The School of Mathematics

THE UNIVERSITY
of EDINBURGH

Parameter Identification in Market-Consistent Calibrations of the SVJD Equity Model

by

Yaojiuwei Cai

Dissertation Presented for the Degree of
MSc in Financial Mathematics

August 2022

Supervised by
Doctor Tamás Mátrai
Doctor Theodoros Assiotis

Abstract

Model validation in the organisation is playing a significant role in guiding model selection decisions. The Stochastic Volatility Jump Diffusion model, an implementable stochastic model which can project realistic market performance, has been using as flagship equity model for many years. We aim to give a detailed introduction of the model itself and COS method pricing under Moody's own Market Consistent Calibration Methodology. As a strong theory-based model, we will apply both theoretical and practical analysis on the basis of parameter risk. We present a partial differential equation for making the assumptions of the redundant parameters, and a new practice-relevant validation that shows more quantitative visualised results will be assembled to strongly support the theory approach. The calibration process has been done by using Moody's Market-Consistent Economic Scenario Generator (ESG), which is a suite stochastic asset modelling tool with high flexibility on parameter setting. This paper is built under the assumption of risk-neutral measure.

Keywords: Stochastic Volatility Jump Diffusion, COS method pricing, Market Consistent Calibration, Economic Scenario Generator, Redundancy.

Acknowledgments

This dissertation concludes my M.Sc. and summer internship work at the University of Edinburgh and Moody's Analytics that started in June 2022 and ended in Sep 2022.

My most sincere gratitude goes to my industrial supervisor Dr.Tamás Mátrai. This is my first time doing research individually, also the first time to be interviewed and got accepted not only academically, but also industrially. Tamas guided me through the relevant process that I need to know to solve the questions, he was patiently answering all my questions in plenty of emails, and during in-person meetings. During these months, I have benefited greatly from his guidance and encouragement. Without his help, this research will never be done successfully.

I'd also like to extend my gratitude to my academic supervisor Dr.Theodoros Assiotis, he helped me with the structures of the thesis. During the limited meetings, he guided me on how to present a strong theoretical based dissertation, also gave me plenty of advice on references and presented language. Without his help, I would never be able to present my work professionally and reach the academic requirements.

I also wish to thank my parents, they were guiding me throughout my entire life and gave me too much love and support. Without my parents, I won't be able to be the person I want to be and do the things I want to do.

Besides, I am indebted to Harri and Matthew, who helped me deal with some technical issues, had many life talks, and always gathered at the main library to study together.

Special thanks also go to all the librarians, they helped me to get the physical book that I need for the dissertation and provided a good study environment for us. A big appreciation should also be delivered to the university, I won't learn these many things if I didn't get the offer.

Finally, thanks to all my colleagues who I was getting along with during these 365 days. Without them, I'd have never gone this far.

Own Work Declaration

This sheet must be filled in, signed and dated - your work will not be marked unless this is done.

Name: Yaojiuwei Cai
Matriculation Number: S2256965
Exam No. B209378
Title of work: Parameter Identification in Market-Consistent Calibrations of the SVJD Equity Model

I confirm that all this work is my own except where indicated, and that I have:

- Clearly referenced/listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Not sought or used the help of any external professional academic agencies for the work
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Complied with any other plagiarism criteria specified in the Course handbook

To be precise, Market Consistent Calibration has been done by using Moody's Economic Scenario Generator (ESG), which is followed up with the given files Scenario Generator - Developer Guide (2021) [6], Scenario Generator - Configuration Guide (2022) [7], Scenario Generator - Methodology Guide (2022) [8], SVJD Market Consistent Calibration Methodology (2013) [3], Calibration Tools Methodology Guide (2019) [5]. Stochastic Volatility Jump Diffusion Model Definition (2013) [4] and Gatheral (2006) [19] has been used as the main paper to define the model, other related papers are cited in the main paragraphs. ESG class of functions are called from Python and the whole generation of the data can be seen in Appendix A.1, function is generalised by my own. The COS pricing method is adapted from Fang (2008) [17], and the related FFT algorithm is adapted from Lewis (2001) [24] and Kwok, Leung and Wong (2012) [23]. Theoretical proof in Section 6.1 is the indicated by my own that I mainly got the ideas from the first derivation of Dr.Tamás Mátrai, the final process of the proof is created by my own. Linear Regression analysis idea is referred from the paper Tamás and Ileana (2022) [28]. All of the python functions of generating data, cleaning data, sorting data, doing regression analysis, and plotting graphs and tables are created by my own, the codes and data will be available on GitHub (https://github.com/RogerCai956/FinalDissertation.git), and the most important codes for related functions will be available in Appendix.

I understand that any false claim for this work will be penalised in accordance with the University regulations.

Signature: 

Date: August 26, 2022

Contents

List of Tables

List of Figures

1 Introduction

In the reality, traders are making tons of transactions every day, as the change in the assets' price fluctuates sometimes dramatically and cannot be anticipated. Therefore, with the further development of finance, mathematics, economics, and statistics, quantitative models are created and applied for making assumptions about real-life data processes. The importance of trading is investors could gain millions of profit at one point, but also have chances of losing everything at the next minute. In this case, a good way of eliminating the potential risk from the quantitative model will be necessary when investors or traders are making the strategies. This becomes possible when various financial derivatives are constructed, i.e., Call, Put options [33]. Nowadays, plenty of derivatives are defined by financial researchers and have been used for the past decades until now. An example of the option derivatives, the holder seems to have the absolute opportunity of gaining sufficient benefit in their preference. In contrast, the trader will sustain an amount of uncertainty, for example, if the underlying price of the Call option is going up, the buyer of the Call will absolutely gain the payoff of the underlying price minus the strike, but the trader has to pay the payoff that has been signed in the option contract. In this case, making a fair price of the derivatives will be very necessary for the derivative traders.

Definition 1.1 (Option) *An option is a financial derivative on an underlying asset and it gives the right but not the obligation to buy or sell the asset at a fixed time, the expiration of the option is called maturity and the exercise price is called the strike price. An option is giving you the right to gain the benefit, thus, they will be charged with a defined fair price.*

Call option: *is a financial contract that gives the right but not the obligation to the investor to buy an asset with a fixed amount of money.*
Put option: *is opposite to the Call option, for which, it gives the right to the option holder to sell an asset with a fixed amount of money.*

Example of two kinds of Option:

European option: *is a type of options contract that limits rights exercise to only the day of expiration.*
American option: *is a type of options contract that allows holders to exercise their rights at any time before and including the expiration date.*

In the case of making the price of the derivatives, scientific tools are used to simulate the price dynamics under suitable stochastic models, and make empirical and well-founded pricing by using methods that have been fully researched by past and present scientists.

This research paper is going to mainly describe a fantastic model for asset price dynamics called Stochastic Volatility Jump Diffusion which is defined as:

$$d\ln(S_t^{SV} \cdot S_t^{JD}) = \left(\mu - \frac{v_t}{2} - \lambda\bar{\mu}\right) dt + \sqrt{v_t}\, dW_t^1 + \ln J\, dN(t), \tag{1.1}$$

where S^{SV}, S^{JD} are two assets represent Stochastic Volatility and Jump Diffusion respectively, μ is presented as the risk premium[1], λ is the expected number of jumps happen in a unit of time. J is following a log-normal distribution with mean μ_J and variance σ_J^2, and we write $\bar{\mu} = \exp\{\mu_J + \frac{1}{2}\sigma_J^2\} - 1$. The number of jumps $N(t)$ that occur over the period $[0, T]$ is determined by a Poisson process with parameter λt. We also have v_t as the volatility which has the dynamics:

$$dv_t = \alpha(\theta - v_t)\, dt + \xi\sqrt{v_t}\, dW_t^2, \tag{1.2}$$

[1] which is equal to the risk free interest rate under the risk-neutral measure.

where α is the speed of the mean reversion, θ represents the mean reversion level, ξ is the volatility of variance, and the initial value is defined as v_0. For above model, W^1 and W^2 are Wiener processes linking by the correlation parameter ρ:

$$d[W^1, W^2]_t = \rho \, dt. \tag{1.3}$$

Then, a method that has been created in previous decades for pricing the options called the Fourier Cosine Expansion method will be described in Section 5. Finally, the goodness of the model fit with eight parameters will be tested by using Moody's Scenario Generator's built-in method called Market Consistent Calibration to get the market-like data, then, linear regression method will be used to test the redundancy of the model. The following Sections will generate the flow of understanding the topics.

2 Implied Volatility and Market Consistent Calibration Methodology

2.1 Implied Volatility (IV)

Black and Scholes in 1973 [10] has developed a good model for pricing the European options (see Equation 2.2) called Black-Scholes model. In the Black-Scholes model, we would only need the information for the deterministic value of volatility, the price of the underlying asset, the strike price of the underlying, the time until expiration of the option, and the risk-free interest rate. However, since the 1987 Black Monday market crash, the industrial average implied volatilities were dropping by almost 25 percent of the daily return [11], this begins to grab people's attention on big changes or jumps are possible to happen. The event also led to a view on the volatility skew and after mapping the implied volatility on the graph with the same maturity date and non-fixed strikes, a smile pattern is obviously shown [1].

Since the standard Black-Scholes model is simulating the stock price by using deterministic parameters in the model, therefore, the volatility is a constant or known function in the model. As we would able to see in Figure 1, the change of the real-life data is following a stochastic pattern and we could also see that the big moves follow big moves and small moves follow small moves, so-called volatility clustering [2].

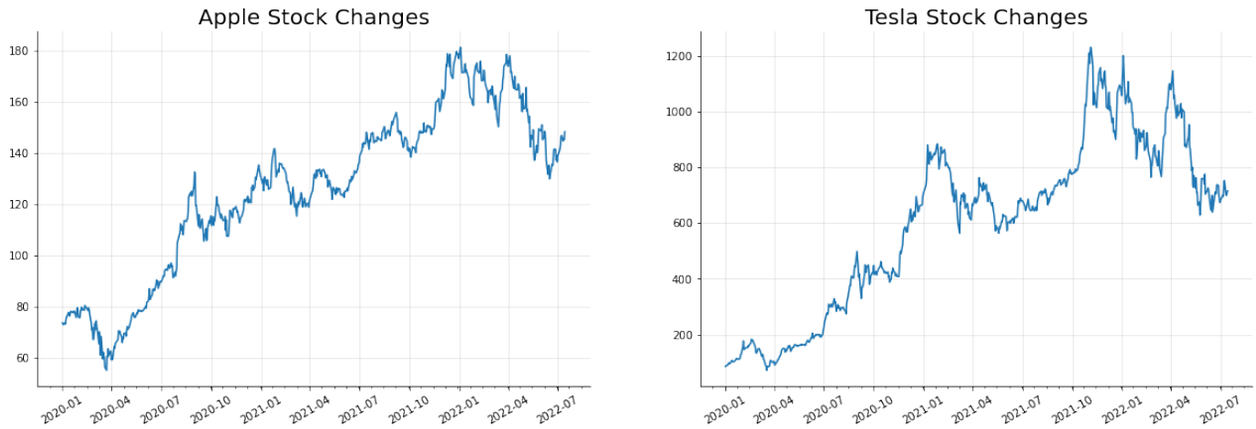

Figure 1: Two plots show how the stock prices of Apple and Tesla changes from January, 2020, to July, 2022 (Data is gotten from Yahoo Finance).

To more clearly show the return from the stock price, a definition of log-return should be introduced here.

Definition 2.1 (Logarithmic Return) *The log-return*[3] *is used to compute the rate of the return on an underlying. In the calculation, we should know the value V_t at the time step t and the value V_{t+1} at the next time step, then we have:*

$$R_t^{log} = \ln\left(\frac{V_{t+1}}{V_t}\right) \times 100\%. \tag{2.1}$$

Log-return plots in Figure 2 are drawn to illustrate the daily percentage shift of Apple and Tesla. In Figure 2 the feature clustering is shown in the time series and jumps are stated obviously.

[2] It is first noted by Mandelbrot (1963)[26]. This feature signifies that the volatility is auto-correlated, and could be simulated as the mean reversion in the Stochastic Model.

[3] The advantage of using this return is normalisation: it measures all variables in a comparable metric, and make a good source of comparing variables that are having different scales of values.

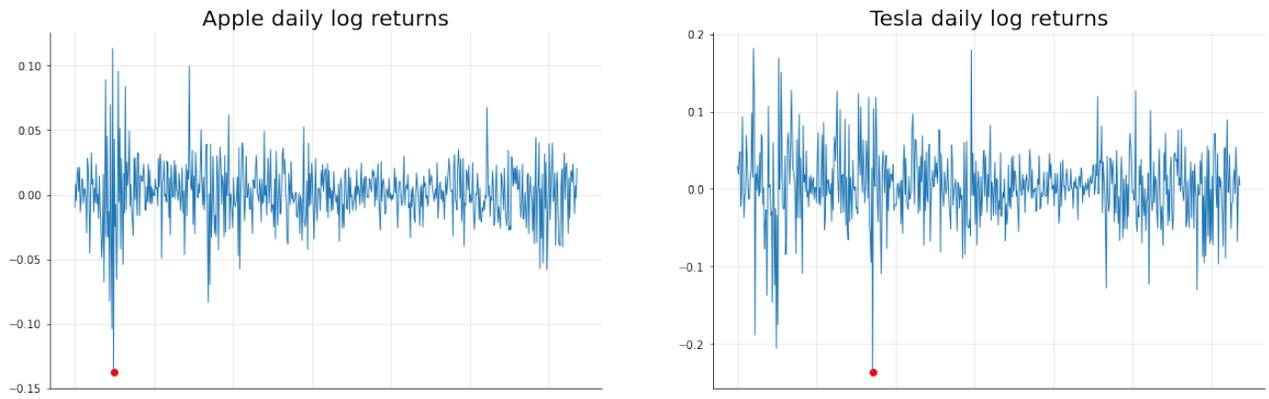

Figure 2: Apple and Tesla daily logarithmic returns from January 2020, to July 2022. Notable minimum Apple log-return with -0.14, and minimum Tesla log-return with -0.24 are shown with the red scatters in the plots.

Looking at the histogram and Q-Q plot of Tesla stock price in Figure 3, we could conclude that the actual distribution can not be simulated just by using classic Black-Scholes model with normally shifted Brownian motion, since the actual data is having a high peak and fat tail compared with the normal distribution which is shown as the orange shadow, and the scatters in Q-Q plot are not having a straight line pattern.

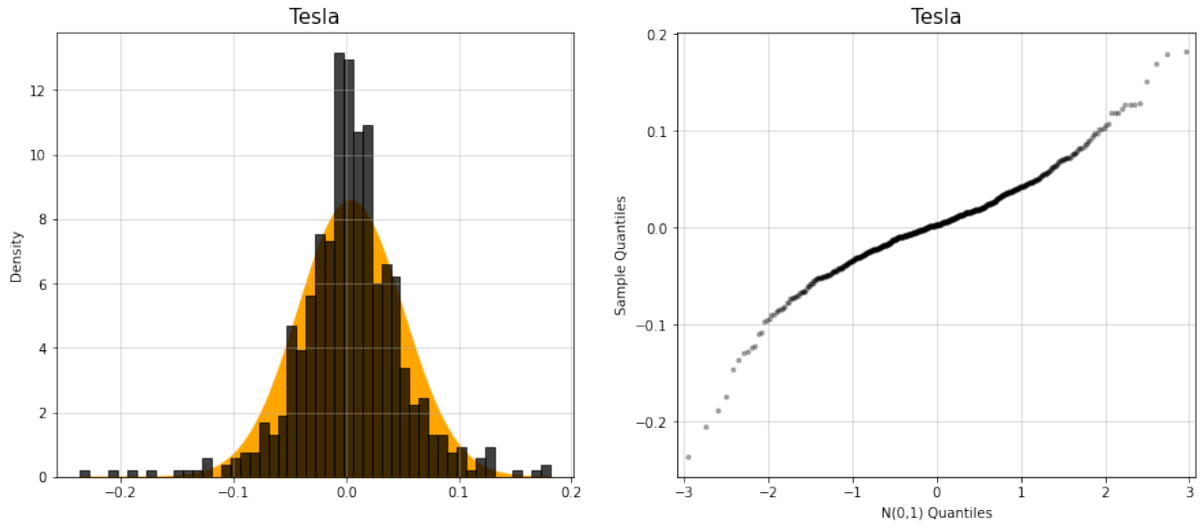

Figure 3: Tesla data from January 2020, to January 2022 has been used to analyse the pattern of the fat tails and volatility difference during the time series. On the left, the histogram is shown the density distribution of real-life data which is compared with the normal distribution (orange shadow). On the right, Q-Q plot is drawn to further support the ideas.

As it is well-known, the pricing of the European option in the Black-Scholes [10] world with deterministic parameters is interpreted by the formula[4]:

$$C(S_t, t; \sigma, r; K, M) = N(d_1)S_t - N(d_2)Ke^{-r(M-t)}, \tag{2.2}$$

where

$$d_1 = \frac{1}{\sigma\sqrt{M-t}}\left[\ln(\frac{S_t}{K}) + (r + \frac{\sigma^2}{2})(M-t)\right] \quad \text{and} \quad d_2 = d_1 - \sigma\sqrt{M-t}, \tag{2.3}$$

[4]Here, call option price is shown, the put option price could be gotten by using Put-Call Parity.

in the above formula, $C(S_t, t; \sigma, r; K, M)$ represents the price of the European Call option, S_t represents the price of the underlying asset at time t, t is the time in the year, r is the annualised risk neutral interest rate, σ is the volatility or variance of the underlying asset, M is the maturity of the option, K is the strike price of the option, which the option buyer or seller hold the right to exercise the option at the maturity, and $N(x)$ is denoted as the standard normal cumulative distribution function.

To calculate the price, all parameters except σ could be easily measured, therefore, as long as the price of the option is known, the σ could be calculated inversely, this σ is so-called **Implied Volatility (IV)**. On the other hand, we can compute the volatility σ if we know the option price C [5] [35]. As we analysed that the volatility should not be deterministic during the life of the stock, therefore, the volatilities do not all lie on the horizontal line in term of the strike price for the same underlying and maturity. However, the pattern of the IV with respect to the strike will then have a smile shaped pattern, which is known as **Volatility Smile** [16].

Figure 4 is presenting a 3D Implied Volatility Surface to show the changes of IV with respect to strikes and maturities at time point zero. It is obviously displaying a smile shaped IV, and the longer the maturity the flatter the IV will become.

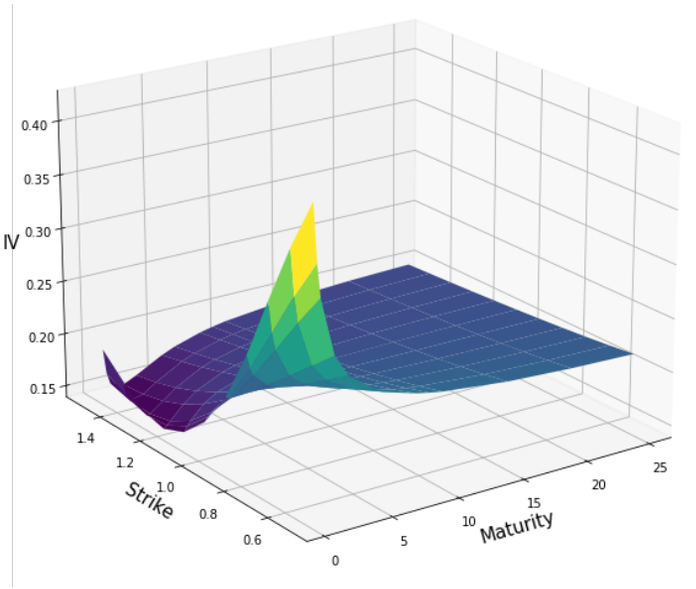

Figure 4: The figure shows a good pattern of Volatility Smile with z-axis represents the value of the implied volatilities, x-axis represents the strikes, and y-axis represents the maturities. The plot is made by using generated data in Table 2.

Therefore, practitioners need equity models which produce IV Surface in line with the output data. In Section 3.1 we will introduce a good model for doing this, which is called the Stochastic Volatility Jump Diffusion Model (SVJD).

Importance of Implied Volatility: Implied volatility is normally used to make the prediction of the future volatility changes of the underlying, since in the option market, the implied volatility could be calculated from the options with different strikes and maturities, this could lead us to estimate or make the prediction that how will the stock price changes in the future, but it could not help us further with the change of the price direction [34]. Additionally, Sara and Julia in 2020 [34] researched and concluded that implied volatility has been used better

[5] Regulated by the value of the option should be bigger than the payoff and less than the Stock price, we could only find the one and only one value for sigma that makes the models calculated price and the real market price the same.

than the realised volatility for prediction. Thus, hedging the risk in the real life is becoming more to estimate the change of the implied volatility to match with the risk aversion of investors.

2.2 Market Consistent Calibration (MC)

As long as a suitable model could be defined, we would need to test its stability with the calibration of parameters that the model contains. Therefore, another methodology will be delivered which is called Market Consistent Calibration (MC), this methodology details are imposed by Moody's [22]. MC is the process of adjusting the model parameter values such that the model could reproduce or produce the price as close as the price that we observe in the market.

Moody's [3] has its own MC methodology to apply to the SVJD model. Using the method, we could find the best fit parameters such that the discrepancies between the model prices and the market prices of the European equity put options are minimised. The aim is to minimise an objective function which is solved by the optimiser called Levenberg-Marquardt [25] to weight the sum of squares of residuals. The processes are shown in the following steps:

1. **Specifying weighting matrix and Vega re-scaling of weights:** Moody's has its specified weighting matrix w defined in the calibration toolkit, which is labeled with a range of strikes and a range of maturities. The weighting matrix illustrates the weight of each entry of the target data set[6]. However, the calibration tool does not fit with the implied volatility as we could get the price directly by inputting a set of parameter values to the formula in COS method (Section 5), in this case, the weighting matrix for the IV will not perform well in the region where the prices are small. Moody's then imposed a technique that is using Vega as the sensitivity element to re-scale the weighting matrix, subsequently, the re-scaled weighting matrix could apply to the price matrix. The Vega is presented as below:

$$v = \phi(d_1)\sqrt{M}, \tag{2.4}$$

where $M = T - t$ is the maturity, $d_1 = \frac{-\ln \tilde{K} + \frac{\sigma^2}{2}(T-t)}{\sigma\sqrt{T-t}}$, $\phi(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{x^2}{2}}$, $\tilde{K}$ is the forward strike and σ is the market implied volatility[7].

For the IV which is having very low sensitivity, in case of maximising the efficiency of the calibration process a threshold is introduced which is Vega will set to zero if it is lower than 0.001.

Then the final re-scaled weighting matrix is defined by $\frac{w}{v^2}$.

2. **Pseudo market prices:** Using the Black-Scholes model to calculate the price of the option under the given implied volatilities. The market price of the option is defined as S^{Market}.

3. **Levenberg-Marquardt optimiser:** The loss objective function should be used for finding the optimised parameter set that minimise the squared norm of the vector of residuals between the market price and the model calculated price[8]. Table 1 shows the parameter values that are given by previous calibration, the model will use these to calculate the price under COS method, the reason that we have boundaries is because it will help to detect similarity on the changes of IV while we choose different parameters value, see example in

[6]Here, target data set means that the whole calibration process will end up giving us the optimised parameters set and also giving out the price of the option which calculated by the method called COS method (see Section 5). Each entry of the matrix represents the implied volatility under specific strike and maturity. Example of cleaned and sorted data is shown in Table 2.

[7]Market IV data are provided by Thomson Reuters, Markit, and SuperDerivatives.

[8]The model calculated prices are getting by using the seed we obtained in the previous calibration period in order to faster the speed of convergence.

Section 6.2.1.

The objective function with parameter set $\varphi = (\lambda, \mu_J, \sigma_J, \rho, \theta, \alpha, v_0, \xi)$ is defined as:

$$\Theta^2(\varphi) = \Theta^2_{Market}(\varphi) + \Theta^2_{Feller}(\xi^2 - 2\alpha\theta), \tag{2.5}$$

where Θ^2_{Feller} is the Feller condition penalty term with $B = 0.001$ and $S = 4$:

$$\Theta^2_{Feller}(x) = \begin{cases} 0 & x < -\frac{B}{2} \\ \left(\frac{x+\frac{B}{2}}{B}\right)^S & x \geq -\frac{B}{2}, \end{cases} \tag{2.6}$$

and the $\Theta^2_{Market}(\varphi)$:

$$\Theta^2_{Market}(\varphi) = \sum_{M,K} \frac{w}{v^2}(S^{Market}_{M,K} - S^{Model}_{M,K}(\varphi))^2. \tag{2.7}$$

Then, we will be able to get the optimised parameters set by using the optimiser.

ParentEquityModel	Median	Min	Max
ArrivalRate (λ)	0.100000	0.090000	0.110000
JumpMean (μ_J)	-0.400000	-0.410000	-0.390000
JumpVolatility (σ_J)	0.300000	0.290000	0.310000
Correlation (ρ)	-0.550000	-0.560000	-0.540000
MeanReversionLevel (θ)	0.026062	0.016062	0.036062
MeanReversionRate (α)	0.649984	0.639984	0.659984
InitialValue (v_0)	0.022392	0.012392	0.032392
Volatility (ξ)	0.165254	0.155254	0.175254

Table 1: Table shows the seeds that we fit into the model calibration process, eight parameter values with Max values and Min values are stressed by Median+0.01 and Median-0.01 respectively. it gives a boundary for fitting the parameters into the process. Detailed IV generation will be mentioned below.

4. **Calculate all model prices and implied volatilities:** After we get the optimised parameters, we would be able to calculate the model prices by using the COS method again for all maturities and strikes. Then, transfer it to implied volatility for assessment of the quality of fit.

However, the more parameters we use, the more the model will be possibly over-fitted, which means that the parameters we defined are predicting the result too well and this will perhaps make the model to be very sensitive to the change of the parameters. Regarding the sensitivity of the model, the model **redundancy** check should also be established. To consider the **parameter redundancy** is very important for the parameterisation of the model, the model is considered to be redundant if it could be re-parameterised by a smaller number of parameters, or the parameter will be defined as non-identifiable if it only ever appears in the model as a product and can not be estimated individually; only the product can be estimated [12].

2.2.1 Moody's Economic Scenario Generator (ESG)

As the market becomes more sophisticated, traders want to be sure the information they are handling is reliable, as they need an entire view of risks and opportunities. Moody's Scenario

Generator is offering a comprehensive solution to further understand not only the short-term but also the long-term impact of market performance on financial products. Moody's ESG has already provided solutions to clients across a wide range of financial areas, including corporations, banks, insurance companies, capital allocators and portfolio managers [2].

ESG is built on stochastic asset models and calibration content that support the models reflect or perform as similar as the realistic asset returns and risk-factor distributions. Under the built-in multi-asset classes, ESG is also strong in multi-risk factor, and multi-time step scenario generating which is used by investors and traders for making decisions. It aims to produce outstanding flexibility in the parameterisation of its models to give more power to the user for expressing their views on the problem. However, parameterisation is only meaningful as long as the calibration target set determines all parameters, otherwise, over-parameterised could cause the model to be unstable.

Advantages of ESG:

1. The realistic scenarios are generated by ESG, which projects the actual distributions for asset returns and risk factors. The scenarios will cover a wide range of time intervals, i.e., 1-5 years, or for 50 years or longer.

2. Benefit from a comprehensive monthly calibration service covering a wide range of economies and asset classes.

3. The flexibility of the calibration methodology let the user implement their own thoughts of the risk and return.

4. Well documented by researchers, which includes calibration reports, model methods, calibration methods, and so on.

5. Less computational cost is achieved by using the advanced methodology.

In the following sections, we will mainly be using the ESG's model to make the simulation of the stock price and ESG's built-in method for calibration will also play a critical role during the process of getting the parameter set that best simulate the forward-looking market data that helps us to have a future view of the price changes under the implied volatility. The following sections will focus on giving a good view of the theoretical model description and the mainly used pricing method that has been commonly implemented by Moody's for market calibration. Then, the theoretical and numerical results in Section 6 will be presented mainly aim to solve the following questions:

1. What impact do the model parameters have on the model prices of options in the calibration target set?

2. For each parameter of the model, which calibration targets are most suitable for its identification?

3. Is the parameterisation of the SVJD model redundant in practice, and if yes, is there an extension of the calibration target set which eliminates the redundancy?

3 SVJD Model

3.1 Models

As we have discussed above, the real-life data is not following the dynamics of the Black-Scholes model. Instead, we should find a good model that could successfully simulate the pattern of the actual process of the market assets. The ideas come up and are analysed by a lot of researchers, for example, Roger, Lars A, and Knut [15] stated that adding jumps directly to the

Black-Scholes model will lead to a very poor estimation of the data since it cannot capture the skewness of the actual data. Therefore, we need a model that can handle both the skewness and the high peak of the data, i.e., the pattern in Figure 3. In the 1980s, the original Stochastic Volatility model had been generated and proved by Hull and White [21], Stein and Stein [32], Heston [20] and following researchers, and later add jump diffusion into the Stochastic Volatility model had been further improved by Bates [9] and Pan [30]. In this section, I will introduce the model called Stochastic Volatility Jump Diffusion (SVJD).

The SVJD model contains two parts, the first one is the Heston Stochastic Volatility model which is a continuous process and the second part is a discrete Jump Diffusion process. But here, instead of assuming that we simulate the process under Geometric Brownian Motion in the time series of no jump, we add the jump part into the Heston Stochastic Volatility model.

3.2 The Heston Stochastic differential equations

Suppose that $(S_t^{SV})_{t\in[0,T]} \in \mathbb{R}$ is a real-valued stochastic process defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Let W_t^1 and W_t^2 be two Wiener processes with respect to $(\mathcal{F}_t)_{t\geq 0}$, and let the μ is the drift which equals the risk-free rate in a risk-neutral world. Then the continuous Heston Stochastic Volatility model will be:

$$dS_t^{SV} = \mu S_t^{SV}\, dt + \sqrt{v_t} S_t^{SV}\, dW_t^1. \tag{3.1}$$

Volatility process can be defined by a familiarised model that has been created originally for modelling interest rate, called Cox-Ingersoll-Ross process [14]:

$$dv_t = \alpha(\theta - v_t)\, dt + \xi\sqrt{v_t}\, dW_t^2, \tag{3.2}$$

where we will have parameter initial value v_0, the two Wiener processes are correlated with the correlation value ρ, as we could write:

$$d[W^1, W^2]_t = \rho\, dt, \tag{3.3}$$

where α represents the speed of the Mean Reversion or Mean Reversion Rate, $\theta > 0$ defined as the mean reversion level, and ξ is the volatility of variance.

3.2.1 Numerical Simulation

To generate the process of the model, the Euler Scheme left most point has been used.

First, we use Itô's formula to get the explicit solution:

$$d\ln S_t^{SV} = \left(\mu - \frac{v_t}{2}\right) dt + \sqrt{v_t}\, dW_t^1, \tag{3.4}$$

using Euler method we can get implicit solution:

$$S_{t+\Delta t}^{SV} = S_t^{SV} exp\left\{\left(\mu - \frac{(v_t)_+}{2}\right)\Delta t + \sqrt{(v_t)_+}\sqrt{\Delta t}\, W^1\right\}, \tag{3.5}$$

where $(v)_+ = \max(v, 0)$.

Then,

$$v_{t+\Delta t} = v_t + \alpha(\theta - (v_t)_+)\,\Delta t + \xi\sqrt{(v_t)_+}\sqrt{\Delta t}\, W^2, \tag{3.6}$$

to generate $W = \{W^1, W^2\}$, we use Cholesky Decomposition with the normal distribution:

$$\begin{pmatrix} W^1 \\ W^2 \end{pmatrix} \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right). \tag{3.7}$$

3.3 The Merton Jump Stochastic differential equations

Here, the discrete part of the model will be illustrated by using the concept of the Merton Jump Diffusion model. The jump will be added directly to the Stochastic Volatility model later.

Suppose that $(S_t^{JD})_{t\in[0,T]} \in \mathbb{R}$ is a real-values discrete process, and $S_0^{JD} = 1$. Then the Merton Jump model will be defined as below:

$$\frac{dS_t^{JD}}{S_{t_-}^{JD}} = -\lambda\bar{\mu}dt + (J-1)\, dN(t), \tag{3.8}$$

where $S_{t_-}^{JD} = \lim_{t\to t_-} S_t^{JD}$, $\bar{\mu} = exp\{\mu_J + \frac{1}{2}\sigma_J^2\} - 1$ and $dN(t) \equiv N(t+dt) - N(t)$, J is representing a log-normal distributed random jump value such that $\ln J \sim N(\mu_J, \sigma_J^2)$, where μ_J is the mean of the jump size, and σ_J^2 is the variance/volatility of the jump size. The number of random jumps $N(t)$ that occur over the period $[0,t]$ is determined by a Poisson process with parameter λt, where λ is the jump arrival rate (the expected number of the jumps per unit time).

3.3.1 Numerical Simulation

Again, to generate the process of the jump, we use the Euler Scheme left most point.

First, We use the Itô's lemma in the version of the Jump Diffusion model[9], we have:

$$d\ln S_t^{JD} = -\lambda\bar{\mu}\, dt + \ln J\, dN(t), \tag{3.9}$$

then implicit solution/numerical Euler form will be:

$$S_{t+\Delta t}^{JD} = S_t^{JD} exp\{-\lambda\bar{\mu}\,\Delta t\} \prod_{u=1}^{N(\Delta t)} J_u. \tag{3.10}$$

It can also be written as:

$$\ln S_{t+\Delta t}^{JD} = \ln S_t^{JD} - \lambda\bar{\mu}\,\Delta t + \sum_{u=1}^{N(\Delta t)} \ln J_u, \tag{3.11}$$

where,

$$\begin{aligned} N(\Delta t) &\sim Poisson(\lambda\Delta t), \\ \ln J_u &\sim N(\mu_J, \sigma_J^2). \end{aligned} \tag{3.12}$$

3.4 Stochastic Volatility Jump Diffusion (SVJD)

Two stochastic processes have already been defined above. Since in the time series, the process of discrete jump will be added directly at the point that jump happens, therefore, the jump process will be added to the stochastic volatility process. The final model will be generalised as

[9] See the definition in the Section 4.1

follow:

$$\ln S_{t+\Delta t}^{SV} + \ln S_{t+\Delta t}^{JD} = \ln S_t^{SV} + \ln S_t^{JD} + (\mu - \frac{v_t}{2} - \lambda\bar{\mu})\,\Delta t + \sqrt{v_t}\sqrt{\Delta t}\,W^1 + \sum_{u=1}^{N(\Delta t)} \ln J_u$$
$$= \ln S_t^{SV} + (\mu - \frac{v_t}{2})\,\Delta t + \sqrt{v_t}\sqrt{\Delta t}\,W^1 + \ln S_t^{JD} - \lambda\bar{\mu}\,\Delta t + \sum_{u=1}^{N(\Delta t)} \ln J_u, \tag{3.13}$$

and

$$v_{t+\Delta t} = v_t + \alpha(\theta - (v_t)_+)\,\Delta t + \xi\sqrt{(v_t)_+}\sqrt{\Delta t}\,W^2, \tag{3.14}$$

with

$$\begin{pmatrix} W^1 \\ W^2 \end{pmatrix} \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}\right). \tag{3.15}$$

Finally, let S_t^{XS} be the combined process, we have:

$$S_t^{XS} = S_t^{SV} S_t^{JD} \tag{3.16}$$

as the final presentation, and the model contains eight parameters:

1. θ is the mean reversion level of the volatility process.
2. α is the speed of mean reversion of the volatility process.
3. ξ is the volatility of variance of the volatility process.
4. ρ is the correlation of two Wiener processes.
5. λ is the jump arrival rate.
6. μ_J is the jump mean.
7. σ_J^2 is the jump variance.
8. v_0 is the initial value of the volatility process.

4 Theorems we use to support the thesis

4.1 Itô's formula and Jump process

Here, we introduce Itô's techniques to help the model creation and new theoretical analysis of the option price under the partial differential equation that I will define in Section 6.1. Itô's formula presented by Øksendal (1998) [29] is the one we normally see in life, but a new way of using Itô's ideas under the Jump Diffusion will be mainly focused on.

Definition 4.1 *A stochastic process X is known as an Itô process if X_0 is $\mathcal{F}_0$-measurable and X can be written in the form (Definition 4.1.1 [29]):*

$$X_t = X_0 + \int_0^t F_s\,ds + \int_0^t G_s\,dW_s,\ t \in [0,T], \tag{4.1}$$

Here, $G \in \mathcal{H}^2$ and F are two continuous adapted processes, W_s is the Wiener process.

The first integral is classical: the area under the random curve F_t. The second integral is an Itô integral.

Note: To fully specify X_t, we also need to know F_s, G_s, and the initial value X_0. Since this means we will be dealing with integrals of the form $\int_0^t F_u du$.

Theorem 4.1 (Product rule) *Let X and Y be Itô processes. Then $X \cdot Y$ has the stochastic differential [31]:*

$$d(X_t \cdot Y_t) = Y_t\, dX_t + X_t\, dY_t + dX_t\, dY_t. \tag{4.2}$$

In the simple Black-Scholes model or the Stochastic Volatility model, the process are following the martingale property. But in the jump case, we are having an unpredictable jump at the discrete time point, therefore, in the time interval of not having jumps, the process could also have a martingale property, but there won't be a martingale at the jump point. Therefore, we will introduce a newly defined process which is known as **Semimartingale**. The definition is defined by Cont and Tankov (2004) [13].

Definition 4.2 (Semimartingale) *An unanticipated cadlag process X defined on the filtered probability space $(\Omega, (\mathcal{F}_t)_{t\in[0,T]}, \mathbb{P})$ is called a semi-martingale if the stochastic integral of simple predictable processes Z with respect to X (Definition 8.2 [13]):*

$$Z = Z_0\, \mathbb{I}_{t=0} + \sum_{i=0}^{n} Z_i\, \mathbb{I}_{[T_i, T_{i+1}]} \mapsto \int_0^T Z\, dX = Z_0 X_0 + \sum_{i=0}^{n} Z_i (X_{T_{i+1}} - X_{T_i}), \tag{4.3}$$

where $\mathbb{I}$ represents the indicator function, satisfies the following continuity property: for every Z^n, $Z \in \mathbb{S}([0,T])$, if

$$\sup_{(t,\omega)\in[0,T]\times\Omega, n\to\infty} |Z_t^n(\omega) - Z_t(\omega)| \to 0, \tag{4.4}$$

then

$$\int_0^T Z^n\, dX \to \int_0^T Z\, dX, \ \text{as } n \to \infty. \tag{4.5}$$

Definition 4.3 (Quadratic Variation) *For an unpredictable semi-martingale process X, the quadratic variation process could be defined by (Definition 8.3 [13]):*

$$\sum_{t_i\in\pi^n, n\to\infty}^{0\le t_i<t} (X_{t_{i+1}} - X_{t_i})^2 \to [X,X]_t, \tag{4.6}$$

such that $\pi^n = (t_0^n = 0 < t_1^n < ... < t_{n+1}^n = T)$ is a sequence of partitions of $[0,T]$, and $\pi^n = \sup_k |t_k^n - t_{k-1}^n| \to 0$ as $n \to \infty$, where the convergence is uniform in t.

Theorem 4.2 (Itô's formula) *Suppose that, for $t \in [0,T]$ a real value and $x \in \mathbb{R}$, $f(t,x)$ is a deterministic function that is differentiable in t and twice differentiable in x. Then $Z_t = f(t, X_t)$ is an Itô process with stochastic differential (Definition 4.1.2 [29]):*

$$dZ_t = \frac{\partial f}{\partial t}(t, X_t)\, dt + \frac{\partial f}{\partial x}(t, X_t)\, dX_t + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}(t, X_t)\, d[X,X]_t, \tag{4.7}$$

and we have

$$d[t,t] = 0,\ d[t,W]_t = 0,\ d[W,W]_t = dt. \tag{4.8}$$

Thus, Itô's formula is constructed below:

$$dZ_t = \left\{ \frac{\partial f}{\partial t}(t, X_t) + F_t \frac{\partial f}{\partial x} + \frac{1}{2} G_t^2 \frac{\partial^2 f}{\partial x^2}(t, X_t) \right\} dt + G_t \frac{\partial f}{\partial x}(t, X_t)\, dW_t. \tag{4.9}$$

Theorem 4.3 (Itô's formula for the Jump-Diffusion processes) *Suppose X is a diffusion process with jumps, which is defined by the composition of the drift term, the volatility term, and a compound Poisson process (Proposition 8.14 [13]):*

$$X_t = X_0 + \int_0^t F_s\,ds + \int_0^t G_s\,dW_s + \sum_{i=1}^{N_t} \Delta X_i,\ t \in [0,T], \tag{4.10}$$

where F_t and G_t are continuously unpredictable process with

$$\mathbb{E}\left[\int_0^T G_s^2\,dt\right] < \infty. \tag{4.11}$$

Then, for any function $f(t, X_t)$ that is differentiable in t and twice differentiable in x, the process $Z_t = f(t, X_t)$ can be represented as:

$$\begin{aligned} Z_t - Z_0 = &\int_0^t \left[\frac{\partial f}{\partial s}(s, X_s) + \frac{\partial f}{\partial x}(s, X_s)F_s\right] ds \\ &+ \frac{1}{2}\int_0^t G_s^2 \frac{\partial^2 f}{\partial x^2}(s, X_s)\,ds + \int_0^t \frac{\partial f}{\partial x}(s, X_s)G_s\,dW_s \\ &+ \sum_{\{i\geq 1, T_i \leq t\}} [f(X_{T_{i_-}} + \Delta X_i) - f(X_{T_{i_-}})]. \end{aligned} \tag{4.12}$$

In differential notation:

$$dZ_t = \frac{\partial f}{\partial t}(t, X_t)\,dt + F_t\frac{\partial f}{\partial x}(t, X_t)\,dt + \frac{G_t^2}{2}\frac{\partial^2 f}{\partial x^2}(t, X_t)\,dt + \frac{\partial f}{\partial x}(t, X_t)G_t\,dW_t + [f(X_{t_-} + \Delta X_t) - f(X_{t_-})]. \tag{4.13}$$

4.2 The Feynman-Kac formula

The Feynman-Kac formula is established to represent solutions to a partial family of partial differential equations (Section 3.2.3 [36]).

Theorem 4.4 *We define a Stochastic Differential Equation, let X to be a stochastic process that satisfies:*

$$dX_t = \alpha_t dt + \beta_t dW_t, \tag{4.14}$$

where α_t and β_t are deterministic. We will need to consider solutions to this SDE where we vary the initial value of x, and also the time at which the 'initial' value occurs. For given $x \in \mathbb{R}$ and $t \in [0,T]$, we define $\mathbb{E}_{t,x}$ to specify the conditional expectation with the initial value $X_t = x$ is known. Then, we are interested in X_s during the time $s \in [t,T]$, first we interpret a second-order parabolic PDE with a terminal boundary condition.

Consider $\Phi(t,x)$ where $t \in [0,T]$ and $x \in \mathbb{R}$. We will begin by looking at the partial differential equation

$$\begin{aligned} \frac{\partial \Phi}{\partial t}(t,x) + \alpha(t,x)\frac{\partial \Phi}{\partial x}(t,x) + \frac{1}{2}\beta(t,x)^2\frac{\partial^2 \Phi}{\partial x^2}(t,x) = 0, \\ \Phi(T,x) = g(x), \end{aligned} \tag{4.15}$$

here, $\alpha(t,x)$ and $\beta(t,x)$ are deterministic continuous functions, and $g(x)$ is a function known as the boundary condition. Then, the connection will be illustrated by the Lemma 4.1.

Lemma 4.1 *Suppose that Φ is a solution of (4.15), and also that $\beta(t,x)\frac{\partial \Phi}{\partial x}(t, X_t)$ is in $\mathcal{H}^2$. Then,*

$$\Phi(t,x) = \mathbb{E}_{t,x}[\Phi(X_T)] \tag{4.16}$$

for all $x \in \mathbb{R}$ and $t \in [0,T]$.

4.3 Other

Definition 4.4 (Lévy Process) *An stochastic real-valued process $X = \{X_t : t > 0\}$ is called Lévy process if (Definition 3.1 [13]):*

1. *$X_0 = 0$ almost surely.*
2. *Independent increments: for any $0 \leq t_0 < t_1 < ... < t_n$, the random variables X_{t_0}, $X_{t_1} - X_{t_0}, ..., X_{t_n} - X_{t_{n-1}}$ are independent.*
3. *Stationary increments: For any $s < t$, $X_t - X_s$ is equal in distribution to X_{t-s}.*
4. *Continuity in probability: For any constant $\epsilon > 0$ and $t \geq 0$, it holds that $\mathbb{P}(|X_{t+\Delta t} - X_t| > \epsilon) \to 0$ as $\Delta t \to 0$.*
5. *X_t is a cadlag process.*

Theorem 4.5 (Fourier Transform) *(Section 3 [24]) Let $f(x)$ be the probability density function over $(-\infty, \infty)$ of the random variable X and $F_X(x) = \mathbb{P}(X \leq x)$ be the distribution function. Then, we have:*

$$\int_{-\infty}^{\infty} |f(x)|\, dx < \infty. \tag{4.17}$$

The Fourier transform of $f(x)$ will then be defined as:

$$F_f(\omega) = \int_{-\infty}^{\infty} e^{i\omega y} f(y)\, dy. \tag{4.18}$$

After Fourier inversion, we will get:

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-i\omega x} F_f(\omega)\, d\omega \tag{4.19}$$

By the above definition, we could also define the characteristic function to be:

$$F_f(\omega) = \mathbb{E}[e^{i\omega y}]. \tag{4.20}$$

When we choose ω to be complex, the Fourier inversion will become:

$$f(x) = \frac{1}{2\pi} \int_{i \cdot Im(\omega) - \infty}^{i \cdot Im(\omega) + \infty} e^{-i\omega x} F_f(\omega)\, d\omega. \tag{4.21}$$

5 Pricing Method

As we have mentioned the main pricing method of this paper is called Fourier Cosine Expansion (COS) method. To fully understand the concept of this method, I will firstly describe the background of this method which is linking to the Fast Fourier Transform (FFT) Algorithms.

5.1 FFT Algorithms for pricing the options

Lewis in 2001 [24] illustrated the FFT Algorithm formally and Kwok, Leung and Wong in 2012 [23] furthered this idea, I am going to describe this algorithm closely with referencing their descriptions in the article.

We first define $V(S_t)$ to be the current price of a European-style option with payoff function $V_T(x) \geq 0$, where $x = \ln S_T$. Then, we assume that $V_T(x)$ is Fourier integrable with Fourier transform $F_{V_T}(\omega)$, where $\omega \in \mathbb{C}$. In addition, we let

$$S_t = S_0 \exp\{(r - q)t + X_t\}, t > 0, \tag{5.1}$$

where X_t is defined as a Lévy process, r is the risk-free interest rate, and $\exp(X_t)$ is a martingale. This definition has also been defined by Filipović (2009) [18] for Heston Stochastic Volatility Model.

Then, following the property of the martingale pricing, and the Fourier inversion that we denoted in equation 4.21, the European option value would be defined as below:

$$V(S_t, t) = e^{-r(T-t)}\mathbb{E}[V_T(x)], \tag{5.2}$$

we then substitute $V_T(x)$ to be the transformed formula:

$$V(S_t, t) = \frac{e^{-r(T-t)}}{2\pi}\mathbb{E}\left[\int_{i\cdot Im(\omega)-\infty}^{i\cdot Im(\omega)+\infty} S_T^{-i\omega} F_{V_T}(\omega)\, d\omega\right], \tag{5.3}$$

plug the S_t that we define in 5.1 into 5.3, we get:

$$V(S_t, t) = \frac{e^{-r(T-t)}}{2\pi}\mathbb{E}\left[\int_{i\cdot Im(\omega)-\infty}^{i\cdot Im(\omega)+\infty} \exp\{-i\omega[\ln S_0 + (r-q)T]\} \cdot \exp\{-i\omega X_T\} F_{V_T}(\omega)\, d\omega\right]. \tag{5.4}$$

Lewis [24] proved that the swapping of expectation and integration can be made by proving the option valuation theorem is true. We then make the swap and get the final pricing formula to be:

$$V(S_t, t) = \frac{e^{-r(T-t)}}{2\pi}\int_{i\cdot Im(\omega)-\infty}^{i\cdot Im(\omega)+\infty} e^{-i\omega[\ln S_0 + (r-q)T]}\phi_{X_T}(-\omega) F_{V_T}(\omega)\, d\omega, \tag{5.5}$$

where, by definition, we have $\phi_{X_T}(-\omega)$ as a Characteristic function of X_T.

Until now, the FFT Algorithm has been described properly, and this is building up our understanding of the next section, where I will further up to illustrate another algorithm that has been used to faster the pricing speed of options [17].

5.2 Fourier Cosine Expansion (COS) method

Based on the FFT Algorithm, an alternative COS method is used by Moody's ESG to make a highly efficient recovering density function for pricing derivatives and is used in the market calibration tool. We follow up the descriptions from Fang and Oosterlee (2008) [17], we describe the change of the density first, then the pricing process will be illustrated afterward.

First of all, from the Fourier-cosine series expansion, we extract series coefficients directly from the integrand:

$$f(\theta) = \sum_{k=0}^{\infty} q_k A_k \cos(k\theta) \quad \text{with} \quad A_k = \frac{2}{\pi}\int_0^{\pi} f(\theta)\cos(k\theta)\, d\theta, \tag{5.6}$$

where $q_0 = 1/2$, and $q_k = 1$ represents the first term in the summation of the expansion is weighted by one-half. To make a flexible range selection, we could use the technique that is shown in 5.7 below to let the integration support any finite interval, say $[a, b] \in \mathbb{R}$,

$$\theta := \frac{x-a}{b-a}\pi \quad \text{with} \quad x = \frac{b-a}{\pi}\theta + a. \tag{5.7}$$

Therefore, the density function will be presented as:

$$f(x) = \sum_{k=0}^{\infty} q_k A_k \cos\left(k\pi \frac{x-a}{b-a}\right) \quad \text{with} \quad A_k = \frac{2}{(b-a)} \int_b^a f(x) \cos\left(k\pi \frac{x-a}{b-a}\right) dx. \tag{5.8}$$

Fang and Oosterlee [17] give the fact that any real function has a cosine expansion when it is finitely supported. Under this fact, we would assume that $[a, b] \in \mathbb{R}$ is chosen s.t. the truncated integral approximates the infinite counterpart, for example,

$$\phi'(\omega) := \int_a^b e^{i\omega x} f(x)\, dx \approx \int_{-\infty}^{\infty} e^{i\omega x} f(x)\, dx = \phi(\omega). \tag{5.9}$$

Then, by replacing the terms in 5.8 with the Fourier transform or Characteristic function in the interval $[a, b]$, we will obtain the cos coefficient:

$$A_k \equiv \frac{2}{(b-a)} Re\left\{\phi'\left(\frac{k\pi}{b-a}\right) \exp\left\{-i\frac{ka\pi}{b-a}\right\}\right\}, \tag{5.10}$$

where $Re\{\cdot\}$ represents taking the real part of the input.

5.2.1 Pricing under COS method

Under the martingale pricing we have,

$$V(S_t, t) = e^{-r(T-t)} \mathbb{E}[V(S_T)|S_t] = e^{-r(T-t)} \int_a^b V(S_T) f(S_T|S_t)\, dS_T, \tag{5.11}$$

as the density function $f(S_T|S_t)$ normally not known, but we know the characteristic function, thus, we replace the density function by its cosine expansion in S_T:

$$f(S_T|S_t) = \sum_{k=0}^{+\infty} q_k A_k \cos\left(k\pi \frac{S_T - a}{b-a}\right) \quad \text{with} \quad A_k := \frac{2}{b-a} \int_a^b f(S_T|S_t) \cos\left(k\pi \frac{S_T - a}{b-a}\right) dS_T, \tag{5.12}$$

thus, we would be able to do the substitution of the density function and indicate the valuation function to be:

$$V(S_t, t) = e^{-r(T-t)} \int_a^b V(S_T) \sum_{k=0}^{+\infty} q_k A_k \cos\left(k\pi \frac{S_T - a}{b-a}\right) dS_T, \tag{5.13}$$

as we have some terms in the equation 5.13 as constants and in terms of S_T, so they would be able to be integrated directly. Therefore, we define,

$$H_k := \frac{2}{b-a} \int_a^b V(S_T) \cos\left(k\pi \frac{S_T - a}{b-a}\right) dS_T. \tag{5.14}$$

By interchanging the integration and summation in equation 5.13, we will have

$$V(S_t, t) = \frac{1}{2}(b-a) e^{-r(T-t)} H_k \sum_{k=0}^{+\infty} q_k A_k, \tag{5.15}$$

and so, the truncated series summation will be:

$$V'(S_t, t) = \frac{1}{2}(b-a) e^{-r(T-t)} H_k \sum_{k=0}^{N-1} q_k A_k. \tag{5.16}$$

Then, we recall the coefficients A_k in terms of the characteristic function, resulting

$$V(S_t,t) \approx V'(S_t,t) = e^{-r(T-t)} H_k \sum_{k=0}^{N-1} q_k Re\left\{\phi\left(\frac{k\pi}{b-a}; S_t\right) e^{-ik\pi\frac{a}{b-a}}\right\}. \tag{5.17}$$

5.3 Coefficient $\mathbf{H}_k$

We have defined the coefficient H_k above, in case of applying the pricing formula in practice, we would need to properly solve or define this unknown coefficient in this subsection.

As we assumed that the characteristic function is known, then defining the payoff function to be in terms of log formed underlying:

$$V(\ln(S_T)) \equiv [\alpha K(e^{\ln(S_T)} - 1)]^+, \tag{5.18}$$

where, $\alpha = 1$ if we are pricing call option, $\alpha = -1$ if we are pricing a put option, and K is the strike price. For simplicity, we let $\varepsilon = \ln(S_T)$ for later presentation.

In case of the call option pricing, we have:

$$H_k^{Call} = \frac{2}{b-a}\int_a^b K(e^{\varepsilon} - 1)\cos\left(k\pi\frac{\varepsilon - a}{b-a}\right)\, d\varepsilon, \tag{5.19}$$

we then for simplicity, define:

$$\chi_k(c,d) := \int_c^d e^{\varepsilon}\cos\left(k\pi\frac{\varepsilon - a}{b-a}\right)\, d\varepsilon, \tag{5.20}$$

which has the result by using simple calculus [5][17] and we let $\zeta_k = \frac{k\pi}{(b-a)}$:

$$\chi_k(c,d) := \frac{1}{1+\zeta_k^2}\left[e^d\cos\left(\zeta_k(d-a)\right) - e^c\cos\left(\zeta_k(c-a)\right) + e^d\zeta_k\sin\left(\zeta_k(d-a)\right) - e^c\zeta_k\sin(\zeta_k(c-a))\right]. \tag{5.21}$$

And we also define:

$$\psi_k(c,d) := \int_c^d \cos\left(k\pi\frac{\varepsilon - a}{b-a}\right)\, d\varepsilon, \tag{5.22}$$

which has the result:

$$\psi_k(c,d) := \begin{cases} \frac{1}{\zeta_k}\left[\sin(\zeta_k(d-a)) - \sin(\zeta_k(c-a))\right] & k > 0 \\ d - c & k = 0. \end{cases} \tag{5.23}$$

Then, the Call and Put option's value of coefficients are presented below, for the call option:

$$H_k^{Call} = \frac{2}{b-a}K(\chi_k(a^+,b) - \psi_k(a^+,b)), \tag{5.24}$$

and for put option:

$$H_k^{Put} = \frac{2}{b-a}K(-\chi_k(a,b^-) + \psi_k(a,b^-)). \tag{5.25}$$

Finally, we have got and fully defined everything that we need for the COS method of pricing the options. In the next section, I will start giving out the COS method result of pricing under the SVJD model, and as we assumed that the characteristic functions are known at the beginning, we will also illustrate the characteristic functions of the SVJD model.

5.4 COS method under Stochastic Volatility Jump Diffusion Model

SVJD as the built-in model of Moody's Scenario Generator is using the COS method to calibrate with the market data. In this section, the result of the pricing process will be illustrated by using the COS method that we discussed above.

First of all, as the above discussions are based on the characteristic functions of the asset price are known, thus, we need to define the characteristic function ϕ of the SVJD model.

Let X_T to be the logarithmic asset dynamics:

$$X_T = \ln\left(\frac{S_T}{K}\right) = \ln\left(\frac{S_T^{XS}}{\tilde{K}}\right) \quad \text{with} \quad \tilde{K} = Ke^{-rT} \text{ is the forward strike price.} \tag{5.26}$$

Then, define $\phi\left(\zeta; X_T\right)$ to be the characteristic function of the X_T, we have:

$$\begin{aligned}
\phi\left(\zeta; X_T\right) &= \mathbb{E}\left[e^{i\zeta X_T} | X_t = x, v_t = v_0\right] \\
&= \mathbb{E}\left[e^{i\zeta \ln\left(\frac{S_T^{SV}}{\tilde{K}}\right)} e^{i\zeta \ln\left(S_T^{JD}\right)} | X_t^{SV} = x^{SV}, X_t^{JD} = x^{JD}, v_t = v_0, \right] \\
&= \mathbb{E}\left[e^{i\zeta \ln\left(\frac{S_T^{SV}}{\tilde{K}}\right)} | X_t^{SV} = x^{SV}\right] \mathbb{E}\left[e^{i\zeta \ln\left(S_T^{JD}\right)} | X_t^{JD} = x^{JD}\right] \\
&= \phi^{SV}(\zeta)\phi^{JD}(\zeta).
\end{aligned} \tag{5.27}$$

For getting characteristic functions for both the Heston model and Merton model, the proof can be found in Moody's (2013) [4]. The proof of the Heston characteristic function applied the basic martingale, and tower properties and used Itô's lemma to define the characteristic function in the form of the partial differential equation (PDE). Then make a good guess of terminal value, and substitute it back to the PDE, the result can be presented in terms of parameters:

$$\phi^{SV}(\zeta) = e^{i\zeta \ln(\frac{S_0^{SV}}{K}) + A + BV_0}, \tag{5.28}$$

with

$$\begin{aligned}
A &= \frac{\alpha\theta}{\xi^2}\left((\beta - D)T - 2\ln\left(\frac{1 - Ge^{-DT}}{1 - G}\right)\right), \qquad B = \frac{1}{\xi^2}\frac{1 - e^{-DT}}{1 - Ge^{-DT}}(\beta - D), \\
G &= \frac{\beta - D}{\beta + D}, \qquad D = \sqrt{\beta^2 + (\zeta^2 + i\zeta)\xi^2}, \qquad \beta = \alpha - i\rho\xi\zeta.
\end{aligned} \tag{5.29}$$

Moody's [4] also provided the proof of getting Merton's characteristic function. The law of iterated expectations, log normal expectation, and Poisson property have been used, the result is presented as:

$$\phi^{JD}(\zeta) = \exp\left(-i\zeta\lambda\bar{\mu}T + \lambda T\left(e^{i\zeta\mu_J - \frac{1}{2}\zeta^2\sigma_J^2 - 1}\right)\right). \tag{5.30}$$

5.5 COS method under SVJD model

Now we have everything we need to eventually present the valuation formula of the options. Assume that we have held a call option that expires at time T with a deterministic interest rate, and we want to simulate the price of the option under the SVJD model, the COS method gives us the result:

$$V_0 = \tilde{K} H_k \sum_{k=0}^{N-1} q_k Re[\phi(\zeta_k) e^{-i\zeta_k a}], \tag{5.31}$$

where

$$H_k = \frac{2}{b-a}(\chi_k(a^+, b) - \psi_k(a^+, b)), \tag{5.32}$$

where $\tilde{K} = Ke^{-rT}$ is the forward strike price, r is the risk-free interest rate, $\zeta_k = \frac{k\pi}{(b-a)}$, $q_k = 1/2$ if $k = 0$, $q_k = 1$ if $k > 0$, $[a, b]$ is the interval of the cosine series truncated after N terms, V_0 represents the value of the call option at initial time point.

6 Parameter Identification and Redundancy check

In this section, we will focus on the explanation of parameter identification and the redundancy check. In the case of looking for the pattern of IV changes with the variation of the values of parameters, market Consistent Calibration in Section 2 has been used to generate the IV data set. Then, particular plots made by using the data we created will visualise an obvious feature of the modeled data.

Recall: The SVJD model has eight parameters that lend themselves for frequent Market Consistent Calibration updates:

SV process: MeanReversionLevel (θ), MeanReversionRate (α), Volatility (ξ), Correlation (ρ), and InitialValue (v_0) are controlling the continuous Stochastic Volatility process;

JD process: ArrivalRate (λ), JumpMean (μ_J), JumpVolatility (σ_J) are controlling the discrete Jump process.

The set of all these parameters is defined as Π.

6.1 Theoretical Approach

Associated with the Section 3, we let $X_t = \ln S_t^{XS}$, $X_t^{SV} = \ln S_t^{SV}$, and $X_t^{JD} = \ln S_t^{JD}$. The SDEs of two processes are presented:

$$dX_t^{SV} = \left(r - \frac{v_t}{2}\right) dt + \sqrt{v_t}\, dW_t^1 \tag{6.1}$$

and

$$dX_t^{JD} = -\lambda m(1)\, dt + \ln J\, dN(t), \tag{6.2}$$

where, $m(z) = \exp(\mu_J \cdot z + \frac{1}{2}\sigma_J^2 z^2) - 1$ evaluated only at $z = 1$ and $z = i\zeta$. In addition, we have:

$$X(T) = X_T^{SV} + X_T^{JD}. \tag{6.3}$$

In Equation 5.27, we defined the Characteristic function as following:

$$\phi\left(\zeta; X_T\right) = \phi^{SV}(\zeta)\phi^{JD}(\zeta), \tag{6.4}$$

use Itô's product rule, we then have:

$$\frac{\partial \phi}{\partial t} = \frac{\partial \phi^{SV}}{\partial t}\phi_t^{JD} + \frac{\partial \phi^{JD}}{\partial t}\phi_t^{SV}. \tag{6.5}$$

We apply Feynman-Kac formula directly to the Heston model, then we get:

$$-\frac{\partial \phi^{SV}}{\partial t} = (r - \frac{v_t}{2})\frac{\partial \phi^{SV}}{\partial X_t^{SV}} + \alpha(\theta - v_t)\frac{\partial \phi^{SV}}{\partial v_t} + \frac{v_t}{2}\frac{\partial^2 \phi^{SV}}{\partial X_t^{SV\,2}} + \rho\xi v_t \frac{\partial^2 \phi^{SV}}{\partial X_t^{SV}\partial v_t} + \frac{1}{2}\xi^2 v_t \frac{\partial^2 \phi^{SV}}{\partial v_t^2}, \quad (6.6)$$

and the Characteristic function of the Jump Diffusion model is what we defined in 5.30, we recall it here:

$$\phi^{JD}(\zeta) = e^{-i\zeta\lambda m(1)(T-t)+\lambda m(i\zeta)(T-t)}, \quad (6.7)$$

then the first derivative of ϕ^{JD} w.r.t. t will be:

$$\frac{\partial \phi^{JD}}{\partial t} = e^{-i\zeta\lambda m(1)(T-t)+\lambda m(i\zeta)(T-t)}(i\zeta\lambda m(1) - \lambda m(i\zeta)). \quad (6.8)$$

Substituting 6.6 and 6.7 back into 6.5, we have:

$$\frac{\partial \phi}{\partial t} = \frac{\partial \phi^{SV}}{\partial t}\phi_t^{JD} + (i\zeta\lambda m(1) - \lambda m(i\zeta))\phi_t. \quad (6.9)$$

6.1.1 Result

Under the presentation of Equation 6.9, it is obvious that parameter ArrivalRate λ is proportional to the function $m(z)$, and $m(z)$ is a function that contains parameters JumpMean μ_J and parameter JumpVolatility σ_J. Thus, if we fix the values of ζ, $\lambda m(1) = y_1$, and $\lambda m(i\zeta) = y_2$, then there always be a pair of μ_J and σ_J to fit the change of λ, i.e., let λ_0, $\mu_{J,0}$, and $\sigma_{J,0}$ be the median parameter values, and $m_0(z) = \exp\{\mu_{J,0}z + \frac{1}{2}\sigma_{J,0}^2 z^2\} - 1$, then if we change λ to λ_0, the values of μ_J and σ_J will align to $\mu_{J,0}$ and $\sigma_{J,0}$ in order to reach the fixed values of y_1 and y_2. Therefore, this change in λ can approximately offset by changes in μ_J and σ_J.

6.2 Implied Volatility Generation

In this subsection, we will cooperate with the calibration process to describe the generation of the implied volatilities. Appendix A.1 shows a powerful python function that we use for making the IV data set for the value variation of each parameter.

By inputting the controlled/stressed parameter's name into the function, the function will then go through the steps we coded which are shown in Appendix A.1 with 1 sample size and 52 annually time steps, and the calibration will be done for each parameter. As the return from the function of each parameter, we will get three 53×178 shaped excel files, 53 includes time 0 and 178 includes two columns of time step and trial index[10]. In the data set, it includes all the market calibrated $IV(K, M, \pi)$ which have been calculated using the ESG's Monte Carlo-based LMM+ SVJD Test Analysis outputs for

1. Those ForwardStrike2 values $K \in \{0.5, 0.6, ..., 1.5\}$, and maturities $M \in \{0.25, 0.50, ..., 1, 2, ..., 10, 15, ..., 25\}$. All IVs are presented as nodes on the target set of the SVJD calibration tool.

   - ForwardStrike2: here the strike price is defined as the proportion of the forward price, which can be seen in the formula:

   $$\text{Strike Price} = \frac{S_T}{K} \quad (6.10)$$

   where S_T represents the terminal price of the underlying asset, K represents the actual strike price. This presentation is also named Moneyness, it is telling investors whether they are making a profit or loss. A loss would be indicated as out of the money, a profit is indicated as in the money, and break-even will be known as at the money.

[10]The data set is too large, the example is available on GitHub.

2. Stressed parameters p, $p \in \Pi$, obtained from base parameters by changing the value of exactly one of the parameters in Π by 0.01 and keeping other parameters with the median values.

Then, using the functions in Appendix A.2 to sort out the target data sets, we would be able to get a cleaned and sorted pivot tables for parameters, one example is shown in Table 2. There are 24 tables created in total by using the function for analysing the parameter changes.

Strike Maturity	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3	1.4	1.5
0.25	0.542848	0.471314	0.388329	0.296148	0.213164	0.167677	0.147530	0.335988	0.183776	0.217468	0.245384
0.50	0.422215	0.372959	0.316274	0.257382	0.207746	0.173776	0.153008	0.144630	0.150345	0.167843	0.187069
0.75	0.368833	0.329921	0.286944	0.244022	0.206775	0.177976	0.157847	0.147067	0.145507	0.151891	0.163297
1.00	0.337864	0.305562	0.271175	0.237166	0.206570	0.181210	0.162076	0.150306	0.145912	0.147419	0.153194
2.00	0.284545	0.264740	0.245037	0.225507	0.206723	0.189461	0.174585	0.162919	0.154928	0.150408	0.148606
3.00	0.264022	0.248964	0.234521	0.220439	0.206828	0.194017	0.182431	0.172487	0.164487	0.158526	0.154464
4.00	0.252799	0.240095	0.228487	0.217406	0.206804	0.196787	0.187538	0.179256	0.172106	0.166178	0.161475
5.00	0.244978	0.234306	0.224531	0.215374	0.206722	0.198574	0.190990	0.184054	0.177846	0.172420	0.167797
6.00	0.239559	0.230198	0.221738	0.213925	0.206626	0.199790	0.193415	0.187525	0.182149	0.177314	0.173030
7.00	0.235467	0.227129	0.219667	0.212848	0.206536	0.200658	0.195183	0.190101	0.185415	0.181130	0.177247
8.00	0.232264	0.224748	0.218072	0.212020	0.206458	0.201305	0.196516	0.192066	0.187939	0.184130	0.180631
9.00	0.229686	0.222848	0.216809	0.211367	0.206393	0.201806	0.197554	0.193603	0.189930	0.186522	0.183365
10.00	0.227566	0.221297	0.215783	0.210839	0.206340	0.202207	0.198384	0.194834	0.191532	0.188458	0.185598
15.00	0.220889	0.216469	0.212636	0.209242	0.206192	0.203419	0.200875	0.198526	0.196344	0.194308	0.192402
20.00	0.217363	0.213956	0.211022	0.208440	0.206131	0.204042	0.202134	0.200376	0.198748	0.197231	0.195811
25.00	0.215187	0.212417	0.210041	0.207958	0.206101	0.204425	0.202897	0.201493	0.200193	0.198984	0.197853

Table 2: ESG generated target sets of implied volatilities for parameter Correlation ρ (The Correlation parameter See Equation 3.3) under the Strikes defined in the range of 0.5 to 1.5, and Maturities 0.25 to 25.

Then the change of the response of option IVs to the parameter stress (p_{Max}, p_{Median}, p_{Min}) was calculated as:

$$\Delta_{Max-Median}(K, M, p) = \frac{IV(K, M, p_{Max}) - IV(K, M, p_{Median})}{||IV(K, M, p_{Max}) - IV(K, M, p_{Median})||_2}, \tag{6.11}$$

and

$$\Delta_{Median-Min}(K, M, p) = \frac{IV(K, M, p_{Median}) - IV(K, M, p_{Min})}{||IV(K, M, p_{Median}) - IV(K, M, p_{Min})||_2}, \tag{6.12}$$

here we used the normalisation to make the magnitude of the response to be the same for every parameter stress at time 0.

6.2.1 Result

Then by using these 24 pivot tables, we can generate pair plots for each parameter. As ESG is possible to misprice some data points, we will delete these points from the original IV target set, thus, individual points or paths can appear in the plots, but the generalisation of the pattern will still be highly informative:

- **Correlation** ρ

  - Two plots for stressed **Correlation** parameter are shown in Figure 5. As we can see that change of ρ will lead the IVs with the longer option maturities to have a more stable increment trend with the increase of the strike. On the other hand, the shorter the maturities, the more unstable the IVs will be, and the pattern seems to change from convex to concave after the strike from 1.0 to 1.1. Therefore, we conclude that the change of the parameter ρ will have more effects on the short-term derivative option, it won't influence the IV very much for the long-term derivative option. In contrast, the two plots are showing that the

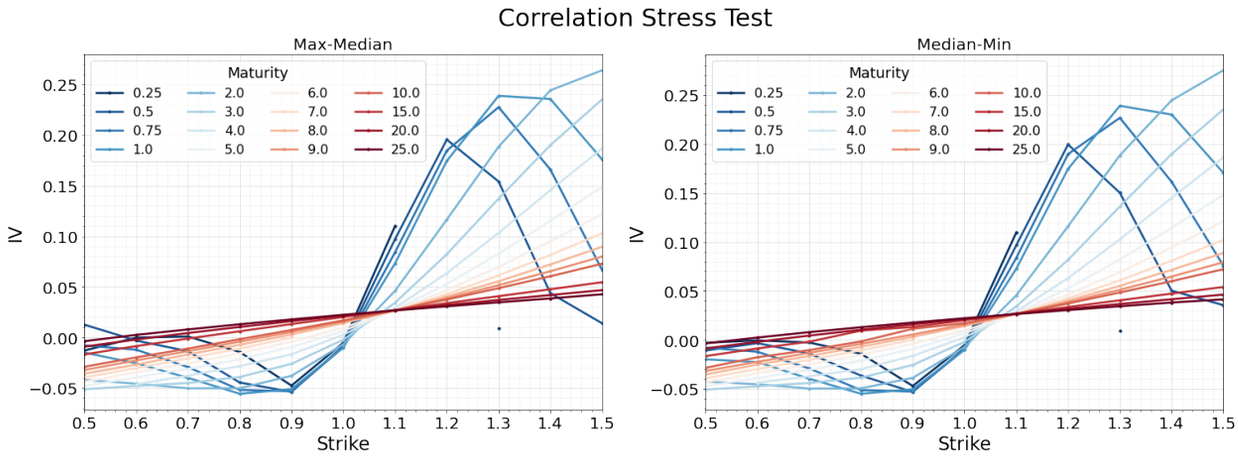

Figure 5: The plots are showing the normalised IV plots of the stressed Correlation ρ. Therefore, how does IV change with stressing the parameter ρ. On the left, the Max stressed with Median, the right shows the stress of Median and Min.

data we got are reliable as they are both having a similar pattern.

Note: the legend in Figure 5 will be shared with the following plots.

- **Jump Arrival Rate** λ

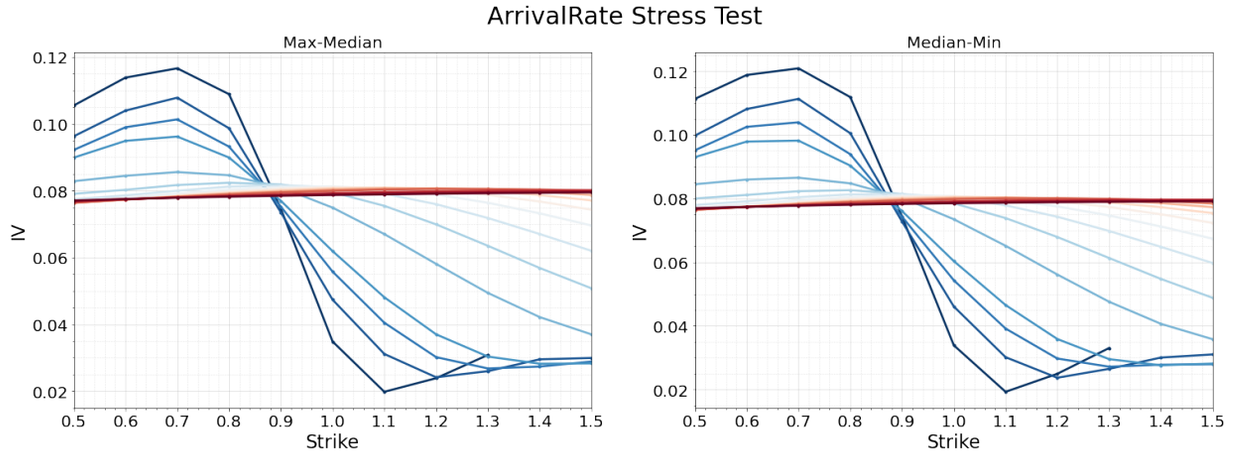

Figure 6: The plots are showing the normalised IV plots of the stressed Arrival Rate λ. Therefore, how does IV change with stressing the parameter λ. On the left, the Max is stressed with Median, the right shows the stress of Median and Min.

- Two plots for stressed **Jump Arrival Rate** parameter are shown in Figure 6. The similar feature of the change of the parameter size will more likely affect the options with shorter maturities is revealed. Besides, the pattern otherwise shows an opposite direction, the IV changes with a concave shape for the strikes lower than about 0.88, and the opposite for the strikes higher than about 0.88. Two plots are illustrating the same pattern, thus, the analysis is reliable.

- **Jump Mean** μ_J

- Two plots for stressed **Jump Mean** parameter are shown in Figure 7. The similar maturity feature is still applied. The difference, in this case, is that IVs changes are

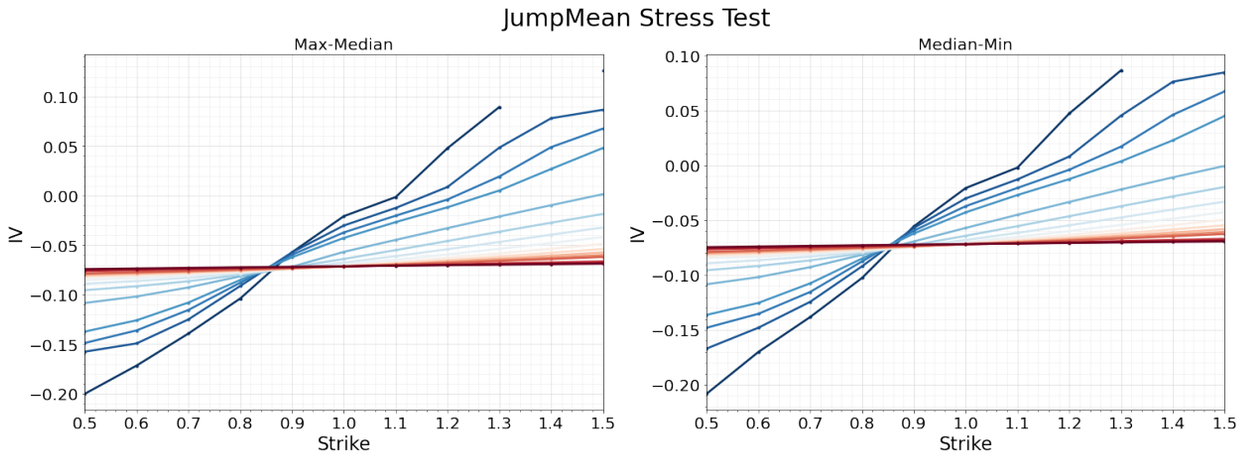

Figure 7: The plots are showing the normalised IV plots of the stressed Jump Mean μ_J. Therefore, how does IV change with stressing the parameter μ_J. On the left, the Max is stressed with Median, the right shows the stress of Median and Min.

following a monotonic increasing trend concerning increasing of strikes. Then, two plots are showing the reliability of the analysis.

- **Jump Volatility** σ_J

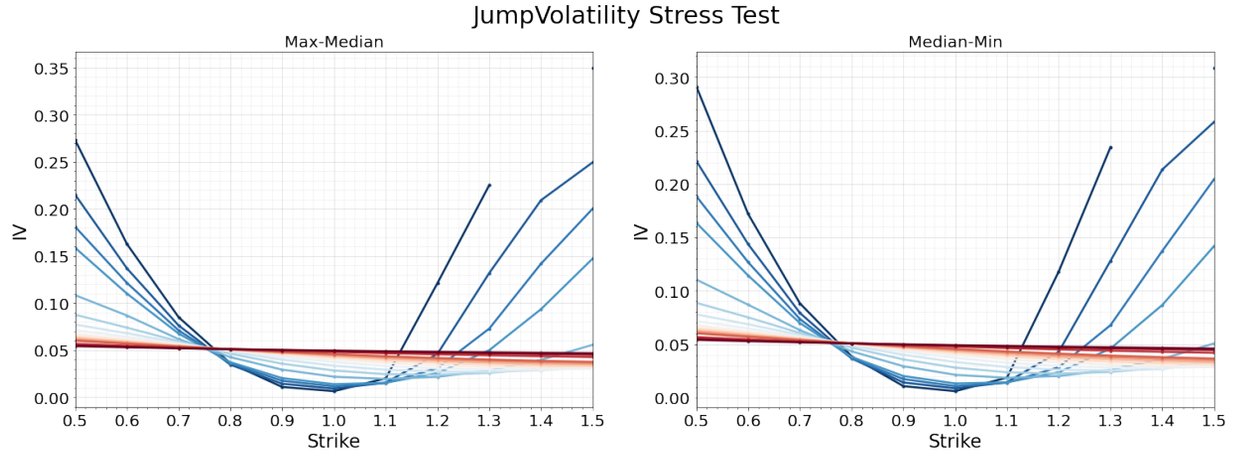

Figure 8: The plots are showing the normalised IV plots of the stressed Jump Volatility σ_J. Therefore, how does IV change with stressing the parameter σ_J. On the left, the Max is stressed with Median, the right shows the stress of Median and Min.

- Two plots for stressed **Jump Volatility** parameter are shown in Figure 8. The similar maturity feature is still applied. Additionally, the diagram has a convex shape, therefore, the lowest IV change is happening when the strike is around 1.0, therefore, σ_J will have the lowest effect on the options which have short maturities but strikes around 1.0. Two plots are showing the reliability of the analysis.

- **Mean Reversion Level** θ

- Two plots for stressed **Mean Reversion Level** parameter are shown in Figure 9. The change of IVs for the longer maturities seem still having a stable change, but the change become larger and larger while the maturities increase. And IVs vary with the trends of concave shape while keep in a shorter maturities. The change of parameter seems unlikely

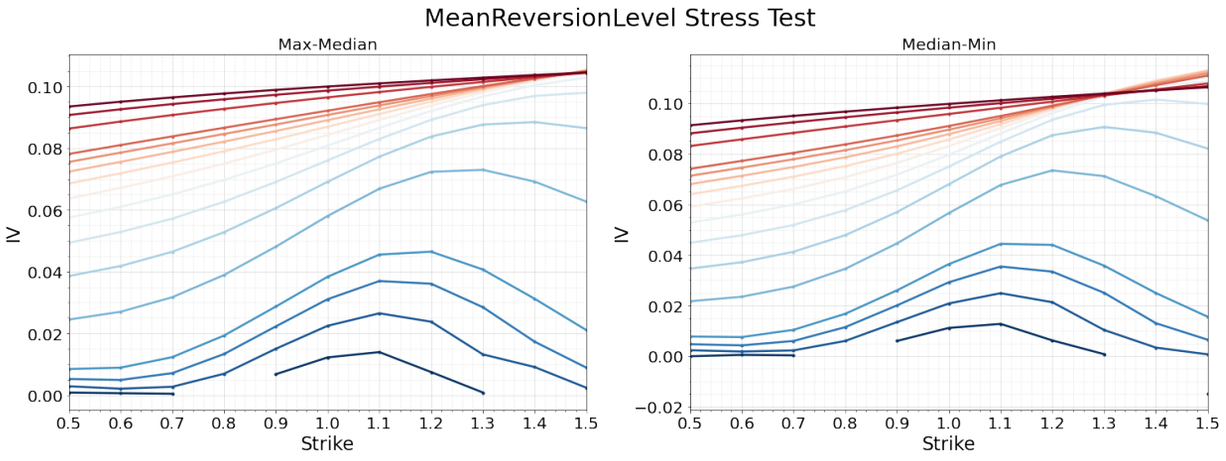

Figure 9: The plots are showing the normalised IV plots of the stressed Mean Reversion Level θ. Therefore, how does IV change with stressing the parameter θ. On the left, the Max is stressed with Median, the right shows the stress of Median and Min.

to affect options that have small strike and shorter maturities. Two plots present the reliability of the analysis.

- **Mean Reversion Rate** α

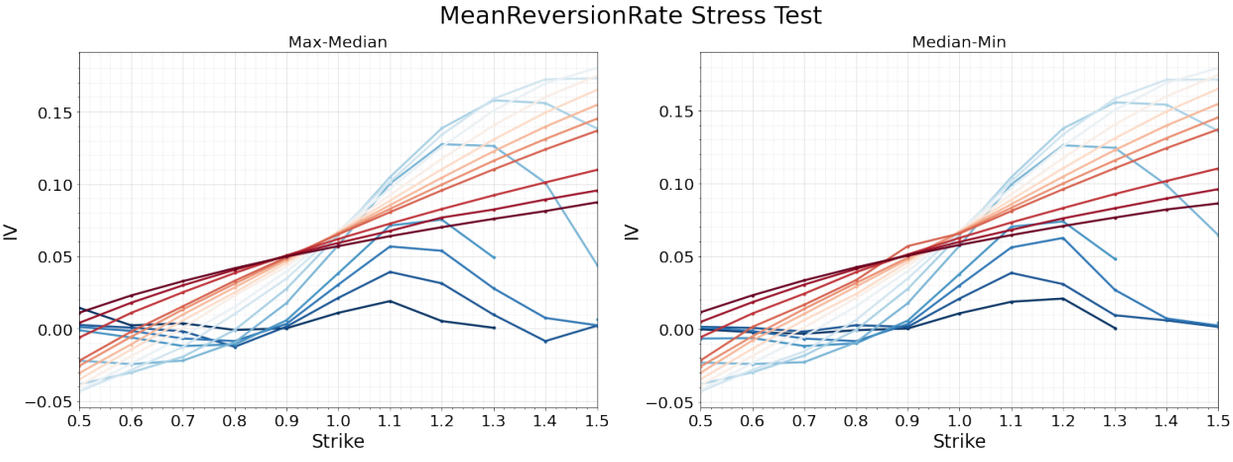

Figure 10: The plots are showing the normalised IV plots of the stressed Mean Reversion Rate α. Therefore, how does IV change with stressing the parameter α. On the left, the Max is stressed with Median, the right shows the stress of Median and Min.

- Two plots for stressed **Mean Reversion Rate** parameter are shown in Figure 10. Stressing α seems IV will alter with a big range, but the longer the maturities the change of the IVs will not have a horizontal like pattern, instead, it has a straight line but larger gradient trends. For the options with shorter maturities, IVs look very unstable. Therefore, whether α would be a redundant variable will be discussed in later section. Reliability is presented by two plots.

- **Initial Value** v_0

- Two plots for stressed **Initial Value** parameter are shown in Figure 11. The plot shows an extremely good result for pricing the longer maturities as the IV changes are almost zero for all strikes. But the bell-shaped pattern is appearing for shorter maturities. Reliability is confirmed.

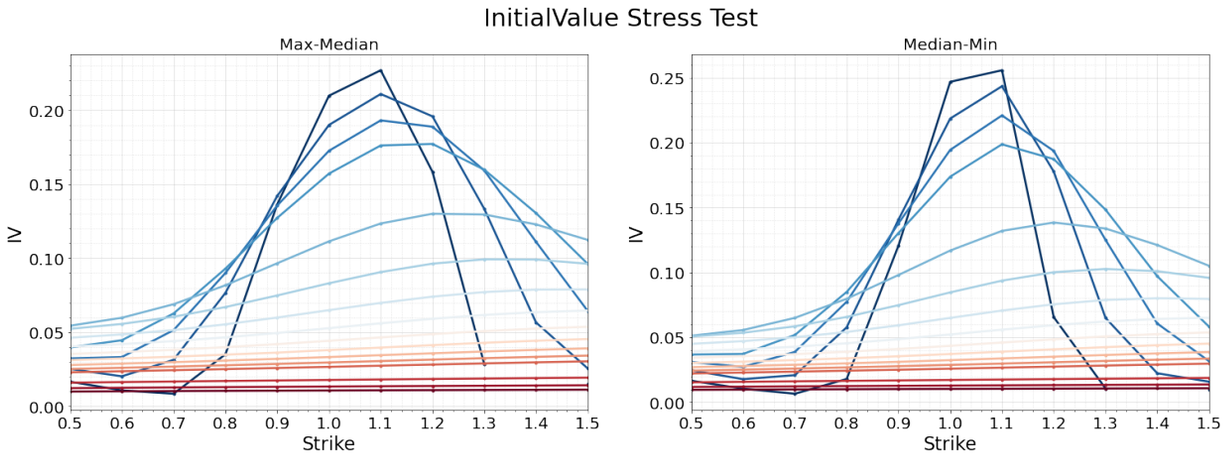

Figure 11: The plots are showing the normalised IV plots of the stressed Initial Value v_0. Therefore, how does IV change with stressing the parameter v_0. On the left, the Max is stressed with Median, and the right shows the stress of Median and Min.

- **Volatility** ξ

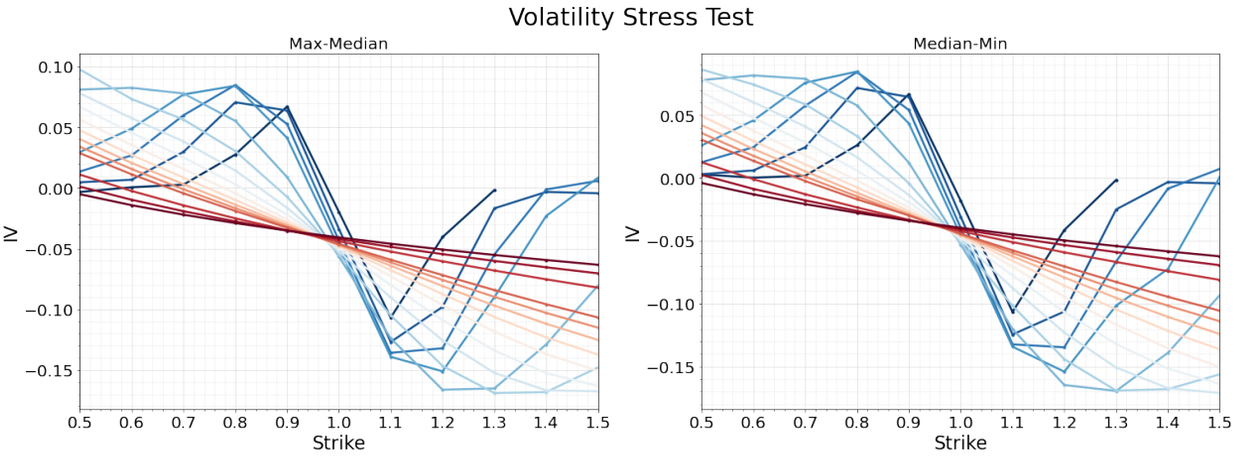

Figure 12: The plots are showing the normalised IV plots of the stressed Volatility ξ. Therefore, how does IV change with stressing the parameter ξ. On the left, the Max is stressed with Median, and the right shows the stress of Median and Min.

- Two plots for stressed **Volatility** parameter are shown in Figure 12. The shape of the plot is similar to the Arrival Rate, but it turns out to have a downtrend pattern with longer maturities. For the short-term maturities, IV has a concave shape before the strike with a value of about 1.0, and a convex shape afterward. Reliability has been ensured.

To conclude, eight plots are clearly showing that parameter value changes are having a more influence on derivative options which are having a short maturity. For the long-term maturity options, we are more likely to see a stable or straight pattern. Parameter changes leading to the IV changes are considered an optimistic way of saying that the parameter applies to the model, in other words, the parameter is not redundant or non-identifiable. If the parameter is redundant, this means that the model will maybe fit with the market data too well, this will lead to a problem that the model will not get an accurate output if we change the value of the parameter next time. From the above results, it is difficult to see which parameter is redundant, since the variation of them are all leading to an obvious change of the IV. Therefore, we will introduce another way for checking out the possible redundancy of the model in the next section.

6.3 Parameter Redundancy

To check the parameter redundancy, the linear regression analysis is implemented, and the interpretation is shown as the equation below:

$$\Delta_*(p) = \sum_{\tau \in \Pi(p)} \beta(\tau)\Delta_*(\tau) + r_*(p), \tag{6.13}$$

where $p \in \Pi$ is the parameter we want to get the residual from as the input parameter of the dependent variable, $\Pi(p) = \Pi \backslash \{p\}$, r is the residual of the parameter p that is known as the local, linearised contribution of p to the calibration that no other parameter can substitute, and $* = Max - Min$ represents the stressed delta value we chose. As you maybe notice here, we are using linear regression without the intercept, the reason is that the target sets of dependent or independent variables are all representing the ESG generated IV, and they all have similar features, thus, we will force the model to pass the origin. By using this method here, we could see the possibility of the redundancy at the hand of checking residual sizes. A detailed report of results will be generated in the next result section.

6.3.1 Result

In this section, the result of the regression analysis will be presented. Figure 13, 14, and 15 are plotted to present the pattern of the results under stressed parameters p. The top x-axis is defined as the maturities of the option, each slot represents one maturity period, the bottom x-axis is defined as the strikes, each slot represents the price changes under a range of strikes with a particular maturity, and colours are referring to each parameter.

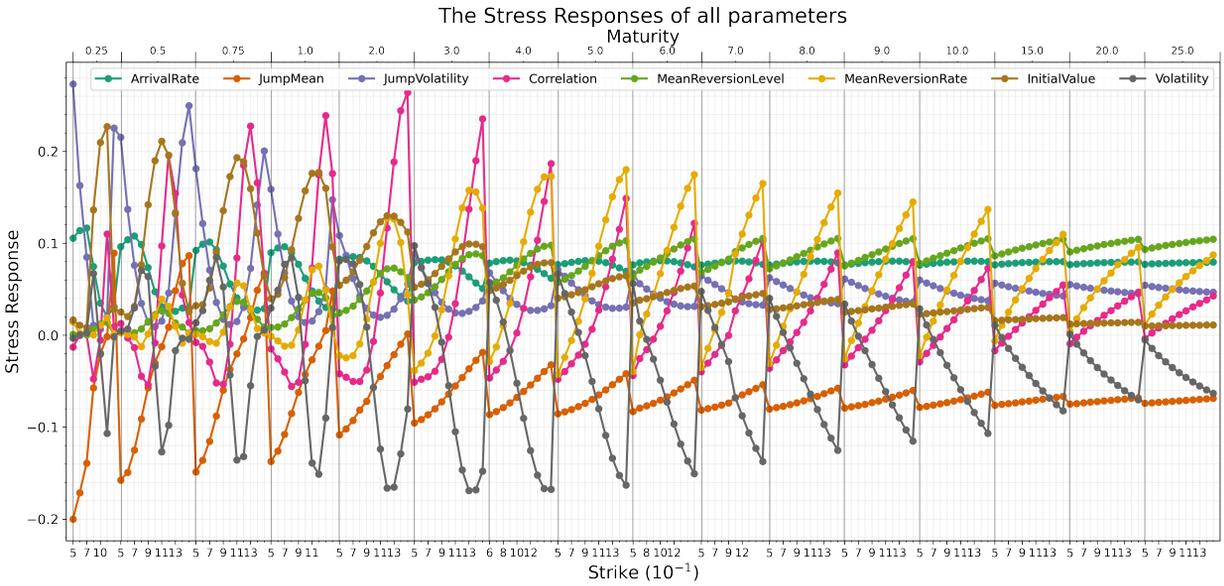

Figure 13: This plot is showing the delta response of the stressed parameters, eight parameters line are plotted the bottom x-axis represents the strike range for the priced option, they are presented recursively with different maturity period which is shown on the top x-axis. y-axis represents the response of the IV.

Figure 13 is showing Δ_* response of each parameter. The plot is concluding the previous results, as we can see that the longer the maturities, the more stable the pricing will be. Also, it is absolutely obvious that some parameters are having boundaries within a small range, this could interpret that they may have less effect on the price of the options during the whole period, thus,

we would maybe consider that they are redundant parameters. From the plot, Arrival Rate and Mean Reversion Level are showing the patterns we discussed. The stress response of the Arrival Rate always varies in a small range between 0.00 and 0.12, and consecutively, the boundaries are becoming smaller and smaller until it turns to a straight line-like trend with the value fixed around 0.08. For the Mean Reversion Level, it shows shorter maturities pricing won't have any effects by the parameter variation, but as the maturity becomes larger and larger, the price under the stressed parameter will have an increment path up to around 0.1. As we discussed before in Figure 10, the Mean Reversion Rate will possibly be not redundant as the boundaries are having a large range over time.

By using the functions in Appendix B, we could fit the linear regression model with the value of IV variation, under stressed parameter, to generate the residuals for each parameter which are also labeled with particular strike and maturity, one example of the output table is shown in Table 3 (Note: This is shown as the first output of the function).

	ArrivalRate	JumpMean	JumpVolatility	Correlation	MeanReversionLevel	MeanReversionRate	InitialValue	Volatility	Strike	Maturity
0	0.004193	-0.001675	0.066364	0.007682	0.024681	-0.007907	0.034074	0.007839	0.5	0.25
1	0.003758	0.008722	0.010315	0.039806	0.000568	-0.023084	0.004366	-0.040015	0.6	0.25
2	0.001678	0.010623	-0.035322	0.04055	-0.020191	-0.023432	-0.021441	-0.064776	0.7	0.25
3	-0.004123	-0.012554	-0.025442	-0.031782	-0.013097	0.029806	-0.012848	0.037939	0.9	0.25
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
164	-0.001437	-0.005072	-0.006608	-0.005844	-0.00258	0.004197	-0.005645	0.002286	1.2	25.0
165	-0.001533	-0.005447	-0.006881	-0.006163	-0.002813	0.00527	-0.005878	0.003518	1.3	25.0
166	-0.001617	-0.005776	-0.007112	-0.00632	-0.003014	0.006246	-0.006077	0.00468	1.4	25.0
167	-0.001785	-0.006441	-0.007535	-0.006845	-0.003339	0.007869	-0.006433	0.006777	1.5	25.0

Table 3: This table is showing 168 residuals for each parameter with labeled strike and maturity in the last two columns. Regression has done under no excluded parameter stressing. Created by function in Appendix B.

Also, by using the function in Appendix B, the Euclidean norm could be computed for each parameter under their regression residual. This could give an overview of the performance of each parameter. In Figure 14, under the regression of no excluded parameter[11], we will get the norm residuals of each parameter in Table 4 (Note: This is shown as the second output of the function).

	ArrivalRate	JumpMean	JumpVolatility	Correlation	MeanReversionLevel	MeanReversionRate	InitialValue	Volatility
No Exclude	0.028021	0.111567	0.142471	0.358988	0.074628	0.135536	0.098569	0.202311
Exclude ArrivalRate	0.028021	0.341280	0.219393	0.445111	0.122365	0.232256	0.183600	0.215448
Difference	0.000000	0.229713	0.076922	0.086122	0.047736	0.096720	0.085031	0.013137

Table 4: Table is presenting the Euclidean norm of each parameter which is calculated by using the function in Appendix B.

In Figure 14 we plotted each parameter's residual under the setup of the maturities and strikes. Here we see that the overall path trend is turning from a dramatic vibration to a flat shape, as we discussed that the longer the maturity, the more stable the pricing will be. Therefore, it is not surprising that the error become smaller as the maturity becomes larger. As you may see that there is an extremely vibrated pink line in the first half maturity range which represented as Correlation ρ, then it becomes extremely small in the second half maturity range. Under this observation, we would assume that Correlation is definitely not a redundant parameter as changing this parameter will lead a big change of the IV responses. On the other hand, there is a dark green line represented as parameter Arrival Rate λ hiding behind most of the lines, but we could easily see that during the whole period it doesn't have too much changes, as the regression model could handle the precise prediction for the most of the time, and other parameters could almost cover the attributes that created from parameter λ. Therefore, λ could be allocated in the pool as the first redundant parameter. Besides, some other parameters are also

[11] No excluded means we fit all parameters in the regression model 6.13.

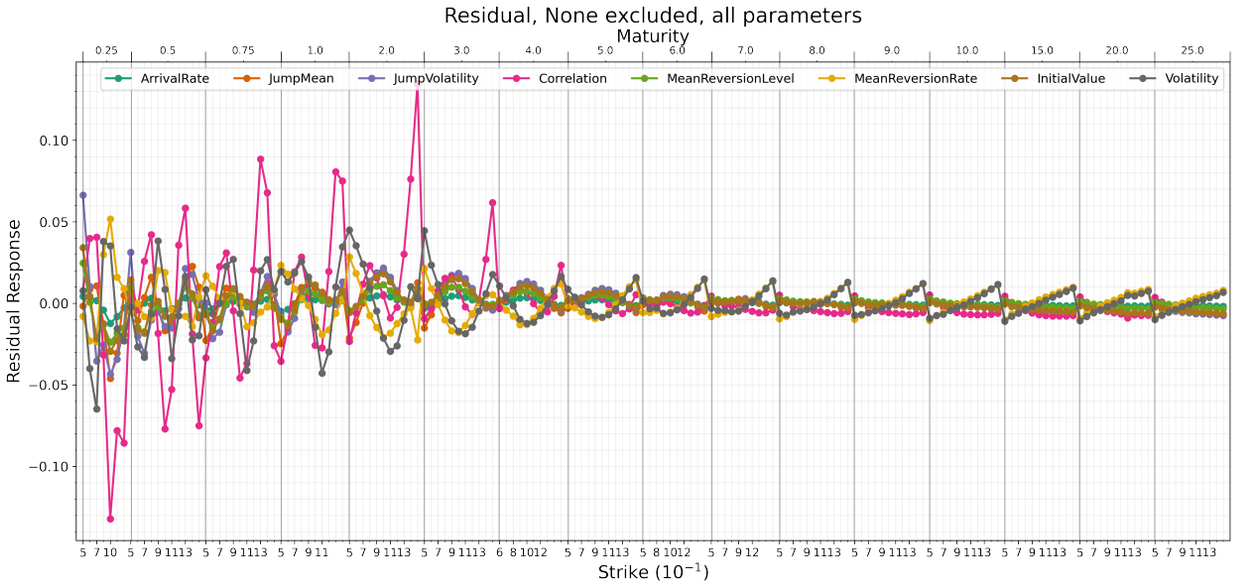

Figure 14: We plotted residuals for all parameters that we fit into the linear regression model. Each colour represents one parameter.

visible of having small vibrations, like Mean Reversion Level, and Initial Value. By looking at the norm result of Table 4, the above assumption has been proved as the Correlation has the largest Euclidean norm, and Arrival Rate has the smallest Euclidean norm.

However, it is possible that a model not only has one redundant parameter but also a few redundancies could be detected. Therefore, we will exclude the Arrival Rate λ in the regression fitting for all parameters, then to see what is going to happen and whether there are more redundant parameters.

Figure 15 is featuring the residual plot that we exclude the Arrival Rate in the regression model[12], what we can see is the Correlation is still handling the significance during the most of the maturity periods. But two undeniable increasing significance are shown by the orange and purple lines which represents the Jump Mean μ_J and Jump Volatility σ_J respectively. This could bring up the idea that λ is having a relationship with μ_J and σ_J. We extract only the residuals of Jump Mean and Jump Volatility before and after the parameter Arrival Rate is excluded from the model, from the Figure 16 we could obviously see the weights are changed largely, thus, theoretical proof in Section 6.1 is shown practically. We also discover a feature that change the parameter is also changing the weight of other parameters. One interesting feature is also obvious here, after excluding the λ, Mean Reversion Level θ and Initial Value v_0 gained an amount of weight, but as they are still having the smallest norm which shown in Table 4, we would keep them in the consideration of redundancy.

6.4 Short-term Maturities

In the Section 6.3.1, we are mostly focusing on long-term maturity products, and we see an obvious possible redundant parameter λ. Then, a question is coming out, are there going to be more redundancies discovered or λ will no longer be redundant if we trade more frequently. To answer this question, we designed the calibration to go through 16 maturities which are evenly split from 0.0834 to 1 year, but keep strikes to be the same as in Section 6.3.1.

[12]Excluding the Arrival Rate means we do not fit the Arrival Rate in the regression model, but we will still keep the original residual values of Arrival Rate in the purpose of making the comparison.

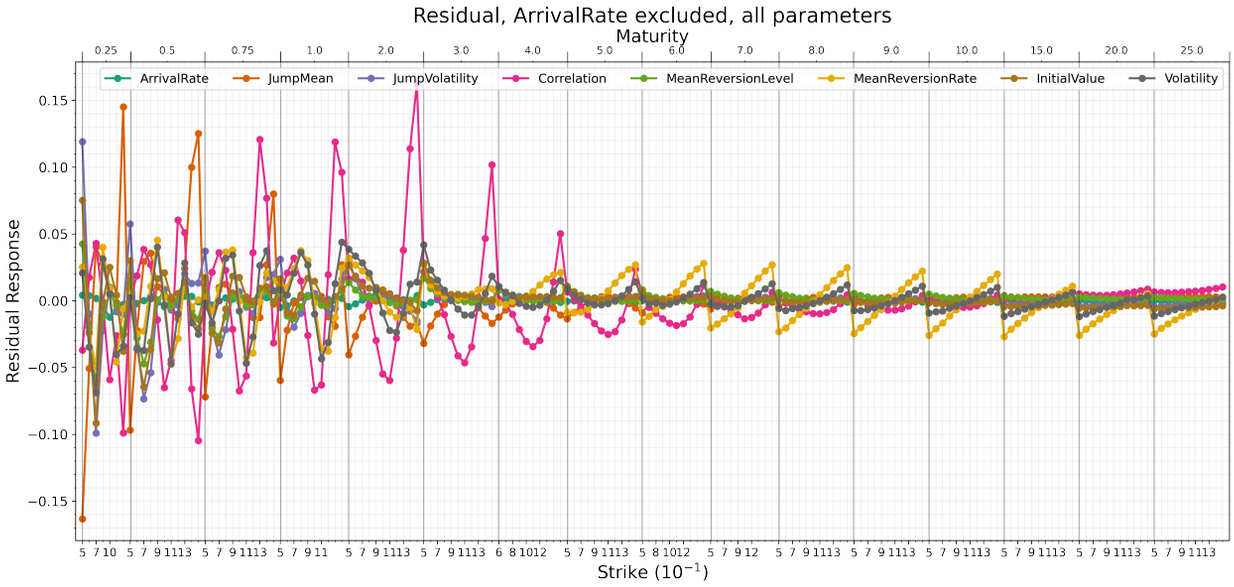

Figure 15: We plotted residuals for all parameters that we fit into the linear regression model. Each colour represents one parameter. In this case, we exclude the Arrival Rate from the model fitting.

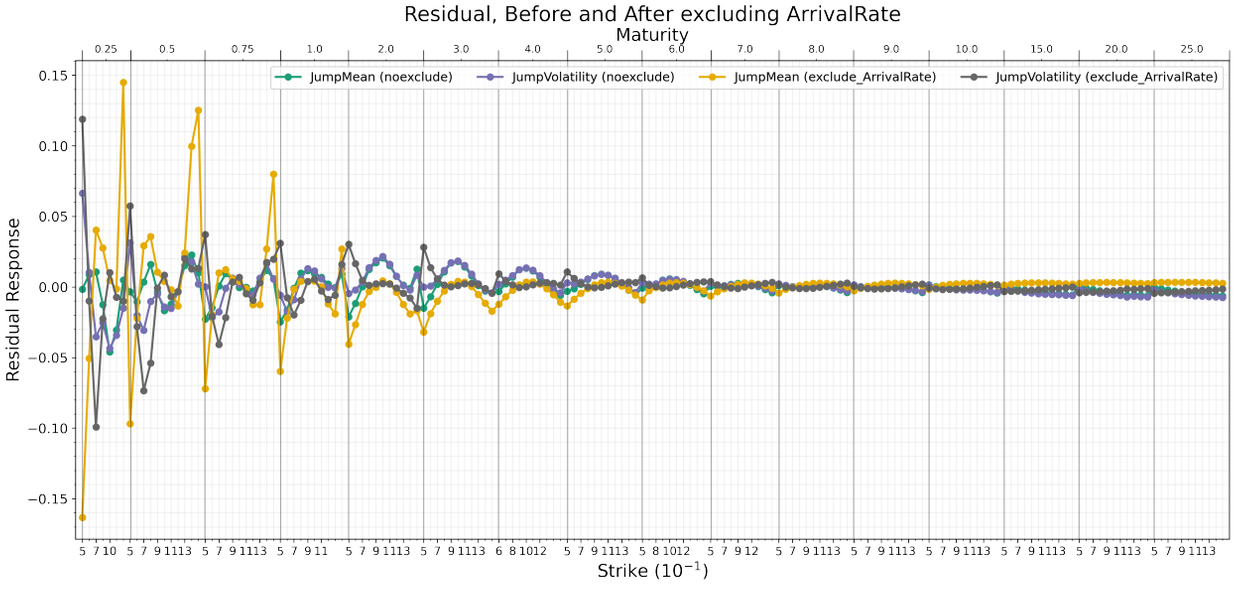

Figure 16: We plotted residuals of parameters Jump Mean and Jump Volatility before and after excluding parameter Arrival Rate.

The decrement of the IV stability is shown in Figure 20[13] as the maturity approach to 1 will no longer have a straight-line like pattern, but they are all having the similar shapes as in the long-term. On the other hand, all parameters are gaining significance in the short-term maturity range.

Applying the linear regression analysis again, we observe in Figure 17 that all parameters are not very negligible, even λ is obtaining more weights. Table 5 is supporting the analysis above, but λ is still the mostly ignorable parameter.

[13]See the Appendix C

Excluding the parameter λ, the residuals of fitting without λ is plotted in Figure 18. It is interesting that one outstanding orange line is gaining a lot of weight, the line represents Jump Mean μ_J. This is a massive discovery, as in Section 6.1, we theoretically indicated that ArrivalRate's change will significantly affect the change of either μ_J or σ_J. To numerically support the idea, we look at Table 5, μ_J is indisputably having the most changes compared with other parameters in $\Pi \setminus \{\mu_J\}$.

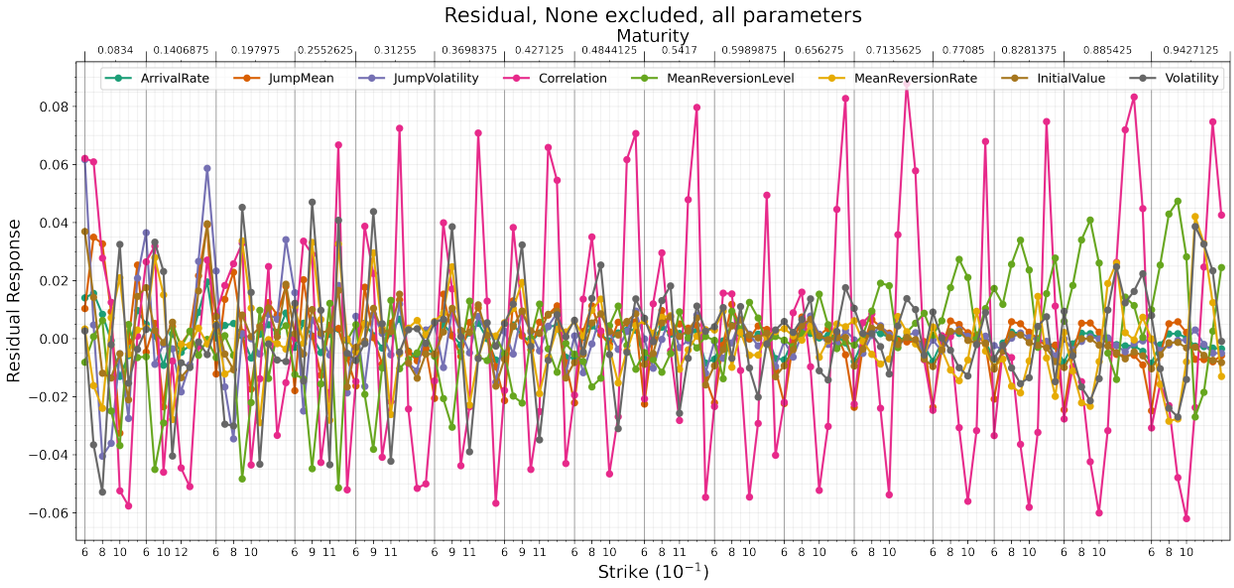

Figure 17: The residuals for all parameters that we fit into the linear regression model under the short-term maturities are plotted here.

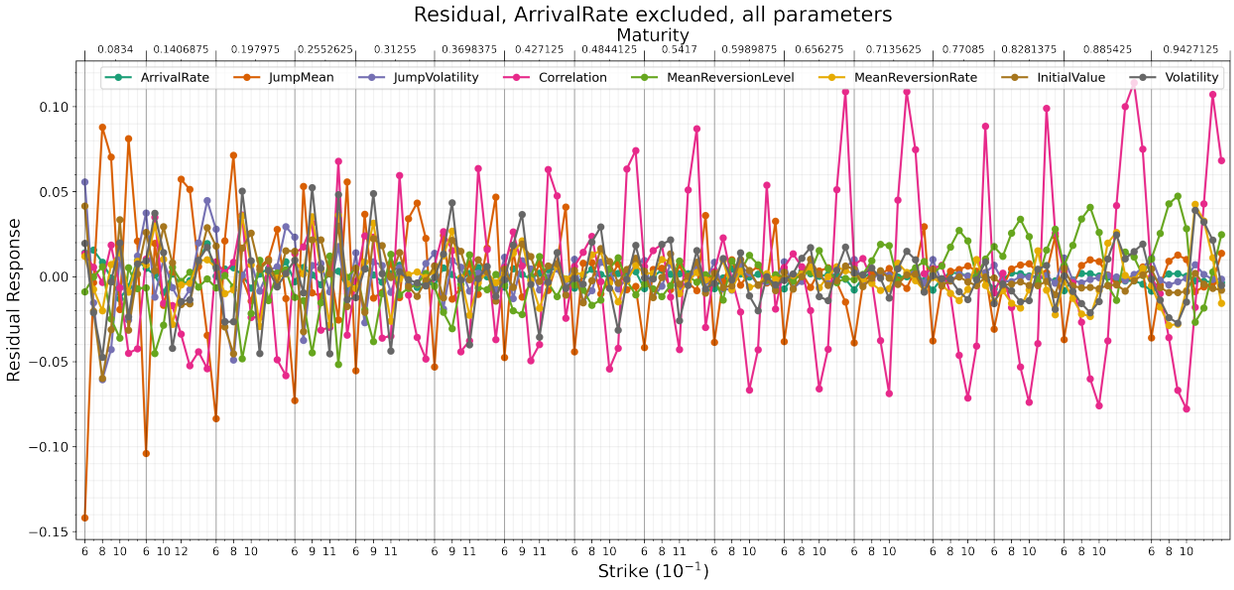

Figure 18: The residuals for all parameters that we fit into the linear regression model under the short-term maturities are plotted here. In this case, we exclude the Arrival Rate from the model fitting.

	ArrivalRate	JumpMean	JumpVolatility	Correlation	MeanReversionLevel	MeanReversionRate	InitialValue	Volatility
No Exclude	0.055496	0.137548	0.148221	0.459638	0.197410	0.153571	0.103101	0.213059
Exclude ArrivalRate	0.055496	0.351356	0.163261	0.511297	0.197431	0.157042	0.162614	0.222245
Difference	0.000000	0.213808	0.015040	0.051659	0.000021	0.003471	0.059512	0.009186

Table 5: Table is presenting the Euclidean norm of each parameter which is calculated by using the function in Appendix B.

6.5 Reduced Calibration Target Set

In this section, we continue to discuss a little bit more on the Market Consistent Calibration Methodology. In Moody's [3], there is one more step that Moody's used before applying the Vega to the weight matrix, which is applying the local linear regression to the target data set. As Moody's assumed that using this method the size of the original data set could be reduced, and the reduced data set can be used to present a similar performance as the non-reduced one in the calibration process.

What we will analyse here is to use the practical method and the fact we get from the results to show that the reduced target data set cannot perform as good as the original one.

We used the same data that we used in Section 6.3 here, however, we filtered the data to only contain the data that Moody's [3] used as the reduced target data set. Table 6 has the last two columns to show 10 pairs of strike and maturity in the reduced data set. Then, we used the filtered data to construct the residuals and norms by using the function in Appendix B. The results of the residuals and norms are shown in Table 6 and Table 7.

We then conclude that the original calibration target data set couldn't be replace by the reduced data set, since the norms in Table 7 (norms of reduced data set) and Table 4 (norms of original data set) are extremely different. In reduced data, the Mean Reversion Level parameter gained the most of the significance with norm 0.000155, but it has the second smallest norm in the original data with norm 0.074628. The Initial Value parameter becomes the second largest in reduced data with 0.000062, but it is the third smallest in original data with 0.098569. The Correlation parameter becomes the second smallest in the reduced data with 0.000001, but it has the highest norm in original data. As we concluded before, parameter Arrival Rate should be defined as a redundant parameter as it has the smallest norm with the value 0.028021, although the norm in reduced data becomes 0.000013 that is higher than Jump Mean, Jump Volatility, Correlation, and Mean Reversion Rate.

Figure 19 presents the residuals of the reduced data, it obviously displays the fact that reduced data cannot be used in the calibration process.

ArrivalRate	JumpMean	JumpVolatility	Correlation	MeanReversionLevel	MeanReversionRate	InitialValue	Volatility	Strike	Maturity
-4.888211e-08	2.243797e-08	-1.827042e-08	2.205798e-09	-2.899729e-06	-6.241880e-10	-6.445934e-07	7.928225e-08	1.0	0.25
2.340725e-08	1.869339e-08	-1.305162e-08	3.460265e-09	-2.881292e-06	-2.378509e-09	-5.457842e-07	-1.462182e-07	0.7	0.50
4.746203e-09	9.800784e-09	4.744183e-09	-1.658812e-09	-1.576925e-07	8.212282e-10	1.815349e-08	-1.991017e-08	1.3	0.50
-3.091842e-07	-2.737978e-07	1.190381e-07	-2.400331e-08	1.482052e-05	2.515628e-08	4.215779e-06	1.023432e-06	0.8	3.00
-2.487434e-07	-2.138728e-07	8.572704e-08	-1.606356e-08	1.017367e-05	9.682183e-09	2.907712e-06	-5.421840e-07	1.2	3.00
1.045709e-06	5.436158e-07	-2.036376e-07	4.828353e-08	-1.904210e-05	-4.532478e-08	-5.659564e-06	-5.881848e-07	1.0	4.00
4.841199e-06	2.043458e-06	-1.160942e-06	4.708405e-07	-5.948860e-05	-2.840792e-07	-2.491960e-05	5.423131e-06	0.8	10.00
-1.042120e-05	-4.314640e-06	2.399658e-06	-9.744703e-07	1.246879e-04	5.599315e-07	5.027562e-05	-1.416175e-05	1.0	10.00
4.971139e-06	2.143765e-06	-1.210985e-06	4.884540e-07	-6.553840e-05	-2.597383e-07	-2.597528e-05	9.026467e-06	1.2	10.00
1.418094e-07	2.053916e-08	-2.279647e-09	2.951900e-09	3.257494e-07	-3.446178e-09	3.275566e-07	-9.406837e-08	1.0	25.00

Table 6: The table shows the residuals for each parameter that we calculated for the reduced calibration target data set (We did not normalise the data in here, the analysis will not be affected).

ArrivalRate	JumpMean	JumpVolatility	Correlation	MeanReversionLevel	MeanReversionRate	InitialValue	Volatility
0.000013	0.000005	0.000003	0.000001	0.000155	6.815334e-07	0.000062	0.000018

Table 7: This table shows the Euclidean norm of the residuals for the reduced calibration target data set.

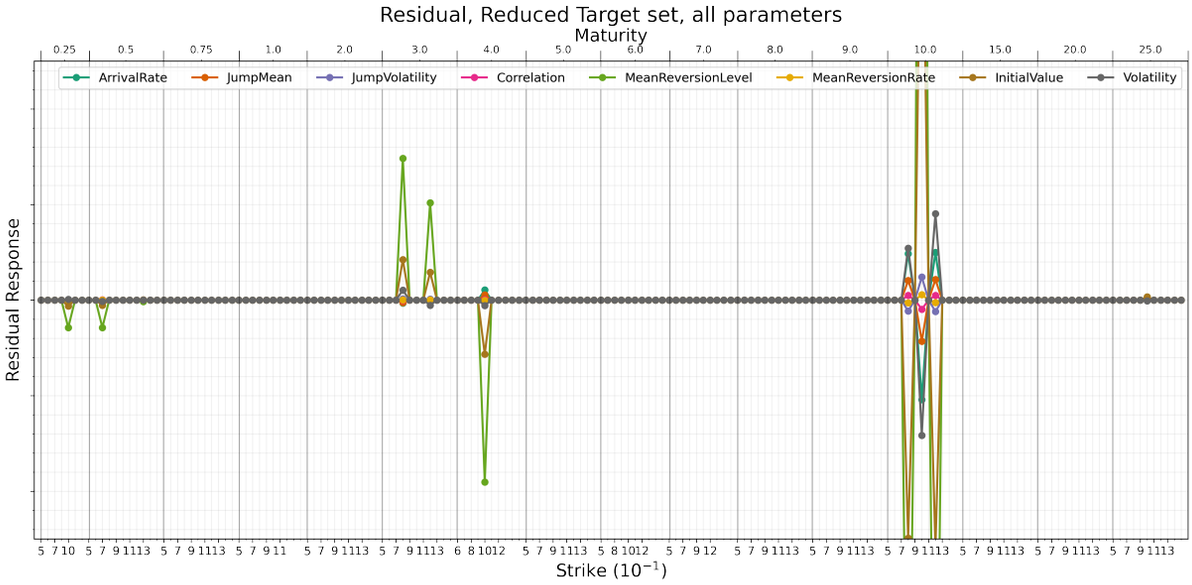

Figure 19: We plot the residuals for each parameter under the reduced calibration target data set. The limit of y-axis is reduced, since this can present a more visible pattern. y-axis range referred to the Table 6, we hide it for convenience.

7 Conclusions

In this paper, we presented the Stochastic Volatility Jump Diffusion (SVJD) model and discussed an advanced option pricing method, called the COS method, based on Fourier cosine series expansion. Moody's has been using COS method in the Market-Consistent Calibration (MC) process of the SVJD model to speed up the option pricing. Moody's Economic Scenario Generator (ESG) software implements the SVJD model and COS pricing, and we used this software to generate option implied volatility (IV) surfaces for multiple parameterizations of the SVJD model. We analysed the dependence of model IVs on model parameters in Section 6.

Based on theoretical considerations, see Equation 6.9, we identified the Arrival Rate parameter λ as a possibly redundant parameter in practice, as a change in λ could be approximately offset by changing the Jump Mean μ_J and Jump Volatility σ_J parameters.

In Section 6.2.1, we also analysed parameter redundancy by linear regression to see to what extent the linearised contribution of a parameter to the calibration can be substituted by other parameters. We used the residual of the regression as a measure of redundancy. In line with the theoretical results, we found that λ has the smallest residual hence appears as a redundant parameter in practice. We also found that excluding λ from the calibration increases the residuals of μ_J and σ_J, hence they are significant.

In Section 6.4 we performed the analogous analysis for options with very short time-to-maturity, and found that the gain in significance for μ_J and σ_J is even more pronounced when λ is excluded from the calibration.

In Section 6.5 we repeat the analysis of Section 6.2.1 for the reduced calibration target set used by Moody's in its MC calibration process. We found that the Mean Reversion Rate and Correlation parameters show some of the smallest residuals, which suggests the full calibration target set should be used for a stable calibration of the SVJD model.

Future Research: As in Equation 6.10, we have mentioned that the decision of the strike can lead us to look at the market by in the market, at the market, and out of the market concepts. Therefore, we could use Moody's MC Calibration process for extended target set to confirm the above findings are correct.

References

[1] Black-scholes model. https://www.investopedia.com/terms/b/blackscholes.asp. Accessed: 2022-08-15.

[2] Why granular data is essential for climate risk insights. https://capitalmonitor.ai/partner-content/climate-risk-insights-granular-data-essential/. Accessed: 2022-08-22.

[3] M. Analytics. Stochastic volatility jump diffusion market consistent calibration methodology. *Available at barrie+hibbert*, 2013.

[4] M. Analytics. Stochastic volatility jump diffusion model definition. *Available at barrie+hibbert*, 2013.

[5] M. Analytics. Calibration tools methodology guide. *Available at barrie+hibbert*, 2019.

[6] M. Analytics. 9.7.2 - scenario generator - developer guide. *Available at barrie+hibbert*, 2021.

[7] M. Analytics. 9.8.0 - scenario generator - configuration guide. *Available at barrie+hibbert*, 2022.

[8] M. Analytics. 9.8.0 - scenario generator - methodology guide. *Available at barrie+hibbert*, 2022.

[9] D. S. Bates. Jumps and stochastic volatility: Exchange rate processes implicit in deutsche mark options. *The Review of Financial Studies*, 9(1):69–107, 1996.

[10] F. Black and M. Scholes. The pricing of options and corporate liabilities. *The Journal of Political Economy*, 81(3):637–654, 1973.

[11] J. C. Bogle. Black monday and black swans. *Financial Analysts Journal*, 64(2):30–40, 2008.

[12] D. J. Cole. *Parameter redundancy and identifiability*. Chapman and Hall/CRC, 2020.

[13] R. Cont and P. Tankov. *Financial modelling with jump processes*. Chapman and Hall/CRC, 2004.

[14] J. C. Cox, J. E. Ingersoll, and S. A. Ross. A theory of the term structure of interest rates. *Econometrica*, 53(2):385–407, 1985.

[15] R. Craine, L. A. Lochstoer, and K. Syrtveit. Estimation of a stochastic-volatility jump-diffusion model. *Revista de Analisis Economico*, 15(1):61–87, 2000.

[16] B. Dumas, J. Fleming, and R. E. Whaley. Implied volatility functions: Empirical tests. *The Journal of Finance*, 53(6):2059–2106, 1998.

[17] F. Fang and C. W. Oosterlee. A novel pricing method for european options based on fourier-cosine series expansions. *SIAM Journal on Scientific Computing*, 31(2):826–848, 2008.

[18] D. Filipović. *Term-structure models: A graduate course*. Springer Science & Business Media, 2009.

[19] J. Gatheral. *The volatility surface: A practitioner's guide*. John Wiley & Sons, 2006.

[20] S. L. Heston. 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, 1993.

[21] J. Hull and A. White. The pricing of options on assets with stochastic volatilities. *The journal of finance*, 42(2):281–300, 1987.

[22] M. Kemp. *Market consistency: model calibration in imperfect markets*. John Wiley & Sons, 2009.

[23] Y. K. Kwok, K. S. Leung, and H. Y. Wong. Efficient options pricing using the fast fourier transform. In *Handbook of computational finance*, pages 579–604. Springer, 2012.

[24] A. L. Lewis. A simple option formula for general jump-diffusion and other exponential lévy processes. *Available at SSRN 282110*, 2001.

[25] K. Madsen, H. B. Nielsen, and O. Tingleff. Methods for non-linear least squares problems. 2004.

[26] B. Mandelbrot. The variation of some other speculative prices. *The Journal of Business*, 40(4):393–413, 1967.

[27] R. N. S. A. d. Matos. *Stochastic volatility jump-diffusion models as time-changed Lévy processes*. PhD thesis, 2014.

[28] T. Mátrai and I. Naish-Guzmán. Revisiting the lmm+ model. *Available at barrie+hibbert*, 2013.

[29] B. Øksendal. *Stochastic differential equations: An introduction with applications*. Springer Science & Business Media, 1998.

[30] J. Pan. The jump-risk premia implicit in options: Evidence from an integrated time-series study. *Journal of financial economics*, 63(1):3–50, 2002.

[31] S. Pourghanbar and M. Ranjbar. From itô and stratonovich to black-scholes. 2016.

[32] E. M. Stein and J. C. Stein. Stock price distributions with stochastic volatility: An analytic approach. *The review of financial studies*, 4(4):727–752, 1991.

[33] H. R. Stoll. The relationship between put and call option prices. *The Journal of Finance*, 24(5):801–824, 1969.

[34] S. Vikberg and J. Björkman. How well does implied volatility predict future stock index returns and volatility?: A study of option-implied volatility derived from omxs30 index options, 2020.

[35] P. Wilmott. *Derivatives: The theory and practice of financial engineering*. Wiley, 1998.

[36] J. Zhu. *Applications of Fourier transform to smile modeling: Theory and implementation*. Springer Science & Business Media, 2010.

Appendices

A Data

A.1 Data Generation

```
def ControlParameter_version2(control="Correlation", ScaleStress=0.01, ValueFix="Median", maturities = None):

    '''
    The function is used to generate the calibrated implied volatilities under multi-maturities and multi-strikes.
    The whole processes have been done by using Moody's ESG with some side calibration(.bhc) and simulation(.bhs) files.
    The function will help us to generate three excel files for each parameter, which we fit Max, Median, and Min values
    of the parameter and fit values of other parameters to be the Median value. The IV target sets we get from the ESG
    are for derivatives with multi-maturities and multi-strikes. Each simulation should be done individually from the very
    beginning, otherwise the simulation output files will overlap.

    Inputs:
        control: str, default 'Correlation'
            the parameter that we decide to fit with different values, and fix others with Median value.

        ScaleStress: int, default '0.01'
            the scaled values of the max and min for each parameter.

        ValueFix: str, default 'Median'
            the parameter that we want to fix from the range of the parameter value table.

        maturities: list
            this parameter should be a list with size 16 which indicates the range of maturities that we want to price.


    Outputs: excel file
            the generated data will be sorted out into three different excel files with name Max..., Median..., and Min... .
    '''

    # clr should be installed, sample code to build and run a ESG simulation using pythonnet. http://pythonnet.github.io/
    # system setting, connect to the ESG
    assembly_path = r"C:\Program Files\Moody's Analytics\SG\9.8.0"
    sys.path.append(assembly_path)
    clr.AddReference("BarrHibb.Sim.API")
    from BarrHibb.Sim.API import Simulation

    # Activate the simulation file
    simulation = Simulation()

    # get the excel file for the range parameter's values (user specific)
    df = pd.read_excel(r'C:\Users\Roger Cai\OneDrive\Desktop\Moodys_project\Parameter_range.xlsx')
    df = pd.DataFrame(df)

    # preparing the input parameter's values
    ParaName = ["ArrivalRate", "JumpMean", "JumpVolatility", "Correlation", "MeanReversionLevel", "MeanReversionRate", "InitialValue", "Volatility"]
    df.iloc[:8, 0] = ParaName
    df = df.set_index('Unnamed: 0')
```

```
    df.index.name = 'ParentEquityModel'
    df_ = df.iloc[:, 0:3]
    df_.loc[:, "Max"] = df_.loc[:, "Median"].copy() + ScaleStress
    df_.loc[:, "Min"] = df_.loc[:, "Median"].copy() - ScaleStress

    # Fix globally the values of the parameters that we want to input into the
    model
    Choice_dic = {'ArrivalRate' : df_.loc["ArrivalRate", ValueFix]
                  , 'JumpMean' : df_.loc["JumpMean", ValueFix]
                  , 'JumpVolatility' : df_.loc["JumpVolatility", ValueFix]
                  , 'Correlation' : df_.loc["Correlation", ValueFix]
                  , 'MeanReversionLevel' : df_.loc["MeanReversionLevel",
    ValueFix]
                  , 'MeanReversionRate' : df_.loc["MeanReversionRate", ValueFix
    ]
                  , 'InitialValue' : df_.loc["InitialValue", ValueFix]
                  , 'Volatility' : df_.loc["Volatility", ValueFix]}

    # ****** These paths are user specific *******
    dataPath = (r"C:\Users\<user>\AppData\Local\Moody's_Analytics,_Inc\BarrHibb.
    Sim.UI.exe_StrongName_e1wsydkdo5eocko4vyo3jjpaaqod4bxp\9.8.0.0")
    modelPath = r"C:\Program Files\Moody's Analytics\SG\9.8.0\Models"
    outputPath = r"C:\Users\Roger Cai\OneDrive\Desktop\output of My Simulations\
    SecondPartOfTheResult"
    licencePath = r"C:\Users\Roger Cai\OneDrive\Desktop\Moodys_project\Training-
    Moodys Analytics Research Group-120522-010922.lic"
    sim_file = r"C:\Users\Roger Cai\OneDrive\Desktop\Moodys_project\
    Simulation_File\SampleScriptSim.bhs"
    CalibrateFilePath = r"C:\Users\Roger Cai\OneDrive\Desktop\Moodys_project\
    EndJun2021_MC_SWAP_Global_Standard_SVJD.bhc"
    # ************************************************

    # Activate the licence of running Moody's confidential ESG
    simulation.InitialiseWithLicence(modelPath, '', dataPath, licencePath)

    # Setting up the environment in ESG, the method that we use, model type,
    sample size, and time step
    simulation.Create("ESG")
    economy = simulation.AddModelWithName('Economy', "GBP")
    nycs = economy.SubModel("NominalYieldCurves")
    nyc = nycs.SubModel("NominalYieldCurve")
    nyc.Type = "AnnualLMMPlus"
    PEAC = simulation.AddModelWithName("ParentEquityAssetCorrelationModel", "
    E_GBP")
    PEAC.Parameter('Economy').Value = economy.Name
    Sigma = PEAC.SubModel("Sigma")
    Sigma.Type = "SVJD"
    Jump = Sigma.SubModel("Jump")
    Variance = Sigma.SubModel("Variance")
    simulation.Parameter('BaseEconomy').Value = economy.Name
    simulation.Parameter('BaseDate').Value='16/08/2022'
    simulation.Parameter('NumberOfTrials').Value = 1
    timeSteps = simulation.ParameterCollection("Timesteps")
    # timestep 52 means, hedging once every year, continue hedging 52 years
    timeSteps["Annual"] = 52
    simulation.Parameter("RandomNumberGenerator").Value = 'MersenneTwister'
    simulation.Parameter("UseAntithetics").Value = False
    simulation.Parameter("UseRiskNeutralValuation").Value = True
    simulation.Parameter("EquityAssetCorrelationMethod").Value = "
    CorrelationMatrix"

    if control == "Correlation":

        # The name list of the columns in the output excel file
        selectedOutputNames = []
```

```
        # Add output
        outputFile = simulation.AddOutputFile('RNAP_new')
        outputFile.Folder = outputPath

        # Fix the range of the maturities and strikes that we want to price
        forwardStrike = PEAC.Output("EquityOptionImpliedVolatility")
        Parameters_ = forwardStrike.Parameters
        Parameters_[1].Value = "ForwardStrike2"
        strikes = np.arange(0.5,1.6,0.1)
        maturities = maturities
        for i in strikes:
            for j in maturities:
                Parameters_[0].Value = float(j)
                Parameters_[2].Value = float(i)
                selectedOutput_ = outputFile.AddOutputWithParameters(forwardStrike, Parameters_)[0]
                selectedOutputNames.append(selectedOutput_.FullName)

        # Calibration(.bhc)
        simulation.Calibrate(CalibrateFilePath)

        # Save the simulation file(.bhs)
        simulation.Save(sim_file)

        # Set up the parameter's values that we want to control
        Correlation_value = [df.loc["Correlation", "Max"], df.loc["Correlation", "Median"], df.loc["Correlation", "Min"]]

        # Iterating through and do the simulation for each pair of maturity and strike
        for idx, i in enumerate(Correlation_value):
            for name in Choice_dic.keys():
                if name in ["ArrivalRate", "JumpMean", "JumpVolatility"]:
                    Jump.Parameter(name).Value = float(Choice_dic[name])
                elif name == "Correlation":
                    Variance.Parameter(name).Value = float(i)
                elif name in ["MeanReversionLevel","MeanReversionRate","InitialValue","Volatility"]:
                    Variance.Parameter(name).Value = float(Choice_dic[name])

            # Set the name of the output file
            name = ["Max", "Median", "Min"]
            outputFile.Name = 'RNAP_Correlation_'+str(name[idx])

            # Save simulation file:
            simulation.Save(sim_file[0:-4]+'_'+'Correlation'+str(i)+'.bhs')
            simulation.RunLocally(0, True)

    elif control == "ArrivalRate":

        # The name list of the columns in the output excel file
        selectedOutputNames = []

        # Add output
        outputFile = simulation.AddOutputFile('RNAP_new')
        outputFile.Folder = outputPath

        # Fix the range of the maturities and strikes that we want to price
        forwardStrike = PEAC.Output("EquityOptionImpliedVolatility")
        Parameters_ = forwardStrike.Parameters
        Parameters_[1].Value = "ForwardStrike2"
        strikes = np.arange(0.5,1.6,0.1)
        maturities = maturities
        for i in strikes:
            for j in maturities:
                Parameters_[0].Value = float(j)
```

```
                Parameters_[2].Value = float(i)
                selectedOutput_ = outputFile.AddOutputWithParameters(forwardStrike, Parameters_)[0]
                selectedOutputNames.append(selectedOutput_.FullName)

        # Calibration(.bhc)
        simulation.Calibrate(CalibrateFilePath)

        # Save the simulation file(.bhs)
        simulation.Save(sim_file)

        # Set up the parameter's values that we want to control
        ArrivalRate_value = [df.loc["ArrivalRate", "Max"], df.loc["ArrivalRate", "Median"], df.loc["ArrivalRate", "Min"]]

        # Iterating through and do the simulation for each pair of maturity and strike
        for idx, i in enumerate(ArrivalRate_value):
            for name in Choice_dic.keys():
                if name in ["JumpMean", "JumpVolatility"]:
                    Jump.Parameter(name).Value = float(Choice_dic[name])
                elif name == "ArrivalRate":
                    Jump.Parameter(name).Value = float(i)
                elif name in ["Correlation", "MeanReversionLevel","MeanReversionRate","InitialValue","Volatility"]:
                    Variance.Parameter(name).Value = float(Choice_dic[name])

            # Set the name of the output file
            name = ["Max", "Median", "Min"]
            outputFile.Name = 'RNAP_ArrivalRate_'+str(name[idx])

            # Save simulation file:
            simulation.Save(sim_file[0:-4]+'_'+'ArrivalRate'+str(i)+'.bhs')
            simulation.RunLocally(0, True)

    elif control == "JumpMean":

        # The name list of the columns in the output excel file
        selectedOutputNames = []

        # Add output
        outputFile = simulation.AddOutputFile('RNAP_new')
        outputFile.Folder = outputPath

        # Fix the range of the maturities and strikes that we want to price
        forwardStrike = PEAC.Output("EquityOptionImpliedVolatility")
        Parameters_ = forwardStrike.Parameters
        Parameters_[1].Value = "ForwardStrike2"
        strikes = np.arange(0.5,1.6,0.1)
        maturities = maturities
        for i in strikes:
            for j in maturities:
                Parameters_[0].Value = float(j)
                Parameters_[2].Value = float(i)
                selectedOutput_ = outputFile.AddOutputWithParameters(forwardStrike, Parameters_)[0]
                selectedOutputNames.append(selectedOutput_.FullName)

        # Calibration(.bhc)
        simulation.Calibrate(CalibrateFilePath)

        # Save the simulation file(.bhs)
        simulation.Save(sim_file)

        # Set up the parameter's values that we want to control
        JumpMean_value = [df.loc["JumpMean", "Max"], df.loc["JumpMean", "Median"
```

```
    ], df.loc["JumpMean", "Min"]]

        # Iterating through and do the simulation for each pair of maturity and strike
        for idx, i in enumerate(JumpMean_value):
            for name in Choice_dic.keys():
                if name in ["ArrivalRate", "JumpVolatility"]:
                    Jump.Parameter(name).Value = float(Choice_dic[name])
                elif name == "JumpMean":
                    Jump.Parameter(name).Value = float(i)
                elif name in ["Correlation", "MeanReversionLevel","MeanReversionRate","InitialValue","Volatility"]:
                    Variance.Parameter(name).Value = float(Choice_dic[name])

            # Set the name of the output file
            name = ["Max", "Median", "Min"]
            outputFile.Name = 'RNAP_JumpMean_'+str(name[idx])

            # Save simulation file:
            simulation.Save(sim_file[0:-4]+'_'+'JumpMean'+str(i)+'.bhs')
            simulation.RunLocally(0, True)

    # Following comments are the same as above

    elif control == "JumpVolatility":

        selectedOutputNames = []

        # Add output
        outputFile = simulation.AddOutputFile('RNAP_new')
        outputFile.Folder = outputPath

        forwardStrike = PEAC.Output("EquityOptionImpliedVolatility")
        Parameters_ = forwardStrike.Parameters
        Parameters_[1].Value = "ForwardStrike2"
        strikes = np.arange(0.5,1.6,0.1)
        maturities = maturities
        for i in strikes:
            for j in maturities:
                Parameters_[0].Value = float(j)
                Parameters_[2].Value = float(i)
                selectedOutput_ = outputFile.AddOutputWithParameters(forwardStrike, Parameters_)[0]
                selectedOutputNames.append(selectedOutput_.FullName)

        # Calibration
        simulation.Calibrate(CalibrateFilePath)

        simulation.Save(sim_file)

        JumpVolatility_value = [df.loc["JumpVolatility", "Max"], df.loc["JumpVolatility", "Median"], df.loc["JumpVolatility", "Min"]]

        for idx, i in enumerate(JumpVolatility_value):
            for name in Choice_dic.keys():
                if name in ["ArrivalRate", "JumpMean"]:
                    Jump.Parameter(name).Value = float(Choice_dic[name])
                elif name == "JumpVolatility":
                    Jump.Parameter(name).Value = float(i)
                elif name in ["Correlation", "MeanReversionLevel","MeanReversionRate","InitialValue","Volatility"]:
                    Variance.Parameter(name).Value = float(Choice_dic[name])

            name = ["Max", "Median", "Min"]
            outputFile.Name = 'RNAP_JumpVolatility_'+str(name[idx])

```

```
            # Save simulation file:
            simulation.Save(sim_file[0:-4]+'_'+'JumpVolatility'+str(i)+'.bhs')
            simulation.RunLocally(0, True)

    elif control == "MeanReversionLevel":

        selectedOutputNames = []

        # Add output
        outputFile = simulation.AddOutputFile('RNAP_new')
        outputFile.Folder = outputPath

        forwardStrike = PEAC.Output("EquityOptionImpliedVolatility")
        Parameters_ = forwardStrike.Parameters
        Parameters_[1].Value = "ForwardStrike2"
        strikes = np.arange(0.5,1.6,0.1)
        maturities = maturities
        for i in strikes:
            for j in maturities:
                Parameters_[0].Value = float(j)
                Parameters_[2].Value = float(i)
                selectedOutput_ = outputFile.AddOutputWithParameters(
    forwardStrike, Parameters_)[0]
                selectedOutputNames.append(selectedOutput_.FullName)

        # Calibration
        simulation.Calibrate(CalibrateFilePath)
        simulation.Save(sim_file)

        MeanReversionLevel_value = [df.loc["MeanReversionLevel", "Max"], df.loc[
    "MeanReversionLevel", "Median"], df.loc["MeanReversionLevel", "Min"]]

        for idx, i in enumerate(MeanReversionLevel_value):
            for name in Choice_dic.keys():
                if name in ["ArrivalRate", "JumpMean", "JumpVolatility"]:
                    Jump.Parameter(name).Value = float(Choice_dic[name])
                elif name in ["Correlation","MeanReversionRate","InitialValue","
    Volatility"]:
                    Variance.Parameter(name).Value = float(Choice_dic[name])
                elif name == "MeanReversionLevel":
                    Variance.Parameter(name).Value = float(i)

            name = ["Max", "Median", "Min"]
            outputFile.Name = 'RNAP_MeanReversionLevel_'+str(name[idx])

            # Save simulation file:
            simulation.Save(sim_file[0:-4]+'_'+'MeanReversionLevel'+str(i)+'.bhs
    ')
            simulation.RunLocally(0, True)

    elif control == "MeanReversionRate":

        selectedOutputNames = []

        # Add output
        outputFile = simulation.AddOutputFile('RNAP_new')
        outputFile.Folder = outputPath

        forwardStrike = PEAC.Output("EquityOptionImpliedVolatility")
        Parameters_ = forwardStrike.Parameters
        Parameters_[1].Value = "ForwardStrike2"
        strikes = np.arange(0.5,1.6,0.1)
        maturities = maturities
        for i in strikes:
            for j in maturities:
                Parameters_[0].Value = float(j)
```

```
                Parameters_[2].Value = float(i)
                selectedOutput_ = outputFile.AddOutputWithParameters(forwardStrike, Parameters_)[0]
                selectedOutputNames.append(selectedOutput_.FullName)

        # Calibration
        simulation.Calibrate(CalibrateFilePath)
        simulation.Save(sim_file)

        MeanReversionRate_value = [df.loc["MeanReversionRate", "Max"], df.loc["MeanReversionRate", "Median"], df.loc["MeanReversionRate", "Min"]]

        for idx, i in enumerate(MeanReversionRate_value):
            for name in Choice_dic.keys():
                if name in ["ArrivalRate", "JumpMean", "JumpVolatility"]:
                    Jump.Parameter(name).Value = float(Choice_dic[name])
                elif name in ["Correlation","MeanReversionLevel","InitialValue","Volatility"]:
                    Variance.Parameter(name).Value = float(Choice_dic[name])
                elif name == "MeanReversionRate":
                    Variance.Parameter(name).Value = float(i)

            name = ["Max", "Median", "Min"]
            outputFile.Name = 'RNAP_MeanReversionRate_'+str(name[idx])

            # Save simulation file:
            simulation.Save(sim_file[0:-4]+'_'+'MeanReversionRate'+str(i)+'.bhs')
            simulation.RunLocally(0, True)

    elif control == "InitialValue":

        selectedOutputNames = []

        # Add output
        outputFile = simulation.AddOutputFile('RNAP_new')
        outputFile.Folder = outputPath

        forwardStrike = PEAC.Output("EquityOptionImpliedVolatility")
        Parameters_ = forwardStrike.Parameters
        Parameters_[1].Value = "ForwardStrike2"
        strikes = np.arange(0.5,1.6,0.1)
        maturities = maturities
        for i in strikes:
            for j in maturities:
                Parameters_[0].Value = float(j)
                Parameters_[2].Value = float(i)
                selectedOutput_ = outputFile.AddOutputWithParameters(forwardStrike, Parameters_)[0]
                selectedOutputNames.append(selectedOutput_.FullName)

        # Calibration
        simulation.Calibrate(CalibrateFilePath)
        simulation.Save(sim_file)

        InitialValue_value = [df.loc["InitialValue", "Max"], df.loc["InitialValue", "Median"], df.loc["InitialValue", "Min"]]

        for idx, i in enumerate(InitialValue_value):
            for name in Choice_dic.keys():
                if name in ["ArrivalRate", "JumpMean", "JumpVolatility"]:
                    Jump.Parameter(name).Value = float(Choice_dic[name])
                elif name in ["Correlation","MeanReversionLevel","MeanReversionRate","Volatility"]:
                    Variance.Parameter(name).Value = float(Choice_dic[name])
                elif name == "InitialValue":
```

```
                    Variance.Parameter(name).Value = float(i)

            name = ["Max", "Median", "Min"]
            outputFile.Name = 'RNAP_InitialValue_'+str(name[idx])

            # Save simulation file:
            simulation.Save(sim_file[0:-4]+'_'+'InitialValue'+str(i)+'.bhs')
            simulation.RunLocally(0, True)

    elif control == "Volatility":

        selectedOutputNames = []

        # Add output
        outputFile = simulation.AddOutputFile('RNAP_new')
        outputFile.Folder = outputPath

        forwardStrike = PEAC.Output("EquityOptionImpliedVolatility")
        Parameters_ = forwardStrike.Parameters
        Parameters_[1].Value = "ForwardStrike2"
        strikes = np.arange(0.5,1.6,0.1)
        maturities = maturities
        for i in strikes:
            for j in maturities:
                Parameters_[0].Value = float(j)
                Parameters_[2].Value = float(i)
                selectedOutput_ = outputFile.AddOutputWithParameters(
    forwardStrike, Parameters_)[0]
                selectedOutputNames.append(selectedOutput_.FullName)

        # Calibration
        simulation.Calibrate(CalibrateFilePath)
        simulation.Save(sim_file)

        InitialValue_value = [df.loc["Volatility", "Max"], df.loc["Volatility",
    "Median"], df.loc["Volatility", "Min"]]

        for idx, i in enumerate(InitialValue_value):
            for name in Choice_dic.keys():
                if name in ["ArrivalRate", "JumpMean", "JumpVolatility"]:
                    Jump.Parameter(name).Value = float(Choice_dic[name])
                elif name in ["Correlation","MeanReversionLevel","
    MeanReversionRate","InitialValue"]:
                    Variance.Parameter(name).Value = float(Choice_dic[name])
                elif name == "Volatility":
                    Variance.Parameter(name).Value = float(i)

            name = ["Max", "Median", "Min"]
            outputFile.Name = 'RNAP_Volatility_'+str(name[idx])

            # Save simulation file:
            simulation.Save(sim_file[0:-4]+'_'+'Volatility'+str(i)+'.bhs')
            simulation.RunLocally(0, True)
```

A.2 Sort Data and Create Pivot Table

```
def SortOutput(OutputPath):
    '''
    This function will create a Tensor Matrix with shape (maturity, strike, TrialNumber, TimeStep)

    Inputs:
        Path: the path of the excel file of the particular IV data

    Output: numpy array
        the shape of (maturity, strike, TrialNumber, TimeStep) numpy array will be generated
    '''
    # Get the data from local excel
    df_output_ = pd.read_csv(OutputPath, index_col = False)

    # Using regex to read the maturities and strikes
    df_output_n = df_output_.iloc[:, 2:]
    maturity = [float(re.split("[(,)\s]", i)[1]) for i in df_output_n.columns]
    strike = [float(re.split("[(,)\s]", i)[5]) for i in df_output_n.columns]

    # Transposed form in case of adding maturities and strikes to the columns helping to match with the IV
    df_output_n = df_output_n.T
    df_output_n["Maturity"] = maturity
    df_output_n["Strike"] = strike

    # Getting a list of pivot table each represents one time point
    df_pivot = [df_output_n.pivot(index="Maturity", columns="Strike", values=i) for i in range(len(df_output_))]

    # Reshaping the list to get the final output
    df_output_n = np.stack(df_pivot, 2).reshape(len(set(maturity)), len(set(strike)), len(df_output_['Trial'].unique()), len(df_output_['Timestep'].unique()))

    return df_output_n


def Pivot(path = OutputPath_stress, parameter = "Correlation", value = "Max", timeStep = None):
    '''
    This function is coorperating with the SortOutput() function to generate a pivot table at time 0.

    Input:
        path: the direction to the local file

        parameter: str, default 'Correlation'
            the specific parameter's name that we want to get the pivot table

        value: str, default 'Max'
            as there are three files for each parameter with Max, Median, and Min fitted, we need to define which one we want

        timeStep: int, default None
            the timestep we are looking is at t=0, since we want to see the price of the option at initial point. But we set
            this choice to be flexible, just in case of particular needs.

    Output: Pandas.DataFrame
        the output will be a DataFrame that has columns of all the Strikes and Indexes of all the Maturities. IVs are the nodes
        on the surface.

```

```
    '''

    # Inputting the outputfile and get the maturity and strike
    OutputPath = path + '\RNAP_' + '{}'.format(parameter) + '_' + '{}'.format(
    value) + '.csv'
    temp_df = pd.read_csv(OutputPath, index_col = False)

    # Get the file with the columns only contain the information we need
    temp_df_n = temp_df.iloc[:, 2:]

    # Get the name of all the maturities and strikes
    maturity = [float(re.split("[(,)\s]", i)[1]) for i in temp_df_n.columns]
    strike = [float(re.split("[(,)\s]", i)[5]) for i in temp_df_n.columns]

    #getting the pivot table only at time step 0 and change the columns name and
     index name
    if timeStep==None:
        time_0 = SortOutput(OutputPath)[:,:,:,0]
    else:
        time_0 = SortOutput(OutputPath)[:,:,:,timeStep]
    time_0 = time_0.reshape(len(set(maturity)), len(set(strike)))
    df_time = pd.DataFrame(time_0)
    df_time.columns = np.unique(strike)
    df_time.index = np.unique(maturity)
    df_time.index.name = "Maturity"
    df_time.columns.name = "Strike"
    return df_time
```

B Linear Regression and Residual Calculation

```
def Get_Residual_Norm(paramIgnore):
    '''
    This function is to do the regression fit by deciding which Delta we are going to use to fit the model.

    Input: str, list of str, None
        str: the input of the function is the parameter's name that we want to exclude from the regression fitting.
        None: represents there are no parameters that we want to exclude
        list of str: a list of parameters' name that we want to exclude from the regression model

    Output: DataFrame_1, DataFrame_2
        DataFrame_1 : the residuals of all parameters include the labels of strikes and maturities
        DataFrame_2 : the Euclidean norm of residuals of each parameter which we get from DataFrame_1
    '''

    def baseModel():
        '''
        This function is helping us to fit the parameters' Delta into the linear regression model, and calculate the
        residual and norm

        Output: DataFrame_1, DataFrame_2
            Two DataFrames are defined the same as previous definition
        '''

        residual = []

        # temp_df is the DataFrame that contains all the IV variation of stressed parameters
        # loop through regression fitting of all parameter's IV variation
        for i, idx in enumerate(temp_df.columns[0:8]):
            y = temp_df.loc[:, idx]
            x = temp_df.drop([idx, 'Strike', 'Maturity'], axis=1)

            # fit model:
            reg = LinearRegression().fit(x,y)

            # predicted value of the IV:
            y_predict = reg.predict(x)
            residual_ = y - y_predict
            residual.append(residual_)

        # change the residual list to DataFrame, and labelling
        residual = np.array(residual)
        residual = pd.DataFrame(residual).T
        residual['Strike'], residual['Maturity'] = temp_df['Strike'], temp_df['Maturity']
        name = Name_Parameters.copy()
        residual.columns = name

        # calculate the Euclidean norm of the residuals:
        norm = []
        for j in Name_Parameters[0:-2]:
            norm_ = LA.norm(np.array(residual.loc[:, j]), 2)
            norm.append(norm_)

        norm_df = pd.DataFrame(norm).T
        norm_df.columns = Name_Parameters[0:-2].copy()
        return residual, norm_df

```

```
    # Detailed comments have done above, the comments ignored below will be the same as above as the process above

    # if there is no excluded parameter
    if paramIgnore == None:
        # call the function above
        residual, norm = baseModel()
        return residual, norm

    else:
        # if data type is a list, we do this:
        if type(paramIgnore) == list:
            column_names = temp_df.columns
            temp_ignored = temp_df.iloc[:, :8]
            temp_ignored = temp_ignored.drop(paramIgnore, axis=1)

            residual=[]
            for i, idx in enumerate(temp_ignored.columns):
                y = temp_ignored.loc[:, idx]
                x = temp_ignored.drop(idx, axis=1)

                # fit model:
                reg = LinearRegression().fit(x,y)

                # predicted value of the IV:
                y_predict = reg.predict(x)
                residual_ = y - y_predict
                residual.append(residual_)

            residual = np.array(residual)
            residual = pd.DataFrame(residual).T
            residual['Strike'], residual['Maturity'] = temp_df['Strike'], temp_df['Maturity']

            temp_new_name = Name_Parameters.copy()
            for eles in temp_new_name:
                for j in paramIgnore:
                    if j in temp_new_name:
                        temp_new_name.remove(j)

            residual.columns = temp_new_name

            res, _ = baseModel()
            for z in residual.columns:
                res[z] = residual[z]


            # calculate the Euclidean norm of the residuals:
            norm = []
            for j in Name_Parameters[0:-2]:
                norm_ = LA.norm(np.array(res.loc[:, j]), 2)
                norm.append(norm_)

            norm_df = pd.DataFrame(norm).T
            norm_df.columns = Name_Parameters[0:-2]

            return res, norm_df

        # if the data type is a single str, we do this:
        else:
            column_names = temp_df.columns
            temp_ignored = temp_df.iloc[:, :8]
            temp_ignored = temp_ignored.drop(paramIgnore, axis=1)

            residual=[]
            for i, idx in enumerate(temp_ignored.columns):
```

```
                y = temp_ignored.loc[:, idx]
                x = temp_ignored.drop(idx, axis=1)

                # fit model:
                reg = LinearRegression().fit(x,y)

                # predicted value of the IV:
                y_predict = reg.predict(x)
                residual_ = y - y_predict
                residual.append(residual_)

            residual = np.array(residual)
            residual = pd.DataFrame(residual).T
            residual['Strike'], residual['Maturity'] = temp_df['Strike'],
    temp_df['Maturity']
            name = Name_Parameters.copy()
            name.remove(paramIgnore)
            residual.columns = name

            res, _ = baseModel()
            for z in residual.columns:
                res[z] = residual[z]


            # calculate the Euclidean norm of the residuals:
            norm = []
            for j in Name_Parameters[0:-2]:
                norm_ = LA.norm(np.array(res.loc[:, j]), 2)
                norm.append(norm_)

            norm_df = pd.DataFrame(norm).T
            norm_df.columns = Name_Parameters[0:-2]

            return res, norm_df
```

C All parameters IV stressed plots

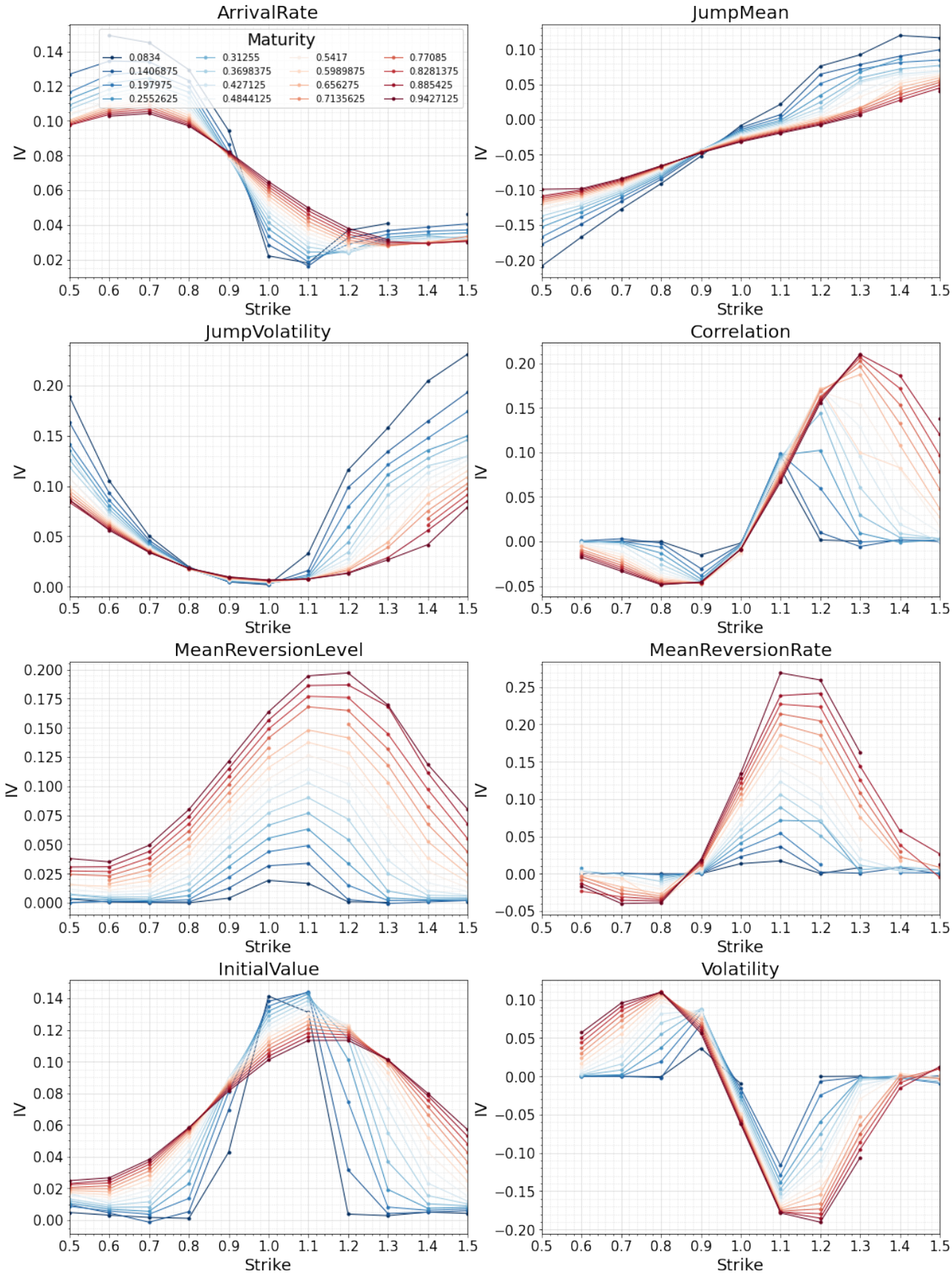

Figure 20: This figure shows the normalised IV plots for each parameter. Therefore, how does IV change with stressing the parameters in Π. All plots are stressed by using Max-Median parameters' value.