QF602 - Homework 6

Question

- Let's assume $r = q = 0$, $S_0 = 1$ and the implied volatility for $T = 4$ are given by the following formula:

$$\Sigma(K) = 0.510 - 0.591K + 0.376K^2 - 0.105K^3 + 0.011K^4 \qquad (1)$$

  with an upper limit, which is given by $\Sigma(K) = \Sigma(3)$ for $K > 3$.

- Any payoff that only depends on S_T can be priced with the following formula,

$$\begin{aligned} V_0 &= e^{-rT}V_T(F_0(T)) + \int_0^{F_0(T)} Put(K,T)\frac{\partial^2 V_T(K)}{\partial K^2}dK \\ &\quad + \int_{F_0(T)}^{\infty} Call(K,T)\frac{\partial^2 V_T(K)}{\partial K^2}dK \end{aligned}$$

  where $Call(K,T)$ and $Put(K,T)$ is computed by using the Black Scholes formula using the volatility $\Sigma(K)$ obtained in (1).

- Using the B-L formula, compute numerically the option prices at time 0, for the following payoffs:

  1. $V_T(S_T) = \sqrt{S_T}$
  2. $V_T(S_T) = S_T^3$

  You need to submit a Jupyter Note Book which contains executable Python code. You can modify the Python code for static replication for the square payoff in the lecture note 6.

  Answer.

  1. $\frac{\partial^2 V_T}{\partial S_T^2} = -\frac{1}{4}S_T^{-1.5}$

$$V_0 = \sqrt{F_0(T)} - \int_0^{F_0(T)} Put(K,T)\frac{K^{-1.5}}{4}dK - \int_{F_0(T)}^{\infty} Call(K,T)\frac{K^{-1.5}}{4}dK$$

2. $\frac{\partial^2 V_T}{\partial S_T^2} = 6 S_T$

$$V_0 = \left(F_0(T)\right)^3 + 6\int_0^{F_0(T)} Put(K,T) K dK + 6\int_{F_0(T)}^{\infty} Call(K,T) K dK$$

```python
import numpy as np
import scipy.integrate as integrate

def ivol_helper(K):
    return 0.510 - 0.591*K + 0.376*K**2 - 0.105*K**3 + 0.011*K**4

def ivol_HW6(K):
    if K>=3:
        return ivol_helper(3)
    else:
        return ivol_helper(K)

def black_with_smile(f, k, t, df, callorput):
    vol = ivol_HW6(k)
    return Black(f, k, t, vol, callorput) * df

def numerical_integration_HW6Q1(S0, r, q, T, SD):
    DF = np.exp(-r*T)
    DivF = np.exp(-q*T)
    f = S0*DivF/DF
    vol_for_range = ivol_HW6(f)
    maxS = f * np.exp(vol_for_range * SD * np.sqrt(T))
    forward_part = np.sqrt(f) * DF
    integrand_put = lambda y: y**(-1.5)/4 * black_with_smile(f, y, T, DF, EUROPEAN_PUT)
    put_part, error = integrate.quad(integrand_put, 0.0001, f)
    integrand_call = lambda x: x**(-1.5)/4 * black_with_smile(f, x, T, DF, EUROPEAN_CALL)
    call_part, error = integrate.quad(integrand_call, f, maxS)
    return forward_part - put_part - call_part

def numerical_integration_HW6Q2(S0, r, q, T, SD):
    DF = np.exp(-r*T)
    DivF = np.exp(-q*T)
    f = S0*DivF/DF
    vol_for_range = ivol_HW6(f)
    maxS = f * np.exp(vol_for_range * SD * np.sqrt(T))
    forward_part = f * f * f * DF
    integrand_put = lambda y: 6 * y * black_with_smile(f, y, T, DF, EUROPEAN_PUT)
    put_part, error = integrate.quad(integrand_put, 0, f)
    integrand_call = lambda x: 6 * x * black_with_smile(f, x, T, DF, EUROPEAN_CALL)
    call_part, error = integrate.quad(integrand_call, f, maxS)
    return forward_part + put_part + call_part

# driver routine for Q1 and Q2
q = 0.0; r = 0.0; T = 4; S0 = 1
SDs = np.linspace(1, 6, 6)
Q1numIntResults = [numerical_integration_HW6Q1(S0, r, q, T, sd) for sd in SDs]
Q2numIntResults = [numerical_integration_HW6Q2(S0, r, q, T, sd) for sd in SDs]
```