Chia-Wei Lai | clai2

Simulation Methods

46-773

Homework #5b

In [ ]:
```
import numpy as np
import scipy.stats as stats
```

In [ ]:
```
# np.random.seed(100000)
K  = 40
r = 0.06
S0_values = [38, 40]
sigma = 0.2
T = 1
n_av = 50000
time_steps = 50 * T
dt = T / time_steps

def gbm_path(S0, T, r, sigma, time_steps, n_av):
    dt = T / time_steps
    z = np.random.normal(size = (n_av, time_steps))

    S_path1 = np.zeros((n_av, time_steps + 1))
    S_path1[:, 0] = S0
    S_path1[:, 1:] = S0 * np.exp(np.cumsum((r - 0.5 * sigma**2) * dt + sigma * np.sqrt(dt) * z, axis = 1))

    S_path2 = np.zeros((n_av, time_steps + 1))
    S_path2[:, 0] = S0
    S_path2[:, 1:] = S0 * np.exp(np.cumsum((r - 0.5 * sigma**2) * dt - sigma * np.sqrt(dt) * z, axis = 1))

    return np.append(S_path1, S_path2, axis = 0)

def bs_put(S0, K, T, r, sigma):
    d1 = (np.log(S0 / K) + (r + sigma**2 / 2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    return K * np.exp(-r * T) * stats.norm.cdf(-d2) - S0 * stats.norm.cdf(-d1)


def put_payoff(S, K):
    return np.maximum(K - S, 0)

# Longstaff and Schwartz

def L0(x):
    return np.exp(-x / 2)
def L1(x):
    return np.exp(-x / 2) * (1 - x)
def L2(x):
    return np.exp(-x / 2) * (1 - 2 * x + x**2 / 2)

for S0 in S0_values:
    euro_put_price = bs_put(S0, K, T, r, sigma)
    S = gbm_path(S0, T, r, sigma, time_steps, n_av)
    n_path = S.shape[0]

    discount_payoff = np.exp(-r * dt) * put_payoff(S[:, time_steps], K)
    for j in range(time_steps - 1, 0, -1):

        inTheMoney_idx = np.where(put_payoff(S[:, j], K) > 0)
        X1 = L0(S[inTheMoney_idx, j])
        X2 = L1(S[inTheMoney_idx, j])
        X3 = L2(S[inTheMoney_idx, j])

        X = np.concatenate((np.ones((1, len(inTheMoney_idx[0]))), X1, X2, X3), axis = 0).T
        Y = discount_payoff[inTheMoney_idx]

        betas = np.linalg.lstsq(X, Y, rcond=None)[0]
        Continuation_value = np.zeros(n_path)
        y_hat = betas[0] + (betas[1:] * X[:, 1:]).sum(axis = 1)
        Continuation_value[inTheMoney_idx] = y_hat
        exercise = put_payoff(S[:, j], K).reshape(-1)

        replace_idx = np.where(exercise > Continuation_value)
        discount_payoff[replace_idx] = exercise[replace_idx]
        discount_payoff = np.exp(-r * dt) * discount_payoff

    price = discount_payoff.mean()
    stdErr = np.std((discount_payoff[:n_av] + discount_payoff[n_av:]) / 2, ddof = 1) / np.sqrt(n_av)

    print(f"S0 = {S0}, price = {price:.4f}, stdErr = {stdErr:.4f}, closed-form European = {euro_put_price:.4f}, Early exercise value = {price - euro_put_price:.4f}")
```

```
S0 = 38, price = 3.1769, stdErr = 0.0046, closed-form European = 2.8519, Early exercise value = 0.3249
S0 = 40, price = 2.2546, stdErr = 0.0042, closed-form European = 2.0664, Early exercise value = 0.1882
```

In [ ]: