Case Study 4: Computational Methods in Finance

Charles Laferte - cl4249

In [3]:
```
import modulesForCalibration as mfc

import warnings
warnings.filterwarnings("ignore")

import math
import numpy as np
import scipy.integrate as integrate
import pandas as pd

from scipy.optimize import fmin, fmin_bfgs, minimize
from scipy.stats import norm

import cmath
import math

from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
%matplotlib inline

from datetime import datetime
from tqdm import tqdm
from matplotlib import cm
```

Simulate the market data with Heston Stochastic Volatility Model

Let's set the parameters we will consider for the Heston model:

In [4]:
```
S0 = 4100
r = 0.0485
mu = r
params = [0.02, 1.5, 0.05, 0.18, 0.5, 0.04]
```

Here, we chose to considerate the parameter we obtain working on the SP500 data during the last assignment.

In [36]:
```
N = 10*252 # 10 years * 252 trading day per year
T = 10
dt = T/N
```

In [37]:
```
def heston_simulated_prices(params, N, T, S0, r, mu, plot = False):
    kappa  = params[0]
    theta  = params[1]
    sigma  = params[2]
    rho    = params[3]
```

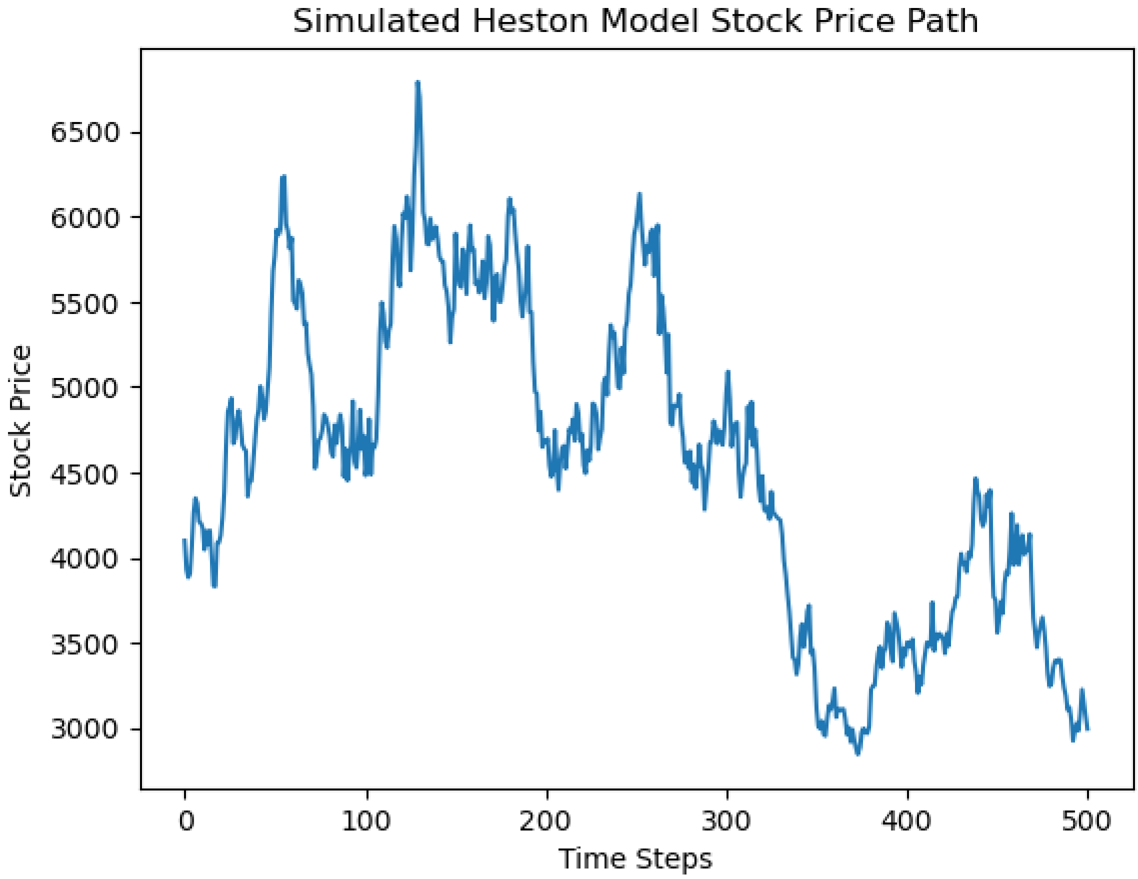
```
    v0     = params[4]

    # Define discretization parameters
    dt = T/N        # time increment
    M = 1           # number of simulations
    print('T:',T,'  N:', N, '  dt:', dt)
    #print(v0)

    # Generate random numbers
    Z1 = norm.rvs(size=(N, M))
    Z2 = rho*Z1 + np.sqrt(1-rho**2)*norm.rvs(size=(N, M))

    # Define arrays to store stock price and volatility paths
    S = np.zeros((N+1, M))
    v = np.zeros((N+1, M))

    # Set initial values
    S[0,:] = S0
    v[0,:] = v0 #theta
    #print(v0)

    # Calculate paths
    for i in range(N):
        v[i+1,:] = np.maximum(0, v[i,:] + kappa*(theta-v[i,:])*dt + sigma*np
        #print(v[i+1,:])
        S[i+1,:] = S[i,:] * np.exp((mu - 0.5*v[i,:])*dt + np.sqrt(v[i,:])*np
        #print(S[i+1,:])

    # Plot results
    if plot == True:
        plt.plot(S)
        plt.title('Simulated Heston Model Stock Price Path')
        plt.xlabel('Time Steps')
        plt.ylabel('Stock Price')
        plt.show()

        plt.plot(v)
        plt.title('Simulated Heston Model volatility Path')
        plt.xlabel('Time Steps')
        plt.ylabel('Volatility')
        plt.show()

    # Reshaping the outputs
    y = np.log(S)
    S = S.T
    S = S[0]
    v = v.T
    v = v[0]
    y = y.T
    y = y[0]

    return S, v, y
```

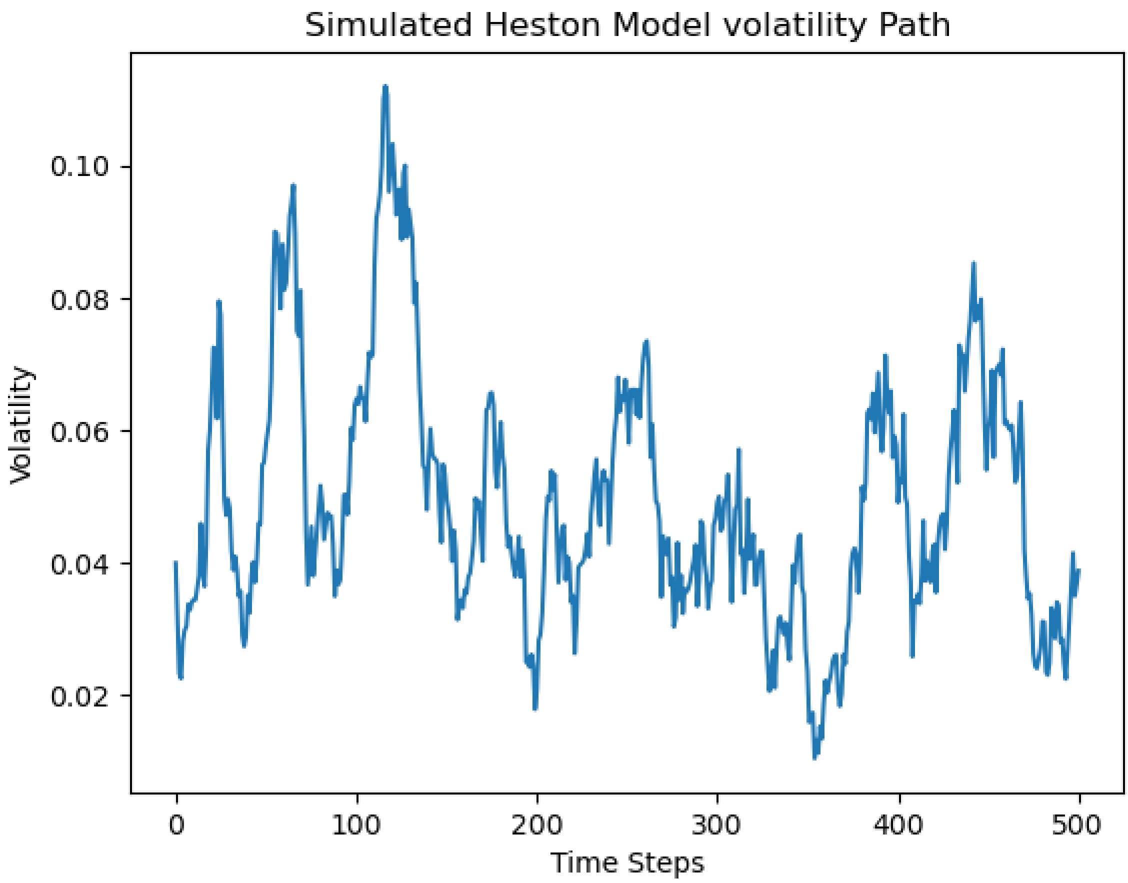
In [38]:

```
#prices, v_, y = heston_simulated_prices(params, N, T, S0, r, mu, plot = Tru
S0 = 4100
r = 0.0485
mu = r
params = [mu, 1.5, 0.05, 0.18, 0.5, 0.04]
#.        mu, kappa, theta, lambda_, rho , v0
N = 500
```

```
T = 10

params_sim = np.zeros(5)
params_sim[0] = params[1]# kappa
params_sim[1] = params[2]#theta
params_sim[2] = params[3]#sigma
params_sim[3] = params[4]#rho
params_sim[4] = params[5]#v0
#print(params_sim)
prices, v_, y = heston_simulated_prices(params_sim, N, T, S0, r, mu, plot =
```

```
T: 10   N: 500   dt: 0.02
```





1. Extended Kalman Filter

In [211…

```
def Extended_Kalman_Filter(params):
    # Declare global variables
    global y_EKF, v_EKF

    # Extract parameters
    mu, kappa, theta, lbda, rho, v0 = params

    dt = T / N  # Calculate time step

    # Initialize matrices
    P = np.matrix([[0.01, 0], [0, 0.01]])
    I = np.identity(2)
    F = np.matrix([[1, -0.5*dt], [0, 1-kappa*dt]])
    U = np.matrix([[np.sqrt(v0*dt), 0], [0, lbda*np.sqrt(v0*dt)]])
    Q = np.matrix([[1, rho], [rho, 1]])
    H = np.matrix([1, 0])

    # Initialize state variables
    x_update = np.matrix([np.log(S0), v0]).T

    # Initialize arrays for storing results
    y_EKF = np.zeros(N)
    v_EKF = np.zeros(N)
    y_EKF[0] = np.log(S0)
    v_EKF[0] = v0

    loglk = 0  # Initialize objective function value
    for i in range(1, N):
        # Prediction step
        x_pred = np.matrix([0, 0], dtype=np.float64).T
```

```
        x_pred[0, 0] = x_update[0, 0] + (mu - 1/2*x_update[1, 0]) * dt
        x_pred[1, 0] = x_update[1, 0] + kappa * (theta - x_update[1, 0]) * c

        P_1 = F * P * F.T + U * Q * U.T

        # Update step
        S = H * P_1 * H.T
        S = S[0, 0]

        e = y[i] - x_pred[0, 0]
        loglk += np.log(abs(S)) + e**2 / S

        K = P_1 * H.T / S
        x_update = x_pred + K * e

        # Apply full truncate scheme to avoid negative volatility values
        x_update[1, 0] = max(1e-5, x_update[1, 0])

        vk = x_update[1, 0]
        U = np.matrix([[np.sqrt(vk*dt), 0], [0, lbda*np.sqrt(vk*dt)]])
        P = (I - K * H) * P_1

        y_EKF[i] = x_update[0, 0]
        v_EKF[i] = x_update[1, 0]

    return loglk #/ N
```

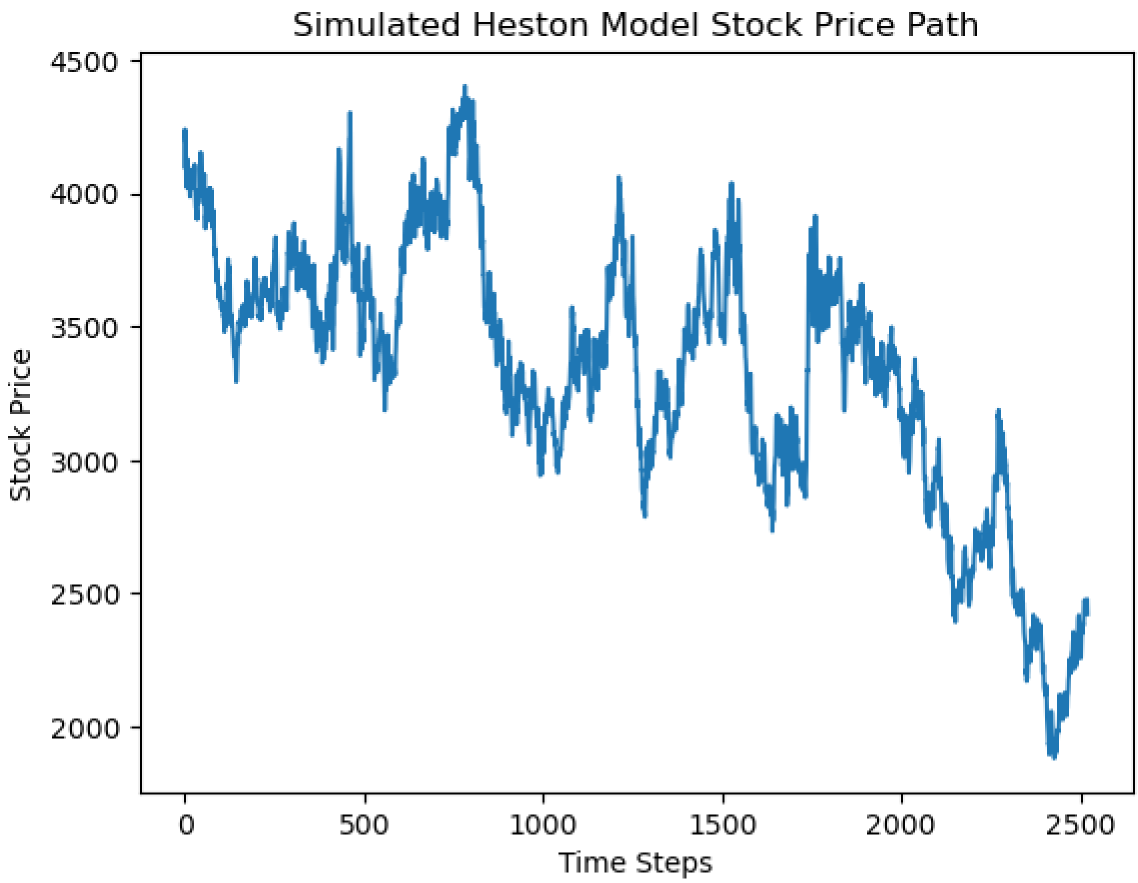
In [199…

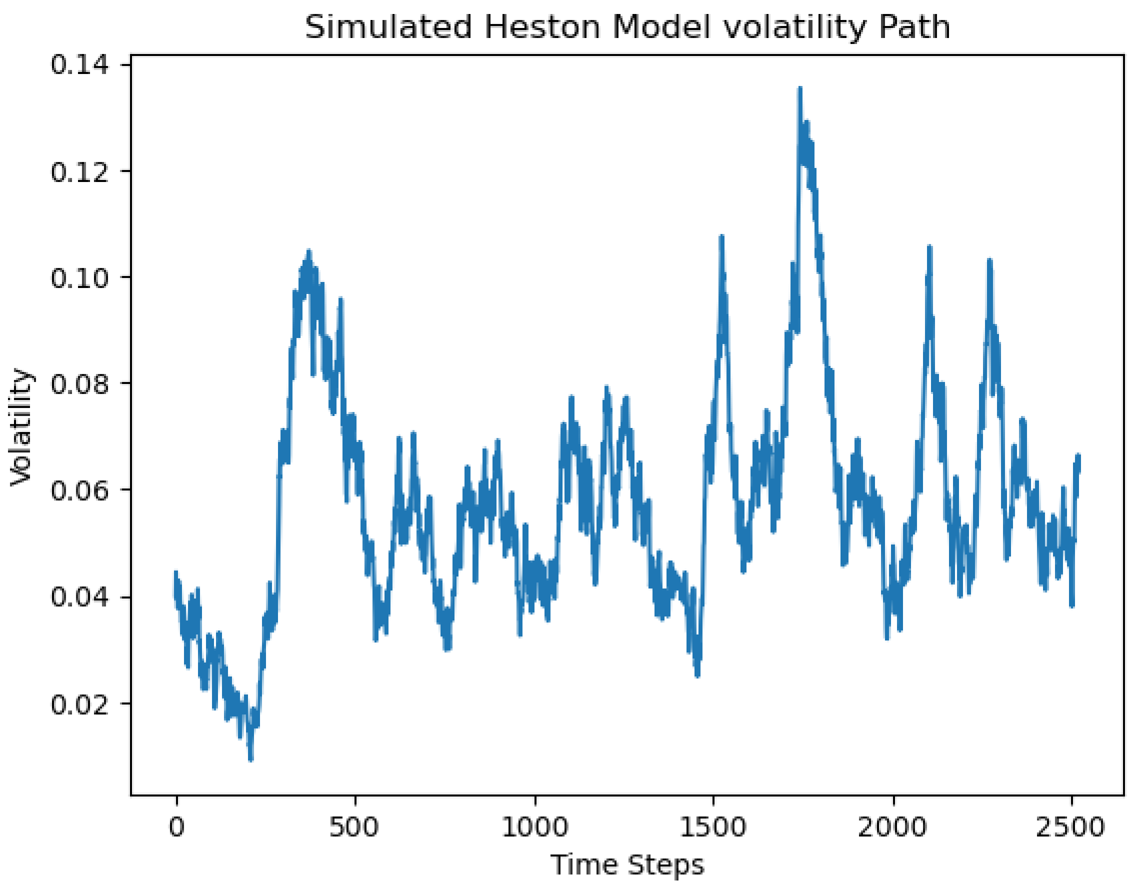
```
# Define the objective function for optimization with noise
def objective(params):
    noise = np.random.normal(scale=0.01, size=len(params))  # Add random noi
    params_with_noise = params + noise
    return Extended_Kalman_Filter(params_with_noise)
```

In [212…

```
S0 = 4100
r = 0.0485
mu = r
params = [mu, 1.5, 0.05, 0.18, 0.5, 0.04]
#.        mu, kappa, theta, lambda_, rho , v0
N = 10*252 # 10 years * 252 trading day per year
T = 10

params_sim = np.zeros(5)
params_sim[0] = params[1]# kappa
params_sim[1] = params[2]#theta
params_sim[2] = params[3]#sigma
params_sim[3] = params[4]#rho
params_sim[4] = params[5]#v0
#print(params_sim)
prices, v_, y = heston_simulated_prices(params_sim, N, T, S0, r, mu, plot =

dt = T/N
#print(dt)
```

```
T: 10   N: 2520   dt: 0.003968253968253968
```





Set the initial guess for the optimization :

In [219…

```
# true : [0.0485, 1.5, 0.05, 0.18, 0.5, 0.04]
params_0 = [0.1, 0.1, 0.1, 0.1, 0.1, 0.05]
params_0 = [0.02, 1.5, 0.05, 0.18, 0.5, 0.1]
params_0 = [0.06, 1.3, 0.07, 0.3, 0.6, 0.04]
```

Use the Nelder-Mead algorithm to minimize the function

In [220…
```
def opt_param_research():

    def callback(x):
        print("Current parameter vector:", x)
        print("Function value: ", Extended_Kalman_Filter(x))

    #constraint1 = {'type': 'ineq', 'fun': lambda x: x}#[0] - 1}
    #constraint2 = {'type': 'ineq', 'fun': lambda x: 2 - x[0]}
    #constraints = [constraint1]#, constraint2]
    bounds = [(0.000001, 30)]*6
    #bounds[1] = (0,2)
    bounds[4] = (-1,1)
    xopt = minimize(Extended_Kalman_Filter, params_0, callback=callback, met
    #xopt_noise =minimize(objective, params_0, callback=callback, method='Ne

    #result_2 = minimize(obj_function_ext_KF_m, params_0, bounds=bounds, cal
    #xopt, fopt, _, _, _ = fmin(ext_Kalman_filter, params_0, maxiter=100, ca

    #result = fmin(ext_Kalman_filter, params_0, callback=callback)
    print(80*'=')
    print('Optimal parameter set:')
    print(xopt)
    print(80*'=')

    return xopt
```

In [221…
```
result_EKF = opt_param_research()
```

```
Current parameter vector: [0.0615  1.3325  0.07175 0.3075  0.54    0.041  ]
Function value:  -18574.696118761647
Current parameter vector: [0.0615  1.3325  0.07175 0.3075  0.54    0.041  ]
Function value:  -18574.696118761647
Current parameter vector: [0.0615  1.3325  0.07175 0.3075  0.54    0.041  ]
Function value:  -18574.696118761647
Current parameter vector: [0.0605     1.38666667 0.07466667 0.29666667 0.54
666667 0.03766667]
Function value:  -18583.5072747409
Current parameter vector: [0.0605     1.38666667 0.07466667 0.29666667 0.54
666667 0.03766667]
Function value:  -18583.5072747409
Current parameter vector: [0.0605     1.38666667 0.07466667 0.29666667 0.54
666667 0.03766667]
Function value:  -18583.5072747409
Current parameter vector: [0.06133333 1.45527778 0.07369444 0.29111111 0.45
777778 0.04088889]
Function value:  -18588.43820846982
Current parameter vector: [0.06133333 1.45527778 0.07369444 0.29111111 0.45
777778 0.04088889]
Function value:  -18588.43820846982
Current parameter vector: [0.06133333 1.45527778 0.07369444 0.29111111 0.45
777778 0.04088889]
Function value:  -18588.43820846982
Current parameter vector: [0.06133333 1.45527778 0.07369444 0.29111111 0.45
777778 0.04088889]
Function value:  -18588.43820846982
Current parameter vector: [0.06133333 1.45527778 0.07369444 0.29111111 0.45
777778 0.04088889]
Function value:  -18588.43820846982
Current parameter vector: [0.06133333 1.45527778 0.07369444 0.29111111 0.45
777778 0.04088889]
Function value:  -18588.43820846982
Current parameter vector: [0.06429321 1.54555556 0.07155556 0.29924897 0.42
853909 0.03717764]
Function value:  -18596.043725327778
Current parameter vector: [0.06429321 1.54555556 0.07155556 0.29924897 0.42
853909 0.03717764]
Function value:  -18596.043725327778
Current parameter vector: [0.06429321 1.54555556 0.07155556 0.29924897 0.42
853909 0.03717764]
Function value:  -18596.043725327778
Current parameter vector: [0.06429321 1.54555556 0.07155556 0.29924897 0.42
853909 0.03717764]
Function value:  -18596.043725327778
Current parameter vector: [0.06429321 1.54555556 0.07155556 0.29924897 0.42
853909 0.03717764]
Function value:  -18596.043725327778
Current parameter vector: [0.06429321 1.54555556 0.07155556 0.29924897 0.42
853909 0.03717764]
Function value:  -18596.043725327778
Current parameter vector: [0.06263072 1.61933813 0.06854287 0.28614074 0.43
100385 0.03671291]
Function value:  -18600.15329127393
Current parameter vector: [0.0696436  1.72784979 0.06583745 0.29696831 0.31
173211 0.03460983]
Function value:  -18603.637949274245
Current parameter vector: [0.0696436  1.72784979 0.06583745 0.29696831 0.31
173211 0.03460983]
Function value:  -18603.637949274245
Current parameter vector: [0.0696436  1.72784979 0.06583745 0.29696831 0.31
```

```
173211 0.03460983]
Function value:  -18603.637949274245
Current parameter vector: [0.0666243  1.80824531 0.06456596 0.29542065 0.33
90924  0.03398798]
Function value:  -18604.56171581265
Current parameter vector: [0.0666243  1.80824531 0.06456596 0.29542065 0.33
90924  0.03398798]
Function value:  -18604.56171581265
Current parameter vector: [0.0666243  1.80824531 0.06456596 0.29542065 0.33
90924  0.03398798]
Function value:  -18604.56171581265
Current parameter vector: [0.0666243  1.80824531 0.06456596 0.29542065 0.33
90924  0.03398798]
Function value:  -18604.56171581265
Current parameter vector: [0.0666243  1.80824531 0.06456596 0.29542065 0.33
90924  0.03398798]
Function value:  -18604.56171581265
Current parameter vector: [0.0666243  1.80824531 0.06456596 0.29542065 0.33
90924  0.03398798]
Function value:  -18604.56171581265
Current parameter vector: [0.06806589 1.78531062 0.06378107 0.29667082 0.35
099712 0.03196741]
Function value:  -18604.607333898377
Current parameter vector: [0.06569758 1.78422826 0.06503129 0.29009689 0.34
155404 0.03308928]
Function value:  -18605.110117266748
Current parameter vector: [0.06941321 1.84537899 0.06412378 0.3017672  0.28
33234  0.03118948]
Function value:  -18605.142688659787
Current parameter vector: [0.06941321 1.84537899 0.06412378 0.3017672  0.28
33234  0.03118948]
Function value:  -18605.142688659787
Current parameter vector: [0.06941321 1.84537899 0.06412378 0.3017672  0.28
33234  0.03118948]
Function value:  -18605.142688659787
Current parameter vector: [0.06941321 1.84537899 0.06412378 0.3017672  0.28
33234  0.03118948]
Function value:  -18605.142688659787
Current parameter vector: [0.07151678 1.79553558 0.06411776 0.29662244 0.29
757002 0.02843466]
Function value:  -18606.641566214697
Current parameter vector: [0.07151678 1.79553558 0.06411776 0.29662244 0.29
757002 0.02843466]
Function value:  -18606.641566214697
Current parameter vector: [0.07151678 1.79553558 0.06411776 0.29662244 0.29
757002 0.02843466]
Function value:  -18606.641566214697
Current parameter vector: [0.07151678 1.79553558 0.06411776 0.29662244 0.29
757002 0.02843466]
Function value:  -18606.641566214697
Current parameter vector: [0.07151678 1.79553558 0.06411776 0.29662244 0.29
757002 0.02843466]
Function value:  -18606.641566214697
Current parameter vector: [0.07536361 1.83626605 0.06394267 0.3079646  0.27
862058 0.02557274]
Function value:  -18606.964633574695
Current parameter vector: [0.07536361 1.83626605 0.06394267 0.3079646  0.27
862058 0.02557274]
Function value:  -18606.964633574695
Current parameter vector: [0.07202834 1.79100564 0.06549153 0.2966501  0.31
344252 0.02632022]
```

```
Function value:  -18607.23513428986
Current parameter vector: [0.07202834 1.79100564 0.06549153 0.2966501  0.31
344252 0.02632022]
Function value:  -18607.23513428986
Current parameter vector: [0.07589744 1.83131994 0.06344312 0.30219409 0.28
406633 0.0227217 ]
Function value:  -18607.300190455928
Current parameter vector: [0.07811693 1.78591659 0.06424787 0.30449672 0.28
067728 0.02282795]
Function value:  -18607.464861650213
Current parameter vector: [0.07727365 1.82186428 0.06431645 0.30477857 0.30
539147 0.02188089]
Function value:  -18607.82460234764
Current parameter vector: [0.07727365 1.82186428 0.06431645 0.30477857 0.30
539147 0.02188089]
Function value:  -18607.82460234764
Current parameter vector: [0.07727365 1.82186428 0.06431645 0.30477857 0.30
539147 0.02188089]
Function value:  -18607.82460234764
Current parameter vector: [0.07512987 1.75148442 0.06545359 0.29476628 0.32
796491 0.0222509 ]
Function value:  -18608.243910302157
Current parameter vector: [0.07512987 1.75148442 0.06545359 0.29476628 0.32
796491 0.0222509 ]
Function value:  -18608.243910302157
Current parameter vector: [0.07512987 1.75148442 0.06545359 0.29476628 0.32
796491 0.0222509 ]
Function value:  -18608.243910302157
Current parameter vector: [0.07512987 1.75148442 0.06545359 0.29476628 0.32
796491 0.0222509 ]
Function value:  -18608.243910302157
Current parameter vector: [0.07512987 1.75148442 0.06545359 0.29476628 0.32
796491 0.0222509 ]
Function value:  -18608.243910302157
Current parameter vector: [0.07512987 1.75148442 0.06545359 0.29476628 0.32
796491 0.0222509 ]
Function value:  -18608.243910302157
Current parameter vector: [0.07512987 1.75148442 0.06545359 0.29476628 0.32
796491 0.0222509 ]
Function value:  -18608.243910302157
Current parameter vector: [0.07512987 1.75148442 0.06545359 0.29476628 0.32
796491 0.0222509 ]
Function value:  -18608.243910302157
Current parameter vector: [0.07512987 1.75148442 0.06545359 0.29476628 0.32
796491 0.0222509 ]
Function value:  -18608.243910302157
Current parameter vector: [0.07512987 1.75148442 0.06545359 0.29476628 0.32
796491 0.0222509 ]
Function value:  -18608.243910302157
Current parameter vector: [0.07512987 1.75148442 0.06545359 0.29476628 0.32
796491 0.0222509 ]
Function value:  -18608.243910302157
Current parameter vector: [0.07512987 1.75148442 0.06545359 0.29476628 0.32
796491 0.0222509 ]
Function value:  -18608.243910302157
Current parameter vector: [0.07542546 1.7455393  0.06555336 0.29483184 0.33
634751 0.02065308]
Function value:  -18608.267910590548
```

```
Current parameter vector: [0.07703999 1.72609785 0.06572555 0.29616824 0.32
716452 0.02052323]
Function value:  -18608.270548974055
Current parameter vector: [0.07480402 1.72165924 0.06568394 0.29197681 0.34
109707 0.02211956]
Function value:  -18608.32979332549
Current parameter vector: [0.07480402 1.72165924 0.06568394 0.29197681 0.34
109707 0.02211956]
Function value:  -18608.32979332549
Current parameter vector: [0.07480402 1.72165924 0.06568394 0.29197681 0.34
109707 0.02211956]
Function value:  -18608.32979332549
Current parameter vector: [0.07480402 1.72165924 0.06568394 0.29197681 0.34
109707 0.02211956]
Function value:  -18608.32979332549
Current parameter vector: [0.07480402 1.72165924 0.06568394 0.29197681 0.34
109707 0.02211956]
Function value:  -18608.32979332549
Current parameter vector: [0.07480402 1.72165924 0.06568394 0.29197681 0.34
109707 0.02211956]
Function value:  -18608.32979332549
Current parameter vector: [0.07480402 1.72165924 0.06568394 0.29197681 0.34
109707 0.02211956]
Function value:  -18608.32979332549
Current parameter vector: [0.07480402 1.72165924 0.06568394 0.29197681 0.34
109707 0.02211956]
Function value:  -18608.32979332549
Current parameter vector: [0.07390152 1.72039658 0.06554802 0.28534762 0.33
50847  0.02150864]
Function value:  -18608.37580534408
Current parameter vector: [0.07390152 1.72039658 0.06554802 0.28534762 0.33
50847  0.02150864]
Function value:  -18608.37580534408
Current parameter vector: [0.07482587 1.68186128 0.06606293 0.28513297 0.34
760359 0.02110112]
Function value:  -18608.415801042436
Current parameter vector: [0.07482587 1.68186128 0.06606293 0.28513297 0.34
760359 0.02110112]
Function value:  -18608.415801042436
Current parameter vector: [0.07482587 1.68186128 0.06606293 0.28513297 0.34
760359 0.02110112]
Function value:  -18608.415801042436
Current parameter vector: [0.07192639 1.69499858 0.06573318 0.27839671 0.35
208214 0.02128199]
Function value:  -18608.49617309586
Current parameter vector: [0.07192639 1.69499858 0.06573318 0.27839671 0.35
208214 0.02128199]
Function value:  -18608.49617309586
Current parameter vector: [0.07192639 1.69499858 0.06573318 0.27839671 0.35
208214 0.02128199]
Function value:  -18608.49617309586
Current parameter vector: [0.07021446 1.69202272 0.06526045 0.26816781 0.35
308311 0.0208137 ]
Function value:  -18608.572212699808
Current parameter vector: [0.07021446 1.69202272 0.06526045 0.26816781 0.35
308311 0.0208137 ]
Function value:  -18608.572212699808
Current parameter vector: [0.06989052 1.65770223 0.06543858 0.26243645 0.35
839839 0.01948642]
Function value:  -18608.599638505904
Current parameter vector: [0.06472443 1.6537023  0.06508875 0.24694094 0.36
```

```
81922  0.02106688]
Function value:  -18608.83067363608
Current parameter vector: [0.06472443 1.6537023  0.06508875 0.24694094 0.36
81922  0.02106688]
Function value:  -18608.83067363608
Current parameter vector: [0.06472443 1.6537023  0.06508875 0.24694094 0.36
81922  0.02106688]
Function value:  -18608.83067363608
Current parameter vector: [0.06472443 1.6537023  0.06508875 0.24694094 0.36
81922  0.02106688]
Function value:  -18608.83067363608
Current parameter vector: [0.06472443 1.6537023  0.06508875 0.24694094 0.36
81922  0.02106688]
Function value:  -18608.83067363608
Current parameter vector: [0.06157667 1.62050758 0.06482133 0.22937226 0.38
326566 0.01865047]
Function value:  -18608.83545885182
Current parameter vector: [0.06157667 1.62050758 0.06482133 0.22937226 0.38
326566 0.01865047]
Function value:  -18608.83545885182
Current parameter vector: [0.05655496 1.60936155 0.06477468 0.2154934  0.40
441994 0.02083958]
Function value:  -18609.164230657734
Current parameter vector: [0.05655496 1.60936155 0.06477468 0.2154934  0.40
441994 0.02083958]
Function value:  -18609.164230657734
Current parameter vector: [0.05655496 1.60936155 0.06477468 0.2154934  0.40
441994 0.02083958]
Function value:  -18609.164230657734
Current parameter vector: [0.05655496 1.60936155 0.06477468 0.2154934  0.40
441994 0.02083958]
Function value:  -18609.164230657734
Current parameter vector: [0.05406724 1.60132648 0.06448772 0.20666383 0.40
35325  0.02091033]
Function value:  -18609.16616366312
Current parameter vector: [0.05406724 1.60132648 0.06448772 0.20666383 0.40
35325  0.02091033]
Function value:  -18609.16616366312
Current parameter vector: [0.04977622 1.57244363 0.06430123 0.18940962 0.42
144585 0.02197114]
Function value:  -18609.16911898547
Current parameter vector: [0.05626329 1.56533358 0.06532619 0.21345232 0.41
54124  0.02174456]
Function value:  -18609.203612469883
Current parameter vector: [0.04492611 1.6024584  0.06350905 0.1795651  0.43
94318  0.02160711]
Function value:  -18609.34277877319
Current parameter vector: [0.04492611 1.6024584  0.06350905 0.1795651  0.43
94318  0.02160711]
Function value:  -18609.34277877319
============================================================================
=====
Optimal parameter set:
 final_simplex: (array([[0.04492611, 1.6024584 , 0.06350905, 0.1795651 , 0.
4394318 ,
        0.02160711],
       [0.04798195, 1.57002185, 0.06428683, 0.18426726, 0.43152123,
        0.02130408],
       [0.05626329, 1.56533358, 0.06532619, 0.21345232, 0.4154124 ,
        0.02174456],
       [0.04977622, 1.57244363, 0.06430123, 0.18940962, 0.42144585,
```

```
 0.02197114],
[0.05406724, 1.60132648, 0.06448772, 0.20666383, 0.4035325 ,
 0.02091033],
[0.05655496, 1.60936155, 0.06477468, 0.2154934 , 0.40441994,
 0.02083958],
[0.05036551, 1.55956357, 0.06423068, 0.18622377, 0.4232059 ,
 0.01932867]]), array([-18609.34277877, -18609.26307107, -18609.2036
1247, -18609.16911899,
       -18609.16616366, -18609.16423066, -18609.16193958]))
           fun: -18609.34277877319
       message: 'Maximum number of iterations has been exceeded.'
          nfev: 159
           nit: 100
        status: 2
       success: False
             x: array([0.04492611, 1.6024584 , 0.06350905, 0.1795651 , 0.43
94318 ,
       0.02160711])
============================================================================
=====
```

In [222…
```
print(result_EKF.x)
```

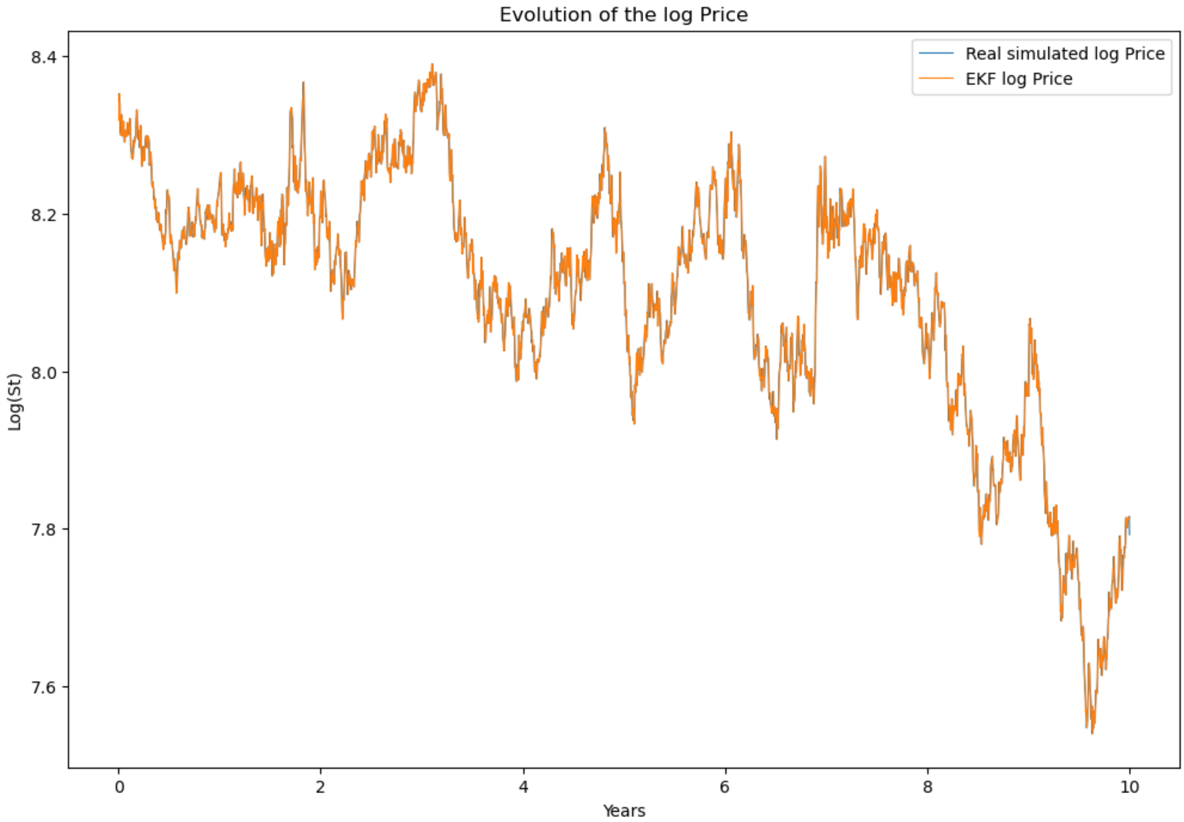
```
[0.04492611 1.6024584  0.06350905 0.1795651  0.4394318  0.02160711]
```

In [180…
```
print(params)
```

```
[0.0485, 1.5, 0.05, 0.18, 0.5, 0.04]
```

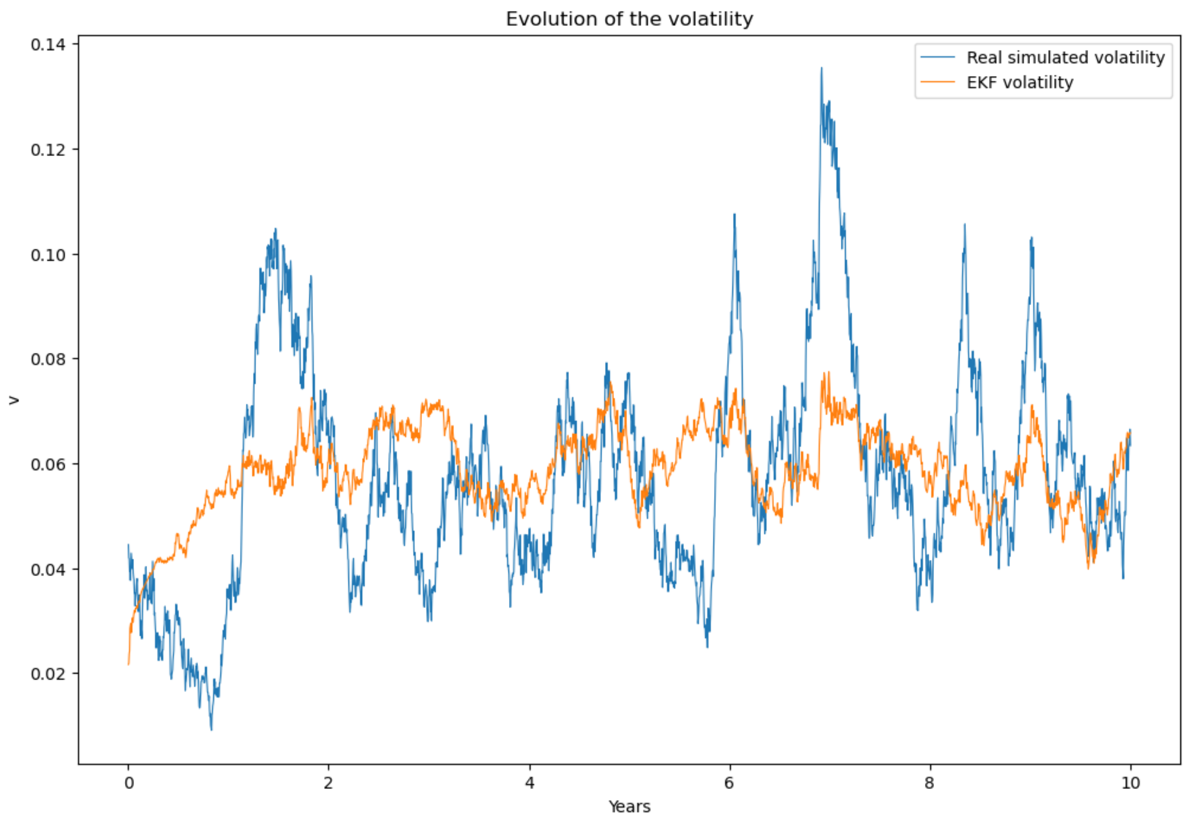
Plotting the results:

In [223…
```
plt.figure(figsize=(12,8))
years = np.arange(y.shape[-1]) * dt
plt.plot(years[1:], y[1:], label = 'Real simulated log Price', linewidth=0.8
plt.plot(years[1:], y_EKF, label = 'EKF log Price', linewidth=0.8)
plt.plot()
plt.title('Evolution of the log Price')
plt.ylabel('Log(St)')
plt.xlabel('Years')
plt.legend()
plt.show()
```



In [224…

```
plt.figure(figsize=(12,8))
plt.plot(years[1:], v_[1:], label = 'Real simulated volatility', linewidth=0
plt.plot(years[1:], v_EKF, label = 'EKF volatility', linewidth=0.8)
plt.plot()
plt.title('Evolution of the volatility')
plt.ylabel('v')
plt.xlabel('Years')
plt.legend()
plt.show()
```



2. Particle Filtering

2.1 Preparing the different functions

In [57]:
```
def proposal_distribution(N, v_prev, dy, params):
    mu, kappa, theta, lbda, rho = params
    # Calculate mean and standard deviation of the proposal distribution
    m = v_prev + kappa * (theta - v_prev) * dt + lbda * rho * (dy - (mu - 1/
    s = lbda * np.sqrt(v_prev * (1 - rho**2) * dt)
    # Sample N particles from the proposal distribution
    return norm.rvs(m, s, N)

def likelihood(y, x, v_prev, y_prev, params):
    mu, kappa, theta, lbda, rho = params
    # Calculate the mean and standard deviation of the measurement distribut
    m = y_prev + (mu - 1/2 * x) * dt
    s = np.sqrt(v_prev * dt)
    # Calculate the likelihood
    return norm.pdf(y, m, s)

def transition_proba(x, v_prev, params):
    mu, kappa, theta, lbda, rho = params
    # Calculate the mean and standard deviation of the transition distributi
    m = 1 / (1 + 1/2 * lbda * rho * dt) * (v_prev + kappa * (theta - v_prev)
    s = 1 / (1 + 1/2 * lbda * rho * dt) * lbda * np.sqrt(v_prev * dt)
    # Calculate the transition probability
    return norm.pdf(x, m, s)

def propo(x, v_prev, dy, params):
    mu, kappa, theta, lbda, rho = params
    # Calculate the mean and standard deviation of the proposal distribution
    m = v_prev + kappa*(theta-v_prev)*dt + lbda*rho*(dy - (mu-1/2*v_prev)*dt
    s = lbda*np.sqrt(v_prev*(1-rho**2)*dt)
    return norm.pdf(x, m, s)

def parameter_states_init(N, bounds):
    current_params = np.zeros((len(bounds), N))
    b0, b1, b2, b3, b4 = bounds
    # Initialize each parameter state using uniform random values within the
    current_params[0] = np.random.rand(N) * (b0[1] - b0[0]) + b0[0]
    current_params[1] = np.random.rand(N) * (b1[1] - b1[0]) + b1[0]
    current_params[2] = np.random.rand(N) * (b2[1] - b2[0]) + b2[0]
    current_params[3] = np.random.rand(N) * (b3[1] - b3[0]) + b3[0]
    current_params[4] = np.random.rand(N) * (b4[1] - b4[0]) + b4[0]

    return current_params

def resample_state(particles, w):
    N = len(particles)
    c_sum = np.cumsum(w)
    c_sum[-1] = 1.
    # Select new particles by randomly sampling from the cumulative sum
    idx = np.searchsorted(c_sum, np.random.rand(N))
    particles[:] = particles[idx]
    # Assign equal w to the new particles
    new_w = np.ones(len(w)) / len(w)

    return particles, new_w
```

```
def resample(v_pred, w, current_params):
        current_params[0], _ = resample_state(current_params[0], w)
        current_params[1], _ = resample_state(current_params[1], w)
        current_params[2], _ = resample_state(current_params[2], w)
        current_params[3], _ = resample_state(current_params[3], w)
        current_params[4], _ = resample_state(current_params[4], w)
        v_pred, w = resample_state(v_pred, w)
        return v_pred, w, current_params

def prediction_density(y, y_prev, x, mu):
    m = y_prev + (mu-1/2*x)*dt
    s = np.sqrt(x*dt)
    return norm.pdf(y, m, s)

def prediction_density_v(v, v_prev, dy, lbda,rho, theta, kappa):
    # Transition
    m = 1/(1+1/2*lbda*rho*dt) * (v_prev + kappa*(theta-v_prev)*dt + 1/2*lbda
    #print('m',m)
    s = (1/(1+1/2*lbda*rho*dt) * lbda * np.sqrt(v_prev*dt))
    return norm.pdf(v, m, s)

def inv_froeb(w):
        return 1. / np.sum(np.square(w))
```

In [58]:

```
def predict(v_pred, particles, y_prev, mu):
    # Generate predicted observations based on the predicted states
    y_hat = y_prev + (mu - 1 / 2 * v_pred) * dt + np.sqrt(particles * dt) *
    # Calculate the prediction density for each predicted observation
    prediction_densities = [prediction_density(y_hat[k], y_prev, v_pred, mu)
    # Calculate the average prediction density for each predicted observatic
    pdf_y_hat = np.array([np.mean(density) for density in prediction_densiti
    # Normalize the prediction densities
    pdf_y_hat = pdf_y_hat / np.sum(pdf_y_hat)
    # Calculate the weighted sum of the predicted observations
    return np.sum(pdf_y_hat * y_hat)

def predict_v(v_pred, particles, v_prev, mu, lbda,rho, theta, kappa, w, para
    v_hat = v_prev + (theta-1/2*particles)*dt + lbda*rho*(((mu-1/2*particles
    pdf_v_hat = np.array([np.mean(prediction_density_v(v_hat[k], particles[k
    pdf_v_hat = pdf_v_hat/sum(pdf_v_hat)
    return np.sum(pdf_v_hat * v_hat)
```

In [166…

```
def particle_filtering(params, N):
    global y_PF, v_PF, v_PF_bis

    # Unpack the model parameters
    mu, kappa, theta, lbda, rho, v0 = params

    print(params[:-1])

    # Initialize the current parameter states
    current_params = parameter_states_init(N, params[:-1])

    # Initialize the arrays to store the estimated states
    y_PF = np.zeros(N)
    v_PF = np.zeros(N)
    v_PF_bis = np.zeros(N)

    y_PF[0] = y[0]
    v_PF[0] = v0
```

```
    v_PF_bis[0] = v0

    # Initialize the weights
    w = np.array([1 / N] * N)

    # Initialize the particles
    particles = norm.rvs(v0, 0.02, N)
    particles = np.maximum(1e-4, particles)

    # Initialize the array to store the parameter steps
    params_steps = np.zeros((len(params) - 1, len(y)))
    params_steps.transpose()[0] = np.mean(current_params, axis=1)
    print(N)

    for i in range(1, N):
        dy = y[i] - y[i - 1]

        # Particle prediction step
        v_pred = proposal_distribution(N, particles, dy, current_params)
        v_pred = np.maximum(1e-3, v_pred)

        # Likelihood calculation
        Li = likelihood(y[i], v_pred, particles, y[i - 1], current_params)
        I = propo(v_pred, particles, dy, current_params)
        T = transition_proba(v_pred, particles, current_params)

        # Update the weights
        w = w * (Li * T / I)
        w = w / np.sum(w)

        # Resampling step
        if inv_froeb(w) < 0.75 * N:
            print('Resampling')
            v_pred, w, current_params = resample(v_pred, w, current_params)

        # State estimation step
        y_hat = predict(v_pred, particles, y[i - 1], np.mean(current_params[
        y_PF[i] = y_hat
        v_PF_bis[i] = predict_v(v_pred, particles, v_PF[i - 1], np.mean(curr
                                np.mean(current_params[3]), np.mean(current_
                                np.mean(current_params[2]), np.mean(current_
        v_PF[i] = np.sum(v_pred * w)
        particles = v_pred
        params_steps.transpose()[i] = np.sum(np.multiply(current_params, w[r

        print("Iteration {} completed".format(i))
    last_param = np.sum(np.multiply(current_params, w[np.newaxis, :]), axis=
    return (v_PF, v_PF_bis, params_steps, y_PF, last_param)
```

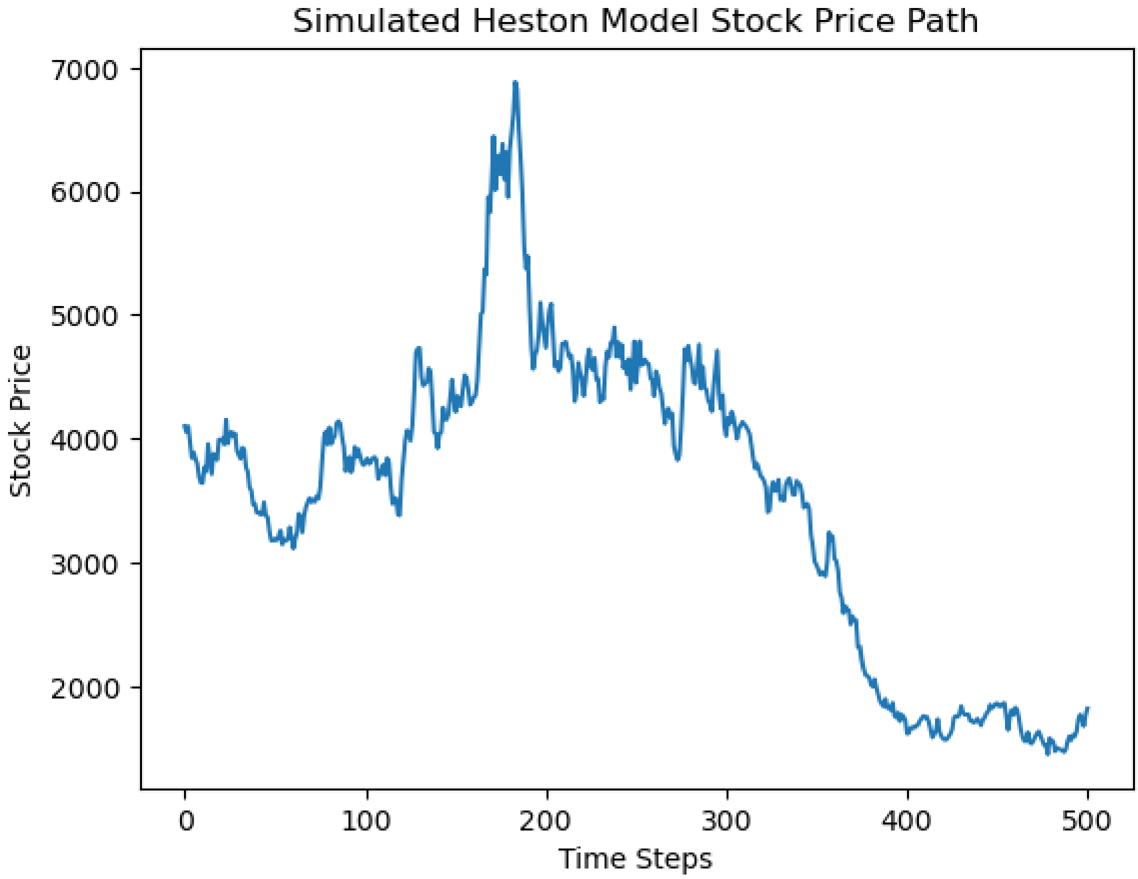
2.2 Running the algorithm

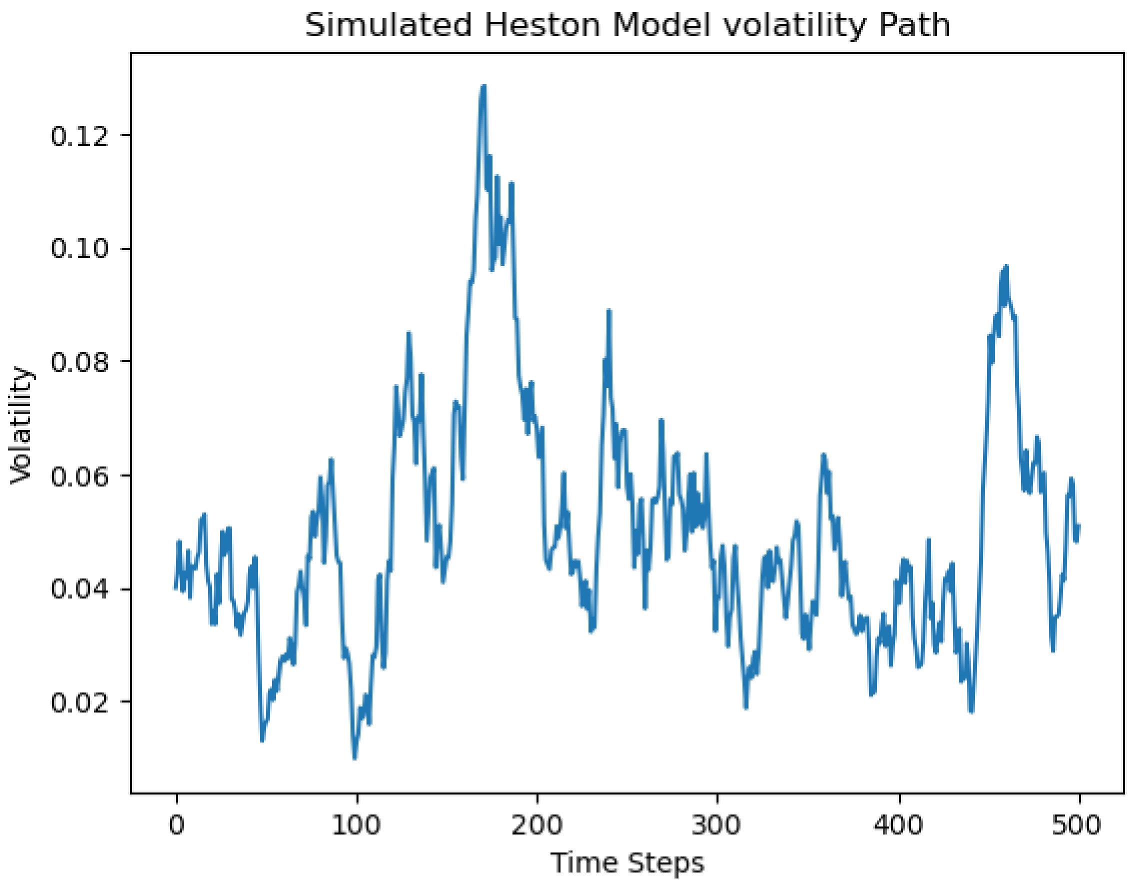
In [232…

```
S0 = 4100
r = 0.0485
mu = r
params = [mu, 1.5, 0.05, 0.18, 0.5, 0.04]
#.          mu, kappa, theta, lbda, rho , v0
N = 500
T = 10

params_sim = np.zeros(5)
```

```
params_sim[0] = params[1]# kappa
params_sim[1] = params[2]#theta
params_sim[2] = params[3]#sigma
params_sim[3] = params[4]#rho
params_sim[4] = params[5]#v0
#print(params_sim)
prices, v_, y = heston_simulated_prices(params_sim, N, T, S0, r, mu, plot =

dt = T/N
#print(dt)
```

```
T: 10   N: 500   dt: 0.02
```





In [233…
```
print(params)
```

```
[0.0485, 1.5, 0.05, 0.18, 0.5, 0.04]
```

In [234…
```
mu = (0.01, 0.05)
kappa = (0.5, 3)
theta = (0.02, 0.2)
lambda_ = (0.01, 0.91)
rho = (-0.5, 1)
v_0 = params[-1]

params_0 = np.array([[mu, kappa, theta, lambda_, rho, v_0]])
params_0 =[mu, kappa, theta, lambda_, rho, v_0]
```

In [242…
```
v_PF_opt, v_bis, param_steps, y_PF_opt, last_param = particle_filtering(para
```

```
[(0.01, 0.05), (0.5, 3), (0.02, 0.2), (0.01, 0.91), (-0.5, 1)]
500
Resampling
Iteration 1 completed
Resampling
Iteration 2 completed
Resampling
Iteration 3 completed
Iteration 4 completed
Resampling
Iteration 5 completed
Resampling
Iteration 6 completed
Resampling
Iteration 7 completed
Resampling
Iteration 8 completed
Iteration 9 completed
Resampling
Iteration 10 completed
Resampling
Iteration 11 completed
Resampling
Iteration 12 completed
Resampling
Iteration 13 completed
Resampling
Iteration 14 completed
Resampling
Iteration 15 completed
Resampling
Iteration 16 completed
Resampling
Iteration 17 completed
Resampling
Iteration 18 completed
Resampling
Iteration 19 completed
Resampling
Iteration 20 completed
Resampling
Iteration 21 completed
Resampling
Iteration 22 completed
Resampling
Iteration 23 completed
Resampling
Iteration 24 completed
Resampling
Iteration 25 completed
Resampling
Iteration 26 completed
Resampling
Iteration 27 completed
Resampling
Iteration 28 completed
Resampling
Iteration 29 completed
Resampling
Iteration 30 completed
Resampling
```

```
Iteration 31 completed
Resampling
Iteration 32 completed
Resampling
Iteration 33 completed
Resampling
Iteration 34 completed
Resampling
Iteration 35 completed
Resampling
Iteration 36 completed
Resampling
Iteration 37 completed
Resampling
Iteration 38 completed
Resampling
Iteration 39 completed
Resampling
Iteration 40 completed
Resampling
Iteration 41 completed
Resampling
Iteration 42 completed
Resampling
Iteration 43 completed
Resampling
Iteration 44 completed
Resampling
Iteration 45 completed
Resampling
Iteration 46 completed
Resampling
Iteration 47 completed
Resampling
Iteration 48 completed
Resampling
Iteration 49 completed
Resampling
Iteration 50 completed
Resampling
Iteration 51 completed
Resampling
Iteration 52 completed
Resampling
Iteration 53 completed
Resampling
Iteration 54 completed
Resampling
Iteration 55 completed
Resampling
Iteration 56 completed
Resampling
Iteration 57 completed
Resampling
Iteration 58 completed
Resampling
Iteration 59 completed
Resampling
Iteration 60 completed
Resampling
Iteration 61 completed
```

```
Resampling
Iteration 62 completed
Resampling
Iteration 63 completed
Resampling
Iteration 64 completed
Resampling
Iteration 65 completed
Resampling
Iteration 66 completed
Resampling
Iteration 67 completed
Resampling
Iteration 68 completed
Resampling
Iteration 69 completed
Resampling
Iteration 70 completed
Resampling
Iteration 71 completed
Resampling
Iteration 72 completed
Resampling
Iteration 73 completed
Resampling
Iteration 74 completed
Resampling
Iteration 75 completed
Resampling
Iteration 76 completed
Resampling
Iteration 77 completed
Resampling
Iteration 78 completed
Resampling
Iteration 79 completed
Resampling
Iteration 80 completed
Resampling
Iteration 81 completed
Resampling
Iteration 82 completed
Resampling
Iteration 83 completed
Resampling
Iteration 84 completed
Resampling
Iteration 85 completed
Resampling
Iteration 86 completed
Resampling
Iteration 87 completed
Resampling
Iteration 88 completed
Resampling
Iteration 89 completed
Resampling
Iteration 90 completed
Resampling
Iteration 91 completed
Resampling
```

```
Iteration 92 completed
Resampling
Iteration 93 completed
Resampling
Iteration 94 completed
Resampling
Iteration 95 completed
Resampling
Iteration 96 completed
Resampling
Iteration 97 completed
Resampling
Iteration 98 completed
Resampling
Iteration 99 completed
Resampling
Iteration 100 completed
Resampling
Iteration 101 completed
Resampling
Iteration 102 completed
Resampling
Iteration 103 completed
Resampling
Iteration 104 completed
Resampling
Iteration 105 completed
Resampling
Iteration 106 completed
Resampling
Iteration 107 completed
Resampling
Iteration 108 completed
Resampling
Iteration 109 completed
Resampling
Iteration 110 completed
Resampling
Iteration 111 completed
Resampling
Iteration 112 completed
Resampling
Iteration 113 completed
Resampling
Iteration 114 completed
Resampling
Iteration 115 completed
Resampling
Iteration 116 completed
Resampling
Iteration 117 completed
Resampling
Iteration 118 completed
Resampling
Iteration 119 completed
Resampling
Iteration 120 completed
Resampling
Iteration 121 completed
Resampling
Iteration 122 completed
```

```
Resampling
Iteration 123 completed
Resampling
Iteration 124 completed
Resampling
Iteration 125 completed
Resampling
Iteration 126 completed
Resampling
Iteration 127 completed
Resampling
Iteration 128 completed
Resampling
Iteration 129 completed
Resampling
Iteration 130 completed
Resampling
Iteration 131 completed
Resampling
Iteration 132 completed
Resampling
Iteration 133 completed
Resampling
Iteration 134 completed
Resampling
Iteration 135 completed
Resampling
Iteration 136 completed
Resampling
Iteration 137 completed
Resampling
Iteration 138 completed
Resampling
Iteration 139 completed
Resampling
Iteration 140 completed
Resampling
Iteration 141 completed
Resampling
Iteration 142 completed
Resampling
Iteration 143 completed
Resampling
Iteration 144 completed
Resampling
Iteration 145 completed
Resampling
Iteration 146 completed
Resampling
Iteration 147 completed
Resampling
Iteration 148 completed
Resampling
Iteration 149 completed
Resampling
Iteration 150 completed
Resampling
Iteration 151 completed
Resampling
Iteration 152 completed
Resampling
```

```
Iteration 153 completed
Resampling
Iteration 154 completed
Resampling
Iteration 155 completed
Resampling
Iteration 156 completed
Resampling
Iteration 157 completed
Resampling
Iteration 158 completed
Resampling
Iteration 159 completed
Resampling
Iteration 160 completed
Resampling
Iteration 161 completed
Resampling
Iteration 162 completed
Resampling
Iteration 163 completed
Resampling
Iteration 164 completed
Resampling
Iteration 165 completed
Resampling
Iteration 166 completed
Resampling
Iteration 167 completed
Resampling
Iteration 168 completed
Resampling
Iteration 169 completed
Resampling
Iteration 170 completed
Resampling
Iteration 171 completed
Resampling
Iteration 172 completed
Resampling
Iteration 173 completed
Resampling
Iteration 174 completed
Resampling
Iteration 175 completed
Resampling
Iteration 176 completed
Resampling
Iteration 177 completed
Resampling
Iteration 178 completed
Resampling
Iteration 179 completed
Resampling
Iteration 180 completed
Resampling
Iteration 181 completed
Resampling
Iteration 182 completed
Resampling
Iteration 183 completed
```

```
Resampling
Iteration 184 completed
Resampling
Iteration 185 completed
Resampling
Iteration 186 completed
Resampling
Iteration 187 completed
Resampling
Iteration 188 completed
Resampling
Iteration 189 completed
Resampling
Iteration 190 completed
Resampling
Iteration 191 completed
Resampling
Iteration 192 completed
Resampling
Iteration 193 completed
Resampling
Iteration 194 completed
Resampling
Iteration 195 completed
Resampling
Iteration 196 completed
Resampling
Iteration 197 completed
Resampling
Iteration 198 completed
Resampling
Iteration 199 completed
Resampling
Iteration 200 completed
Resampling
Iteration 201 completed
Resampling
Iteration 202 completed
Resampling
Iteration 203 completed
Resampling
Iteration 204 completed
Resampling
Iteration 205 completed
Resampling
Iteration 206 completed
Resampling
Iteration 207 completed
Resampling
Iteration 208 completed
Resampling
Iteration 209 completed
Resampling
Iteration 210 completed
Resampling
Iteration 211 completed
Resampling
Iteration 212 completed
Resampling
Iteration 213 completed
Resampling
```

```
Iteration 214 completed
Resampling
Iteration 215 completed
Resampling
Iteration 216 completed
Resampling
Iteration 217 completed
Resampling
Iteration 218 completed
Resampling
Iteration 219 completed
Resampling
Iteration 220 completed
Resampling
Iteration 221 completed
Resampling
Iteration 222 completed
Resampling
Iteration 223 completed
Resampling
Iteration 224 completed
Resampling
Iteration 225 completed
Resampling
Iteration 226 completed
Resampling
Iteration 227 completed
Resampling
Iteration 228 completed
Resampling
Iteration 229 completed
Resampling
Iteration 230 completed
Resampling
Iteration 231 completed
Resampling
Iteration 232 completed
Resampling
Iteration 233 completed
Resampling
Iteration 234 completed
Resampling
Iteration 235 completed
Resampling
Iteration 236 completed
Resampling
Iteration 237 completed
Resampling
Iteration 238 completed
Resampling
Iteration 239 completed
Resampling
Iteration 240 completed
Resampling
Iteration 241 completed
Resampling
Iteration 242 completed
Resampling
Iteration 243 completed
Resampling
Iteration 244 completed
```

```
Resampling
Iteration 245 completed
Resampling
Iteration 246 completed
Resampling
Iteration 247 completed
Resampling
Iteration 248 completed
Resampling
Iteration 249 completed
Resampling
Iteration 250 completed
Resampling
Iteration 251 completed
Resampling
Iteration 252 completed
Resampling
Iteration 253 completed
Resampling
Iteration 254 completed
Resampling
Iteration 255 completed
Resampling
Iteration 256 completed
Resampling
Iteration 257 completed
Resampling
Iteration 258 completed
Resampling
Iteration 259 completed
Resampling
Iteration 260 completed
Resampling
Iteration 261 completed
Resampling
Iteration 262 completed
Resampling
Iteration 263 completed
Resampling
Iteration 264 completed
Resampling
Iteration 265 completed
Resampling
Iteration 266 completed
Resampling
Iteration 267 completed
Resampling
Iteration 268 completed
Resampling
Iteration 269 completed
Resampling
Iteration 270 completed
Resampling
Iteration 271 completed
Resampling
Iteration 272 completed
Resampling
Iteration 273 completed
Resampling
Iteration 274 completed
Resampling
```

```
Iteration 275 completed
Resampling
Iteration 276 completed
Resampling
Iteration 277 completed
Resampling
Iteration 278 completed
Resampling
Iteration 279 completed
Resampling
Iteration 280 completed
Resampling
Iteration 281 completed
Resampling
Iteration 282 completed
Resampling
Iteration 283 completed
Resampling
Iteration 284 completed
Resampling
Iteration 285 completed
Resampling
Iteration 286 completed
Resampling
Iteration 287 completed
Resampling
Iteration 288 completed
Resampling
Iteration 289 completed
Resampling
Iteration 290 completed
Resampling
Iteration 291 completed
Resampling
Iteration 292 completed
Resampling
Iteration 293 completed
Resampling
Iteration 294 completed
Resampling
Iteration 295 completed
Resampling
Iteration 296 completed
Resampling
Iteration 297 completed
Resampling
Iteration 298 completed
Resampling
Iteration 299 completed
Resampling
Iteration 300 completed
Resampling
Iteration 301 completed
Resampling
Iteration 302 completed
Resampling
Iteration 303 completed
Resampling
Iteration 304 completed
Resampling
Iteration 305 completed
```

```
Resampling
Iteration 306 completed
Resampling
Iteration 307 completed
Resampling
Iteration 308 completed
Resampling
Iteration 309 completed
Resampling
Iteration 310 completed
Resampling
Iteration 311 completed
Resampling
Iteration 312 completed
Resampling
Iteration 313 completed
Resampling
Iteration 314 completed
Resampling
Iteration 315 completed
Resampling
Iteration 316 completed
Resampling
Iteration 317 completed
Resampling
Iteration 318 completed
Resampling
Iteration 319 completed
Resampling
Iteration 320 completed
Resampling
Iteration 321 completed
Resampling
Iteration 322 completed
Resampling
Iteration 323 completed
Resampling
Iteration 324 completed
Resampling
Iteration 325 completed
Resampling
Iteration 326 completed
Resampling
Iteration 327 completed
Resampling
Iteration 328 completed
Resampling
Iteration 329 completed
Resampling
Iteration 330 completed
Resampling
Iteration 331 completed
Resampling
Iteration 332 completed
Resampling
Iteration 333 completed
Resampling
Iteration 334 completed
Resampling
Iteration 335 completed
Resampling
```

```
Iteration 336 completed
Resampling
Iteration 337 completed
Resampling
Iteration 338 completed
Resampling
Iteration 339 completed
Resampling
Iteration 340 completed
Resampling
Iteration 341 completed
Resampling
Iteration 342 completed
Resampling
Iteration 343 completed
Resampling
Iteration 344 completed
Resampling
Iteration 345 completed
Resampling
Iteration 346 completed
Resampling
Iteration 347 completed
Resampling
Iteration 348 completed
Resampling
Iteration 349 completed
Resampling
Iteration 350 completed
Resampling
Iteration 351 completed
Resampling
Iteration 352 completed
Resampling
Iteration 353 completed
Resampling
Iteration 354 completed
Resampling
Iteration 355 completed
Resampling
Iteration 356 completed
Resampling
Iteration 357 completed
Resampling
Iteration 358 completed
Resampling
Iteration 359 completed
Resampling
Iteration 360 completed
Resampling
Iteration 361 completed
Resampling
Iteration 362 completed
Resampling
Iteration 363 completed
Resampling
Iteration 364 completed
Resampling
Iteration 365 completed
Resampling
Iteration 366 completed
```

```
Resampling
Iteration 367 completed
Resampling
Iteration 368 completed
Resampling
Iteration 369 completed
Resampling
Iteration 370 completed
Resampling
Iteration 371 completed
Resampling
Iteration 372 completed
Resampling
Iteration 373 completed
Resampling
Iteration 374 completed
Resampling
Iteration 375 completed
Resampling
Iteration 376 completed
Resampling
Iteration 377 completed
Resampling
Iteration 378 completed
Resampling
Iteration 379 completed
Resampling
Iteration 380 completed
Resampling
Iteration 381 completed
Resampling
Iteration 382 completed
Resampling
Iteration 383 completed
Resampling
Iteration 384 completed
Iteration 385 completed
Iteration 386 completed
Resampling
Iteration 387 completed
Resampling
Iteration 388 completed
Resampling
Iteration 389 completed
Resampling
Iteration 390 completed
Resampling
Iteration 391 completed
Resampling
Iteration 392 completed
Resampling
Iteration 393 completed
Resampling
Iteration 394 completed
Resampling
Iteration 395 completed
Resampling
Iteration 396 completed
Resampling
Iteration 397 completed
Resampling
```

```
Iteration 398 completed
Resampling
Iteration 399 completed
Resampling
Iteration 400 completed
Resampling
Iteration 401 completed
Resampling
Iteration 402 completed
Resampling
Iteration 403 completed
Resampling
Iteration 404 completed
Resampling
Iteration 405 completed
Resampling
Iteration 406 completed
Resampling
Iteration 407 completed
Resampling
Iteration 408 completed
Resampling
Iteration 409 completed
Resampling
Iteration 410 completed
Resampling
Iteration 411 completed
Resampling
Iteration 412 completed
Resampling
Iteration 413 completed
Resampling
Iteration 414 completed
Resampling
Iteration 415 completed
Resampling
Iteration 416 completed
Resampling
Iteration 417 completed
Resampling
Iteration 418 completed
Resampling
Iteration 419 completed
Resampling
Iteration 420 completed
Resampling
Iteration 421 completed
Resampling
Iteration 422 completed
Resampling
Iteration 423 completed
Resampling
Iteration 424 completed
Resampling
Iteration 425 completed
Resampling
Iteration 426 completed
Resampling
Iteration 427 completed
Resampling
Iteration 428 completed
```

```
Resampling
Iteration 429 completed
Resampling
Iteration 430 completed
Resampling
Iteration 431 completed
Resampling
Iteration 432 completed
Resampling
Iteration 433 completed
Resampling
Iteration 434 completed
Resampling
Iteration 435 completed
Resampling
Iteration 436 completed
Resampling
Iteration 437 completed
Resampling
Iteration 438 completed
Resampling
Iteration 439 completed
Resampling
Iteration 440 completed
Resampling
Iteration 441 completed
Resampling
Iteration 442 completed
Resampling
Iteration 443 completed
Resampling
Iteration 444 completed
Resampling
Iteration 445 completed
Resampling
Iteration 446 completed
Resampling
Iteration 447 completed
Resampling
Iteration 448 completed
Resampling
Iteration 449 completed
Resampling
Iteration 450 completed
Resampling
Iteration 451 completed
Resampling
Iteration 452 completed
Resampling
Iteration 453 completed
Resampling
Iteration 454 completed
Resampling
Iteration 455 completed
Resampling
Iteration 456 completed
Resampling
Iteration 457 completed
Resampling
Iteration 458 completed
Resampling
```

```
Iteration 459 completed
Resampling
Iteration 460 completed
Resampling
Iteration 461 completed
Resampling
Iteration 462 completed
Resampling
Iteration 463 completed
Resampling
Iteration 464 completed
Resampling
Iteration 465 completed
Resampling
Iteration 466 completed
Resampling
Iteration 467 completed
Resampling
Iteration 468 completed
Resampling
Iteration 469 completed
Resampling
Iteration 470 completed
Resampling
Iteration 471 completed
Resampling
Iteration 472 completed
Resampling
Iteration 473 completed
Resampling
Iteration 474 completed
Resampling
Iteration 475 completed
Resampling
Iteration 476 completed
Resampling
Iteration 477 completed
Resampling
Iteration 478 completed
Resampling
Iteration 479 completed
Resampling
Iteration 480 completed
Resampling
Iteration 481 completed
Resampling
Iteration 482 completed
Resampling
Iteration 483 completed
Resampling
Iteration 484 completed
Resampling
Iteration 485 completed
Resampling
Iteration 486 completed
Resampling
Iteration 487 completed
Resampling
Iteration 488 completed
Resampling
Iteration 489 completed
```

```
Resampling
Iteration 490 completed
Resampling
Iteration 491 completed
Resampling
Iteration 492 completed
Resampling
Iteration 493 completed
Resampling
Iteration 494 completed
Resampling
Iteration 495 completed
Resampling
Iteration 496 completed
Resampling
Iteration 497 completed
Resampling
Iteration 498 completed
Resampling
Iteration 499 completed
```

In [243…

```
last_param
```

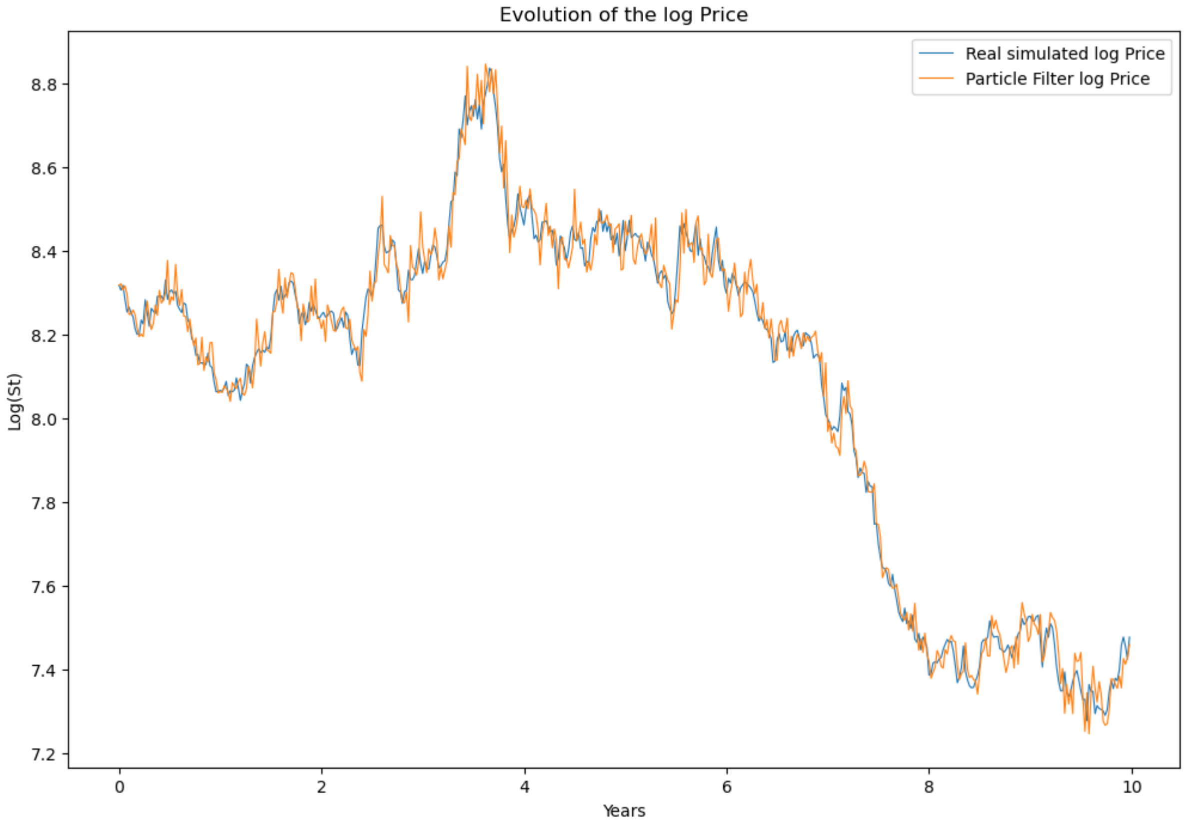
Out[243]:

```
array([0.02730858, 1.62033214, 0.06120653, 0.25795131, 0.93447779])
```

2.3 Plotting the results:

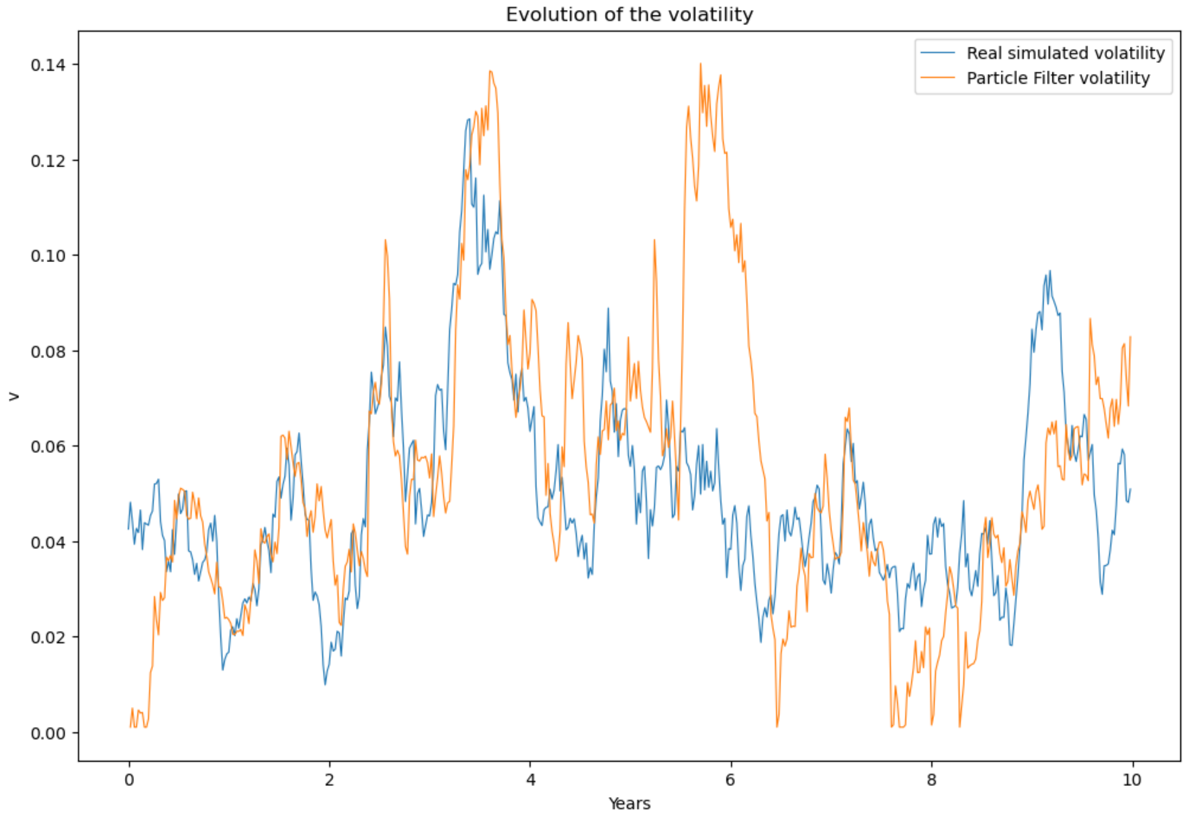
In [244…

```
plt.figure(figsize=(12,8))
years = np.arange(obs.shape[-1]) * (T/N)
plt.plot(years, y[:-1], label = 'Real simulated log Price', linewidth=0.8)
plt.plot(years, y_PF_opt, label = 'Particle Filter log Price', linewidth=0.8
plt.plot()
plt.title('Evolution of the log Price')
plt.ylabel('Log(St)')
plt.xlabel('Years')
plt.legend()
plt.show()
```



In [245…

```
plt.figure(figsize=(12,8))
years = np.arange(v.shape[-1]) * (T/N)
plt.plot(years, v_[1:], label = 'Real simulated volatility', linewidth=0.8)
plt.plot(years[1:], v_PF_opt[1:], label = 'Particle Filter volatility', line
#plt.plot(years[1:], v_PF_bis[1:], label = 'Particle Filter volatility', lin
plt.plot()
plt.title('Evolution of the volatility')
plt.ylabel('v')
plt.xlabel('Years')
plt.legend()
plt.show()
```



Result

In [246…

```
p_PF = np.append(last_param,0.04)
p_EKF = result_EKF.x
```

In [247…

```
params_result = pd.DataFrame({})
params_result['From'] = ['Simulation (true values)','Extended Kalman Filter'
params_name = ['mu', 'kappa', 'theta', 'lambda_', 'rho', 'v_0']

for i in range(len(params)):
    params_result[params_name[i]] = [params[i],p_EKF[i], p_PF[i]]
params_result
```

Out[247]:

	From	mu	kappa	theta	lambda_	rho	v_0
0	Simulation (true values)	0.048500	1.500000	0.050000	0.180000	0.500000	0.040000
1	Extended Kalman Filter	0.044926	1.602458	0.063509	0.179565	0.439432	0.021607
2	Particle filter	0.027309	1.620332	0.061207	0.257951	0.934478	0.040000

We can see that both model get close to real parameters but struggle a bit. For instance the mu estimate of the particle filtering is quite far from the real one. It is worth noting that the two models worked on different data as the two algorithms running on same data would have lead to long computation times. We did not get the parameters back very precisely.

Looking at the graphs, we observe that the two models are able to stick to the price path but are quite noisy around the true volatility paths simulated.

Therefore, in the context of parameter estimation in the Heston stochastic volatility model, both EKF and PF can be used. However, the choice could depend on the specific characteristics of the model, the available data, and the desired accuracy of the estimation. Looking back at the way the models should perform, if the model is relatively linear and Gaussian, and the noise is well-behaved, the EKF may provide accurate estimates. On the other hand, if the model is nonlinear or the noise is non-Gaussian, the PF may be more appropriate to capture the complexity of the distribution and provide robust estimates.