatility surface is a 3D representation showing how an option's implied volatility varies with maturity and strike (or moneyness). In theory, under Black-Scholes, the surface would be flat, but in reality it shows patterns like smiles or skews, reflecting market behavior and risk preferences. This volatility surface appear as market does not think that returns are normally distributed, but the distributions as fatter tails, which means that higher returns are more likely to happen that predicted by Black and Scholes model, explaining the apparition of a volatility surface.

3.3 Heston Model (1993)

We can see that in the equation ??, that σ is constant over time. Consequently, Heston (1993) decided to build a model with a stochastic volatility. The final model is driven by the following equations :

$$dS_t = rS_t dt + \sqrt{V_t} S_t dW_t^{(1)} \quad (3.2)$$

$$dV_t = \kappa(\theta - V_t) dt + \sigma_v \sqrt{V_t} dW_t^{(2)} \quad (3.3)$$

$$dW_t^{(1)} dW_t^{(2)} = \rho dt \quad (3.4)$$

Where:

- dS_t : infinitesimal change in the asset price at time t
- V_t : infinitesimal change in the variance of the asset at time t
- r : risk-free interest rate under the risk-neutral measure
- κ : speed of mean reversion of the variance process
- θ : long-term mean level of variance.
- σ_v : volatility of variance (“vol of vol”)
- $W_t^{(1)}, W_t^{(2)}$: two standard Brownian motions under Q
- ρ : correlation between the asset price and the variance movement
- equation (??) defines the correlation structure between the Brownian motions of the volatility and asset price

The first feature to note in this model is that the variance is modeled by a mean reverting process : $\kappa(\theta - V_t) dt$.

Indeed,

- If $V_t > \theta$ then $\kappa(\theta - V_t) < 0$ and volatility tend to decrease
- If $V_t < \theta$ then $\kappa(\theta - V_t) > 0$ and volatility tend to increase

For instance, if volatility recently rose we may have $V_t > \theta$ and consequently $\kappa(\theta - V_t) < 0$. Thanks to the mean reverting process, volatility will tend to go back to the value of its long-term mean θ . In other words, this process allows the volatility to oscillate around its long-term mean. This behavior reflects the trend of the volatility to revert to "normal" level after market stress period. Figure 3.2 and 3.3 represent price and volatility path according to the Heston model. The parameters used to draw these graphs are as follow,

$$\left\{ \begin{array}{l} T = 1 \\ S_0 = 100 \\ V_0 = 0.15 \\ \kappa = 3 \\ \theta = 0.15 \\ \sigma_v = 0.25 \\ \rho = -0.8 \\ r = 0.05 \end{array} \right.$$

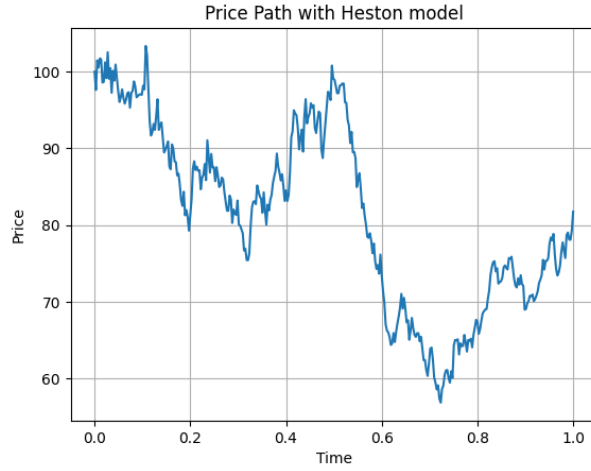


Figure 3.2: Price Path using Heston Model

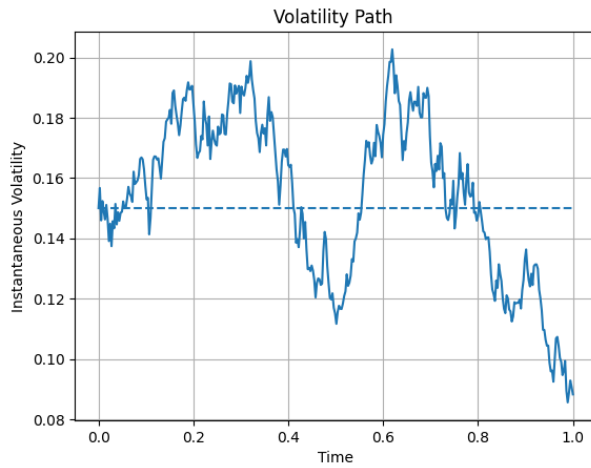


Figure 3.3: Volatility Path using Heston Model

Figure 3.3 shows the mean-reverting process, as we can see that the volatility is oscillating around its long term mean here $\theta = 0.15$.

3.4 Bates Model (1991)

Bates (1991) decided to add jumps in the price path using a Poisson process. Indeed, sometimes we can observe several crash in asset prices charts. Hence he decided to

introduce a term that will try to model erratic's path of asset prices. Similarly, the model equations are as follows,

$$dS_t = S_t \left((r - \lambda \mathbb{E}[J - 1]) dt + \sqrt{V_t} dW_t^{(1)} + (J - 1) dN_t \right) \quad (3.5)$$

$$dV_t = \kappa(\theta - V_t) dt + \sigma_v \sqrt{V_t} dW_t^{(2)} \quad (3.6)$$

$$dW_t^{(1)} dW_t^{(2)} = \rho dt \quad (3.7)$$

Where:

- dS_t : infinitesimal change in the asset price at time t
- V_t : instantaneous variance
- r : risk-free interest rate
- κ : mean reversion rate of the variance
- θ : long-term mean variance
- σ_v : volatility of the variance, "vol of vol"
- ρ : correlation between the Brownian motions $W_t^{(1)}$ and $W_t^{(2)}$
- $W_t^{(1)}, W_t^{(2)}$: two correlated Brownian motions
- N_t : Poisson process with intensity λ
- J : multiplicative jump size factor for the asset price, $J > 0$
- λ : jump intensity
- $\mathbb{E}[J - 1]$: expected relative jump size

Figure 3.4 and 3.5 represent price and volatility path according to the Bates model. The parameters used to draw these graphs are as follow,

$$\left\{ \begin{array}{l} T = 1.0 \\ S_0 = 100.0 \\ V_0 = 0.04 \\ r = 0.03 \\ \kappa = 2.0 \\ \theta = 0.04 \\ \sigma_v = 0.5 \\ \rho = -0.7 \\ \lambda_j = 1 \\ \mu_j = 0 \\ \sigma_j = 0.1 \end{array} \right.$$

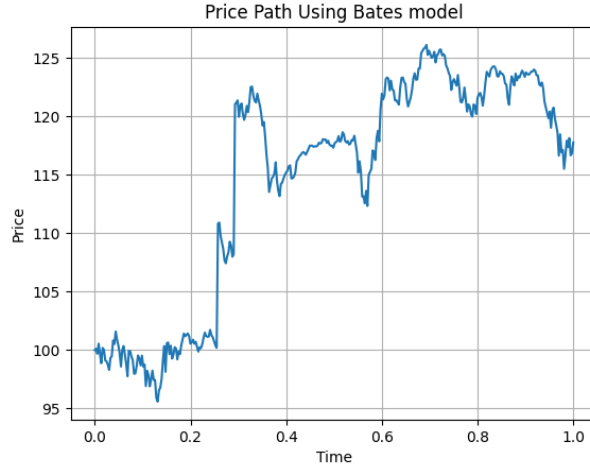


Figure 3.4: Price Path using Bates Model

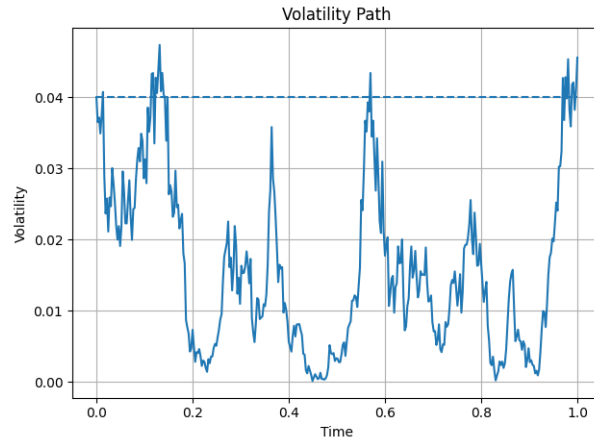


Figure 3.5: Volatility Path using Bates Model

Figure 3.4 shows the erratic path of the asset price. We can see that price can rise quickly and looks like a staircase. Figure 3.5 shows the mean-reverting process, as we can see that the volatility is oscillating around its long term mean here $\theta = 0.04$.

3.5 Global Optimizer : Differential Evolution Algorithm

Previous presented model have multiple parameters that need to be calibrated to the market to be actually operational. As I have briefly explained it in the literature review (section 2.4), the problem of calibrating such models is a non-convex one. Which means that the objective function to minimize present multiple local-minimas that traditional optimizer will fail to escape. From that statement, I have chosen to work with a global optimizer in order to find the best solution without being stuck into a local minimum, and hopefully find the global minimum.

Differential evolution presented by Storn, R. and Price, K. (1997), is the global optimizer that I have chosen to work with in this thesis. The principle of this algorithm is to work with a population of candidates solutions (vectors) that evolves over time through mutation, crossover, and selection. In the following discussion I will work you through the steps of this algorithm.

Step 1: Initialization: Generate an initial population of N candidate solutions $\{\mathbf{x}_i^{(0)}\}_{i=1}^N$ within the bounds of the problem.

Step 2: Mutation: For each target vector $\mathbf{x}_i^{(g)}$ at generation g , create a mutant vector $\mathbf{v}_i^{(g)}$ by adding the weighted difference of two randomly selected distinct population vectors to a third one:

$$\mathbf{v}_i^{(g)} = \mathbf{x}_{i_1}^{(g)} + F \cdot (\mathbf{x}_{i_2}^{(g)} - \mathbf{x}_{i_3}^{(g)})$$

where indices i_1, i_2, i_3 are distinct and different from i , and F is the mutation factor.

Step 3: Crossover: Construct a trial vector $\mathbf{u}_i^{(g)}$ by mixing the target vector $\mathbf{x}_i^{(g)}$ and the mutant vector $\mathbf{v}_i^{(g)}$. For each component j ,

$$u_{i,j}^{(g)} = \begin{cases} v_{i,j}^{(g)} & \text{if } \text{rand}_j \leq CR \text{ or } j = j_{\text{rand}} \\ x_{i,j}^{(g)} & \text{otherwise} \end{cases}$$

where $\text{rand}_j \sim U(0, 1)$, $CR \in [0, 1]$ is the crossover probability, and j_{rand} is a randomly chosen index to ensure at least one component is taken from the mutant vector.

Step 4: Selection: Evaluate the objective function f for $\mathbf{u}_i^{(g)}$ and $\mathbf{x}_i^{(g)}$. The vector with the better fitness survives to the next generation:

$$\mathbf{x}_i^{(g+1)} = \begin{cases} \mathbf{u}_i^{(g)} & \text{if } f(\mathbf{u}_i^{(g)}) \leq f(\mathbf{x}_i^{(g)}) \\ \mathbf{x}_i^{(g)} & \text{otherwise} \end{cases}$$

The steps are repeated until a predefined criterion is satisfied, such as reaching a specified number of iterations or when all elements of the population become very similar. the choice of the control parameters may impact the performance of the algorithm. For instance, a large population size and mutation rate can increase the probability of finding the global minimum. Moreover fixing a minimum rate of improvement at each steps, could be internally initiated to prevent from over searching and optimize search time.

4. Method

4.1 Two Step Approach¹

In this thesis I use a two step approach, composed of the forward and backward pass. The first stage, called forward pass, involves training a neural network on a large synthetic dataset generated from the stochastic model. The objective is to approximate the model's pricing function by using a Neural Network that act as a fast substitute for the original pricing method. In the second stage, known as the 'backward pass', the trained network is used during the calibration process to replace traditional numerical pricing techniques. This speeds up the calibration process considerably by allowing rapid evaluation of option prices. Each of these steps can be formulated as an optimisation problem. In the learning phase, optimisation consists of adjusting the weights of the neural network in order to minimise a loss function that measures the error between the network's predictions and the dataset prices. In the calibration phase, optimisation consists of adjusting the parameters of the stochastic model (neural network input parameters) in order to minimise a cost function that measures the difference between the option prices observed on the market and those predicted by the neural network. Figure 4.1 summarize the process followed in this thesis. First I build a training dataset that will feed my Neural Network. Secondly, I perform the Forward Pass which is the Neural Network training phase. And finally, the Backward Phase which involves the training Neural Networks into the calibration phase.

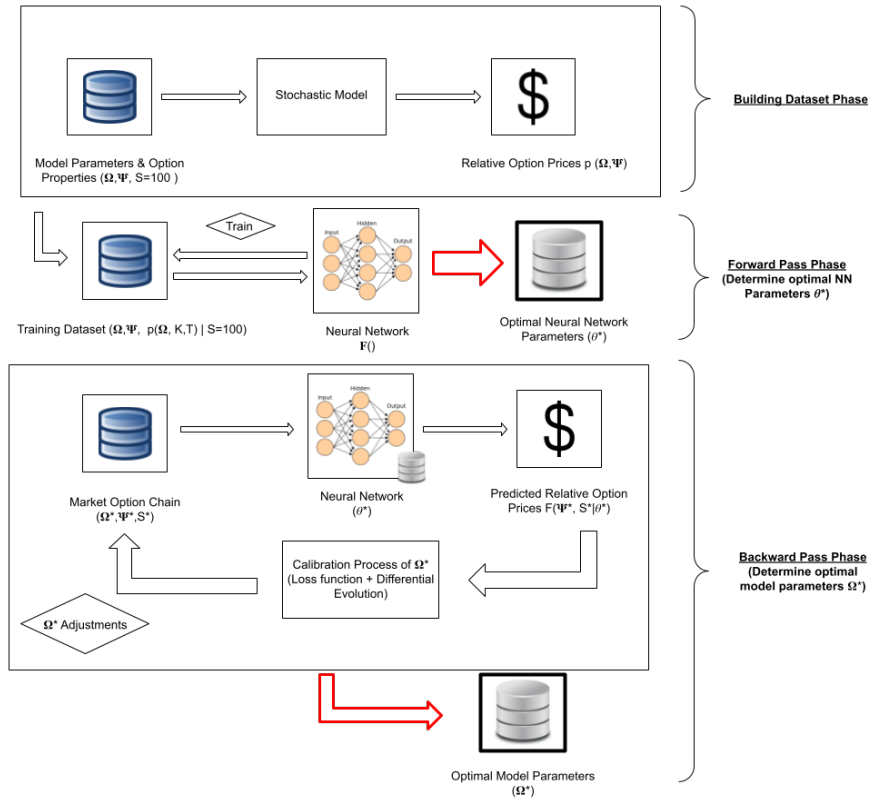


Figure 4.1: Two Step Approach

¹Notation and exposition in this chapter (Section 4.1) are adapted from and closely followed that of Caleiras (2024).

4.1.1 The Forward Pass: Learning the Pricing Function

Calibration of a stochastic model cannot be discussed without first addressing the pricing function, which lies at the core of the calibration process. To calibrate a model effectively, one must compute option prices a large number of times and adjust the model parameters so that the predicted prices align as closely as possible with observed market prices. This repetitive process highlights a key challenge: the pricing function is really important, especially when option prices cannot be directly derived from stochastic models through closed-form analytical solutions. In most cases, computing these prices requires heavy numerical methods such as Monte Carlo simulations or finite difference methods. Since the calibration procedure involves evaluating the pricing function many times, the efficiency and accuracy of this pricing method become crucial. This problem has long been a topic of intensive research, and various numerical approaches have been proposed to approximate option prices under stochastic models. Indeed, Carr and Madan (1999) developed a method called Fast Fourier Transform (FFT) to accurately price European-style options by leveraging the characteristic function of the logarithm of the underlying asset's price. Building upon this fundamental work, Fang and Oosterlee (2009) proposed the COS method, a Fourier-based technique utilizing cosine series expansions to enhance both the accuracy and computational efficiency of option pricing. More recently, Hernandez (2017) proposed a Neural Networks approach to map implied volatility surfaces to model inputs. However, in our approach, I would not use one of the method cited above but I will use a neural network as pricing function. Indeed, this pricing function will be set up in two separated steps *the Forward Pass* and *the Backward Pass*, as proposed by Bayer et al. (2019) and Horvath et al. (2019) and supported by Liu et al. (2019a). The idea is straightforward yet: I replace the traditional, computationally expensive pricing methodology with a trained neural network that can quickly predict option prices based on input parameters (i.e. option properties and model parameters). By doing so, I am able to reduce the computational burden of calibration.

To end up with reliable predictions I need to split this process in two parts.

- Dataset Construction
- Online Training

As previously discussed, the goal is to train a neural network to approximate the pricing behavior of a given stochastic model. Indeed, I want the neural network to learn how to predict the relative price of an option based on the parameters of the stochastic model and the characteristics of the option itself. To achieve this, the first step is to construct a dataset that captures the relationship between model parameters, option properties, and corresponding option prices. This dataset serves as the training ground for the neural network. The quality and accuracy of this dataset are really important, as they directly impact the predictive performance of the model. This data generation phase is fundamental and forms the first stage of the process.

The dataset includes N_p different sets of model parameters, denoted as $\mathbf{\Omega} = (\Omega_1, \dots, \Omega_{N_p})$, option properties denoted as $\Psi_{i,j} = (\Psi_{i,j,k})_{k=1,\dots,N_p}$, and corresponding option prices $P^{(\text{model})}(\mathbf{\Omega}, \Psi_{i,j})$. For each set of parameters $(\Omega_1, \dots, \Omega_{N_p})$, is assigned a set of option properties where $i \in \mathcal{K}$, (where $\mathcal{K}$ is the strike space and $|\mathcal{K}|$ its cardinality) and $j \in \mathcal{T}$ (where $\mathcal{T}$ is the maturity space and $|\mathcal{T}|$ its cardinality). The architecture of the dataset is as follows:

Ω_1	Ψ_1	$\Psi_{1,1}$	$P^{(\text{model})}(\mathbf{\Omega}_1, \Psi_{1,1})$
		$\Psi_{1,2}$	$P^{(\text{model})}(\mathbf{\Omega}_1, \Psi_{1,2})$
		$\vdots$	$\vdots$
		$\Psi_{1, \mathcal{T} }$	$P^{(\text{model})}(\mathbf{\Omega}_1, \Psi_{1, \mathcal{T} })$
Ω_1	Ψ_2	$\Psi_{2,1}$	$P^{(\text{model})}(\mathbf{\Omega}_1, \Psi_{2,1})$
		$\Psi_{2,2}$	$P^{(\text{model})}(\mathbf{\Omega}_1, \Psi_{2,2})$
		$\vdots$	$\vdots$
		$\Psi_{2, \mathcal{T} }$	$P^{(\text{model})}(\mathbf{\Omega}_1, \Psi_{2, \mathcal{T} })$
Ω_1	$\Psi_{ \mathcal{K} }$	$\Psi_{ \mathcal{K} ,1}$	$P^{(\text{model})}(\mathbf{\Omega}_1, \Psi_{ \mathcal{K} ,1})$
		$\vdots$	$\vdots$
		$\Psi_{ \mathcal{K} , \mathcal{T} }$	$P^{(\text{model})}(\mathbf{\Omega}_1, \Psi_{ \mathcal{K} , \mathcal{T} })$
Ω_2	Ψ_1	$\Psi_{1,1}$	$P^{(\text{model})}(\mathbf{\Omega}_2, \Psi_{1,1})$
		$\Psi_{1,2}$	$P^{(\text{model})}(\mathbf{\Omega}_2, \Psi_{1,2})$
		$\vdots$	$\vdots$
		$\Psi_{1, \mathcal{T} }$	$P^{(\text{model})}(\mathbf{\Omega}_2, \Psi_{1, \mathcal{T} })$
Ω_2	$\Psi_{ \mathcal{K} }$	$\Psi_{ \mathcal{K} , \mathcal{T} }$	$P^{(\text{model})}(\mathbf{\Omega}_2, \Psi_{ \mathcal{K} , \mathcal{T} })$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Ω_{N_p}	Ψ_1	$\Psi_{1,1}$	$P^{(\text{model})}(\mathbf{\Omega}_{N_p}, \Psi_{1,1})$
		$\vdots$	$\vdots$
		$\Psi_{ \mathcal{K} , \mathcal{T} }$	$P^{(\text{model})}(\mathbf{\Omega}_{N_p}, \Psi_{ \mathcal{K} , \mathcal{T} })$

The corresponding option prices $P_{i,j}^{(\text{stochastic model})}(\Omega, \Psi_{i,j})$ are computed using a Monte-Carlo simulation. An analysis of this task is conducted in the section 4.2.3. Nonetheless, the Monte Carlo simulation compute a price that is relative to the initial underlying price. Indeed, for simplicity, the Monte Carlo method, calculate a price of the option that is a percentage of the initial underlying price. In fact, in the architecture of the code, the initial underlying price has been kept equal to 100 for every simulation. However, the predicted price must be divided by the initial underlying price (here 100) in the dataset to allow a generalization of the Neural Network, particularly when confronted to underlying prices different from 100. Consequently, during the Backward pass, the option price will not be biased as what will be predicted by the Neural Network will be multiplied by the corresponding initial underlying price (which will provide the real price of the option). Therefore the Neural Network will be able not to directly predict the option price but a relative option price (C/S_0) that is simply a percentage of the initial underlying price.

The second stage is the training phase, often called as the Forward Pass. In this phase, the neural network is trained on the previously constructed dataset, learning to map input parameters (i.e., model parameters $(\Omega_1, \dots, \Omega_{N_p})$ and option properties $\Psi_{i,j}$ with $i \in \mathcal{K}$ and $j \in \mathcal{T}_i$) to output prices ($P_{i,j}^{(\text{stochastic model})}(\Omega, \Psi_{i,j})$). If the network is well trained, it will approximate the behavior of the underlying stochastic pricing model with high accuracy. Therefore, the neural network will be able to mimics the pricing function of the original stochastic model.

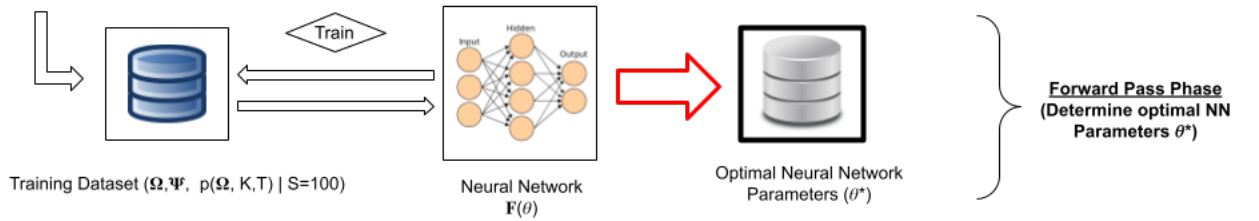


Figure 4.2: Forward Pass

It is important to note that the training phase is performed only once. During actual calibration (discussed in the Backward Pass), the trained Neural Network is only used offline, to predict option prices given input parameters. Which means that Neural Network's weights (Θ^*) are kept fixed during the backward Pass phase as they already have been calculated in the Forward Pass phase. To summarize, the motivation behind this approach is to replace a computationally expensive pricing model with a fast, accurate neural network. While building the training dataset and training the network can be time-consuming, it significantly accelerates the calibration process by avoiding repeated simulations or numerical solutions during calibration.

4.1.2 The Backward Pass: Calibration of Model Parameters

In this phase, I use the previously trained neural network to calibrate the model parameters of the Bates (1991) and Heston (1993) models on real market data. The objective is to identify the set of model parameters (Ω) that allows the pricing model to fit with observed market option prices $\hat{P}_k(\hat{\Psi}_k)$. In other words, given the properties of the options ($\hat{\Psi}_k$), our goal is to determine the set of model parameters (Ω^*) which will enable the neural network to predict option prices that closely align with the actual market prices ones. More precisely, this phase can be seen as an iterative trial-and-error process, where a global optimization algorithm (here Differential Evolution algorithm) adjusts the model

parameters Ω in order to minimize a predefined loss function $\mathcal{L}$. At each iteration, the optimizer updates Ω to reduce the gap between the option prices predicted by the neural network and the actual market prices ($\hat{P}_k(\hat{\Psi}_k)$). Using a Neural Network significantly reduces the computational cost compared to traditional numerical pricing method, hence improving the overall performance of the calibration process. This phase seeks to minimize the following loss function :

$$\min_{\Omega} Loss(\Omega) = \min_{\Omega} \sum_k \mathcal{L}(\hat{P}_k(\hat{\Psi}_k), F(\Omega, \hat{\Psi}_k; \theta^*)) \quad (4.1)$$

where the corresponding optimum is :

$$\Omega^* = \arg \min_{\Omega} \sum_k \mathcal{L}(\hat{P}_k(\hat{\Psi}_k), F(\Omega, \hat{\Psi}_k; \theta^*)) \quad (4.2)$$

where k indexes the set of options considered in the calibration period.

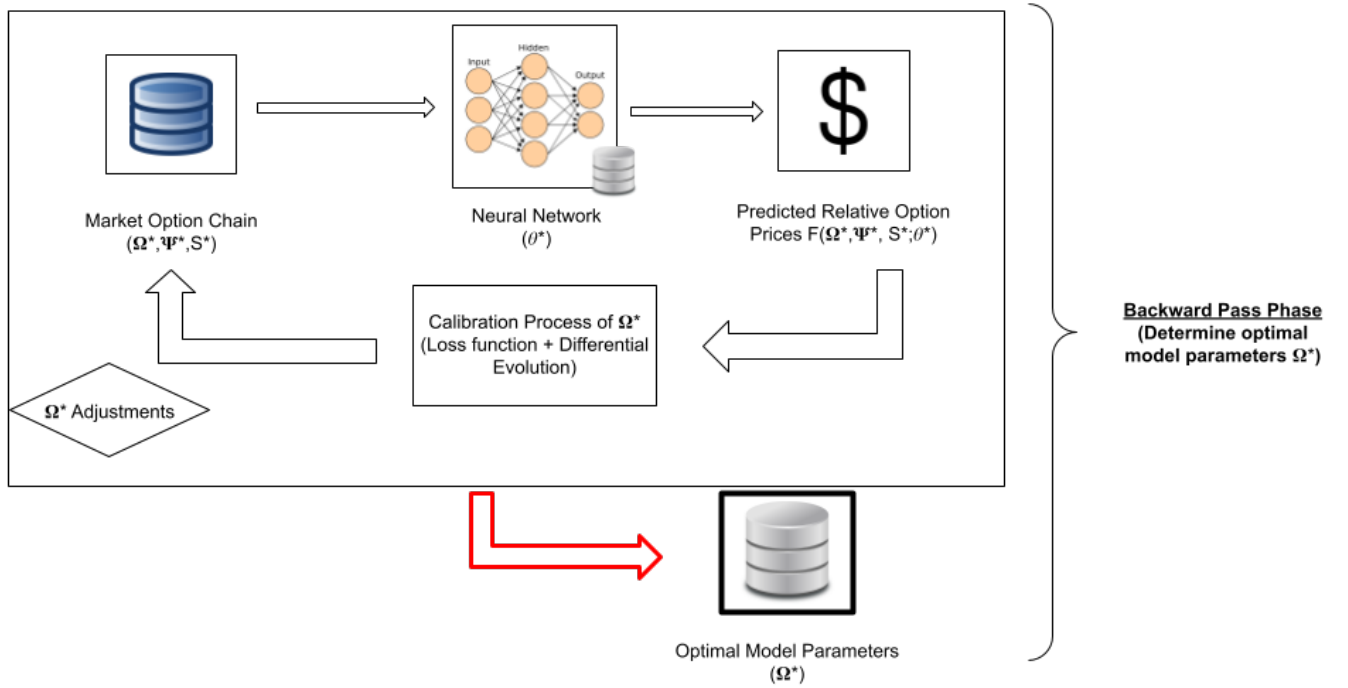


Figure 4.3: Backward Pass

4.2 Forward Pass

4.2.1 Dataset Construction

Sets of model parameters were generated using the Latin Hypercube Sampling (LHS) method. In this approach, each new sample is strategically placed based on the positions of previously selected samples, ensuring that the entire parameter space is explored. This results in a well-distributed set of samples, where each sample contributes to covering a wide part of the specified range. Regardless of the model involved I have chosen to set up $N_p = 100$.

4.2.2 Generation of Model parameters and Option Properties

Heston parameters generation

Concerning Heston model, it works with 5 parameters as follows :

- ζ Long term Variance
- κ Mean Reversion Speed
- δ Volatility of volatility
- ν_0 Initial variance
- ρ Correlation

Ranges provided to LHS method for Heston model are presented in table 4.1 and are the same as used by Caleiras (2024) :

Table 4.1: Sampling range for the Heston model parameters

Parameter	Value Range
Long-term variance, ζ	[0.01, 0.5]
Mean reversion speed, κ	(0, 3.0]
Volatility of volatility, δ	[0.01, 0.8]
Initial variance, ν_0	[0.03, 0.15]
Correlation, ρ	[-0.9, -0.2]
Risk-Free Interest Rate, r	[0.005, 0.05]

At this stage, LHS method generates an equally distributed sample of parameters according to the given range. Since I have chosen a number $N_p = 100$, I have 100 set of parameters $\Omega^{(\text{Heston})}$ composed of $(\zeta, \kappa, \delta, \nu_0, \rho, \mu)$.

Bates parameters generation

Concerning Bates model , it works with 8 parameters as follows :

- ζ Long term Variance
- κ Mean Reversion Speed
- δ Volatility of volatility
- ν_0 Initial variance
- ρ Correlation
- λ_j Jumps Intensity
- μ_j Jumps Logarithmic Mean
- σ_j Jumps Standard Deviation

Table 4.2 presents the ranges provided to LHS method for Bates model based on the value used by Liu et al. (2019a) :

Table 4.2: Sampling range for the Bates model parameters

Parameter	Value Range
Long-term variance, ζ	[0.01, 0.5]
Mean reversion speed, κ	(0, 3.0]
Volatility of volatility, δ	[0.01, 0.8]
Initial variance, ν_0	[0.03, 0.15]
Correlation, ρ	[-0.9, -0.2]
Risk-Free Interest Rate, r	[0.005, 0.05]
Jumps Intensity, λ_j	[0, 3]
Jumps Logarithmic Mean , μ_j	[-0.1, 0.05]
Jumps Standard Deviation, σ_j	[0, 0.3]

At this stage, LHS method have generated an equally distributed sample of parameters according to the given range. Since I have chosen a number $N_p = 100$, I have 100 set of parameters $\Omega^{(\text{Bates})}$ composed of $(\zeta, \kappa, \delta, \nu_0, \rho, \mu, \lambda_j, \mu_j, \sigma_j)$.

N.B. Although r is an exogenous parameters, and consequently not adjustable by the model, the Neural Network still need to be able to deal with it. One should not fixed this value during the training phase, as it puts at risk the the Neural Network's capacity of generalization. Hence, risk-free interest rate will not be kept fixed in the training dataset but it will not be calibratable during the Backward Pass. The Neural Network will learn the parameter behavior during the training phase but r will not be calibratable during the backward pass as it is a macroeconomic parameter and should not help the model to fit to the data.

Option properties generation

To complete these datasets, each set of parameters has been assigned the same option properties. In other words for each Ω_k with $k \in \{1, \dots, N_p\}$, is assigned all strike prices K_i with $i \in \{1, \dots, |\mathcal{K}|\}$ and for each K_i is assigned all maturities T_j with $j \in \{1, \dots, |\mathcal{T}|\}$. This architecture allows to have a wide range of options for each set of parameters. Moneyness and maturities have been generated using Linspace from Numpy. For each I have chosen a number of 40 moneyness and maturities ranging from [0.6, 1.4] (focusing on relatively more liquid options to have accurate prices) and [0.05, 2] (in years) respectively. The moneyness has been defined as S_0/K . Using moneyness instead of strikes and initial underlying price allows us to reduce the number of input parameters for the Neural Networks. Therefore the number of datapoints generated by this method is equal to the $N_p \times |\mathcal{K}| \times |\mathcal{T}|$ which in our case is equal to $100 \times 40 \times 40$. At this stage I have my training dataset almost complete. I have got so far model parameters and option properties. Furthermore, I now will focus on how the price has been derived from the model and dataset parameters and especially what settings have been used to set up the Monte Carlo method. Indeed, the number of data points justifies the study in section 4.2.3 and seek to answer the following questions : how to set up the Monte Carlo method and especially how many simulations are required to end up with an accurate result.

Generated Dataset Summary Statistics

Table 4.3 and Table 4.4 are the descriptive statistics of the generated data for both Bates and Heston model respectively.

Table 4.3: Descriptive Statistics for the generated dataset (count=160000, Bates model)

	zeta	ksi	delta	V0	rho	mu	lambda_j	mu_j	sigma_j	T	moneyiness
mean	0.26	1.50	0.40	0.09	-0.55	2.75	1.50	-0.03	0.15	1.02	1.00
std	0.14	0.87	0.23	0.03	0.20	1.30	0.86	0.04	0.09	0.58	0.24
min	0.01	0.00	0.01	0.03	-0.89	0.53	0.02	-0.10	0.00	0.05	0.60
25%	0.13	0.76	0.21	0.06	-0.72	1.65	0.76	-0.06	0.08	0.54	0.80
50%	0.25	1.51	0.40	0.09	-0.55	2.74	1.51	-0.02	0.15	1.02	1.00
75%	0.38	2.24	0.60	0.12	-0.38	3.87	2.25	0.01	0.22	1.51	1.20
max	0.50	2.98	0.79	0.15	-0.21	4.99	2.98	0.05	0.30	2.00	1.40

Table 4.4: Descriptive Statistics for the generated dataset (count=160000, Heston model)

	zeta	ksi	delta	V0	rho	mu	T	moneyiness
mean	0.26	1.50	0.41	0.09	-0.55	2.75	1.02	1.00
std	0.14	0.87	0.23	0.03	0.20	1.30	0.58	0.24
min	0.01	0.02	0.02	0.03	-0.90	0.53	0.05	0.60
25%	0.13	0.75	0.21	0.06	-0.72	1.65	0.54	0.80
50%	0.26	1.51	0.40	0.09	-0.55	2.76	1.02	1.00
75%	0.38	2.25	0.60	0.12	-0.38	3.87	1.51	1.20
max	0.50	3.00	0.79	0.15	-0.20	4.96	2.00	1.40

The tables show that generated dataset cover a wide range of data, which will allow an efficient generalization ability of the Neural Network.

4.2.3 Monte Carlo setting

The Monte Carlo methodology is a heavy computational method. Indeed, the basic principle is to simulate several path to derive a result. However, here a question arises, how many path should be calculated to obtain an accurate result. Basically, Monte Carlo simulation is a stochastic process, then with a fixed number of simulation, the calculated price will always be different. Hence I would like to know how many iterations are required to end up with a "stable" approximation of the result. To do this, I compare the Monte Carlo price estimate to a stable benchmark, such as the price given by the Black-Scholes formula. Specifically, I calculate the relative error between the Monte Carlo estimate and the Black-Scholes result for different numbers of simulations. However as explained previously, Monte Carlo simulation is a stochastic process, hence the error is stochastic itself making error graphs really noisy. To address this, I repeat the simulation process 50 times for each number of iterations and compute the average of the absolute relative errors to actually have an idea from which number of simulation the Monte Carlo method provide a relatively stable result. Repeating this process and taking the absolute average will help to smooth out the error curve.

Figure 4.4 graph the results showing how the relative error evolve with the number of simulation. I can see on the graph that a number of 3500 simulations start to provide a

relatively stable price prediction. Consequently, I will use this value to set up the Monte Carlo method. Hence for each set of parameters (Ω, Ψ) a Monte Carlo method of 3500 simulations has been run.

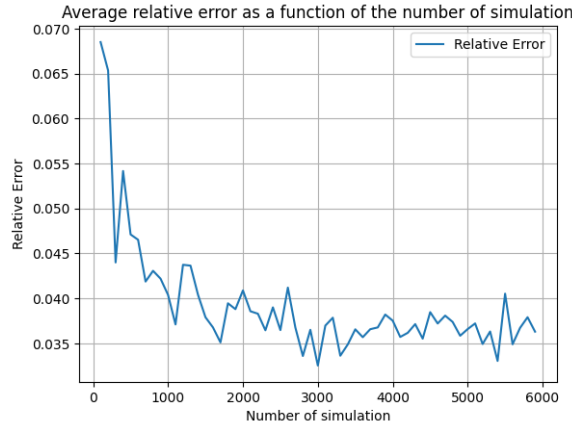


Figure 4.4: Average relative error to BS prediction (N=50)

4.2.4 Neural Network Training

After deriving all the prices from their respective set of parameters and option properties, I can start training the neural networks. Following Liu et al.(2019a) and their results, I decided to work with a FeedForward Neural Network Architecture. The complete data set is randomly divided into two parts, respectively 80% for training and 20% for testing. Moreover, the training dataset will be normalized before being given to the Neural Networks in order to avoid scale-dependent biases. Table 4.5 presents Neural Network architecture.

Parameters	Options
Hidden layers	4
Neurons (each layer)	200
Activation	ReLU
Dropout rate	0.0
Optimizer	Adam
Batch size	32

Table 4.5: Neural network training configuration

For both model, the training did not show any overfitting, as the loss curve of the out of sample test did not show any increase over the epochs. Indeed, a decreasing out of sample error (validation loss error) shows an accurate capacity of generalization from the Neural Network, hence no evidence of overfitting.

Table 4.6: Performance metrics of the trained neural network (Heston)

Metric	Training	Testing
MSE	0.26 %	0.27 %
MAE	3.64 %	3.70 %
MAPE	29.82 %	36.47 %
R^2	99.73 %	99.72 %

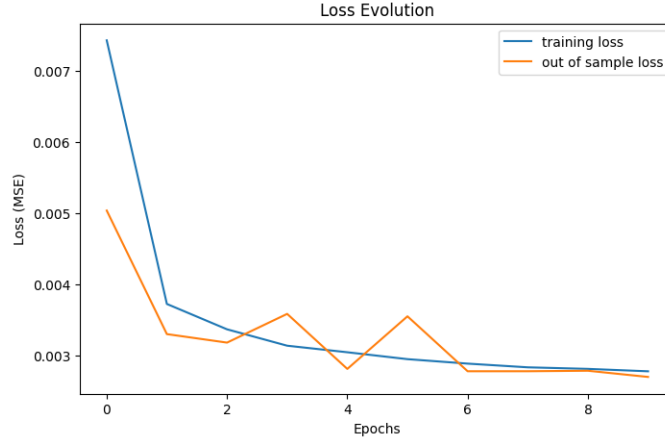


Figure 4.5: Heston Training Loss Evolution (e=10, MSE)

Table 4.7: Performance metrics of the trained neural network (Bates)

Metric	Training	Testing
MSE	0.37 %	0.39 %
MAE	4.26 %	4.38 %
MAPE	38.11 %	36.06 %
R^2	99.62 %	99.60 %

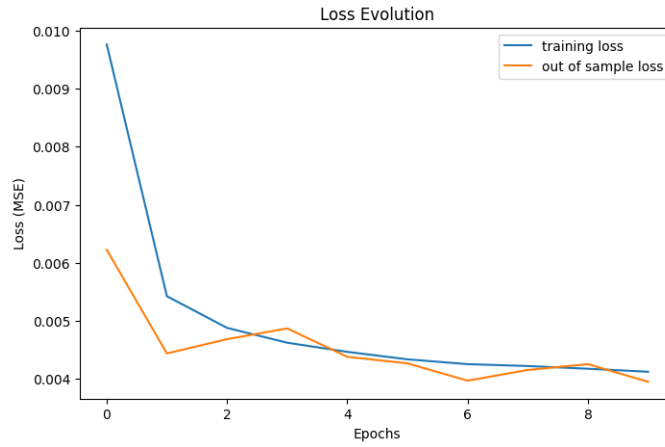


Figure 4.6: Bates Training Loss Evolution (e=10, MSE))

Table 4.6 and 4.7 and Figure 4.5 and 4.6 show the performances observed of each Neural Network, from which I can deduct that the forward pass is able to accurately map the model parameters to option prices in a robust manner. Indeed, the out of sample performance are very similar to the training performance, showing that the ANN is able to generalize well (no-overfitting). Moreover, the results are in accordance with the findings of Liu et al. (2019a) showing good results when training the Neural Networks. The findings in this thesis are slightly less satisfactory. I explain this loss by the size of the training dataset and number of epochs. Indeed, larger the dataset better both sample-in fit and generalization (out-of-sample test) would be. Moreover, the number of epochs used in this thesis is less than 20 compared to Liu et al. (2019a) who ran 8000 epochs.

4.3 Backward Pass

4.3.1 Index Data

In this thesis I have chosen to work with 2 option chains. The first one is a combination of two years of Invesco QQQ Trust Series 1, \$QQQ², options end of day quotes ranging from 01-2021 to 12-2022. The second one is a combination of two years of SPDR S&P 500 ETF Trust (\$SPY) options end of day quotes ranging from 01-2021 to 12-2022. Each row represents the information associated with one contract's strike price and expiration date, relatively to the underlying last price on the quote date. Moreover, to met our requirement of trying to predict one index with an other, I must investigate the tracking performance of these two ETFs.

Invesco QQQ Trust Series 1 (Tracking NASDAQ)

Figure 4.7 and Table 4.8 below are provided by the issuer Invesco and they show that the tracking error is really low, less than 1.5% over 10Y which I have considered as acceptable to use the ETF data as a proxy for the index data.

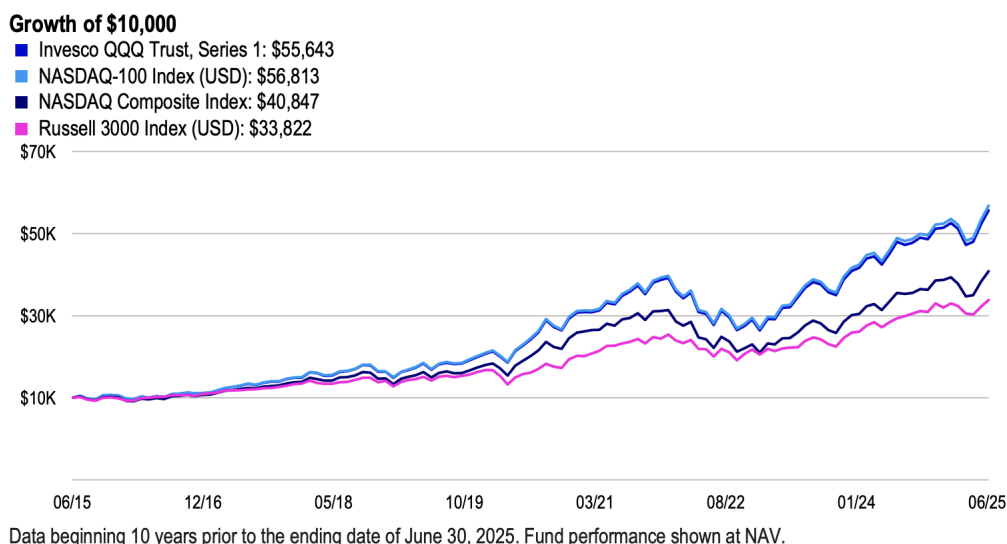


Figure 4.7: Invesco QQQ Trust Series 1 performance in comparison with NASDAQ-100 Index)

Table 4.8: Performance as of June 30, 2025 of Invesco QQQ Trust Series 1 (in %)

Performance (%)	YTD	1Y	3Y	5Y	10Y	Fund Inception
ETF - NAV	8.26	15.82	26.17	18.12	18.72	10.17
ETF - Market Price	8.17	15.79	26.15	18.12	18.70	10.16
Underlying Index	8.35	16.10	26.44	18.36	18.97	10.40

²Invesco QQQ Trust Series performance is available at :<https://www.invesco.com/qqq-etf/en/performance.html>

SPDR S&P 500 ETF Trust (Tracking S&P 500)

³ Similarly, for the SPDR S&P 500 ETF Trust, Figure 4.8 and Table 4.9, provided by the issuer SPDR, indicate that the tracking error is relatively low, approximately 1% over a 10-year period. This level of tracking error is considered acceptable for using the ETF data as a proxy for the index data.



Figure 4.8: SPDR S&P 500 ETF Trust (SPY) performance in comparison with S&P 500 Index (SPX)

Table 4.9: Total Return (As of 06/30/2025)

Metric	QTD	YTD	1 Year	3 Year	5 Year	10 Year
NAV (%)	10.91	6.16	15.05	19.56	16.50	13.50
Market Value (%)	10.84	6.08	14.97	19.54	16.52	13.50
Index (%)	10.94	6.20	15.16	19.71	16.64	13.65

4.3.2 Backward pass on Index Data and Error Prediction

Methodology overview

To proceed to the backward phase, I first reworked the index datasets, by calculating option properties such as moneyness and maturity. As explained in the Section 4.2.2, r has been added in the training dataset but must be kept fixed during the calibration phase. Hence, I have chosen the 10-year US Government Bond as the risk-free rate, because it is widely regarded as the standard proxy for the risk-free rate in USD-denominated rate, further demonstrated by Carleton & Lakonishok (1985) and Fama & Bliss (1987). Its high liquidity, low default risk, and widespread use in both academic and professional finance further support this choice.

³SPDR S&P 500 ETF Trust performance is available at:<https://www.ssga.com/ch/fr/intermediary/etfs/spdr-sp-500-etf-trust-spy>

In this study, I investigate whether it is possible to predict the option chain of a given index using models calibrated on a different index. To do so, I organize the dataset on a weekly basis, assigning each option to its corresponding week. For each week, I calibrate the model parameters separately. Specifically, I perform two types of weekly calibrations over the first 30 weeks: one using the index I aim to predict, and another using a different index. This results in two distinct sets of parameters for each model (Bates and Heston). The core idea is then to assess and compare the predictive errors of these models (calibrated on local and external indexes) over future weeks, keeping the forecast horizon consistent. For example, I use the parameters calibrated at week W_0 to predict option prices of weeks W_1 to W_{50} ; I repeat this procedure using the parameters calibrated at W_1 to predict option chain of W_2 until W_{51} and so on. At each forecast horizon, I obtain two groups of prediction errors, one from the model calibrated on the target index and the other from the model calibrated on the external index, both aiming to predict the same index. These paired error sets, corresponding to the same prediction horizon, allow for a meaningful statistical comparison. In particular, I can perform a *Mann–Whitney test* to determine whether the predictive performance differs significantly between local and external calibrations and for which time horizons (time ahead calibration week). If the results indicate no significant difference, this would suggest that calibrating using the target index or a different one does not impact predictive accuracy. Such a result would suggest the hypothesis that the model’s learned structure exhibits robustness or transferability across indices over similar time horizons. A sample of what the error dataset would look like is presented in table 4.10 for a better understanding,

Table 4.10: RMSE dataset architecture matrix Ω_k^* across forecast windows W_k

W_0	W_1	W_2	W_3	...	W_{0+50}
$\Omega_{0,A}^*$	$RMSE_{\Omega_0,1,AB}$	$RMSE_{\Omega_0,2,AB}$	$RMSE_{\Omega_0,3,AB}$	...	$RMSE_{\Omega_0,(0+50),AB}$
$\Omega_{0,B}^*$	$RMSE_{\Omega_0,1,BB}$	$RMSE_{\Omega_0,2,BB}$	$RMSE_{\Omega_0,3,BB}$	...	$RMSE_{\Omega_0,(0+50),BB}$
	$\Omega_{1,A}^*$	$RMSE_{\Omega_1,2,AB}$	$RMSE_{\Omega_1,3,AB}$	...	$RMSE_{\Omega_1,(1+50),AB}$
	$\Omega_{1,B}^*$	$RMSE_{\Omega_1,2,BB}$	$RMSE_{\Omega_1,3,BB}$	...	$RMSE_{\Omega_1,(1+50),BB}$
		$\Omega_{2,A}^*$	$RMSE_{\Omega_2,3,AB}$	...	$RMSE_{\Omega_2,(2+50),AB}$
		$\Omega_{2,B}^*$	$RMSE_{\Omega_2,3,BB}$	...	$RMSE_{\Omega_2,(2+50),BB}$
			$\Omega_{3,A}^*$	...	$RMSE_{\Omega_3,(3+50),AB}$
			$\Omega_{3,B}^*$	...	$RMSE_{\Omega_3,(3+50),BB}$
				...	...
				...	...

N.B.: A and B here represent the two indexes, hence AB means that the model has been calibrated on A to predict B and BB means that the model has been calibrated on B to predict B.

To illustrate how I will test the error with the same time horizon, I will gather all the RMSE that has the same time horizon (relative to the calibration week). For instance, for 1 week of time horizon I can compare ($RMSE_{\Omega_0,1,AB}$, $RMSE_{\Omega_1,2,AB}$, $RMSE_{\Omega_2,3,AB}, \dots$) and ($RMSE_{\Omega_0,1,BB}$, $RMSE_{\Omega_1,2,BB}$, $RMSE_{\Omega_2,3,BB}, \dots$) as they were all calculated at one-week intervals after the calibration week, thus covering a one-week period.

Calibration Performance

QQQ and SPY datasets are both starting from 2021/01/04 to 2022/12/30. Tables 4.12 and 4.11 are the two summary statistics for both index data. These two tables show how wide are the ranges of options properties values of this dataset, then underlying the challenge posed to the Neural Network in terms of calibration.

Table 4.11: Descriptive Statistics of SPY dataset

	Underlying_Last	Strike	C_Price	Time_delta	Moneyness	r
count	478396	478396	478396	478396	478396	478396
mean	417.24	431.74	32.37	0.64	0.99	1.75
std	29.89	79.34	32.26	0.49	0.16	0.72
min	356.58	275.00	0.00	0.00	0.70	0.91
25%	389.62	370.00	2.20	0.25	0.85	1.34
50%	417.35	420.00	21.60	0.52	0.99	1.56
75%	441.52	487.00	58.34	0.87	1.12	1.72
max	477.77	680.00	128.58	1.99	1.29	4.24

Table 4.12: Descriptive Statistics of QQQ dataset

	Underlying_Last	Strike	C_Price	Time_delta	Moneyness	r
count	388460	388460	388460	388460	388460	388460
mean	338.08	351.36	28.22	0.58	0.99	1.82
std	32.38	63.55	26.21	0.46	0.16	0.75
min	260.05	205.00	0.00	0.00	0.70	0.92
25%	316.00	304.00	3.80	0.22	0.85	1.36
50%	335.01	344.00	20.99	0.47	0.98	1.58
75%	363.10	392.00	48.44	0.79	1.12	1.80
max	403.97	575.00	115.83	2.00	1.30	4.25

To calibrate our model on the previous data, I have chosen in this thesis to implement the Differential Evolution algorithm. The parameters settings for the Differential Algorithm are summarized in the table below. They have been initially chosen following Caleiras, M. B. (2024) and Liu et al. (2019a) and empirically adjusted. I set a maximum number of iterations, denoted N_{iter} , to limit the search process in order to optimize computation time. Moreover, when improvements become negligible across five consecutive iterations, the search stopped earlier.

Table 4.13: Differential Evolution algorithm settings

Parameter	Value
Maximum Iteration N_{iter}	50
Early Termination, N_{early}	5
Population size, N	40
Mutation strategy	best
Differential weight, F	0.8
Crossover probability, Cr	0.9

I have chosen the RMSE to assess the loss of our models as it is in the same units of the

data. Hence the loss function that the Differential Algorithm is seeking to optimize is as follows:

$$\Omega_k^* = \arg \min_{\Omega_k} \sqrt{\frac{1}{|\mathcal{C}_k|} \sum_{i=1}^{|\mathcal{C}_k|} (C_{k,i}(\Psi_{k,i}) - F(\Omega_k, \Psi_{k,i}; \theta^*))^2} \quad (4.3)$$

where k represents the calibration week k and i the datapoint (call prices) belonging to the sub-dataset $\mathcal{C}_k$ indexed by week; with $\mathcal{C}_k \subset \mathcal{C}$, with $\mathcal{C}$ the whole option chain dataset. $\Psi_{k,i}$ is the option properties of week k , number i . Moreover, F represents the Neural Network pricing function.

Therefore I have calibrated models over 30 weeks, for each datasets. It led me to obtain 30 sets of parameters for each model and datasets, as well as their relative loss values (on calibrated week k), which assess models ability to fit to the data for each week. The architecture of the calibrated parameters dataset is presented in table 4.14 :

Table 4.14: Architecture of the calibrated parameters dataset

Week	Ω_k^*	RMSE
0	Ω_0^*	$\text{RMSE}_{\Omega_0^*}$
1	Ω_1^*	$\text{RMSE}_{\Omega_1^*}$
2	Ω_2^*	$\text{RMSE}_{\Omega_2^*}$
$\vdots$	$\vdots$	$\vdots$
29	Ω_{29}^*	$\text{RMSE}_{\Omega_{29}^*}$

Note that this thesis considers two models and two indices. Therefore, the process described above will be repeated four times to build four different files. Specifically, I will calibrate the Bates and Heston models using data from both QQQ and SPY. As a result, there will be four files, each containing the calibrated parameters for one of the two models applied to one of the two indices.

To assess how well our model fits the data, I choose to compare the relative error measured using the Root Mean Squared Error (RMSE) during the calibration week to the average of the option prices used for calibration in that same week. Finally, I computed the mean of these relative errors to obtain a single representative figure. The results of these relative errors are presented in the table 4.15.

Table 4.15: Average relative errors over 30 weeks of calibration

Model	QQQ	SPY
Bates	3.75%	4.85%
Heston	3.21%	3.16%

The results indicate that, over the 30-week calibration period, the Heston model achieved lower average relative errors than the Bates model for both datasets (QQQ and SPY). Specifically, the Heston model outperformed the Bates model by approximately 1% on average, suggesting a better fit to the observed market data. With average errors around 3.5% for both indices, the calibration quality can be considered satisfactory, indicating that both models capture the main dynamics of the option chains during the studied period, although the Heston model appears more efficient in this context.

Error Prediction

From Table 4.14, I am now able to compute forecasting errors. Specifically, for each set of calibrated parameters Ω_k^* , I can evaluate the model's performance in predicting option chains for the subsequent weeks. In this thesis, I assess the prediction accuracy over a 50-week horizon following each calibration week. For example, I begin using calibrated parameters from week $k = 0$, yielding the parameters Ω_0^* . I can then compute the errors made when predicting option chains for week $k = 1$ (one week intervals), $k = 2$ (two weeks intervals), and so on, up to $k = 50$ (fifty weeks intervals). Next, I repeat the process using parameters calibrated on week $k = 1$ i.e. Ω_1^* , and calculate the errors made by the models when predicting option chain of week $k = 2$ (one week intervals) to $k = 51$ (fifty weeks intervals). This procedure is repeated for every available calibrated week.

By applying this methodology to each model and for both indices, I construct the dataset described in Figure 4.10. I finally obtain two such datasets, one per model: each contain two types of error, one when the model is calibrated and evaluated on the same index, and one when it is calibrated on one index and evaluated on the other.

This configuration allows for detailed analysis of cross-index generalization, as well as the effectiveness of the process in terms of calibration accuracy on. By comparing the prediction errors over time horizons (relative to the calibration week), I can determine whether a model trained on one index can effectively forecast another. This allows us to identify which models are best suited for such generalization, and over which time frames this transferability remains possible. An example of such dataset is presented by table 4.16.

This table allow to gather RMSE's (i.e. RMSE with the same prediction week). Consequently, it allows us to compare those RMSE's with the same time horizon, through a *Mann-Whitney test*, Loss QQQ (QQQ) with Loss QQQ (SPY) and Loss SPY (SPY) with Loss SPY (QQQ). Hence for each model, I will be able to know wether a transfer of parameters is achievable and for which time horizon. Moreover, given that the RMSE distributions did not pass normality tests, the *Mann-Whitney U* test was used as the primary method for assessing differences between the two calibration approaches. However, for robustness, a *t-test* was also conducted to compare the means and reinforce conclusions in case of agreements between the two.

Table 4.16: Prediction Error Dataset Architecture for one model

Calibrated Week	Prediction Week	Loss QQQ (QQQ)	Loss QQQ (SPY)
0	1	$\text{RMSE}_{\Omega_{0,1}}^{\text{QQQ} \rightarrow \text{QQQ}}$	$\text{RMSE}_{\Omega_{0,1}}^{\text{SPY} \rightarrow \text{QQQ}}$
0	2	$\text{RMSE}_{\Omega_{0,2}}^{\text{QQQ} \rightarrow \text{QQQ}}$	$\text{RMSE}_{\Omega_{0,2}}^{\text{SPY} \rightarrow \text{QQQ}}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Calibrated Week	Prediction Week	Loss SPY (QQQ)	Loss SPY (SPY)
0	1	$\text{RMSE}_{\Omega_{0,1}}^{\text{QQQ} \rightarrow \text{SPY}}$	$\text{RMSE}_{\Omega_{0,1}}^{\text{SPY} \rightarrow \text{SPY}}$
0	2	$\text{RMSE}_{\Omega_{0,2}}^{\text{QQQ} \rightarrow \text{SPY}}$	$\text{RMSE}_{\Omega_{0,2}}^{\text{SPY} \rightarrow \text{SPY}}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Calibrated Week	Prediction Week	Loss QQQ (QQQ)	Loss QQQ (SPY)
1	1	$\text{RMSE}_{\Omega_{1,1}}^{\text{QQQ} \rightarrow \text{QQQ}}$	$\text{RMSE}_{\Omega_{1,1}}^{\text{SPY} \rightarrow \text{QQQ}}$
1	2	$\text{RMSE}_{\Omega_{1,2}}^{\text{QQQ} \rightarrow \text{QQQ}}$	$\text{RMSE}_{\Omega_{1,2}}^{\text{SPY} \rightarrow \text{QQQ}}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Calibrated Week	Prediction Week	Loss SPY (QQQ)	Loss SPY (SPY)
1	1	$\text{RMSE}_{\Omega_{1,1}}^{\text{QQQ} \rightarrow \text{SPY}}$	$\text{RMSE}_{\Omega_{1,1}}^{\text{SPY} \rightarrow \text{SPY}}$
1	2	$\text{RMSE}_{\Omega_{1,2}}^{\text{QQQ} \rightarrow \text{SPY}}$	$\text{RMSE}_{\Omega_{1,2}}^{\text{SPY} \rightarrow \text{SPY}}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Calibrated Week	Prediction Week	Loss QQQ (QQQ)	Loss QQQ (SPY)
$\vdots$	$\vdots$	$\vdots$	$\vdots$
30	49	$\text{RMSE}_{\Omega_{30,49}}^{\text{QQQ} \rightarrow \text{QQQ}}$	$\text{RMSE}_{\Omega_{30,49}}^{\text{SPY} \rightarrow \text{QQQ}}$
30	50	$\text{RMSE}_{\Omega_{30,50}}^{\text{QQQ} \rightarrow \text{QQQ}}$	$\text{RMSE}_{\Omega_{30,50}}^{\text{SPY} \rightarrow \text{QQQ}}$
Calibrated Week	Prediction Week	Loss SPY (QQQ)	Loss SPY (SPY)
$\vdots$	$\vdots$	$\vdots$	$\vdots$
30	49	$\text{RMSE}_{\Omega_{30,49}}^{\text{QQQ} \rightarrow \text{SPY}}$	$\text{RMSE}_{\Omega_{30,49}}^{\text{SPY} \rightarrow \text{SPY}}$
30	50	$\text{RMSE}_{\Omega_{30,50}}^{\text{QQQ} \rightarrow \text{SPY}}$	$\text{RMSE}_{\Omega_{30,50}}^{\text{SPY} \rightarrow \text{SPY}}$

5. Results

5.1 Same Index Calibration Performance

To assess the predictive performance of each model on its own index, I use the dataset described in Table 4.16. Specifically, I use the RMSEs obtained from predicting the option chain of an index using parameters calibrated on that same index. Then the RMSEs are sorted by prediction horizon, allowing an analysis of both the overall model performance and the distribution of errors, which provides insight into the stability of the models across different time horizons.

5.1.1 Predicting S&P using S&P

Figures 5.1 and 5.3 display candlestick plots of prediction losses (RMSEs) across time horizons for Bates and Heston models. These figures provide an assessment of the inherent predictive performance of each model when both calibration and prediction are performed on the same index. The candlestick representation was chosen because it highlights not only the central tendency (median) of the errors but also their dispersion, thereby offering a clearer picture of error distribution by time horizon. Figures 5.2 and 5.4 allow to zoom-in into figures 5.1 and 5.3 to investigate whether models present errors instability across time horizon. These figures show the interquartile range (IQR) by time horizon for both models (Bates and Heston). I chose to plot the 90th percentile of the IQR values, which I consider as the threshold to determine whether instability can be observed across specific time horizons.

Bates model

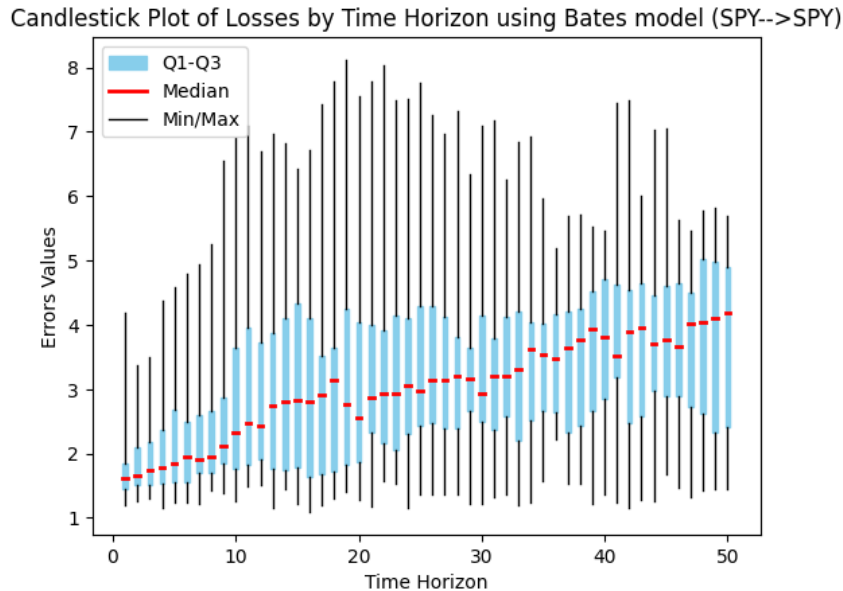


Figure 5.1: Candlestick Plot of Losses by Time Horizon using Bates model (SPY-->SPY)

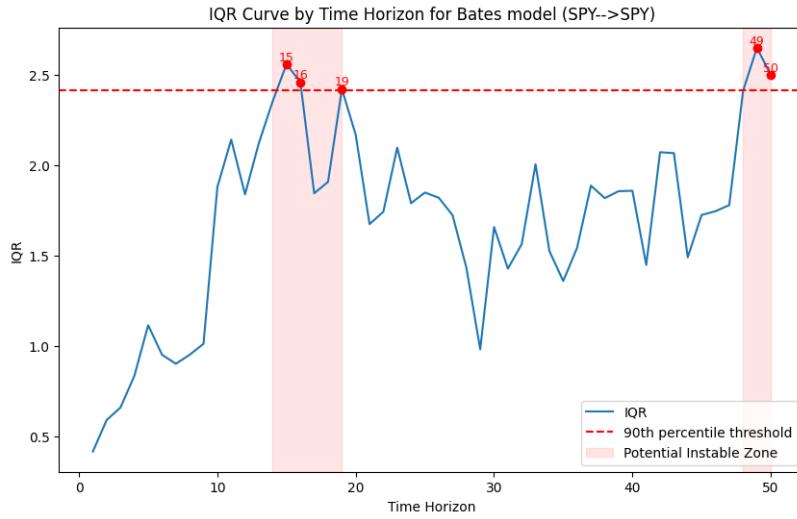


Figure 5.2: IQR Curve by Time Horizon for Bates model (SPY-->SPY)

Figure 5.1 shows that median of errors increase with time horizons. This results was expected as uncertainty becomes larger with longer forecast horizons. Moreover, RMSEs median lies between [1.5:4] which emphasizes strong calibration performance overall but especially for the first 10 weeks where the errors values remain under 2. However, figure 5.1 also highlights periods of high variability in prediction errors, particularly between horizons 10 and 20 and again between horizons 49 and 50. In these intervals, the candlesticks are larger especially in terms of interquartile ranges (blue candles). This suggests that the model presents greater instability during these time spans. These observations are quantitatively supported by Figure 5.2, which shows that horizons 15, 16, 19, 49, and 50 lie above the 90th percentile of IQR values. Such horizons, highlighted in red, can therefore be considered statistically unstable zones where model performance becomes less reliable.

Heston model

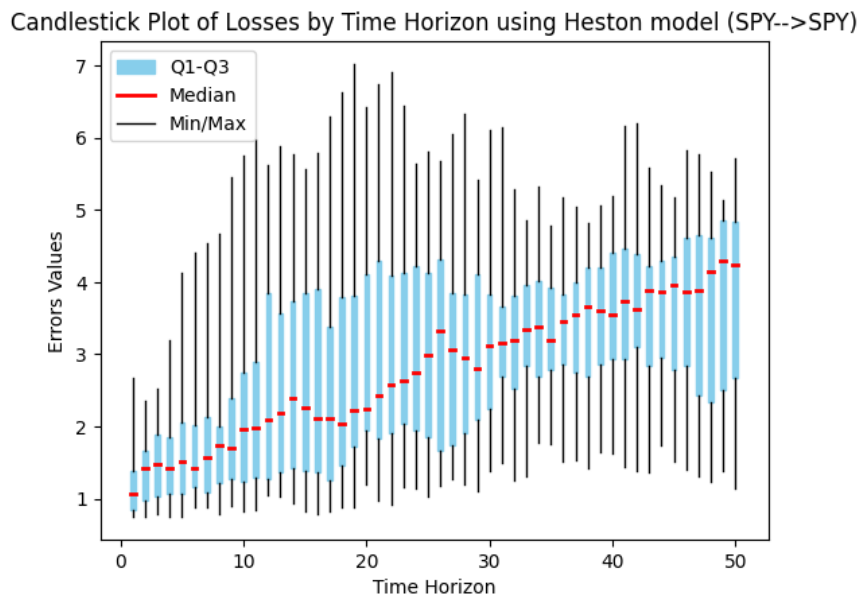


Figure 5.3: Candlestick Plot of Losses by Time Horizon using Heston model (SPY-->SPY)

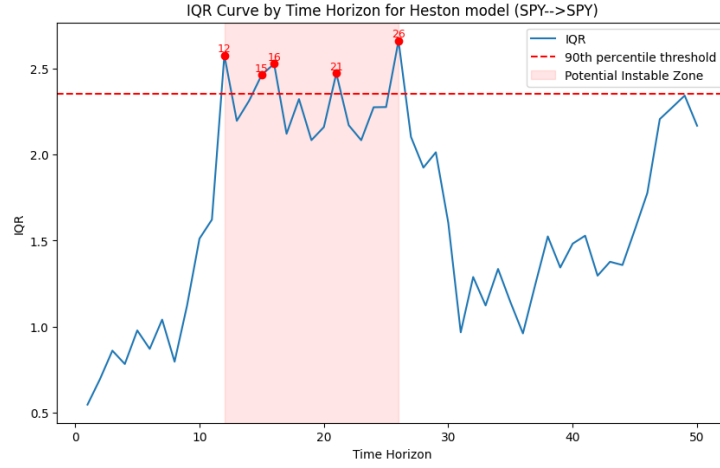


Figure 5.4: IQR Curve by Time Horizon for Heston model (SPY-->SPY)

Similarly figure 5.3 illustrates that the median error increases with the time horizon, remaining within the range [1:4]. Moreover, figure 5.3 highlights periods of unusual high variability in prediction errors particularly in accordance with those spotted in figure 5.1. Indeed, we can observe high dispersion of errors values between horizons 10 and 30 and between horizons 40 and 50. Figure 5.4 support previous observations showing that IQR of horizons 12, 15, 16, 21 and 26 are above the 90th percentile of IQR values, which again suggest potential instable zone. Both figures 5.2 and 5.4 are in accordance to consider that models present greater instability during time horizons lying around 12 to 26. Previous results put at stake the reliability of predicted results by the models for these time horizons.

5.1.2 Predicting Nasdaq using Nasdaq

The same process as described in Section 5.1 has been performed to derived figures 5.5 and 5.6.

Bates model

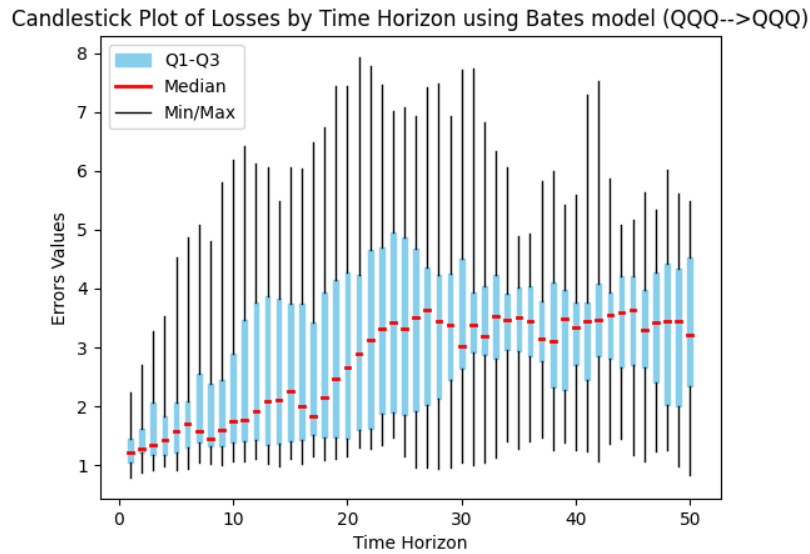


Figure 5.5: Candlestick Plot of Losses by Time Horizon using Bates model (QQQ-->QQQ)

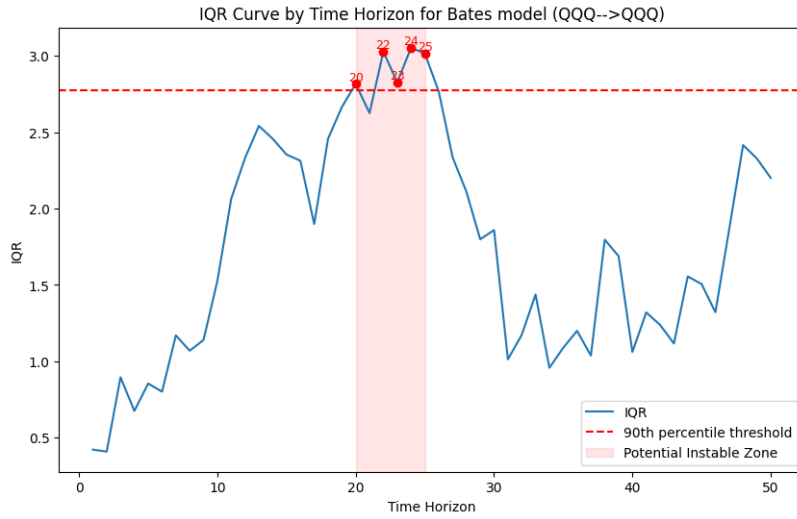


Figure 5.6: IQR Curve by Time Horizon for Bates model (QQQ-->QQQ)

Figure 5.5 shows that the median error increases with the time horizon. The curve appears steeper than those in section 6.1.2, particularly between horizons 18 and 30. In contrast, the median range for the Nasdaq dataset is smaller than in section 5.1.1, lying between 1.2 and 3.1, emphasizing on overall stronger calibration performance. I can remark that calibration performance are particularly good for the first 15 weeks, where errors values remains under 2. Moreover the graph reveals larger dispersion of prediction errors between horizons 10 and 25, both in terms of interquartile ranges and overall min–max ranges. This statement is further supported and precised by Figure 5.6, which indicates that horizons 20 and 22–25 exceed the 90th percentile of IQR values. These findings highlight localized zones of model instability, thereby raising concerns about the reliability of predictions in these horizons.

Heston model

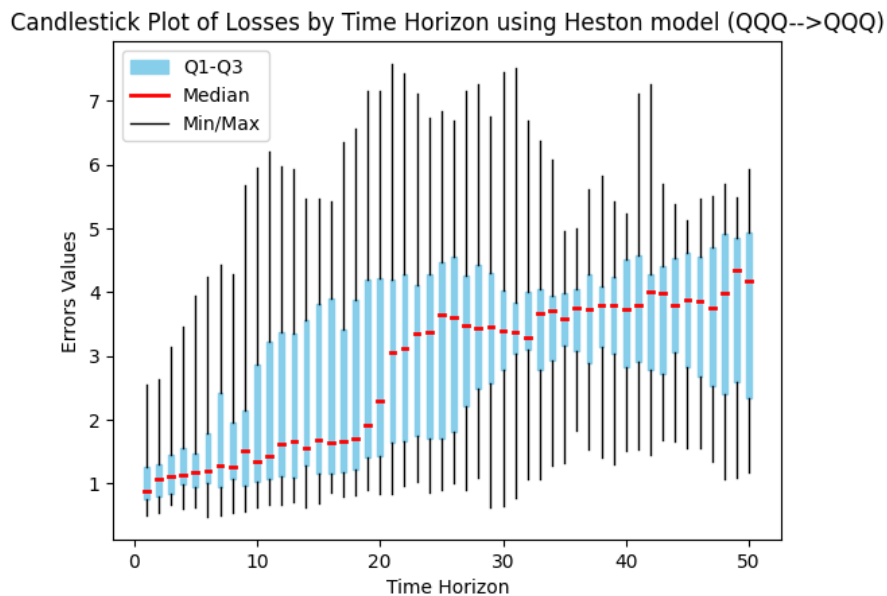


Figure 5.7: Candlestick Plot of Losses by Time Horizon using Heston model (QQQ-->QQQ)

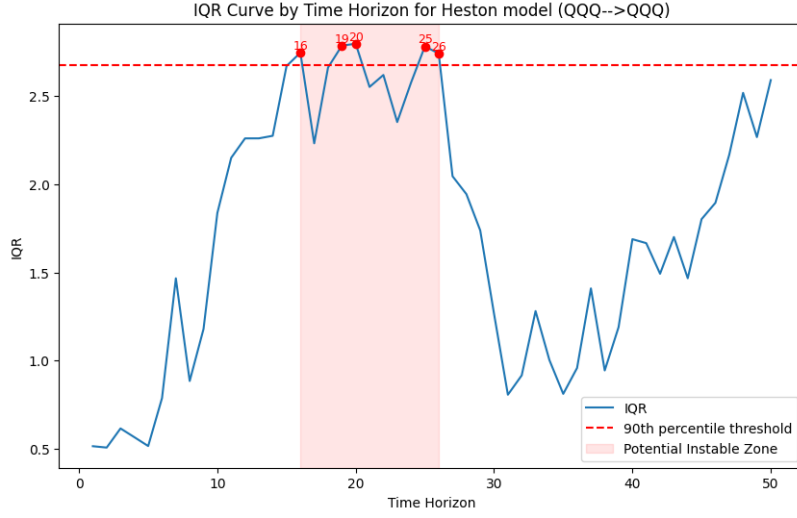


Figure 5.8: IQR Curve by Time Horizon for Heston model (QQQ→QQQ)

Similarly for Heston model, figure 5.7 presents an increasing median error curve, ranging between 0.9 to 4. The figure suggests that model presents good performance for the first 20 weeks as the errors remain relatively stable and under 2. However, the median curve shows instability around time horizon 20, where the curve become discontinuous jumping from 1.8 to 3.5. The instabilities are supported with high dispersion in terms of IQR but also Min/Max candles for time horizons ranging between 10 to 25 to almost 30. The time span of potential instable zones seems more extended according to the Heston model rather than for the Bates model. Indeed, results presented in figure 5.8 are in accordance with those presented in 5.6, pointing out high dispersion above 90th percentile for time horizons between 16 to 26 which put into play the reliability of models predictions for the corresponding time horizons.

5.2 Cross Index Calibration Performance

The following tables 5.1, 5.2, 5.3, and 5.4 summarize the results of the hypothesis tests presented below:

$$\begin{aligned}
 H_0 : \mu_{\text{cross}} &= \mu_{\text{same}} && \text{(no difference in prediction errors)} \\
 H_1 : \mu_{\text{cross}} &\neq \mu_{\text{same}} && \text{(difference in prediction errors exists)}
 \end{aligned}$$

In both calibration–prediction directions (predicting QQQ using SPY-calibrated parameters and vice versa), the absence of evidence to reject the null hypothesis implies that cross-index generalization may be achievable for the considered time horizons after calibration. However, this does not imply that predicting one index using parameters calibrated on another one yields superior results. Indeed, it means that, within these time horizons, the prediction errors from cross-index calibration are not statistically different from those obtained through same-index calibration. The test results indicate statistical equivalence in errors, not necessarily better predictive performance.

Moreover, figures 5.9, 5.11, 5.13 and 5.15 and figures 5.10, 5.12, 5.14 and 5.16 have been generated following the process described in chapter 5. These figures will help to track cross-index calibration performance over time horizons.

5.2.1 Predicting Nasdaq using S&P

Figures 5.9 and 5.11 suggest that models present a great calibration performance during the first 30 weeks after calibration, which attest of a substantial robustness of the calibrated parameters. However, although the median remains relatively stable for the first 30 weeks of time horizon, the errors values remains higher than those presented in figures 5.5 and 5.7, which have been generated from prediction of the Nasdaq using parameters calibrated on the same index. After 30 weeks, we can see that the prediction errors is almost discontinuous, jumping from 4 to 6.5 using Bates model and from 3 to 6 using Heston model, suggesting a greater instability after the 30th weeks. This instability is further emphasized by figures 5.10 and 5.12 both suggesting instable zones from the 30th weeks after calibration.

Bates



Figure 5.9: Candlestick Plot of Losses by Time Horizon using Bates model (SPY->QQQ)

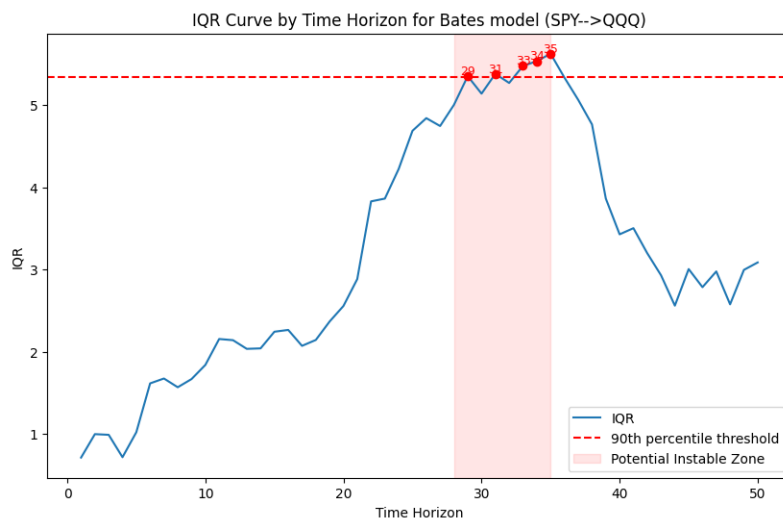


Figure 5.10: IQR Curve by Time Horizon for Bates model (SPY->QQQ)

Candlestick Plot of Losses by Time Horizon using Heston model (SPY-->QQQ)

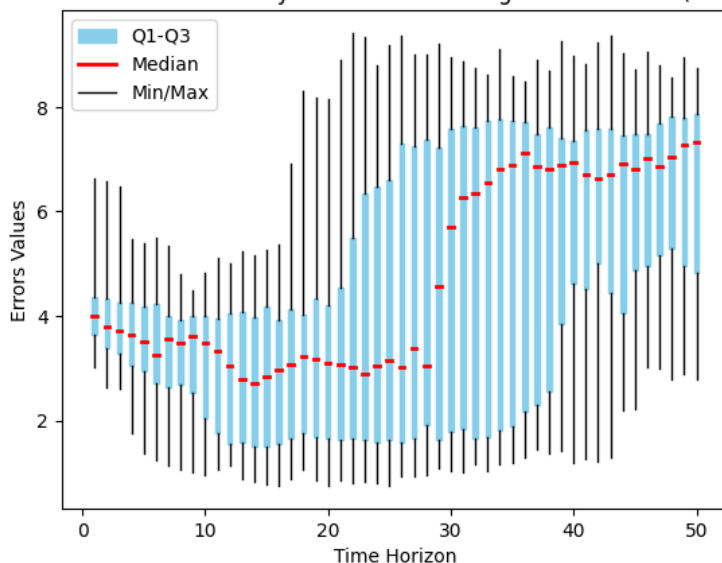


Figure 5.11: Candlestick Plot of Losses by Time Horizon using Heston model (SPY-->QQQ)

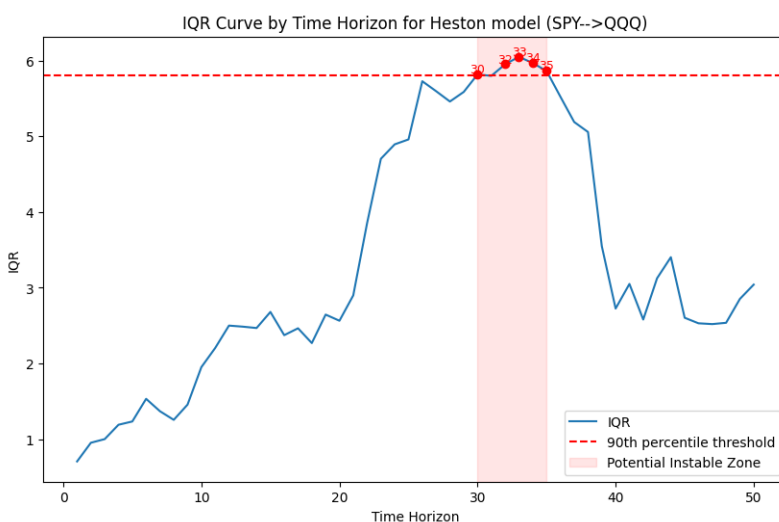


Figure 5.12: IQR Curve by Time Horizon for Heston model (SPY-->QQQ)

Statistics Results

Following tables 5.1 and 5.2 present the results for both models across each time horizon. The last column provides a summary indicating time horizons where both models are in agreement.

By taking the intersection of the results of both test presented in Table 5.1 and 5.2, results show that for time horizons ranging between [1:10] and [33:50] there is a statistics difference in the mean of the errors, which allow to reject the null hypothesis. Consequently, for this dataset and the period of study a cross-index generalization is not achievable via Neural Networks for these time horizons. However, the results indicate that, for time horizons between the 13th and 27th weeks, there is no evidence to reject the null hypothesis. In other words, within this range, cross-index generalization cannot be statistically ruled out.

Table 5.1: Comparison of Heston and Bates models for predicting QQQ using parameters calibrated on SPY (*Mann–Whitney test*)

Heston		Bates		Result intersection ($\cap$)	
Time Horizon	SPY→QQQ	Time Horizon	SPY→QQQ	Time Horizon	Result
[1; 12]	Rejected	[1; 11]	Rejected	[1; 11]	Rejected
[12; 33]	Not Rejected	[12; 32]	Not Rejected	[12; 32]	Not Rejected
[34; 50]	Rejected	[33; 50]	Rejected	[33; 50]	Rejected

Table 5.2: Comparison of Heston and Bates models for predicting QQQ using parameters calibrated on SPY (*t-test*)

Heston		Bates		Result intersection ($\cap$)	
Time Horizon	SPY→QQQ	Time Horizon	SPY→QQQ	Time Horizon	Result
[1; 12]	Rejected	[1; 10]	Rejected	[1; 10]	Rejected
[13; 27]	Not Rejected	[11; 27]	Not Rejected	[13; 27]	Not Rejected
[28; 50]	Rejected	[28; 50]	Rejected	[28; 50]	Rejected

This suggests that QQQ option chain data could be predicted using parameters calibrated on SPY, given that the prediction horizon from the calibration lies between the 13th to the 27th week.

5.2.2 Predicting S&P using Nasdaq

Following figures 5.13 and 5.15 suggest that the median is decreasing over time horizons, which is an unexpected result. Indeed, I would have expect the median to increase over time horizons as longer time horizons come with greater uncertainty which tend to degrade model performance. However, we can see on both figures that errors values are relatively high for short term horizons, almost at the same level as of figure 5.9 and 5.11 after 30 weeks after calibration. This unexpected result could came from a bias in the data and particularly regarding the period of study. However, although I observe a decrease in the median of error value over time horizons, figures 5.10 and 5.12 show that greater instability arise for mid to long term horizons, but especially around 20 weeks to 25 weeks according to both model. This results suggest that despite the median is decreasing over time horizons (better accuracy) the model tend to become less reliable as well.

Candlestick Plot of Losses by Time Horizon using Bates model (QQQ-->SPY)

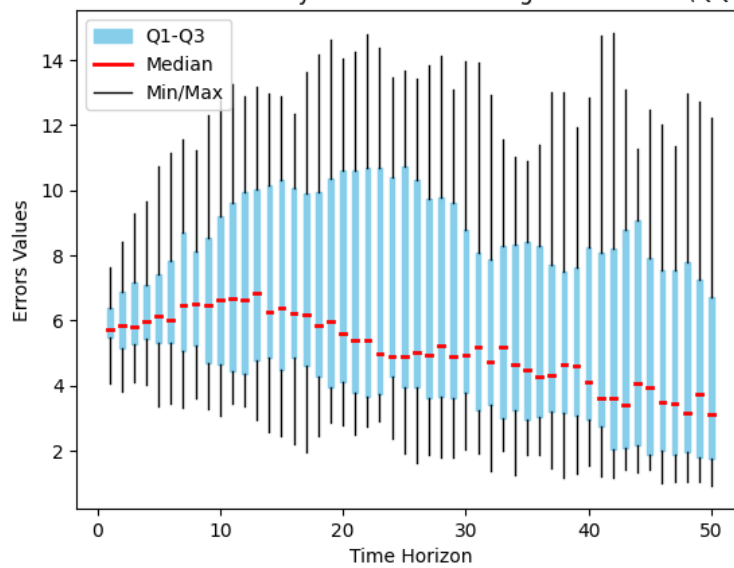


Figure 5.13: Candlestick Plot of Losses by Time Horizon using Bates model (QQQ-->SPY)

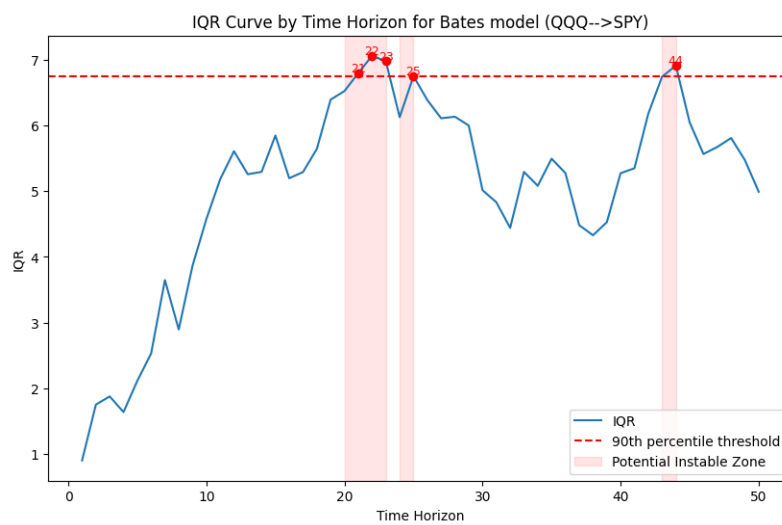


Figure 5.14: IQR Curve by Time Horizon for Bates model (QQQ-->SPY)

Candlestick Plot of Losses by Time Horizon using Heston model (QQQ-->SPY)

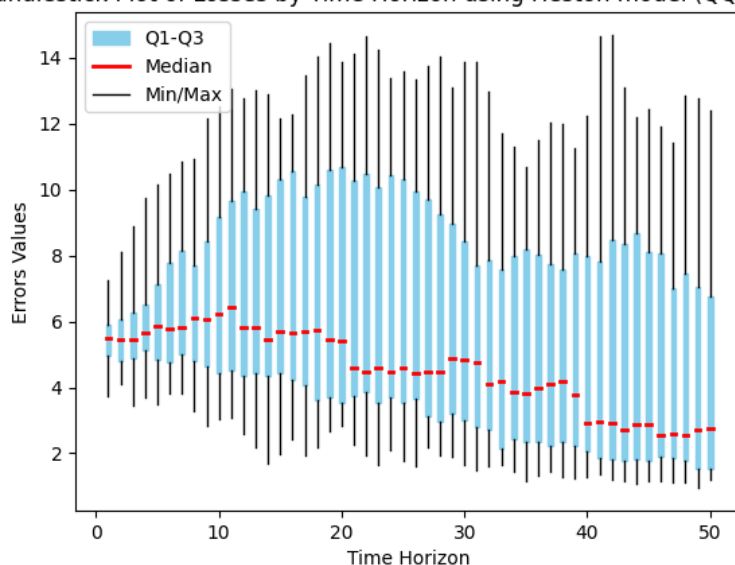


Figure 5.15: Candlestick Plot of Losses by Time Horizon using Heston model (QQQ-->SPY)

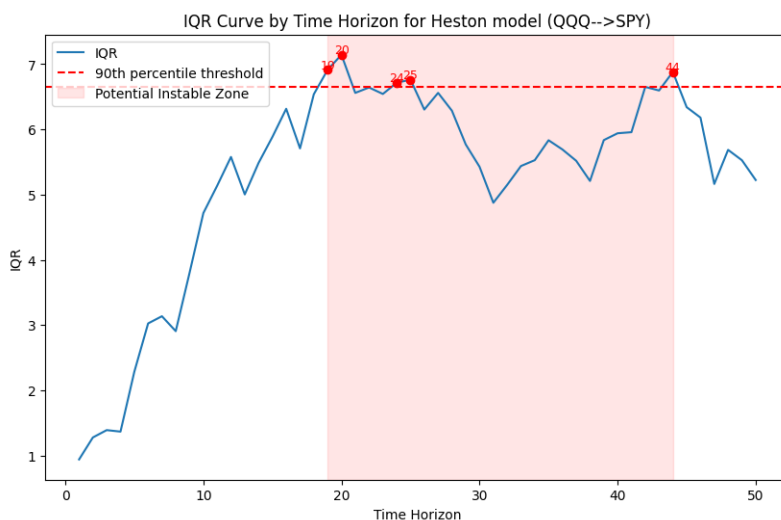


Figure 5.16: IQR Curve by Time Horizon for Heston model (QQQ-->SPY)

Statistics Results

Similarly, following tables 5.3 and 5.4 present the results for both models across each time horizon. The last column provides a summary indicating time horizons where both models are in agreement. Similarly here, by taking the intersection of the results of both test presented in table 5.3 and 5.4, results show that for a time horizon ranging between the 45th week to the 50th week there is no evidence of rejecting the null hypothesis, hence no evidence to exclude a potential cross index generalization for these time horizons (from calibration). However, results demonstrated that a cross index generalization is not achievable for time horizon ranging from 0 to the 32nd week. To conclude, it seems possible to predict SPY data using QQQ, for time horizon ranging between the 45th weeks to the 50th.

Table 5.3: Comparison of Heston and Bates models for predicting SPY using parameters calibrated on QQQ (*Mann–Whitney test*)

Heston		Bates		Result intersection ($\cap$)	
Time Horizon	QQQ→SPY	Time Horizon	QQQ→SPY	Time Horizon	Result
[0; 32]	Rejected	[0; 39]	Rejected	[0; 32]	Rejected
[33; 50]	Not Rejected	[40; 50]	Not Rejected	[40; 50]	Not Rejected

Table 5.4: Comparison of Heston and Bates models for predicting SPY using parameters calibrated on QQQ (*t-test*)

Heston		Bates		Result intersection ($\cap$)	
Time Horizon	QQQ→SPY	Time Horizon	QQQ→SPY	Time Horizon	Result
[0; 40]	Rejected	[0; 44]	Rejected	[0; 40]	Rejected
[41; 50]	Not Rejected	[45; 50]	Not Rejected	[45; 50]	Not Rejected

5.3 Discussions

Discussion of the methodology

Although the construction of the dataset has been done methodologically in this thesis, it potentially could be improved. Indeed, despite I have carefully used the Monte Carlo method to complete my dataset, this work could be further extended using other pricing method such as the COS method demonstrated by Fang and Oosterlee (2009) or the FFT method originally developed by Cooley and Tukey (1965). This would have two major benefits : the first one is that it would provide more accurate prices rather an optimized Monte Carlo method; the second one is that it will improve the rapidity of the calculation process and consequently allow for a larger training dataset. Moreover, it would be worthwhile to conduct this investigation over a longer period of study. A larger dataset would help to reduce errors specific to the sample, consequently improving the reliability of the results. Moreover, although the combination of Neural Networks and a global optimizer such as the Differential Algorithm are considered as a really performing calibration set according to the literature, one could try out another set up perhaps employing other global optimizer such as the Particle Swarm optimizer (Kennedy, J. and Eberhart, R. (1995) and Wang, D., Tan, D., and Liu, L. (2018)) or Simulated Annealing (Kirkpatrick, S. (1984)) that potentially would improve the performance and lead to other findings.

Discussion of the results

One of the strongest outcome arising from the cross-index calibration results is that both test (Mann–Whitney test and t-test) presented in tables 5.1, 5.2, 5.3 and 5.4 are closely agree with each other, as they are really similar across the models and the direction of calibration-prediction. This makes me confident both in terms of data processing and methods. However, I distinguish two explanations regarding the findings. First of all, we could simply conclude that a parameter transfer is achievable for some time horizons and calibration-prediction directions according to the tables 5.1, 5.2, 5.3 and 5.4. Nonetheless, these findings could also extrapolate the inherent limits of each model. Indeed, in that case, the results would show that the errors are effectively close to each other for some time horizons, but not because of a potential cross-index generalization but rather because models reach their predictive limits for these time horizons. This remark questions whether a transfer of parameters is possible or not for the time horizons where the null is not rejected. However results observed in figures 5.6, 5.8 suggest model instability for time

horizons around the 15th to the 25th week. These results put at stake the results observed in section 5.2.1 stating there were no evidence of ruling out a cross-index generalization for time horizon between the 13th and 27th weeks. Although in section 5.2.1 I did not reject the null hypothesis, results in section 5.1.2 suggest instability for the same time horizons which explain why cross-index and same index errors have been statistically considered not different. This is less obvious for the results presented in section 5.1.1 and 5.2.2, as figures 5.2 and 5.4 are closely agree with each other on the instability of the predicted results for time horizons ranging roughly around 15 to 20. However, results presented in tables 5.3 and 5.4 suggest that there is no evidence of rejecting the performance of a cross-index generalization for time horizons between 45 to 50, which do not coincide with the potential instable zone suggested in section 5.1.1. Consequently these observations leaves room for doubt for a potential cross-index generalization for time horizons between 45 to 50.

Nonetheless, section 5.1 have showed that same index calibration performance was very good for the first 10 weeks when predicting the S&P 500 and for the first 15-20 weeks when predicting the the Nasdaq, which is really useful for short term pricing. Indeed, beyond 10 weeks (when predicting the S&P 500) or 15-20 weeks (when predicting the Nasdaq) the 2 step framework presented in this thesis become less accurate. Therefore, results for same index calibration using this process are really promising for short term pricing. However, previous work of cross index calibration have shown promising results as well, especially for the direction S&P 500 to Nasdaq. Indeed, despite errors are statistically different for short term horizons as shown in tables 5.1 and 5.2, figures 5.9 and 5.11 have shown that the medians of the errors were quiet small in comparison for those in the Nasdaq to S&P calibration-prediction direction (figures 5.13 and 5.15) which show that using the S&P to predict the Nasdaq instead of the Nasdaq itself is still possible for short term horizons depending on the level of accuracy required.

To conclude and close this discussion, I fairly think that there are no specific points to enhance this work enabling to derive better results. Indeed, as mentioned earlier I believe that improvements of this work lies in an overall improvements of the details.

6. Conclusion

This thesis seeks to evaluate the feasibility of a cross-index generalization in the calibration of parametric option pricing models using a neural network-based approach. By replacing traditional pricing method with a pre-trained neural network, combined with a Differential Evolution optimizer allow to speed up the process at no extra cost as it preserves high calibration accuracy.

Applying this two-step framework to the Heston and Bates models, and using two years of option data from the S&P 500 and Nasdaq-100, I found that parameters calibrated on the S&P 500 can statistically be used to predict Nasdaq option chains for horizons ranging between the 13th and 27th weeks ahead of calibration week. However, further analysis (Section 5.1.2) revealed that this potential transferability is more likely the result of model instability within these horizons rather than evidence of genuine cross-index generalization. Conversely, using Nasdaq-calibrated parameters to predict the S&P 500 was statistically plausible only for the 45th to 50th weeks ahead of calibration week. Investigation of calibration performance over Nasdaq option in section 5.1.1 has not shown any contradiction of this result, leaving room for doubt as to whether cross-index generalization might occasionally hold in practice. These results were consistent across both models, indicating that cross-index generalization may be possible under very specific conditions.

The findings have multiple implications. On the one hand, under favorable circumstances, calibrating on a single index may suffice to price another, potentially reducing the number of calibrations, lowering computational costs, and enabling faster model implementation. On the other hand, this study also highlights important limitations: the evidence for cross-index generalization remains unsuccessful, and the results may reflect the inherent predictive limits and instabilities of the models rather than a transferable calibration property.

Future work could expand the analysis to different framework involving alternative neural architectures (e.g. recurrent models) and global optimizer (e.g. Simulated Annealing, Particle Swarm). Moreover, using a broader and larger set of assets, would contribute to explore potential other cross-asset generalization.

Appendix

Ethical approval letter

Bibliography

- [1] Amin, K., & Jarrow, R. (1992) “Pricing options on risky assets in a stochastic interest rate economy”. *Mathematical Finance*, 2(3), 217–237. Available at: <https://doi.org/10.1111/j.1467-9965.1992.tb00030.x>
- [2] Anders, U., Korn, O., & Schmitt, C. (1998) “Improving the pricing of options: A neural network approach”. *Journal of Forecasting*, 17(5–6), 369–388. Available at: [https://doi.org/10.1002/\(SICI\)1099-131X\(199809\)17:5/6<369::AID-FOR702>3.0.CO;2-S](https://doi.org/10.1002/(SICI)1099-131X(199809)17:5/6<369::AID-FOR702>3.0.CO;2-S)
- [3] Arıcı, F., & Kaya, E. (2019) “Comparison of meta-heuristic algorithms on benchmark functions”. *Academic Perspective Procedia*, 2(3), 508–517. Available at: <https://www.acperpro.com/document/ISITES2019ID41>
- [4] Bakshi, G., Cao, C., & Chen, Z. (1997) “Empirical performance of alternative option pricing models”. *The Journal of Finance*, 52(5), 2003–2049. Available at: <https://doi.org/10.1111/j.1540-6261.1997.tb02749.x>
- [5] Bates, D. (1991) “The crash of 87: Was it expected? The evidence from options markets”. *Journal of Finance*, 46(3), 1009–1044. Available at: <https://doi.org/10.1111/j.1540-6261.1991.tb03775.x>
- [6] Bates, D. (1996) “Jumps and stochastic volatility: Exchange rate processes implicit in Deutschmark options”. *Review of Financial Studies*, 9(1), 69–108. Available at: <https://doi.org/10.1093/rfs/9.1.69>
- [7] Bayer, C., Horvath, B., Muguruza, A., Stemper, B., & Tomas, M. (2019) “On deep calibration of (rough) stochastic volatility models”. *arXiv preprint*. Available at: <https://arxiv.org/abs/1908.08806>
- [8] Black, F., & Scholes, M. (1973) “The pricing of options and corporate liabilities”. *Journal of Political Economy*, 81(3), 637–654. Available at: <https://doi.org/10.1086/260062>
- [9] Caleiras, M. B. (2024) “Calibrating option pricing models using neural networks and population-based optimization methods”. Available at: https://eg-fr.uc.pt/retrieve/275278/Thesis_Margarida_Biscaia.pdf
- [10] Carleton, W. T., & Lakonishok, J. (1985) “Risk and return on equity: The use and misuse of historical estimates”. *Financial Analysts Journal*, 41(1), 38–47. Available at: <https://doi.org/10.2469/faj.v41.n1.38>
- [11] Carr, P., & Madan, D. (1999) “Option valuation using the fast Fourier transform”. *Journal of Computational Finance*, 2(4), 61–73. Available at: https://www.researchgate.net/publication/2519144_Option_Valuation_Using_the_Fast_Fourier_Transform
- [12] Cooley, J. W., & Tukey, J. W. (1965) “An algorithm for the machine calculation of complex Fourier series”. *Mathematics of Computation*, 19(90), 297–301. Available at: <https://doi.org/10.2307/2003354>
- [13] Cont, R., & Da Fonseca, J. (2002) “Dynamics of implied volatility surfaces”. *Quantitative Finance*, 2(1), 45–60. Available at: <https://doi.org/10.1088/1469-7688/2/1/304>

- [14] Dimitroff, G., Roeder, D., & Fries, C. P. (2018) “Volatility model calibration with convolutional neural networks”. *Available at SSRN* 3252432. Available at: <https://doi.org/10.2139/ssrn.3252432>
- [15] Dogo, E. M., Afolabi, O. J., Nwulu, N. I., Twala, B., & Aigbavboa, C. O. (2018, December) “A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks”. In *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)* (pp. 92–99). IEEE. Available at: <https://ieeexplore.ieee.org/document/8769211>
- [16] Dupire, B. (1994) “Pricing with a smile”. *Risk*, 7(1), 18–20.
- [17] Fama, E. F., & Bliss, R. R. (1987) “The information in long-maturity forward rates”. *The American Economic Review*, 77(4), 680–692.
- [18] Fang, F., & Oosterlee, C. W. (2009) “A novel pricing method for European options based on Fourier-cosine series expansions”. *SIAM Journal on Scientific Computing*, 31(2), 826–848. Available at: <https://doi.org/10.1137/080718061>
- [19] Fournié, E., Lasry, J. M., & Touzi, N. (1997) “Monte Carlo methods for stochastic volatility models”. In *Numerical Methods in Finance* (pp. 146–164). Cambridge University Press.
- [20] Gilli, M., & Schumann, E. (2012) “Calibrating option pricing models with heuristics”. In *Natural Computing in Computational Finance* (pp. 9–37). Springer. Available at: https://link.springer.com/chapter/10.1007/978-3-642-23336-4_2
- [21] Grzelak, L. A., Jablecki, J., & Gatarek, D. (2022) “Efficient pricing and calibration of high-dimensional basket options”. *International Journal of Computer Mathematics*, 101(8), 865–888. Available at: <https://doi.org/10.1080/00207160.2022.2058616>
- [22] Hernandez, A. (2016) “Model calibration with neural networks”. *Available at SSRN* 2812140. Available at: <https://doi.org/10.2139/ssrn.2812140>
- [23] 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. Available at: <https://doi.org/10.1093/rfs/6.2.327>
- [24] Horvath, B., Muguruza, A., & Tomas, M. (2019) “Deep learning volatility: A deep neural network perspective on pricing and calibration in (rough) volatility models”. *Quantitative Finance*, 21(1), 11–27. Available at: <https://doi.org/10.1080/14697688.2019.1677194>
- [25] Hull, J., & White, A. (1998) “Value at risk when daily changes in market variables are not normally distributed”. *Journal of Derivatives*, 5(3), 9–19. Available at: <https://doi.org/10.3905/jod.1998.407961>
- [26] Hutchinson, J. M., Lo, A. W., & Poggio, T. (1994) “A nonparametric approach to pricing and hedging derivative securities via learning networks”. *The Journal of Finance*, 49(3), 851–889. Available at: <https://doi.org/10.1111/j.1540-6261.1994.tb00081.x>
- [27] Isakov, V. (2003) “The inverse problem of option pricing”. In *Recent development in theories and numerics* (pp. 47–55). Springer.

- [28] Itkin, A. (2019) “Deep learning calibration of option pricing models: Some pitfalls and solutions”. *arXiv preprint*. Available at: <https://arxiv.org/abs/1906.03507>
- [29] Kennedy, J., & Eberhart, R. (1995, November) “Particle swarm optimization”. In *Proceedings of ICNN’95 - International Conference on Neural Networks* (Vol. 4, pp. 1942–1948). IEEE. Available at: <https://doi.org/10.1109/ICNN.1995.488968>
- [30] Kennedy, M. C., & O’Hagan, A. (2001) “Bayesian calibration of computer models”. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3), 425–464. Available at: <https://doi.org/10.1111/1467-9868.00294>
- [31] Kirkpatrick, S. (1984) “Optimization by simulated annealing: Quantitative studies”. *Journal of Statistical Physics*, 34(5), 975–986. Available at: <https://doi.org/10.1007/BF01009452>
- [32] Liu, S., Borovykh, A., Grzelak, L. A., & Oosterlee, C. W. (2019) “A neural network-based framework for financial model calibration”. *Journal of Mathematics in Industry*, 9(1), 9. Available at: <https://arxiv.org/abs/1904.10523>
- [33] McGhee, W. A. (2021) “An artificial neural network representation of the SABR stochastic volatility model”. *Journal of Computational Finance*. Available at: <https://doi.org/10.21314/JCF.2021.402>
- [34] Merton, R. (1976) “Option pricing when the underlying stock returns are discontinuous”. *Journal of Financial Economics*, 3(1–2), 125–144. Available at: [https://doi.org/10.1016/0304-405X\(76\)90022-2](https://doi.org/10.1016/0304-405X(76)90022-2)
- [35] Reynolds, C. W. (1987) “Flocks, herds, and schools: A distributed behavioral model”. *Computer Graphics*, 21(4), 25–34. Available at: <https://doi.org/10.1145/37402.37406>
- [36] Scott, L. (1997) “Pricing stock options in a jump-diffusion model with stochastic volatility and interest rates: Application of Fourier inversion methods”. *Mathematical Finance*. Available at: <https://doi.org/10.1111/1467-9965.00039>
- [37] Storn, R., & Price, K. (1997) “A simple and efficient heuristic for global optimization over continuous spaces”. *Journal of Global Optimization*, 11(4), 341–359. Available at: <https://doi.org/10.1023/A:1008202821328>
- [38] Tohme, T., Vanslette, K., & Youcef-Toumi, K. (2020) “A generalized Bayesian approach to model calibration”. *Reliability Engineering & System Safety*, 204, 107141. Available at: <https://doi.org/10.1016/j.ress.2020.107141>
- [39] Wang, D., Tan, D., & Liu, L. (2018) “Particle swarm optimization algorithm: An overview”. *Soft Computing*, 22, 387–408. Available at: <https://doi.org/10.1007/s00500-016-2474-6>