Homework 4 Advanced Derivatives

Albin Henriksson

October 2022

Exercise 1

To price the outperformance option, we generated $M = 10^4$ points using the Gaussian copula method. The price we obtained of the outperformance option was

$$\Pi_0\left(\left(\frac{S_T^{SPX}}{S_0^{SPX}} - \frac{S_T^{AMZN}}{S_0^{AMZN}}\right)^+\right) = 0.05751$$

with a standard error of 0.00088. The code can be seen below.

Code

```
import pandas as pd
from scipy.stats import norm
import numpy as np
import random

"""Parameters and data"""

df=pd.read_excel('Impvols_SPX_AMZN.xlsx')
random.seed(0)
S_0_amzn = 1971
S_0_spx = 2921
strikes_spx = np.array(df.iloc[2:, 0].apply(pd.to_numeric)).reshape
    (-1, 1)
imp_vol_spx = np.array(df.iloc[2:, 1].apply(pd.to_numeric)).reshape
    (-1, 1)
strikes_amzn = np.array(df.iloc[2:, 4].apply(pd.to_numeric)).reshape
    (-1, 1)
imp_vol_amzn = np.array(df.iloc[2:, 5].apply(pd.to_numeric)).reshape
    (-1, 1)
q_amzn = .019
q_spx = .018
r = .024
T = .296
N = 10 ** 4
rho = .5
```

```python
"""Calculating option prices based on the above data"""

def BS_option(S, K, T, sigma, r, q):
    d_1 = (np.log(S / K) + (r - q + 0.5 * sigma ** 2) * T) / (sigma *
          np.sqrt(T))
    d_2 = d_1 - sigma * np.sqrt(T)
    C = S * np.exp(-q * T) * norm.cdf(d_1) - K * np.exp(-r * T) *
        norm.cdf(d_2)
    return C

options_amzn = BS_option(S_0_amzn, strikes_amzn, T, imp_vol_amzn, r,
    q_amzn)
options_spx = BS_option(S_0_spx, strikes_spx, T, imp_vol_spx, r,
    q_spx)

"""Approximating CDF of stock price by approximating the first
    derivative of call option w.r.t. strike"""

cdf_amzn = 1 + np.diff(np.exp(r * T) * options_amzn.T) / np.diff(
    strikes_amzn.T)
cdf_spx = 1 + np.diff(np.exp(r * T) * options_spx.T) / np.diff(
    strikes_spx.T)
cdf_amzn[0, -1] = 1
cdf_spx[0, -1] = 1

"""Generating N multivariate normal r.v.'s with correlation rho in
    order to use the Gaussian copula method"""

Cov = [[1, rho], [rho, 1]]
mu = [0, 0]
X = np.random.multivariate_normal(mu, Cov, (N))
norm_cdf = norm.cdf(X)

"""Simulating stock prices"""

S_T_spx = np.zeros((N, 1))
S_T_amzn = np.zeros((N, 1))
M = len(cdf_amzn.T)
for i in range(N):
    index_amzn = np.where(cdf_amzn > norm_cdf[i, 0])[1][0]
    index_spx = np.where(cdf_spx > norm_cdf[i, 1])[1][0]
    S_T_amzn[i] = strikes_amzn[index_amzn]
    S_T_spx[i] = strikes_spx[index_spx]

"""Calculating outperformance option price along with its standard
    error"""
```

```python
discounted_payoffs = np.exp(-r * T) * np.maximum(S_T_spx / S_0_spx -
    S_T_amzn / S_0_amzn, 0)
outperformance_option_price = np.mean(discounted_payoffs)
standard_error = np.std(discounted_payoffs) / np.sqrt(N)
print("The price of the outperformance option is {0}, with an
    estimated standard error of {1}".format(
     outperformance_option_price,
     standard_error))
```