```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import sparse
from scipy.stats import norm
from scipy.optimize import brentq, fsolve, minimize_scalar
```

Bachelier (1990), Black (1976), QNM (2023) classes

Bachelier (1990)

$$dF_t = \sigma_a dW_t$$

$$C_{BC}(F, K, \sigma_a, r, \tau) = e^{-r\tau}\left[(F-K)N(m_a) + \sigma_a\sqrt{\tau}n(m_a)\right]$$
$$P_{BC}(F, K, \sigma_a, r, \tau) = e^{-r\tau}\left[(K-F)(1-N(m_a)) + \sigma_a\sqrt{\tau}n(m_a)\right]$$

where

$$m_a = \frac{F-K}{\sigma_a\sqrt{\tau}}$$

Black (1976)

$$dF_t = \sigma_G F_t dW_t$$

$$C_{BL}(F, K, \sigma_G, r, \tau) = e^{-r\tau}\left[FN(m_G + \frac{\sigma_G\sqrt{\tau}}{2}) - KN(m_G - \frac{\sigma_G\sqrt{\tau}}{2})\right]$$
$$P_{BL}(F, K, \sigma_G, r, \tau) = e^{-r\tau}\left[K(1-N(m_G - \frac{\sigma_G\sqrt{\tau}}{2})) - F(1-N(m_G + \frac{\sigma_G\sqrt{\tau}}{2}))\right]$$

where

$$m_G = \frac{ln(F-K)}{\sigma_G\sqrt{\tau}}$$

Quadratic Normal Model (Bouchouev, 2023)

$$dF_t = \sigma(F_t)dW_t = (\sigma_{ATM} + a + bF_t + cF_t^2)dW_t$$

$$C(F, K, a, b, c, r, \tau) = C_{BC}(F, K, \sigma_a = \sigma_{ATM}, r, \tau) + e^{-r\tau}U$$
$$P(F, K, a, b, c, r, \tau) = P_{BC}(F, K, \sigma_a = \sigma_{ATM}, r, \tau) + e^{-r\tau}U$$

where

$$U = \sqrt{\tau}n(\frac{F-K}{\sigma_{ATM}\sqrt{\tau}})[a + \frac{b}{2}(F+K) + \frac{c}{3}(F^2 + FK + K^2 + \frac{\sigma_{ATM}^2\tau}{2})]$$

```
[ ]
```
1 個隱藏的儲藏格

IBV and INV function

```
[ ]
```
1 個隱藏的儲藏格

Derman's approximation

In Chapter 16 of The Volatility Smile, Derman approximated local volatility by calculating the average of local volatilities between the current underlying price F and the strike price K. Given the quadratic normal local volatility, i.e. $\sigma(f) = \sigma_{ATM} + a + bf + cf^2$, the implied normal volatility estimated by Derman's approximation becomes:

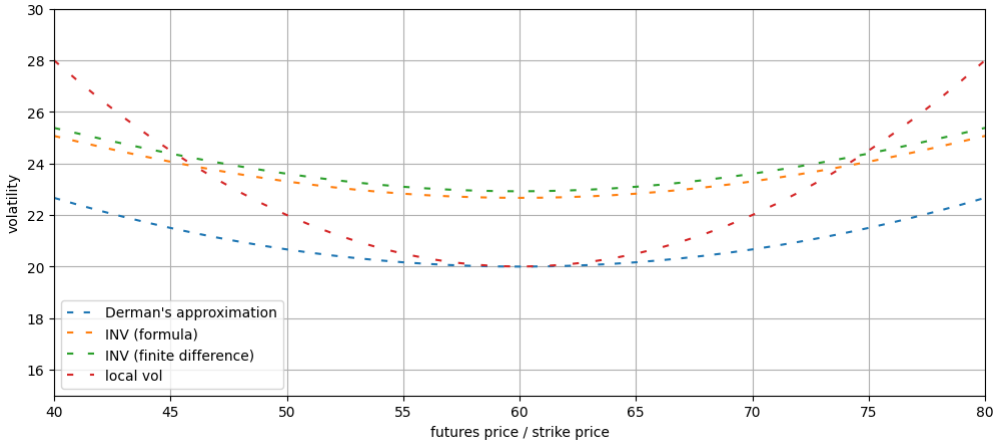
$$\begin{aligned} INV(K, F) &\approx \frac{1}{K-F}\int_F^K \sigma(f)df \\ &= \frac{1}{K-F}\int_F^K (\sigma_{ATM} + a + bf + cf^2)dx \\ &= \sigma_{ATM} + a + \frac{b}{2}(K+F) + \frac{c}{3}(K^2 + KF + F^2) \end{aligned}$$

```
# QNM parameters
sig_atm, a, b, c = 20, 72, -2.4 ,0.02                # 20 + 0.02*(F-60)^2
futures_price = 60.0
risk_free_rate = 0.0
time_to_maturity = 2.0

# local vol
sig_qnm = lambda f : sig_atm + a + b*f + c*f**2

# Derman's approximation
derman = lambda k : sig_atm + a + b/2*(k + futures_price) + c/3*(k**2 + k*futures_price + futures_price**2)
```

```
step_size_f = 1.0
arr_f = np.arange(40, 80 + step_size_f/2, step_size_f)

# compute call prices across strikes
qnm = QNM(futures_price, sig_atm, a, b, c, risk_free_rate, time_to_maturity)
arr_call_formula = qnm.option_pricer(K = arr_f, option_type = 'call')
list_call_fd = []
for k_ in arr_f:
    list_call_fd.append(qnm.implicit_FD(k_, option_type = 'call'))
arr_call_fd = np.array(list_call_fd)

# compute the corresponding INVs across strikes
list_inv_formula = []
list_inv_fd = []
for stirke_, call_formula_, call_fd_ in zip(arr_f, arr_call_formula, arr_call_fd):
    list_inv_formula.append(implied_volatility(call_formula_, futures_price, stirke_, risk_free_rate, time_to_maturity,
```

```
                                                                option_type = 'call', model = 'bachelier', method='brent', disp=True))
    list_inv_fd.append(implied_volatility(call_fd_, futures_price, stirke_, risk_free_rate, time_to_maturity,
                                                                option_type = 'call', model = 'bachelier', method='brent', disp=True))
arr_inv_formula = np.array(list_inv_formula)
arr_inv_fd = np.array(list_inv_fd)
```

```
# plot the local vol, INVs, and Derman's approximation
arr_derman = derman(arr_f)
plt.figure(figsize=(12,5))

plt.plot(arr_f, arr_derman, label = 'Derman\'s approximation', linestyle = (0,(3,5)))
plt.plot(arr_f, arr_inv_formula, label = 'INV (formula)', linestyle = (0,(3,6)))
plt.plot(arr_f, arr_inv_fd, label = 'INV (finite difference)', linestyle = (0,(3,7)))
plt.plot(arr_f, sig_qnm(arr_f), label = 'local vol', linestyle = (0,(3,8)))

plt.xlabel('futures price / strike price')
plt.ylabel('volatility')
plt.xlim(40,80)
plt.ylim(15,30)
plt.grid()
plt.legend()

plt.show()
```



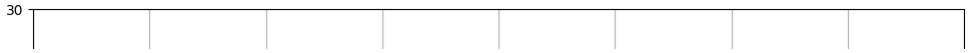
Revised Derman's approximation

One notable limitation of Derman's approximation is its insensitivity to the time to maturity. When we consider the quadratic normal model with the current underlying price corresponding to the vertex (minimum value) of the parabola, the implied normal volatility should increase as the time to maturity extends. This is because, over a longer time frame, the underlying price is more likely to move outward and accumulate higher local volatility. However, Derman's approximation always yields the vertex value, regardless of the time to maturity.

To address this limitation, we have developed a revised version of Derman's approximation by introducing an additional term that is related to the time to maturity. This revised approximation can be readily derived using the quadratic normal formula and Taylor expansion.

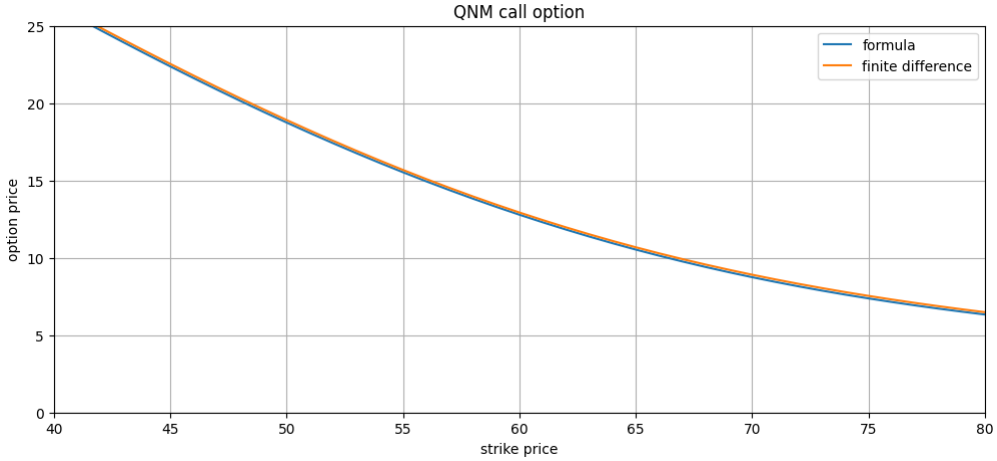
$$INV(K,F) \approx \sigma_{ATM} + a + \frac{b}{2}(K+F) + \frac{c}{3}(K^2 + KF + F^2) + \frac{c}{6}\sigma_{ATM}^2\,\tau$$

```
# Derman's approximation (revised)
derman_revised = lambda k : sig_atm + a + b/2*(k + futures_price) +\
                            c/3*(k**2 + k*futures_price + futures_price**2) + c/6*sig_atm**2*time_to_maturity

# plot the local vol, INVs, and Derman's approximations
arr_derman_revised = derman_revised(arr_f)
plt.figure(figsize=(12,5))

plt.plot(arr_f, arr_derman, label = 'Derman\'s approximation', linestyle = (0,(3,5)))
plt.plot(arr_f, arr_derman_revised, label = 'Derman\'s approximation (revised)', linestyle = (0,(3,6)))
plt.plot(arr_f, arr_inv_formula, label = 'INV (formula)', linestyle = (0,(3,7)))
plt.plot(arr_f, arr_inv_fd, label = 'INV (finite difference)', linestyle = (0,(3,8)))
plt.plot(arr_f, sig_qnm(arr_f), label = 'local vol', linestyle = (0,(3,9)))

plt.xlabel('futures price / strike price')
plt.ylabel('volatility')
plt.xlim(40,80)
plt.ylim(15,30)
plt.grid()
plt.legend()

plt.show()
```

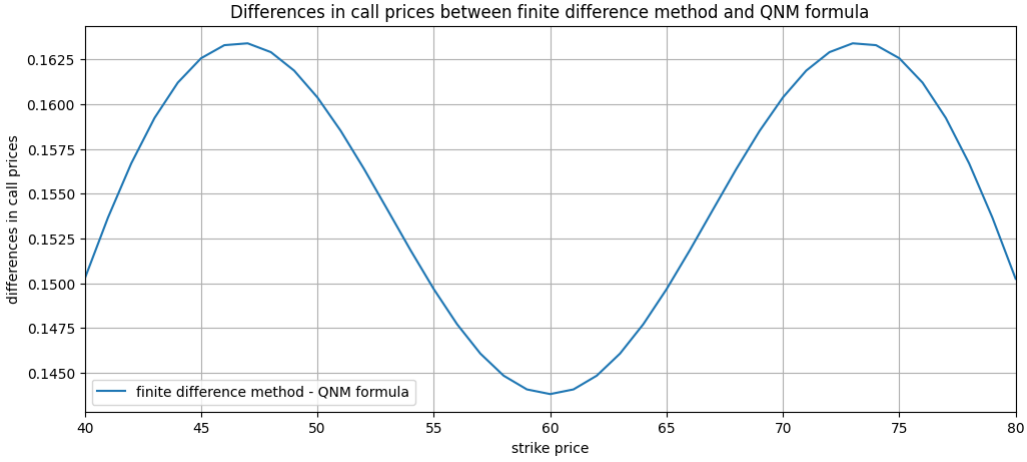


Onging exploration

```
# plot the call prices
plt.figure(figsize=(12,5))
plt.plot(arr_f, arr_call_formula, label = 'formula')
plt.plot(arr_f, list_call_fd, label = 'finite difference')

plt.title('QNM call option')
plt.xlabel('strike price')
plt.ylabel('option price')
plt.xlim(40,80)
plt.ylim(0,25)
plt.grid()
plt.legend()

plt.show()
```




```
# plot the differences in call prices between finite difference method and QNM formula
plt.figure(figsize=(12,5))
plt.plot(arr_f, list_call_fd - arr_call_formula, label = 'finite difference method - QNM formula')

plt.title('Differences in call prices between finite difference method and QNM formula')
plt.xlabel('strike price')
plt.ylabel('differences in call prices')
plt.xlim(40,80)
plt.grid()
plt.legend()

plt.show()
```




```
# compute the corresponding curvatures across strikes for INVs and Derman's approximations
arr_curve_formula = (arr_inv_formula[2:] + arr_inv_formula[:-2] - 2*arr_inv_formula[1:-1])/step_size_f**2
arr_curve_fd = (arr_inv_fd[2:] + arr_inv_fd[:-2] - 2*arr_inv_fd[1:-1])/step_size_f**2
arr_curve_derman = (arr_derman[2:] + arr_derman[:-2] - 2*arr_derman[1:-1])/step_size_f**2
arr_curve_derman_revised = (arr_derman_revised[2:] + arr_derman_revised[:-2] - 2*arr_derman_revised[1:-1])/step_size_f**2

# plot the curvatures for INVs and Derman's approximations
plt.figure(figsize=(12,5))

plt.plot(arr_f[1:-1], arr_curve_derman, label = 'Derman\'s approximation', linestyle = (0,(3,5)))
plt.plot(arr_f[1:-1], arr_curve_derman_revised, label = 'Derman\'s approximation (revised)', linestyle = (0,(3,6)))
plt.plot(arr_f[1:-1], arr_curve_formula, label = 'INV (formula)', linestyle = (0,(3,7)))
plt.plot(arr_f[1:-1], arr_curve_fd, label = 'INV (finite difference)', linestyle = (0,(3,8)))

plt.xlabel('strike')
plt.ylabel('curvature')
plt.xlim(40,80)
plt.ylim(0,0.015)
plt.grid()
plt.legend()
```

```
plt.show()
```