Heston Model Calibration with Markov Chain Monte Carlo Simulation and Option Pricing using Euler method and QE method

By Ken Lau, Ziliang You, Liang Geng, Ziyong Tao

1. Objective

The Heston model is a cornerstone in financial mathematics for option pricing, capturing the complex dynamics of market volatility. This project aims to apply the Markov Chain Monte Carlo (MCMC) method for calibrating the Heston model parameters. The calibrated parameters will be used to simulate stock prices, which in turn are employed for option pricing. These prices will be compared with actual market data to assess the model's accuracy and effectiveness in a real-world setting.

2. Methods

2.1 Heston Model

In the classic Black-Scholes model, volatility is assumed to be constant, while the Heston model allows volatility to vary over time and models this process. This stochastic volatility model offers a more complex but accurate approach to option pricing. Its stochastic volatility feature allows it to better capture actual market dynamics, especially under extreme market conditions. The mean-reverting feature of the stochastic volatility is also more realistic and close to real-world conditions.

The SDEs of the Heston Model is given by:

$$
\begin{aligned}
dS_t &= \mu S_t\, dt + \sqrt{v_t} S_t\, dW_{1t}, \\
dv_t &= \kappa(\theta - v_t)\, dt + \psi\sqrt{v_t}\, dW_{2t}, \\
\rho\, dt &= dW_{1t} dW_{2t}.
\end{aligned}
$$

S_t - Underlying asset price
v_t - The stochastic volatility of the underlying price
μ - Drift of the price process
W_{1t} - The Wiener Process related to the underlying price
W_{2t} - The Wiener Process related to the stochastic volatility
κ - The mean-reverting rate of v_t
θ - Long-term mean level of v_t
ψ - The volatility of v_t
ρ - The correlation of W_{1t} and W_{2t}

2.2 Markov Chain Monte Carlo (MCMC) Method

Instead of using some optimizations techniques such as non-linear optimization to calibrate the model parameters, we decided to adopt MCMC simulation for model calibration due to its versatility and computational efficiency (Cape et al., 2014). MCMC leverages Markov chains, a sequence of random variables where the probability of each variable depends only on the state of the previous variable. This "memoryless" property is key to the efficiency of MCMC

methods. It is particularly useful for sampling from high-dimensional probability distributions, where traditional methods are computationally infeasible.

The basic concept of using MCMC to calibrate Heston model is to find the likelihood function of the discretized process and to find all posterior distributions of the parameters and state space. Then, by the conditional posterior distribution of the posterior parameters, we use MCMC to generate a Markov Chain of the parameters and the state space. The original paper proposed to use Gibbs sampler on the parameters and random walk Metropolis–Hastings approach for the state space. The target parameters will be updated step by step throughout the MCMC process by an acceptance-rejection approach. The details of the Maths behind and steps can be found in the paper by Cape et al. in 2014, so we are not showing them here. The algorithm to achieve this using python is inspired by Imlerith on Github.

2.3 Simulation Discretization Scheme
Adopting an efficient discretization scheme is crucial to the simulation as it directly affects the convergence of the model. We simulated the Heston Model stock price with two discretization schemes: 1. The Quadratic Exponential (QE) Discretization Scheme and 2. Full Truncation Euler Scheme.

2.3.1 QE Scheme (Anderson, 2006)
QE Scheme is an advanced scheme proposed in 2016 for simulating Heston Model. It effectively manages the non-linear nature of the square-root process in the Heston model. The scheme is known for its balance between computational efficiency, accuracy, and stability in simulating paths of the stochastic processes, especially under conditions of high volatility. One of the main reasons why QE is one of the most efficient and accurate schemes for Heston model is that the generation process of the stochastic volatility follows Non-Central Chi-Squared Distribution, which fits the Heston Model where the stochastic volatility is essentially a Cox–Ingersoll–Ross (CIR) process.

The sampling of volatility consists of 5 steps:

1. Given $\hat{V}_t$ the discrete time approximation of V, compute:

$$m = \theta + (\hat{V}_t - \theta)e^{-\kappa\Delta t}$$

$$s^2 = \frac{\hat{V}_t \varepsilon^2 e^{-\kappa\Delta t}}{\kappa}(1 - e^{-\kappa\Delta t}) + \frac{\theta\varepsilon^2}{2\kappa}(1 - e^{-\kappa\Delta t})^2$$

2. Compute: $\Psi = \frac{s^2}{m^2}$

3. Draw a uniform random number U_V

   For $\Psi_c \in [1, 2]$, (we typically use 1.5)

4. If $\Psi \leq \Psi_c$
    a. Compute a and b:
    $b^2 = 2\Psi^{-1} - 1 + \sqrt{2\Psi^{-1}}\sqrt{2\Psi^{-1} - 1} \geq 0$
    $a = \frac{m}{1 + b^2}$
    b. Compute $Z_v = \Phi^{-1}(U_V)$
    c. Set $\hat{V}(t + \Delta t) = a(b + Z_v)^2$

5. Else if $\Psi > \Psi_c$,
    a. Compute β and p:
    $\beta = \frac{1 - p}{m} = \frac{2}{m(\Psi + 1)} > 0$
    $p = \frac{\Psi - 1}{\Psi + 1} \in [0, 1)$
    b. $\Psi^{-1}(u) = \Psi^{-1}(u : p, \beta) = \begin{cases} 0, & 0 \leq u \leq p. \\ \beta^{-1} ln(\frac{1-p}{1-u}), & p < u \leq 1. \end{cases}$
    c. Set $\hat{V}(t + \Delta t) = \Psi^{-1}(u : p, \beta)$

From here we have the $\hat{V}(t + \Delta t)$ and then we use it to simulate the stock price:

$ln\hat{S}(t+\Delta t) = ln\hat{S} + K_1\hat{V}(t) + K_2\hat{V}(t+\Delta t) + \sqrt{K_3\hat{V}(t) + K_4\hat{V}(t + \Delta t)} \cdot Z$

$K_0 = -\frac{\rho\kappa\theta}{\varepsilon}\Delta t$

$K_1 = \gamma_1\Delta t(\frac{\rho\kappa}{\varepsilon} - \frac{1}{2}) - \frac{\rho}{\varepsilon}$

$K_2 = \gamma_2\Delta t(\frac{\rho\kappa}{\varepsilon} - \frac{1}{2}) + \frac{\rho}{\varepsilon}$

$K_3 = \gamma_1\Delta t(1 - \rho^2)$

$K_4 = \gamma_2\Delta t(1 - \rho^2)$

Z is a random standard normal variable

γ_1 and γ_2 are the weights of $\hat{V}(t)$ and $\hat{V}(t + \Delta t)$ and we can typically choose both to be 0.5 (equally weighted)

2.3.2 Fully Truncated Euler Discretization

This is a straightforward time-discretization method which is also suitable for numerical simulation of the Heston model. The basic idea of Euler Discretization is to break down a continuous-time stochastic process into discrete time steps and then approximate the changes in the process over these small intervals. By Fully Truncated, we basically force the stochastic volatility to be non-negative throughout the simulation by taking a maximum function of 0 and the simulated volatility generated by the corresponding randomly generated CIR process.

2.4 Data

We calibrated the Heston Model using real-world data and we chose 2 Index ETFs and their corresponding Call Options with 10 strike levels due to their relatively high volume in options transaction. The 2 assets are SPDR S&P 500 ETF Trust - SPY and Invesco QQQ Trust Series 1 - QQQ. And the expire date of option is on Dec 2023.

2.4.1 Historical Underlying Asset Prices

The data of three underlying assets downloaded from Bloomberg are daily frequency prices in 3 months.

2.4.2 Historical Options Prices

We designed and programmed a Python script to automate the process of fetching and processing financial options data from the Polygon API.

For each underlying ETFs, we fetched the daily OHLCV data of Call Options of 10 strikes around the current stock price. All options have the same maturity.

3. Results

[Presentation of the effectiveness of the MCMC method in estimating Heston model parameters.]

[Comparison of stock price simulations and option pricing using Euler and QE discretization methods.]

[Analysis comparing simulation outcomes with actual market data.]

To implement our algorithm we use Python in Pycharm. In this section, we will present our simulation and observation from comparing simulation results with the true option price qqq in 60 days and 252 days intervals. Since we found out the simulation results are quite different. Thus, the start date would be Oct 31, 2022, and Aug 31, 2023, corresponding to Oct 31, 2023 as the end date and the benchmark date that we use to compare

3.1 Python code and library

In order to make our codes more readable and clear, we create libraries to store our methods and algorithm.

- main file (use these two to run results):main_gld,main_qqq.py
- Polygon API:polygon_fetch.ini.sample.ini, polygon_fetch.py
- Algorithm library(MCMC,QE,Euler):heston_calibrator_simulator.py
- Plotter library:heston_utils.py

3.2 Parameters setting for different methods

- dt=1/252
  We use daily price basis, so as dt, which is the time difference, is =1/252, assuming 252 trade days in one year.

- r = 0.03 and 0.0533*60/360
  The interest rates we use are the 1 year federal funds rate on Oct 31 2022 and Aug 31, 2023

- q = 0
  This is the carry cost. since we are analyzing options of Index ETF, so we set it as 0

Other parameter for algorithm:
# mcmc para
n_mcmc_steps = 6000
burn_in = 3000

#qe para
T = 60/252 # this is maturity in year, changed based on different time interval
n_steps = 60 # in day
n_paths = 1000
gamma1 = 0.5 # averaging factors for the discretization of Xt
gamma2 = 0.5 # averaging factors for the discretization of Xt
phiC = 1.5 # Threshold for the initiation of the two approximate distribution of V(t+1 | Vt)

#b-s
year=1

3.3 Results

3.3.1 QQQ -60 days time zone

We will introduce how to call our function to get simulation results in the first case. For other cases just follows the same process.

First, we import our library

```
from heston_calibrator_simulator import HestonCalibratorSimulator
import heston_utils as hu
```

Initialize objection and then use calibrate() to implement MCMC to simulate Heston parameters.

```
qqq_heston = HestonCalibratorSimulator(cp, cost_of_carry=r-q,
delta_t=1.)
qqq_heston.calibrate(n_mcmc_steps, burn_in)
```

Then using all_params.get() to read parameters

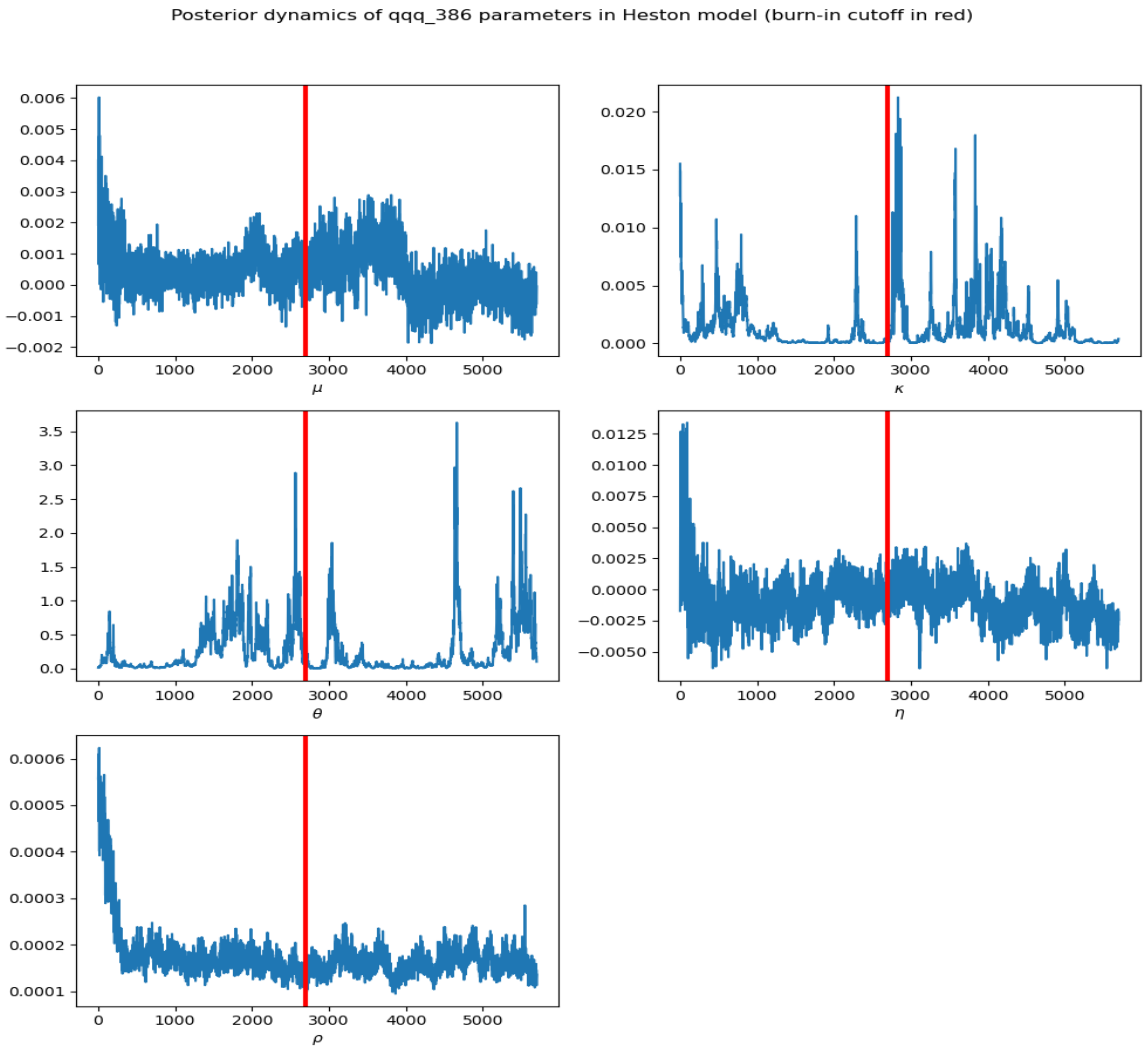
```
all_params = qqq_heston.params_dict
mu = all_params.get( "mu_final" )
kappa = all_params.get( "kappa_final" )
theta = all_params.get( "theta_final" )
eta = all_params.get( "volvol_final" )
rho = all_params.get( "rho_final" )
print(f"{mu:.4f}, {kappa:.4f}, {theta:.4f}, {eta:.4f}, {rho:.4f}")
```

- parameter:

μ = 0.0003, κ = 0.0015, θ = 0.2441, ψ = 0.0127, ρ = -0.0840

- Plot parameters dynamics

```
hu.plot_heston_param_dynamics(param_paths, burn_in_pos,
title='qqq_386')
```

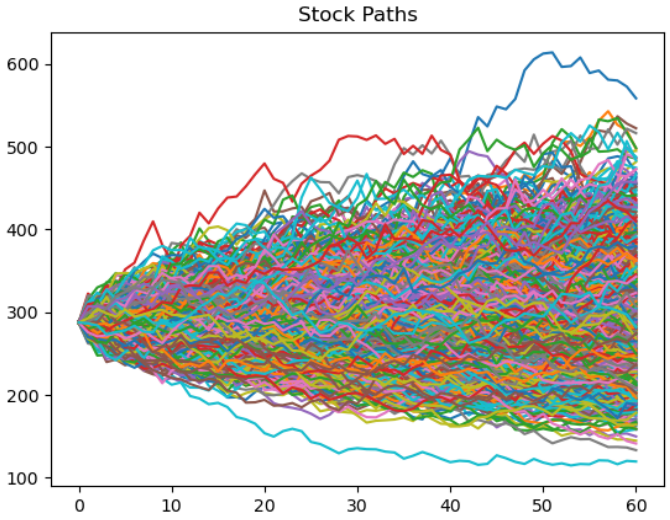


- QE method:

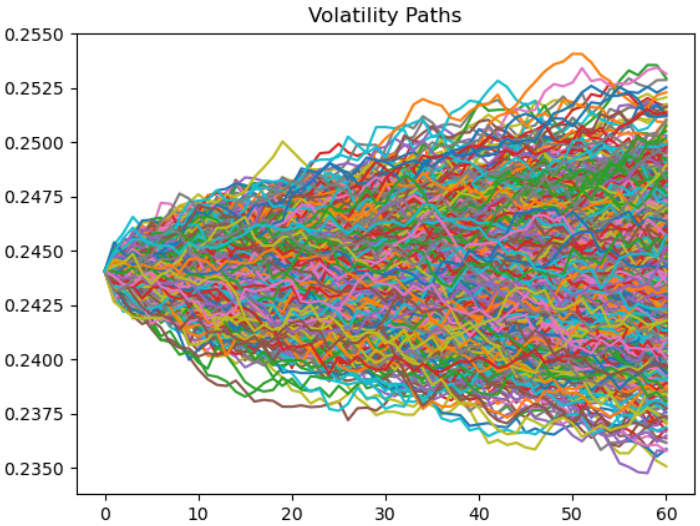
The above functions are only for Heston parameters presentation purpose. Both the QE method and the Euler method are embedded in the same class so they can read parameters from the class. Here we use monte_carlo_qe() to simulate price path and vol path and then use hu.plot_paths() to plot.

```
S_t_qe, V_t_qe = qqq_heston.monte_carlo_qe(T, n_steps, n_paths, s0, v0,
gamma1, gamma2, phiC)
hu.plot_paths(S_t_qe, 'Underlying Paths')
hu.plot_paths(V_t_qe, 'Volatility Paths')
```

price path



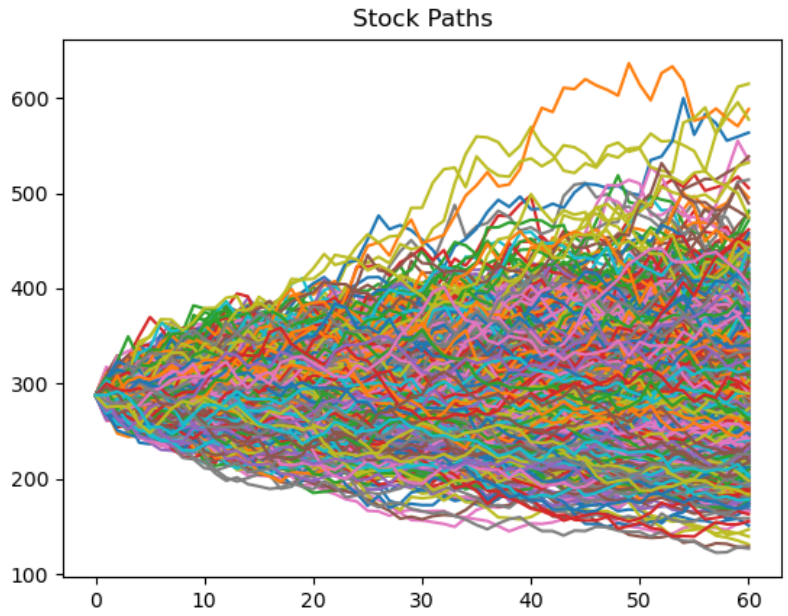
vol path



By checking the underlying path, we observed few potential extreme cases. Thus we want to further see the simulation in a longer time length in the later part.

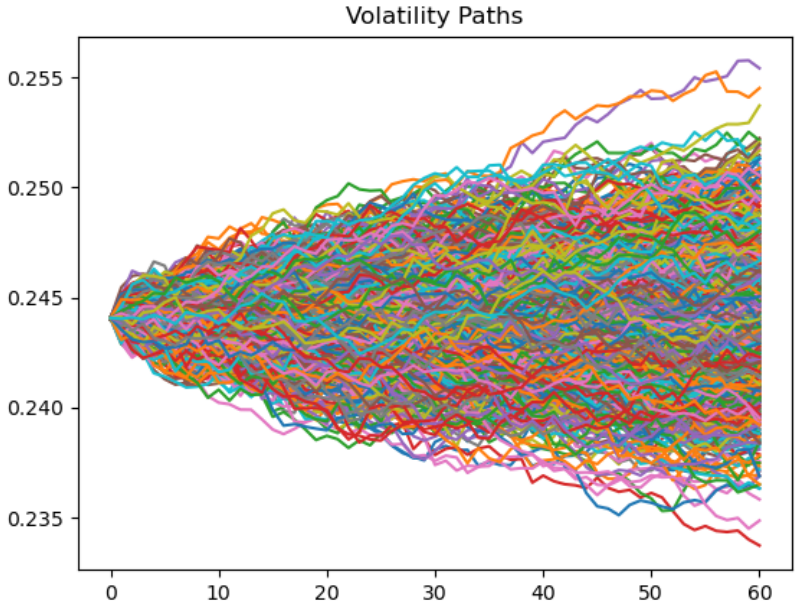
- Euler method:

price path



For the graph, we can see that the Euler method has higher volatility and more extreme cases generated. This is a common situation since the QE method has higher numerical stability and accuracy. We will talk about more in the discussion part later.

vol path



- call option pricing:

Based on different strike prices, we use option_pricing to generate option price

```
Euler_pricing=hu.option_pricing(S_t_euler,K,dt,r,n_steps)
QE_pricing=hu.option_pricing(S_t_qe,K,dt,r,n_steps)
```

qe:

	280	282	284	286	288	290
Price	22.34910	21.24555	20.18287	19.16283	18.19020	17.28671

euler:

	280	282	284	286	288	290
Price	23.36337	22.25687	21.19028	20.16444	19.18279	18.26714

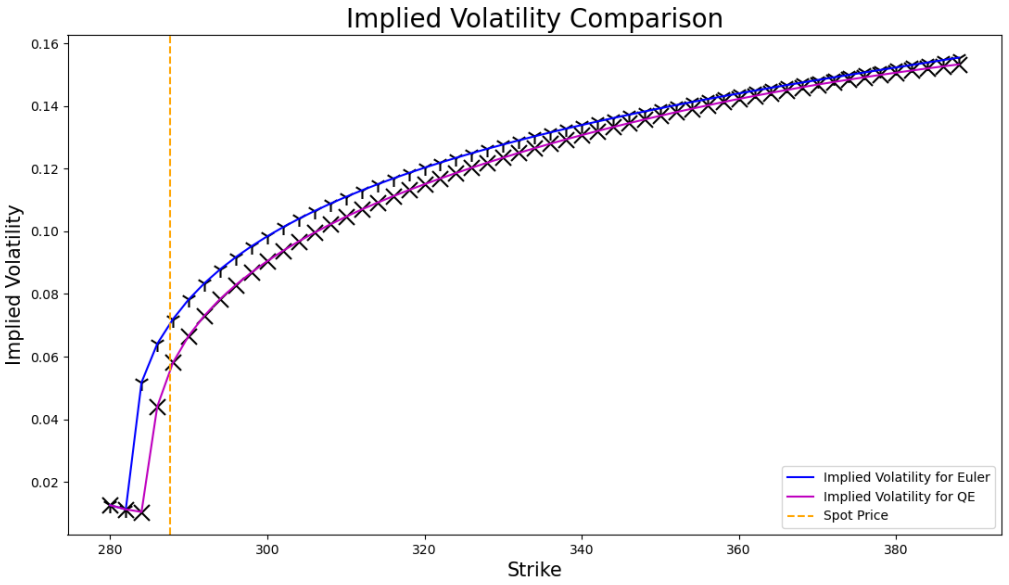
- Implied Vol

  Calculate implied vol and then plot it, using hu.implied_vol() and plot_implied_volatility()

```
imp_vol = [ ]
imp_vol_Euler = np.array([hu.implied_vol(Euler_pricing.iloc[0,
i], InitialP, K[i], year, r) for i in range(len(K))])
imp_vol_QE = np.array([hu.implied_vol(QE_pricing.iloc[0, i],
InitialP, K[i], year, r) for i in range(len(K))])

#%% plot implied vol

hu.plot_implied_volatility(K, imp_vol_Euler, imp_vol_QE,
InitialP)
```

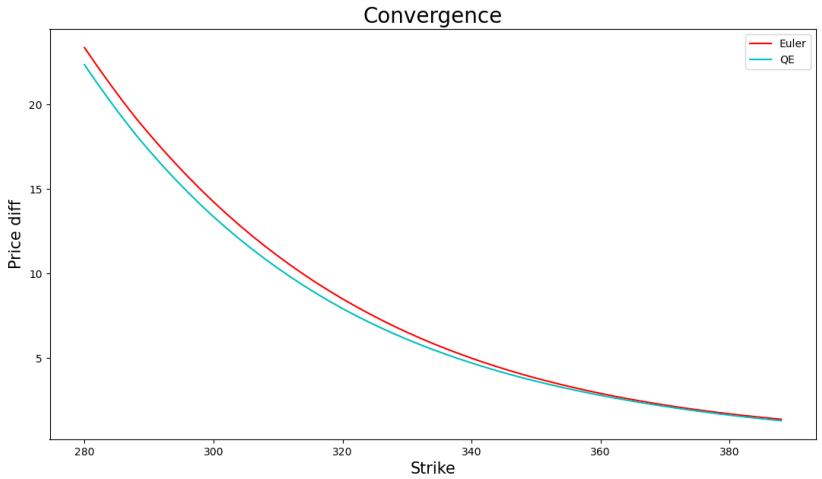


In this implied volatility graph, we observed a “volatility smirk", which refers to a pattern observed in the implied volatility of options as they move away from at-the-money (ATM). In a typical volatility smirk, the implied volatility is higher for out-of-the-money (OTM) options, particularly for put options. Unlike the symmetric 'volatility smile,' a smirk indicates an

asymmetry. This asymmetry is often more pronounced in markets where investors anticipate greater downside risk or during periods of market stress. In general, the appearance of a ‘volatility smirk’ refers to higher downside risks.

- Price convergency

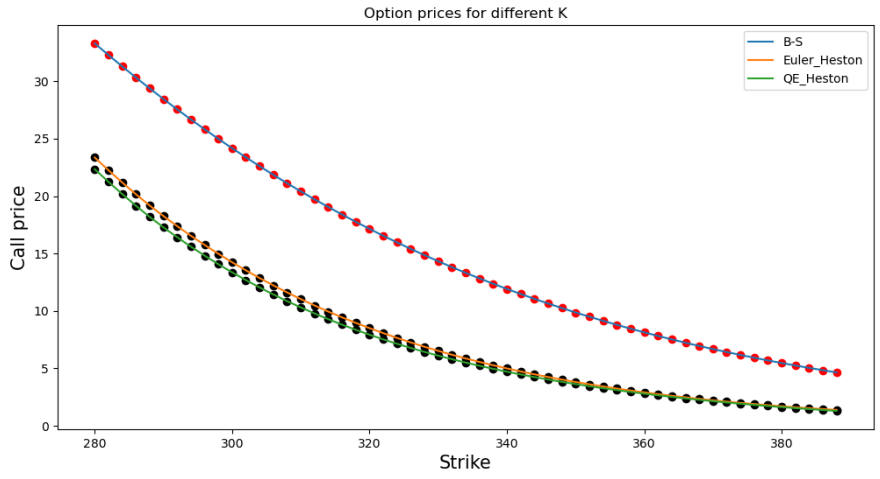
```
hu.plot_convergence(Euler_pricing, QE_pricing, K)
```



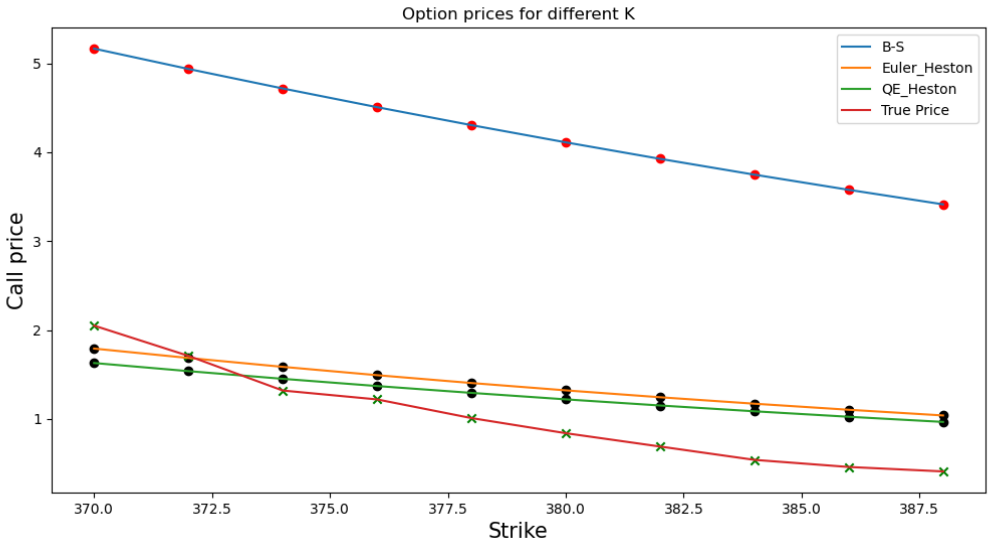
Although the QE method may provide better precision and stability, it can be computationally more complex than the Euler method and might require more computational resources. This is why we want to check the convergency of price difference between these two methods. Under 60 60-day interval, we may conclude that as the strike price increases, we don’t need the additional computational cost for higher accuracy.

- Price Comparison

```
hu.plot_option_prices(K, BS_call_price, Euler_pricing_array,
QE_pricing_array)
```



```
hu.plot_option_prices(K_last_10, BS_call_price_last_10,
Euler_pricing_array_last_10, QE_pricing_array_last_10,true_price_array)
```



We obsered a intersection between True option price, QE method and Euler method. Observation shows a better simulation result of QE and Euler, comparing with B-S.

- Error:

We calculate MSE and MAE for three methods to evaluate their performance

```
#%% cal simulation error
errors_BS = BS_call_price_last_10 - true_price_array
errors_Euler = Euler_pricing_array_last_10 - true_price_array
errors_QE = QE_pricing_array_last_10 - true_price_array

mse_BS = round(np.mean((BS_call_price_last_10 - true_price_array)**2),
4)
mse_Euler = round(np.mean((Euler_pricing_array_last_10 -
true_price_array)**2), 4)
mse_QE = round(np.mean((QE_pricing_array_last_10 -
true_price_array)**2), 4)

mae_BS = round(np.mean(np.abs(BS_call_price_last_10 -
true_price_array)), 4)
mae_Euler = round(np.mean(np.abs(Euler_pricing_array_last_10 -
true_price_array)), 4)
mae_QE = round(np.mean(np.abs(QE_pricing_array_last_10 -
true_price_array)), 4)
```

MSE - Black-Scholes: 10.3543
MSE - Euler: 0.2117
MSE - QE: 0.1612

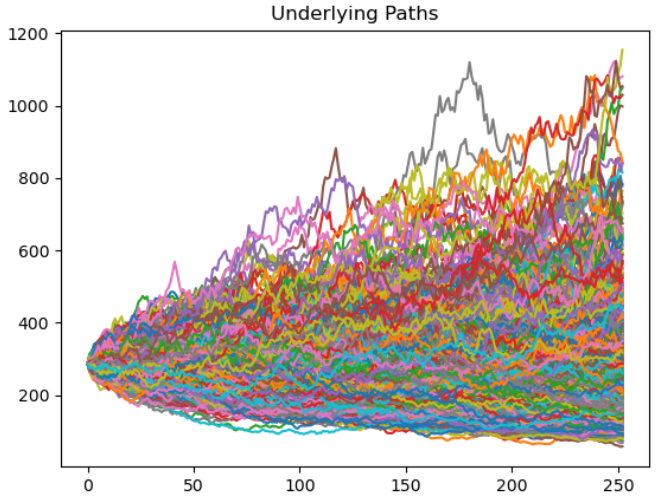
MAE - Black-Scholes: 3.2161
MAE - Euler: 0.4157

MAE - QE: 0.3668

Both MSE and MAE show that QE method had highest accuracy among 3 methods presented. we can get the same result in other cases as well.

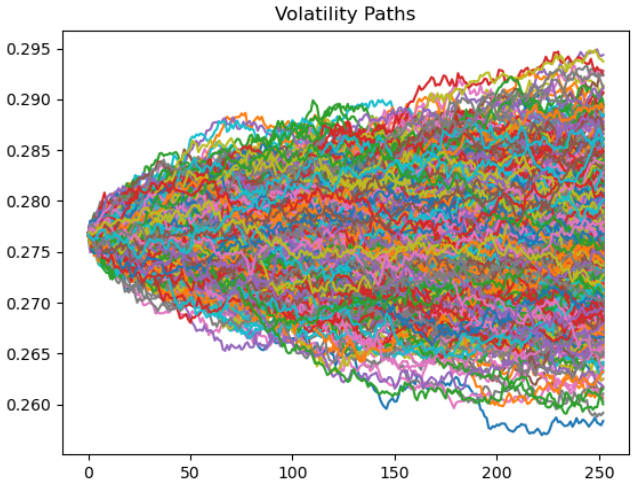
3.3.2 qqq in 252 days (1 year) interval:

- QE method:

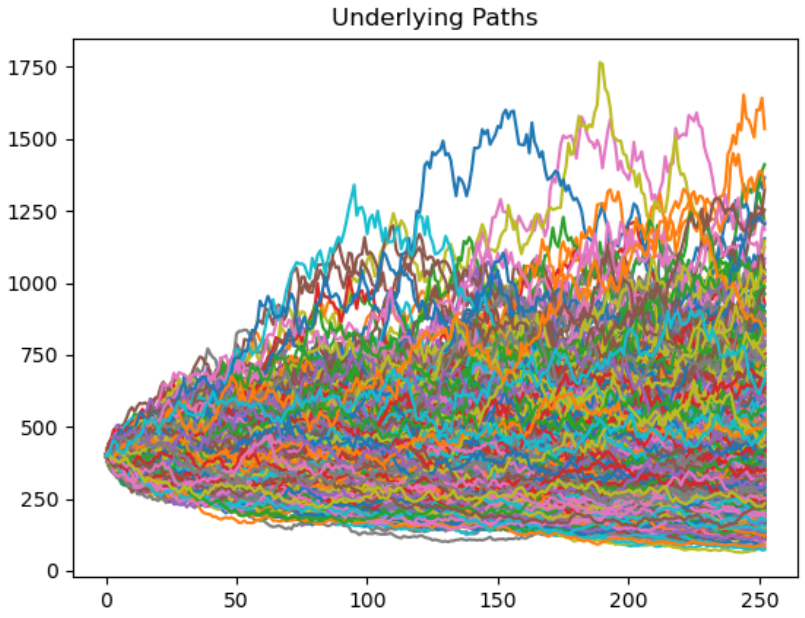


As we mentioned in last case, indeed, we observed an increasing skewness in a long-term interval. This is may due to the assumption of the Heston model, geometric brownie motion.

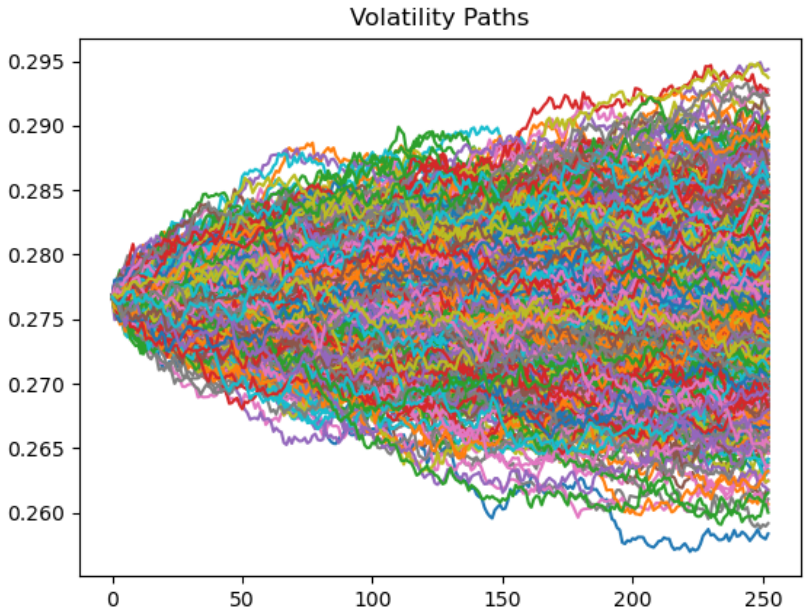
$$
\begin{aligned}
dS_t &= \mu S_t\,dt + \sqrt{v_t} S_t\,dW_{1t}, \\
dv_t &= \kappa(\theta - v_t)\,dt + \psi\sqrt{v_t}\,dW_{2t}, \\
\rho\,dt &= dW_{1t} dW_{2t}.
\end{aligned}
$$



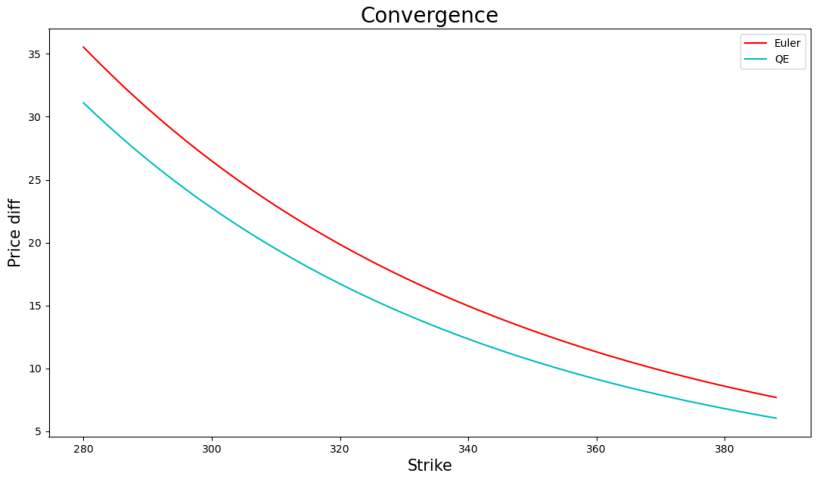
- Euler method:

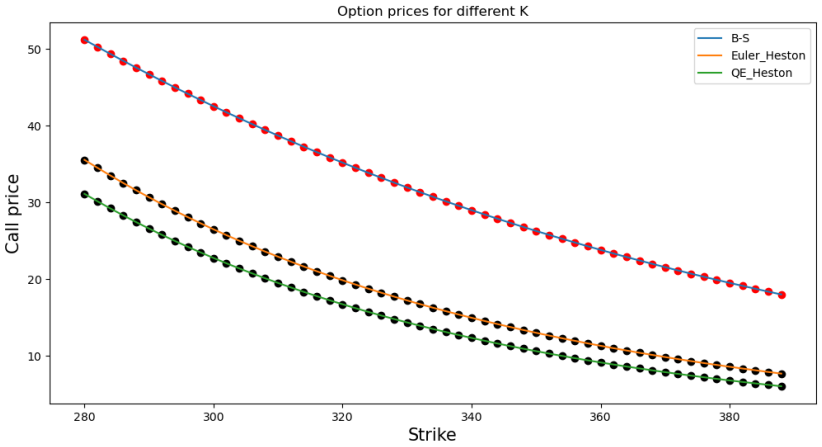


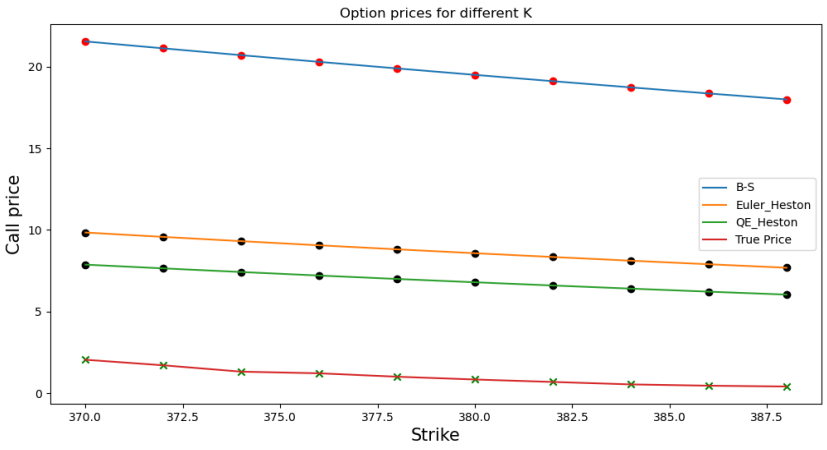
Euler method shows the same increasing trend as well.



- Price comparison







MSE - Black-Scholes: 743.1603
MSE - Euler: 122.1952
MSE - QE: 108.0265
MAE - Black-Scholes: 27.2508
MAE - Euler: 11.0501
MAE - QE: 10.3911

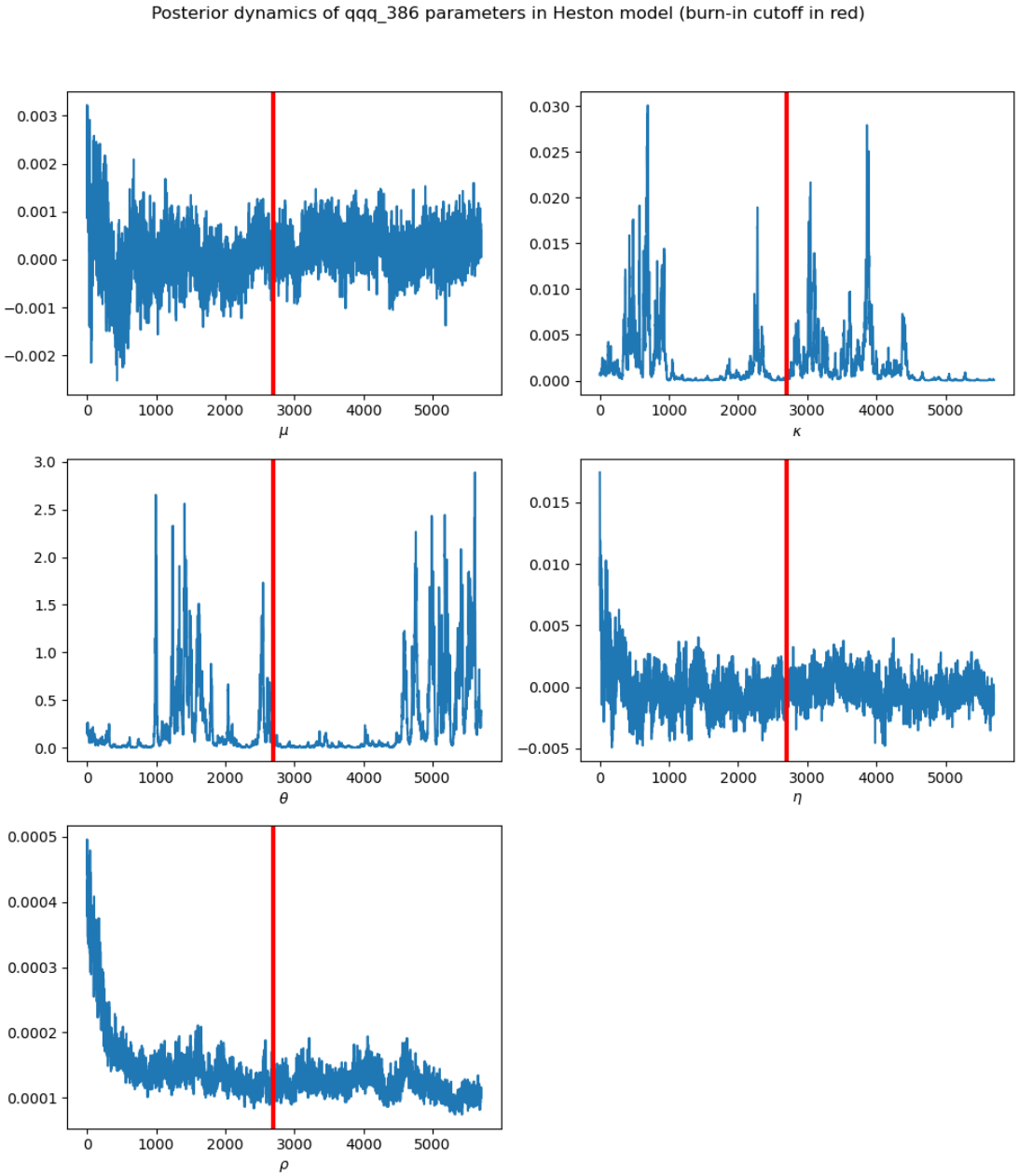
Under 252 days time length, The price predicted by Euler method didn't converge well to QE method. Thus we may say the QE method is worthy in this situation. Although even QE method has high error as well. There is much more space for algorithm improvement.

3.3.3 GLD in 60 interval

- parameter:

print(f"{mu:.4f}, {kappa:.4f}, {theta:.4f}, {eta:.4f}, {rho:.4f}")

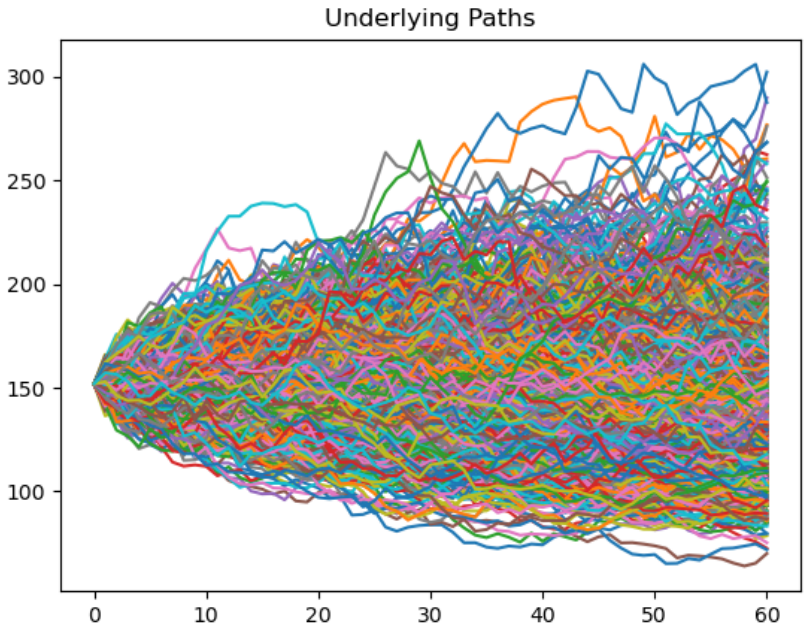
μ = 0.0003, κ = 0.0016, θ = 0.2638, ψ = 0.0110, ρ = -0.0163

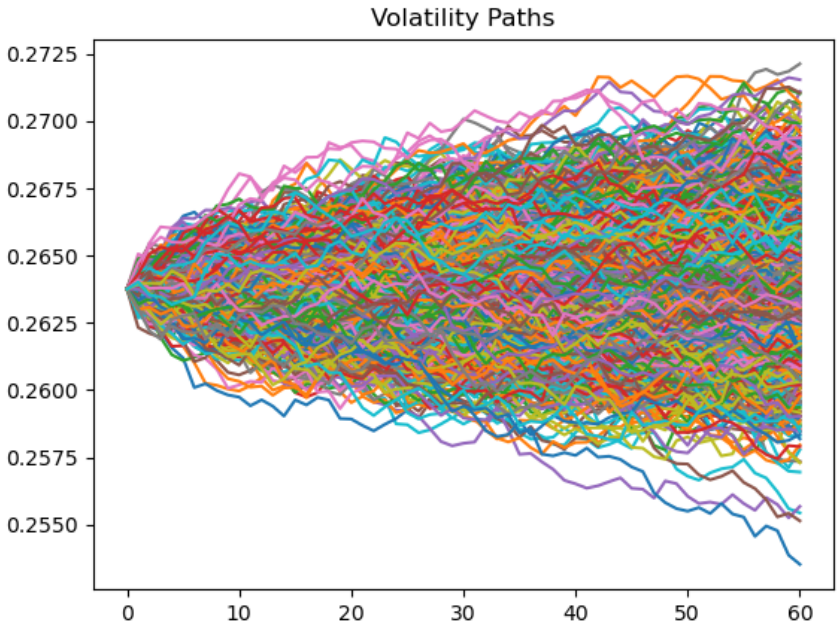


- QE method

underlying path:

```
In [13]: print(QE_pricing)
              182       183       184
Price    2.269989  2.154072  2.043582
```

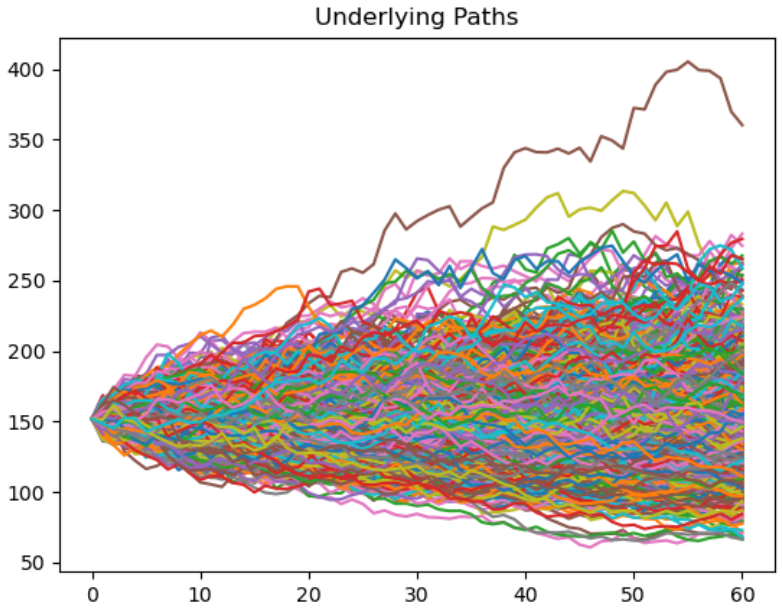


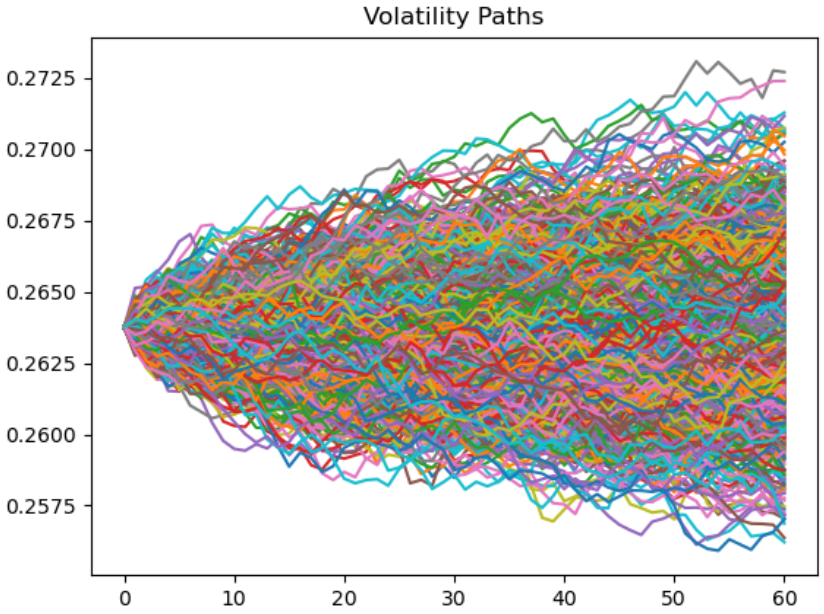


- Euler method

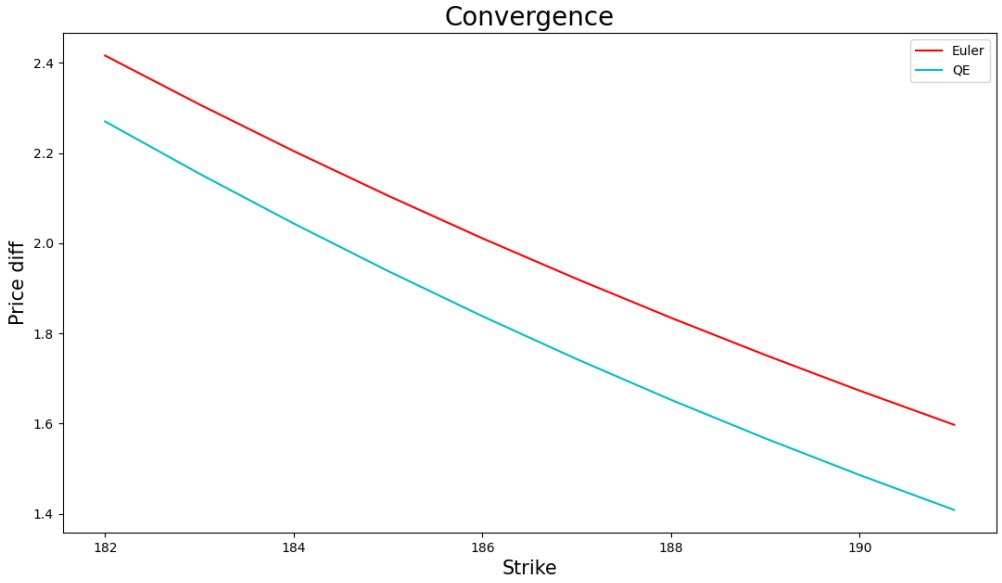
pricing path:

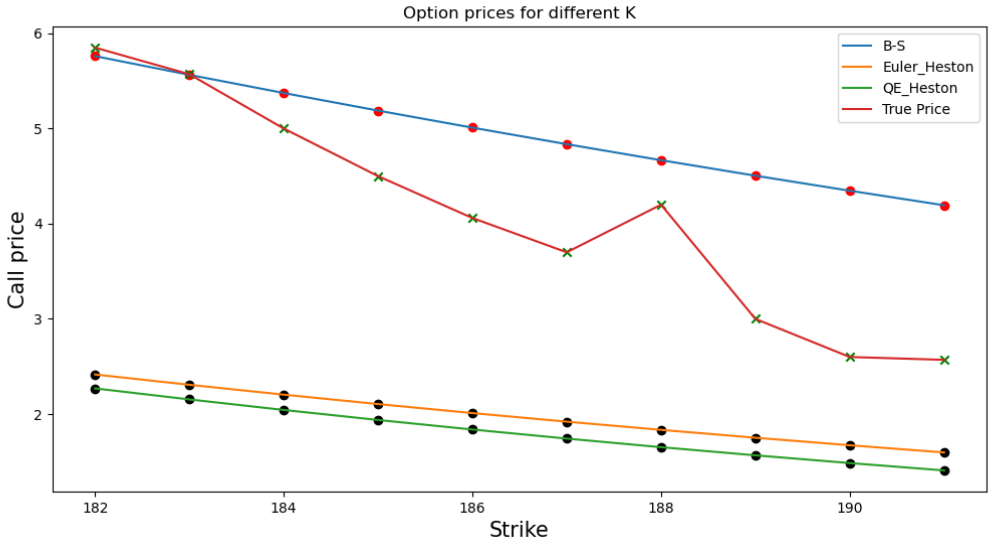
```
In [12]: print(Euler_pricing)
              182       183      184
Price    2.416121  2.307761  2.20418
```





- compare B-S,Euler, QE method with true price:





MSE - Black-Scholes: 0.4747
MSE - Euler: 7.0584
MSE - QE: 8.0321
MAE - Black-Scholes: 0.5985
MAE - Euler: 2.5115
MAE - QE: 2.6941

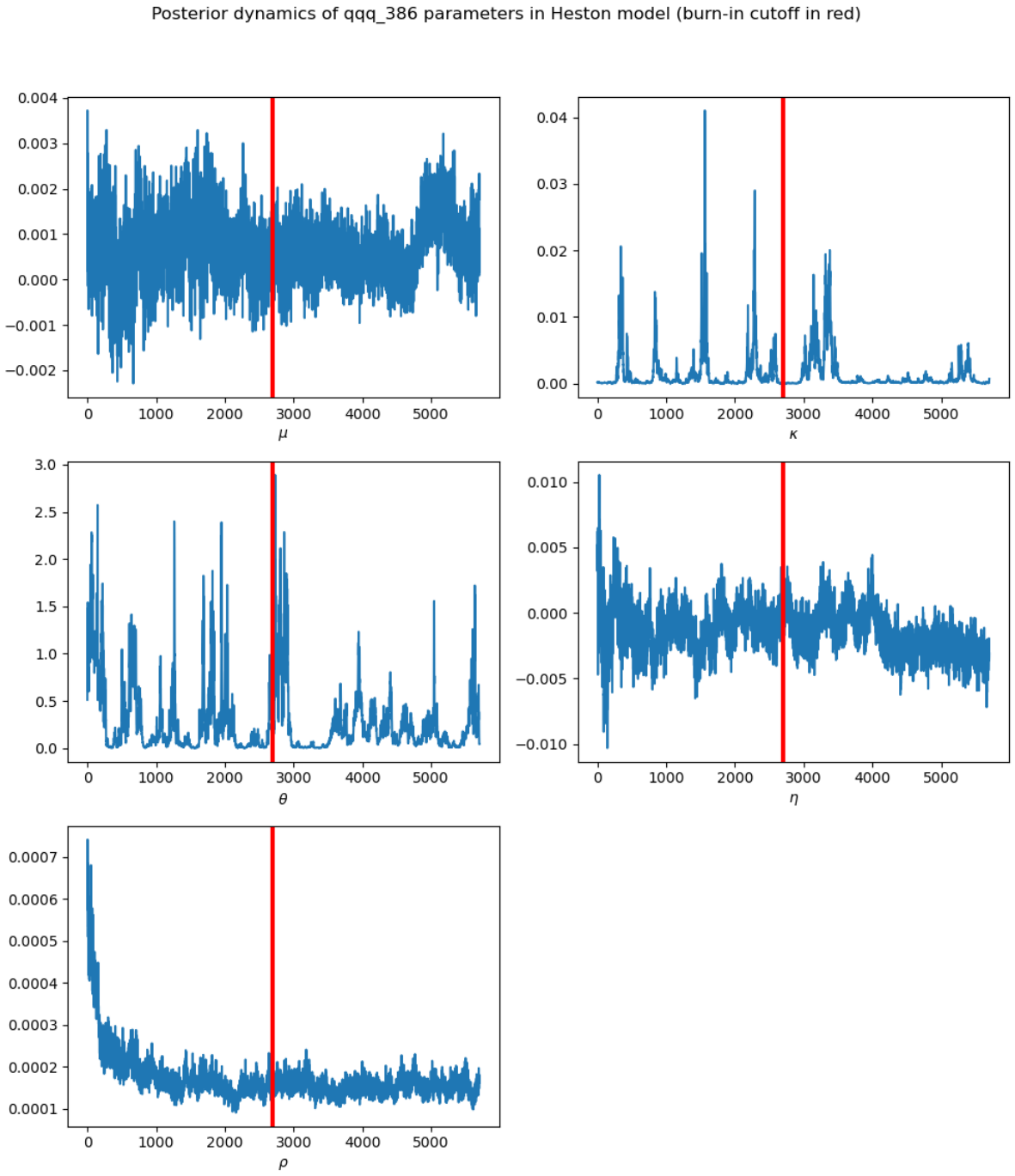
This is the only case that both Euler method and QE method underestimate option price, while price predicted by B-S model intersects with true option price. Thus B-S has the least errors. We don't know what factors lead to this uncommon case since we use the same algorithm. It could be an interest direction of research.

3.3.4 GLD in 252 interval：

- parameter：

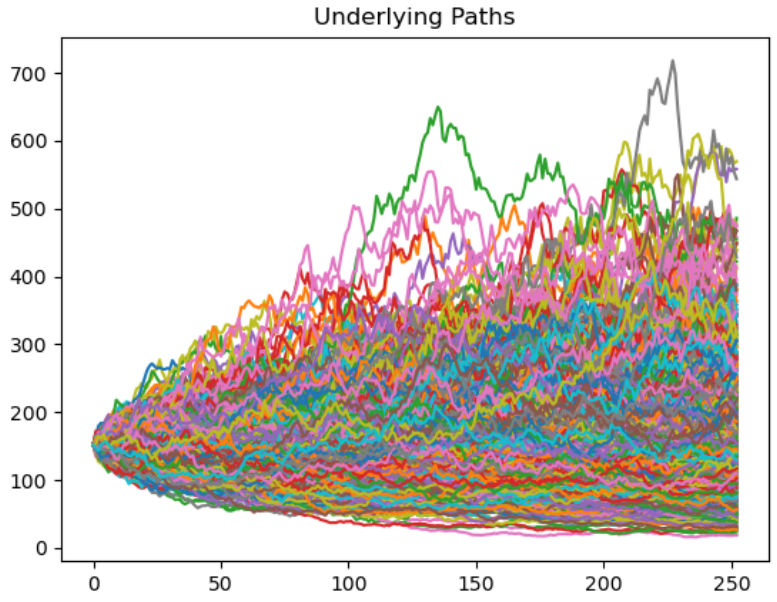
print(f"{mu:.4f}, {kappa:.4f}, {theta:.4f}, {eta:.4f}, {rho:.4f}")

μ = 0.0007, κ = 0.0012, θ = 0.2370, ψ = 0.0124, ρ = -0.1317

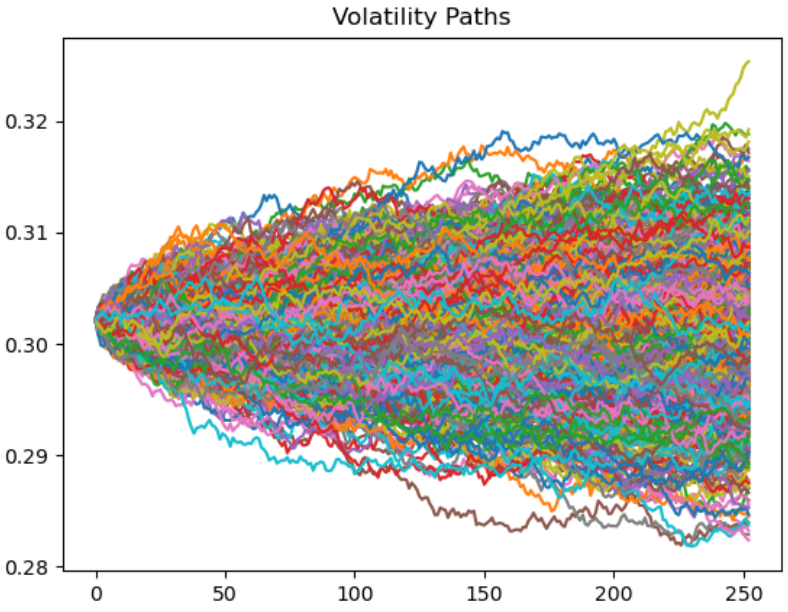


- QE method

underlying path：

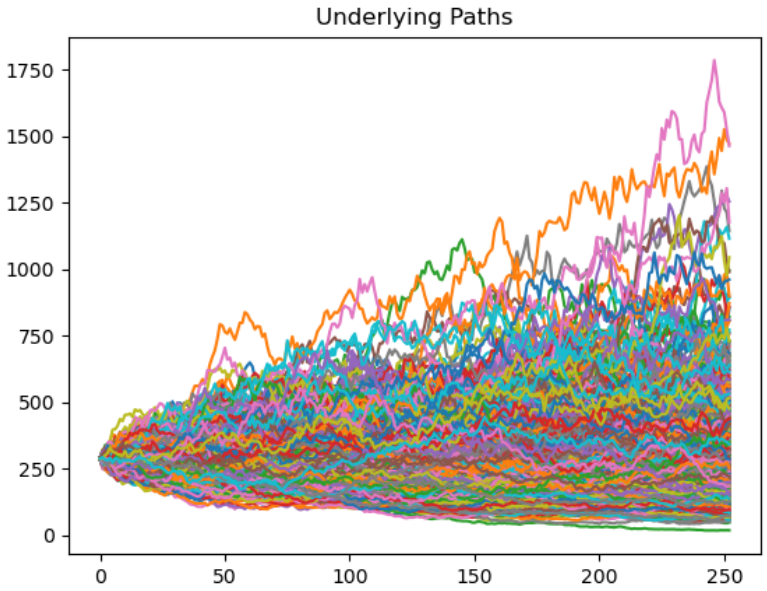


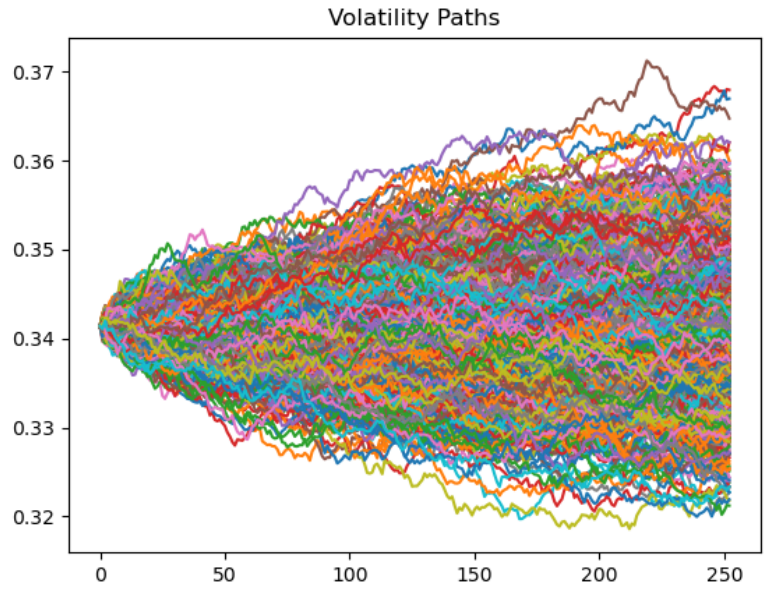
vol path




```
In [9]: print(QE_pricing)
              182       183       184
Price    6.078685  5.902127  5.730397
```

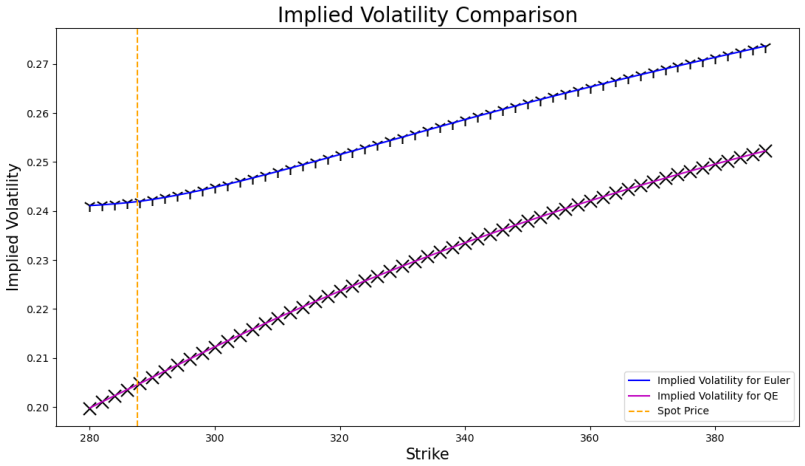
- Euler method:



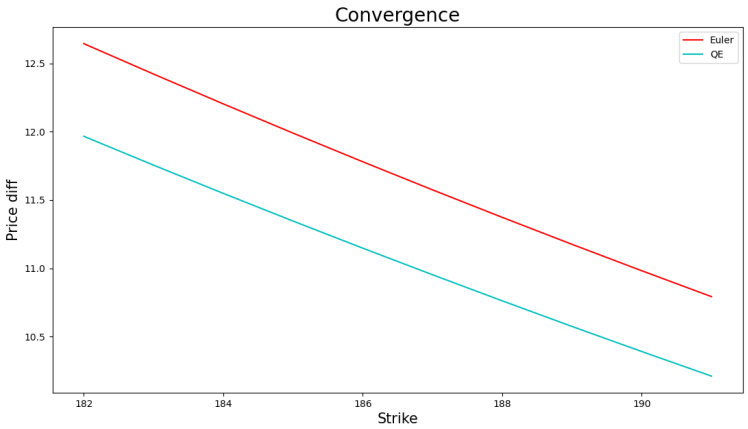


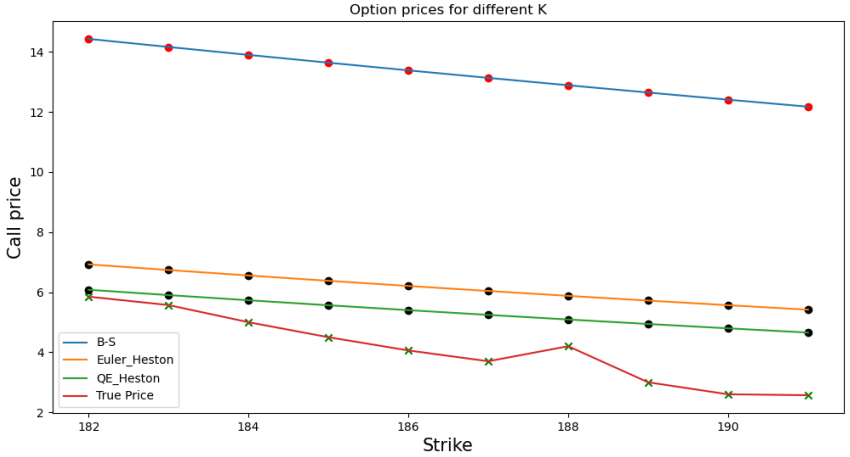

```
In [8]: print(Euler_pricing)
              182       183       184
Price    6.924288  6.736728  6.554542
```

- implied volatility



- Compare B-S,Euler, QE method with true price:





MSE - Black-Scholes: 84.2839
MSE - Euler: 4.5639
MSE - QE: 1.973
MAE - Black-Scholes: 9.1702
MAE - Euler: 2.0367
MAE - QE: 1.2349

QE method again offering the best result this time. The spot price of underlying on Oct 31, 2023, is 184.09. The QE method will intersect with the true price where the strike price equals 182. The 'parallel' price curves mean that the QE method always performs better than Euler method under this circumstance.

4. Discussions

- Strengths of MCMC method:

1. MCMC is well-suited for dealing with the complex posterior distributions that often arise in the Heston model due to the non-linear nature of the stochastic differential equations involved.
2. MCMC methods can handle the correlations between parameters efficiently, which is important in the Heston model as parameters like volatility and mean reversion can be correlated.
3. MCMC not only estimates the parameters but also provides a measure of uncertainty for these estimates, which is crucial for risk management and option pricing.

- limitation of MCMC method:

1. MCMC methods can be computationally intensive, especially when the model involves simulating complex dynamics so that they require a large number of iterations for convergence.
2. The choice of initial values can greatly affect the convergence and efficiency of the MCMC algorithm. Poor choices can lead to slow convergence.
3. MCMC is still based on some assumptions, such as the mean reversion properties of volatility that may not be fully established in actual markets.

- QE method vs Euler method

In complex financial models involving stochastic volatility, such as the Heston model, different numerical methods can significantly impact the accuracy of price

predictions. The QE method usually offers better numerical stability and accuracy in handling high volatility and extreme market conditions.

The core principle of the QE method lies in handling the random volatility process in the Heston model. A key characteristic of the Heston model is that the square root of volatility follows a geometric Brownian motion, leading to theoretical scenarios where volatility can turn negative, especially with larger discretization steps, which is implausible in real markets as volatility cannot be negative. On the other hand, this may be the reason that the simulation generated skewed path sets.

QE method introduces a quadratic term to ensure that volatility remains positive. This method used a positive quadratic term to approximate the original geometric Brownian motion, thus effectively avoiding negative volatilities in numerical simulations.

The Euler method, On the other hand, might be sufficiently accurate in certain scenarios, particularly under low volatility and mild market conditions, it may encounter numerical issues like negative volatilities in high volatility or extreme market situations.

Although the QE method may provide better precision and stability in some situations, it can be computationally more complex than the Euler method and might require more computational resources. The comparison of price convergence helps quantify the additional computational cost needed for higher accuracy.

5. References

Cape, J., Dearden, W., Gamber, W., Liebner, J., Lu, Q., & Nguyen, M. T. (2014). Estimating Heston's and Bates' models parameters using Markov chain Monte Carlo simulation. Journal of Statistical Computation and Simulation, 85(11), 2295–2314. https://doi.org/10.1080/00949655.2014.926899

Andersen, Leif B.G., Efficient Simulation of the Heston Stochastic Volatility Model (January 23, 2007). http://dx.doi.org/10.2139/ssrn.946405

Brownlee, J. (2019, September 24). A gentle introduction to Markov chain Monte Carlo for probability. MachineLearningMastery.com. https://machinelearningmastery.com/markov-chain-monte-carlo-for-probability/

Pollard, M. C. (2007). Markov Chain Monte Carlo Analysis of Option Pricing Models. https://www.semanticscholar.org/paper/Markov-Chain-Monte-Carlo-Analysis-of-Option-Pricing-Pollard-Smith/05b32641b2b63f67e642362403025186f153f91e