Carr-Madan method for pricing and Heston model calibration

BACKGROUND

THE HESTON MODEL

CARR-MADAN PRICING

CALIBRATION

BACKGROUND

THE HESTON MODEL

CARR-MADAN PRICING

CALIBRATION

The Fourier Transform:

$$\mathcal{F}[f(v)] = \int_{-\infty}^{\infty} e^{ixv} f(x)dx$$

And its Inverse:

$$\mathcal{F}^{-1}[f(v)] = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-ixv} f(x)dx$$

If $f \in \mathcal{L}^2(R), \mathcal{F}^{-1}\mathcal{F}f = f$

The Discrete Fourier Transform (DFT) and the FFT:

$$y_k = \sum_{m=0}^{N-1} x_m e^{-2\pi i \frac{mk}{N}} \qquad \forall k = 0, \dots, N-1$$

Evaluating this definition directly requires $O(n^2)$ operations.

An FFT is any method to compute the same results in $O(n\ logn)$ operations.

Characteristic function:

$$S_T = S_0 \exp(rt + X_T) \quad \Longrightarrow \quad X_T = ln\frac{S_T}{S_0} - rT$$

$$\phi_{X_T}(v) = E\left[e^{ivX_T}\right] = \int_R e^{ivX_T} f_{X_T}(x)dx = \mathcal{F}\left[f_{X_T}\right]$$

In the following we will use $\phi_T(v)$ as short for $\phi_{X_T}(v)$
Moreover, in the risk neutral measure, $\phi_T(-i) = 1$ must hold

$$\begin{aligned}\phi(-i) &= E\left[e^{i(-i)X_T}\right] \\ &= E\left[e^{X_T}\right] \\ &= E\left[\frac{S_T}{S_0 e^{rT}}\right] \\ &= \frac{1}{S_0 e^{rT}} S_0 e^{rT} \\ &= 1\end{aligned}$$

BACKGROUND

THE HESTON MODEL

CARR-MADAN PRICING

CALIBRATION

Formulation of the Model:

$$dS_t = \mu S_t dt + \sqrt{v_t} S_t dW_t^1$$
$$dv_t = \kappa(\theta - v_t)dt + \sigma\sqrt{v_t} dW_t^2$$
$$\rho dt = dW_t^1 dW_t^2$$

Its Characteristic Function:

$$\phi_{X_T}(\epsilon) = e^{B(\epsilon)+C(\epsilon)}$$

$$B(\epsilon) = \frac{2\zeta(\epsilon)\left(1 - e^{-\psi(\epsilon)T}\right)V_0}{2\psi(\epsilon) - \left(\psi(\epsilon) - \gamma(\epsilon)\right)\left(1 - e^{-\psi(\epsilon)T}\right)}$$

$$C(\epsilon) = -\frac{2\kappa\theta}{\sigma^2}\left[2\log\left(\frac{2\psi(\epsilon) - \left(\psi(\epsilon) - \gamma(\epsilon)\right)\left(1 - e^{-\psi(\epsilon)T}\right)}{2\psi(\epsilon)}\right) + \left(\psi(\epsilon) - \gamma(\epsilon)\right)T\right]$$

$$\zeta(\epsilon) = -\frac{1}{2}(\epsilon^2 + i\epsilon) \qquad \psi(\epsilon) = \sqrt{\gamma(\epsilon)^2 - 2\sigma^2\zeta(\epsilon)} \qquad \gamma(\epsilon) = \kappa - \rho\sigma\epsilon i$$

BACKGROUND

THE HESTON MODEL

CARR-MADAN PRICING

CALIBRATION

Option valuation using the fast Fourier transform

Peter Carr and Dilip B. Madan

In this paper the authors show how the fast Fourier transform may be used to value options when the characteristic function of the return is known analytically.

1. INTRODUCTION

The Black-Scholes model and its extensions comprise one of the major developments in modern finance. Much of the recent literature on option valuation has successfully applied Fourier analysis to determine option prices (see e.g. Bakshi

Call Option Price: $$C(k) = e^{-rT} E\left[\left(e^{rT+X_T} - e^{k}\right)^{+}\right]$$

Because of Integrability needs we will work with:

$$z_T(k) = e^{-rT} E\left[\left(e^{rT+X_T} - e^{k}\right)^{+}\right] - \left(1 - e^{k-rT}\right)^{+}$$

We take the Fourier Transform of it:

$$\zeta_T(v) = \mathcal{F}[z_T(v)] = \int_{-\infty}^{\infty} e^{ivk}\, z_T(k)dk$$

$$\zeta_T(v) = e^{ivrT}\,\frac{\Phi_T(v-i)-1}{iv(1+iv)}$$

$$z_T(k) = e^{-rT} E\left[\left(e^{rT+X_T} - e^k\right)^+\right] - \left(1 - e^{k-rT}\right)^+$$
$$= e^{-rT}\left(E\left[\left(e^{rT+X_T} - e^k\right)^+\right] - \left(e^{rT} - e^k\right)^+\right)$$
$$= e^{-rT}\left(E\left[\left(e^{rT+X_T} - e^k\right)^+\right] - I_{k\le rT}\left(e^{rT} - e^k\right)\right)$$
$$= e^{-rT}\left(E\left[\left(e^{rT+X_T} - e^k\right)^+\right] - I_{k\le rT} E\left[\left(e^{rT+X_T} - e^k\right)^+\right]\right)$$
$$= e^{-rT}\int_{-\infty}^{+\infty}\left(e^{rT+x} - e^k\right)\left(I_{k\le rT+x} - I_{k\le rT}\right)\rho_T(x)dx$$

Rewriting $z_T(k)$ as:

$$z_T(k) = e^{-rT} E\left[\left(e^{rT+X_T} - e^k\right)^+\right] - \left(1 - e^{k-rT}\right)^+ = e^{-rT} \int_{-\infty}^{\infty} \left(e^{rT+x} - e^k\right)\left(I_{k\leq x+rT} - I_{k\leq rT}\right)\rho_T(x)$$

Therefore:

$$\zeta_T(v) = e^{-rT} \int_{-\infty}^{\infty} dk \int_{-\infty}^{\infty} e^{ivk}\left(e^{rT+x} - e^k\right)\left(I_{k\leq x+rT} - I_{k\leq rT}\right)\rho_T(x)dx$$

$$= e^{-rT} \int_{-\infty}^{\infty} \rho_T(x)dx \int_{x+rT}^{rT} e^{ivk}\left(e^k - e^{rT+x}\right)dk$$

$$= \int_{-\infty}^{+\infty} \left[\frac{e^{ivrT}(1-e^x)}{iv+1} - \frac{e^{x+ivrT}}{iv(iv+1)} + \frac{e^{(iv+1)x+ivrT}}{iv(iv+1)}\right]\rho_T(x)dx = e^{ivrT}\frac{\Phi_T(v-i)-1}{iv(1+iv)}$$

We can find the “modified” price by taking the Inverse Transform:

$$z_T(k) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-ivk}\,\zeta_T(v)dv$$

By Bounding the Integral we get:

$$z_T(k) \approx \frac{1}{\pi}\int_{0}^{A(N-1)/N} e^{-ivk}\,\zeta_T(v)dv$$

We can Numerically Integrate it with:

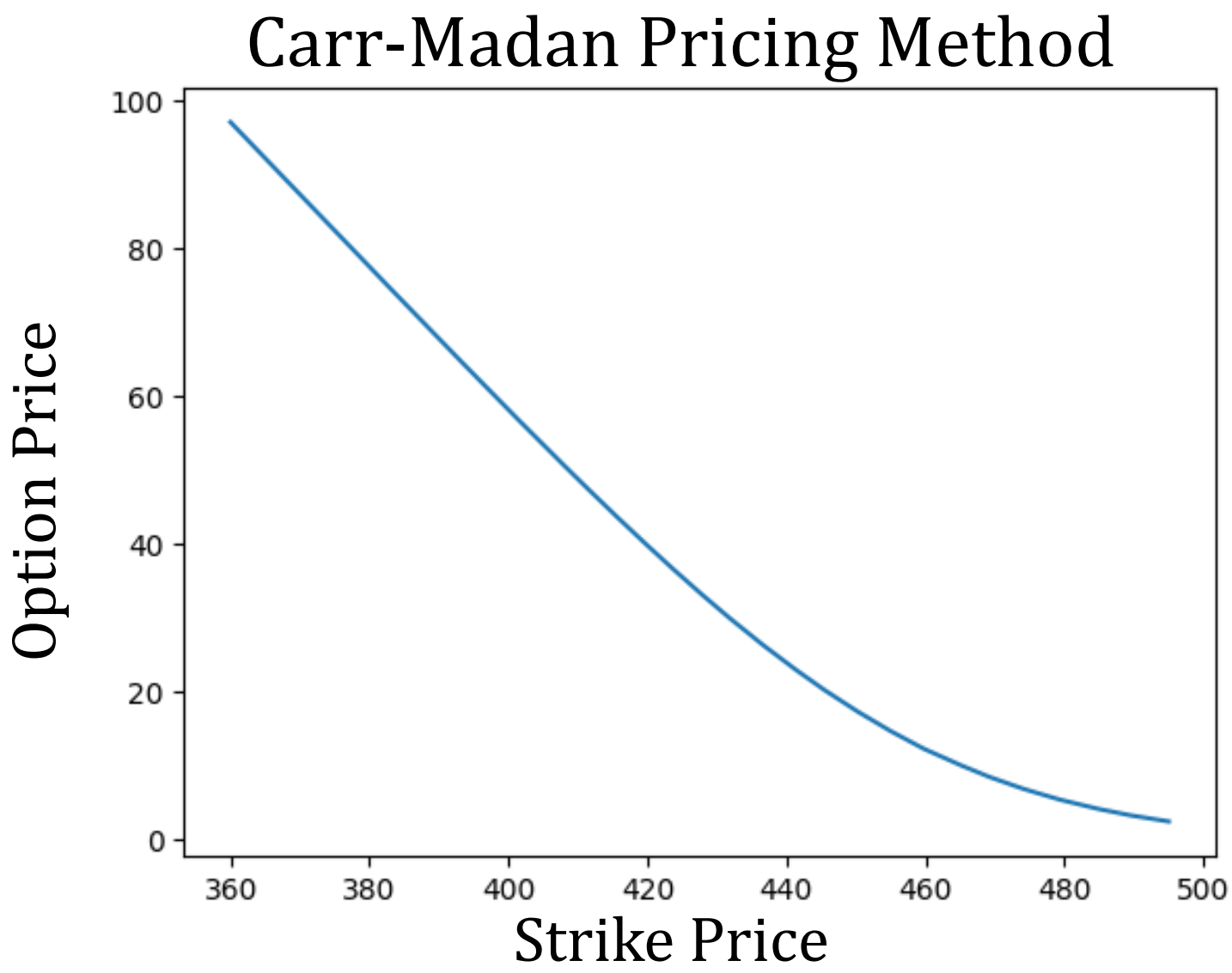
$$z_T(k) \approx \frac{1}{\pi}\sum_{j=0}^{N-1} w_j\, e^{-ijk\eta}\zeta(\eta j)$$

By setting a log-strike grid, $\mathrm{k_l} = -\lambda\frac{N}{2} + \lambda \mathrm{l}$ with $\lambda = \frac{2\pi}{N\eta}$ and $l = 0, \dots, N-1$ we can reach the FFT standard Form:

$$z_T(k_l) \approx \frac{1}{\pi}\sum_{j=0}^{N-1} w_j\, e^{ij\pi} e^{-ijl\frac{2\pi}{N}}\zeta(\eta j)\eta = \frac{1}{\pi} FFT\left[\left\{w_j e^{ij\pi}\zeta(\eta j)\eta\right\}_{j=0}^{N-1}\right]$$

CARR-MADAN PRICING

```python
def C_k(S0, v0, kappa, theta, sigma, rho, tau, r):
    A = 600
    N = 2**13
    eta = A/N
    v = [eta*i for i in range(N)]
    v[0] = 1e-22

    FT_BS = [trasfFur_T(v[i], S0, v0, kappa, theta, sigma, rho, tau, r) for i in range(N)]
    w = np.concatenate(([0.5], np.ones(N-2), [0.5]))
    h = [w[i]*eta*FT_BS[i]*np.exp(i*1j*np.pi) for i in range(N)]

    lambda_val = 2 * np.pi / (N * eta)
    k = -lambda_val * N / 2 + lambda_val * np.arange(N)

    K = []
    maxx = [np.max([0, 1-np.exp(k[i]-r*tau)]) for i in range(N)]
    K = [S0*np.exp(k[i]) for i in range(N)]

    P = S0*np.real(fft(h)/np.pi + maxx)

    #out = [[K[i],P[i]] for i in range(N) if (K[i] > 0.1 * S0 and K[i] < 20 * S0)]
    out = [[K[i],P[i]] for i in range(N)]

    return out
```


Carr-Madan Pricing Method
Option Price
100
80
60
40
20
0
360
380
400
420
440
460
480
500
Strike Price

BACKGROUND

THE HESTON MODEL

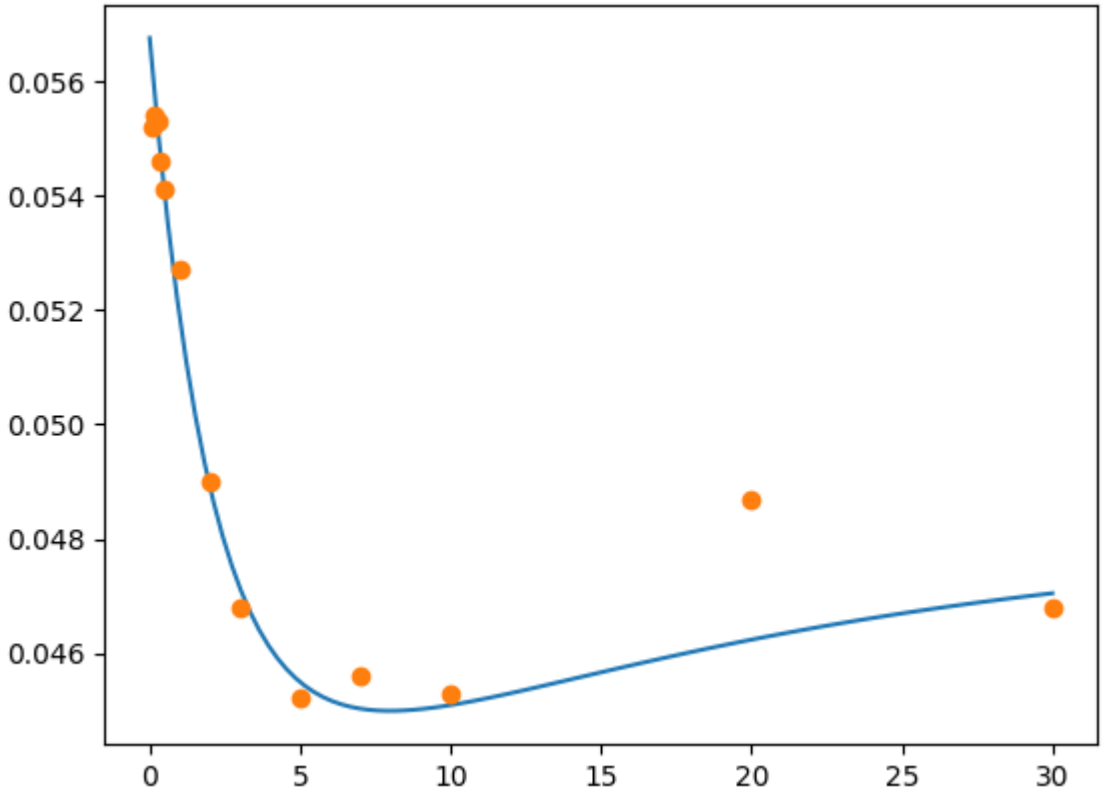
CARR-MADAN PRICING

CALIBRATION

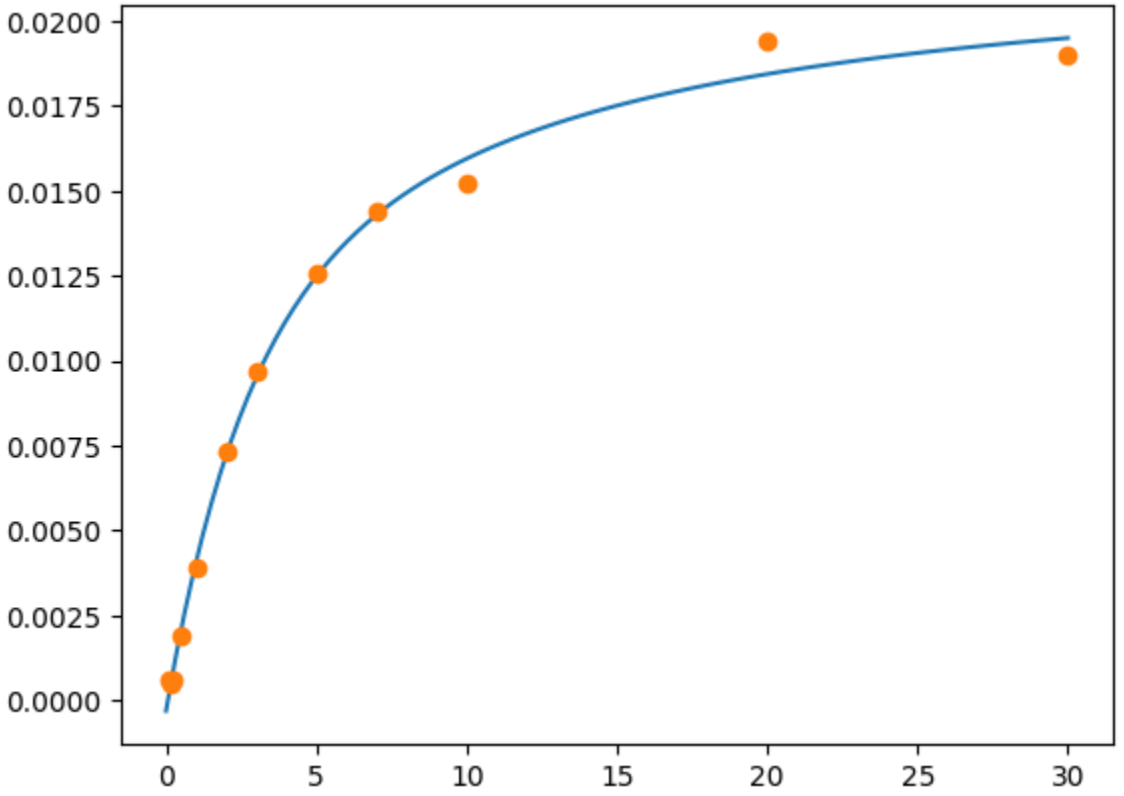
To price derivatives we drop constant the interest rate assumption.

Using Zero Coupon Points Then Interpolate with Nelson-Sieger-Svenson:

$$r(t) = \beta_0 + (\beta_1 + \beta_2)\frac{1 - e^{-\frac{t}{\tau_1}}}{\frac{t}{\tau_1}} - \beta_2 e^{-\frac{t}{\tau_1}} + \beta_3 \frac{1 - e^{-\frac{t}{\tau_2}}}{\frac{t}{\tau_2}} - \beta_3 e^{-\frac{t}{\tau_2}}$$



Inverted Interest Rate Curve - Today



Normal Interest Rate Curve – 31/12/2021

Pulled options data from Yahoo Finance and considered only the ETF tracking the S&P 500, because of its widespread usage.

Dropped any data of options that were traded in a volume of less than 10 units a day.

Made a Training and Test Dataset for our model, by applying a random 80/20 split between all options.

The goal of the calibration is to find:

$$\widehat{\Theta} = arg \min_{\Theta \in U_\Theta} SqErr(\Theta)$$

Where:

$$SqErr(\Theta) = \sum_{i=1}^{N} \sum_{j=1}^{M} w_{ij} \left(C_{MP}(K_i, T_j) - C_{HP}(S_0, K_i, T_j, r_j, \Theta) \right)^2$$

And: $\Theta = (v0, \kappa, \theta, \sigma, \rho)$ $2\kappa\theta - \sigma^2 > 0$

CALIBRATION

```
# Define variables to be used in optimization
S0 = stock_data.history(period="1d")["Close"][0] # initial asset price
r = Training_Table['rate'].to_numpy('float')
K = Training_Table['strike'].to_numpy('float')
tau = Training_Table['maturity'].to_numpy('float')
P = Training_Table['lastPrice'].to_numpy('float')

params = {"v0": {"x0": 0.018293, "boundaries": [1e-3,0.1]},
          "kappa": {"x0": 0.001, "boundaries": [1e-3,5]},
          "theta": {"x0": 0.1, "boundaries": [1e-3,0.1]},
          "sigma": {"x0": 0.05830981, "boundaries": [1e-2,1]},
          "rho": {"x0": 0.0687171, "boundaries": [-1,1]},
          }

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

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

def Feller(x):
    v0, kappa, theta, sigma, rho = [param for param in x]
    return 2*kappa*theta - sigma**2 - 0.01

cons = {'type':'ineq', 'fun': Feller}
```

CALIBRATION

```
def SqErr(x):
    err = 0

    v0, kappa, theta, sigma, rho = [param for param in x]

    new_r = np.array(list(set(r)))
    new_r = np.reshape(new_r, (len(new_r), 1))

    for j in list(new_r):

        id = np.where(r == j)
        new_DF = Training_Table[Training_Table["rate"] == j[0]]
        #print(new_DF)
        strike = new_DF['strike'].tolist()
        P_new = new_DF["lastPrice"].to_numpy()

        PP = priceCont(S0, v0, j[0], kappa, theta, sigma, rho, tau[id][0])

        err = err + np.sum(np.square(np.subtract(P_new, PP(strike))))

    err = err/len(P)
    return err

result = minimize(SqErr, x0, tol = 1e-2, method='SLSQP', bounds=bnds, constraints
```

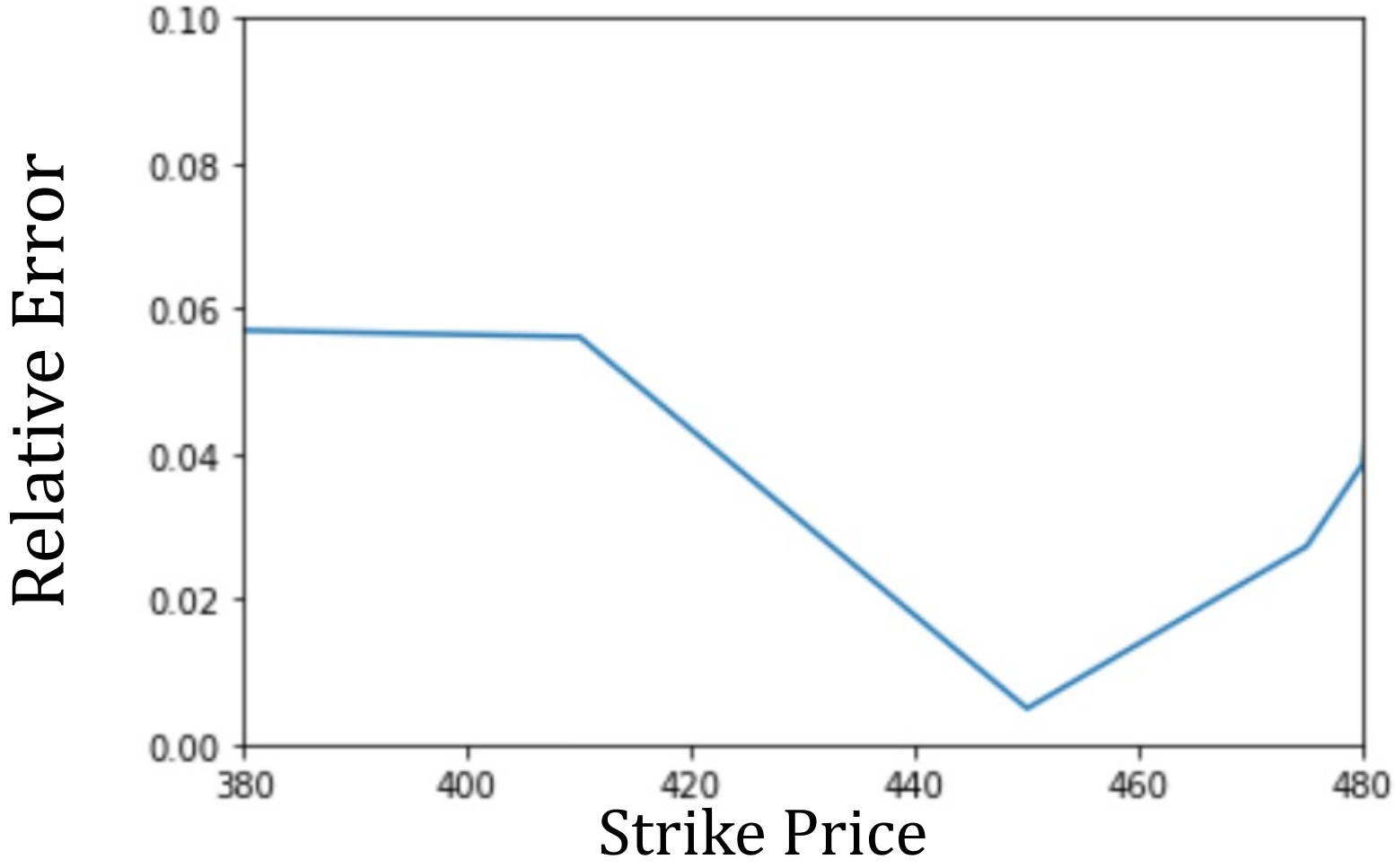
```
    print(result)
    print()
    print("Feller condition:", Feller(result.x)+0.01, " > 0")
```

```
 message: Optimization terminated successfully
 success: True
  status: 0
     fun: 62.57157059619531
       x: [ 1.794e-02  1.623e-02  1.000e-01  3.915e-02  6.637e-02]
     nit: 3
     jac: [-3.547e+00  5.244e-02 -6.072e-02  2.393e-02 -1.135e+01]
    nfev: 25
    njev: 3

Feller condition: 0.0017127935557986 89  > 0
```

```python
def SqErr_Test(x):
    err = 0

    v0, kappa, theta, sigma, rho = [param for param in x]

    size = len(K)*len(tau)

    err = 0

    new_r = np.array(list(set(r)))
    new_r = np.reshape(new_r, (len(new_r), 1))

    for j in list(new_r):

        id = np.where(r == j)
        new_DF = Test_Table[Test_Table["rate"] == j[0]]
        #print(new_DF)
        strike = new_DF['strike'].tolist()
        P_new = new_DF["lastPrice"].to_numpy()

        PP = priceCont(S0, v0, j[0], kappa, theta, sigma, rho, tau[id][0])

        err = err + np.sum(np.square(np.subtract(P_new, PP(strike))))

    return err/len(P)
```

```
Mean squared error: 37.07922512652723
```

Relative Error of all call option of 2026-01-16



THANK YOU