Project 5

By - Ameya Shete

In [1]:
```
import numpy as np
import math
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
```

Question 1

Consider the following information on the stock of company XYZ: The current stock price is \$50, and the volatility of the stock price is $\sigma = 30\%$ per annum. Assume the prevailing risk-free rate is $r = 6\%$ per annum. Use the following method to price the specified option and compute standard errors:

In [2]:
```
# Below, I will define the general functions that will be used across
```

In [3]:
```
s0, sigma, r, K, t1, t2, t3 = 50, 0.3, 0.06, 50, 0.25, 1, 3
n, N = 100, 100000
z = np.random.normal(0, 1, size=50000)
z_neg = -z
N = 100000
n = 100
delta_1, delta_2, delta_3 = t1/n, t2/n, t3/n
```

In [158]:
```
# The question has specified to use delta = 1/np.sqrt(100000). However
# the total maturities given in the questions - 0.25, 1, 3. Thus, I us
def stock_evolution(S0, sigma, r, T, n, z):
    '''
    s0: Initial underlying price
    sigma: volatility of underlying
    T: time to maturity
    n: number of time steps
    z: standard normal variable
    '''
    delta_t = T/n
    return S0 * np.exp((r - (sigma**2)/2)*(delta_t) + (sigma)*(np.sqrt


def stock_matrix(S0, sigma, r, T, n, K):
    z = np.random.normal(0, 1, size=n*int(N/2))
    z = np.reshape(z, (int(N/2), n))
    z_neg = -z
    z = np.concatenate([z, z_neg], axis=0)
    stock_matrix = np.zeros((N, n+1))
    stock_matrix[:, 0] = s0
    for i in range(n):
        stock_matrix[:, i+1] = stock_evolution(stock_matrix[:, i], sig
    stock_matrix = stock_matrix[:, 1:]/K
    return stock_matrix
```

In [159]:
```
stock_matrix_1 = stock_matrix(s0, sigma, r, t1, n, K)
stock_matrix_2 = stock_matrix(s0, sigma, r, t2, n, K)
stock_matrix_3 = stock_matrix(s0, sigma, r, t3, n, K)
```

a)

Use the LSMC method with N=100,000 paths simulations (50,000 plus 50,000 antithetic variates) and a time step of Δ= 1 $\sqrt{N}$ to price an American Put option with strike price of $X =$ \$50 and maturity of 0.25-years, 1-year, and 3-years. Use the first k of the Laguerre Polynomials for $k = 2, 3, 4$. That is, you will compute 9 prices here. Compute the standard errors to assess and comment on the convergence of the 3 cases: $k = 2, 3, 4$.

In [167]:
```
# Laguerre Polynomials
def lsmc_laguerre(k, time, stock_matrix):
    index = np.zeros((N, n))
    index[:, n-1] = np.ones(N)
    cash_flows = np.zeros((N, n))
    cash_flows[:, n-1] = np.maximum(0, 1 - stock_matrix[:, n-1])
    delta_t = time/n
    disc_factor = np.exp(-r * delta_t)
    for i in reversed(range(n-1)):
        payoffs = disc_factor * cash_flows[:, i+1]
        stock_path = stock_matrix[:, i]
        f_term1 = np.exp(-stock_path/2)
        f_term2 = np.exp(-stock_path/2)*(1-stock_path)
        f_term3 = np.exp(-stock_path/2)*(1 - (2*stock_path) + (stock_p
        f_term4 = np.exp(-stock_path/2)*(1 - (3*stock_path) + (3*stock
        term = k
        if term == 2:
            X = np.stack([f_term1, f_term2], axis = 1)
        elif term == 3:
            X= np.stack([f_term1, f_term2,f_term3], axis = 1)
        elif term == 4:
            X = np.stack([f_term1, f_term2, f_term3, f_term4], axis =

        A = np.dot(X.T,X)
        b = np.dot(X.T,payoffs)
        Y = np.dot(np.linalg.inv(A), b)
        cont_val = np.dot(X, Y)
        exercise_value = np.maximum(0, 1 - stock_matrix[:, i])
        cash_flows[:, i] = np.maximum(cont_val, exercise_value)
        index[:, i] = np.where(exercise_value > cont_val, 1, 0)
    exercise_index = []
    for i in range(len(index)):
        exercise_index.append(np.argmax(index[i]))
    index = np.zeros((N, n))
    for i in range(N):
        index[i, exercise_index[i]] = 1
    sum_payoffs = 0
    discount_factor = np.tile(np.exp(-r*delta_t*np.arange(1, n+1, 1)),
    premium = np.mean(np.sum(index*cash_flows*K*np.exp(discount_factor
    return premium
```

In [168]:
```
print("Premium at t = 0.25 and k = 2", lsmc_laguerre(2, t1, stock_matr
print("Premium at t = 0.25 and k = 3", lsmc_laguerre(3, t1, stock_matr
print("Premium at t = 0.25 and k = 4", lsmc_laguerre(4, t1, stock_matr
```

```
Premium at t = 0.25 and k = 2 7.130722851866574
Premium at t = 0.25 and k = 3 7.102735288889802
Premium at t = 0.25 and k = 4 7.109563748137335
```

In [169]:
```
print("Premium at t = 1 and k = 2", lsmc_laguerre(2, t2, stock_matrix_
print("Premium at t = 1 and k = 3", lsmc_laguerre(3, t2, stock_matrix_
print("Premium at t = 1 and k = 4", lsmc_laguerre(4, t2, stock_matrix_
```

```
Premium at t = 1 and k = 2 12.300735100938113
Premium at t = 1 and k = 3 12.703016515798284
Premium at t = 1 and k = 4 12.606841174375054
```

In [170]:
```
print("Premium at t = 3 and k = 2", lsmc_laguerre(2, t2, stock_matrix_
print("Premium at t = 3 and k = 3", lsmc_laguerre(3, t2, stock_matrix_
print("Premium at t = 3 and k = 4", lsmc_laguerre(4, t2, stock_matrix_
```

```
Premium at t = 3 and k = 2 18.337159664027315
Premium at t = 3 and k = 3 19.31172343644629
Premium at t = 3 and k = 4 19.623885584193232
```

b)

In [171]:
```
# Hermite Polynomials
def lsmc_hermite(k, time, stock_matrix):
    index = np.zeros((N, n))
    index[:, n-1] = np.ones(N)
    cash_flows = np.zeros((N, n))
    cash_flows[:, n-1] = np.maximum(0, 1 - stock_matrix[:, n-1])
    delta_t = time/n
    disc_factor = np.exp(-r * delta_t)
    for i in reversed(range(n-1)):
        payoffs = disc_factor * cash_flows[:, i+1]
        stock_path = stock_matrix[:, i]
        f_term1 = 2*stock_path
        f_term2 = 4*stock_path**2 - 2
        f_term3 = 8*stock_path**3 - 12 *stock_path
        f_term4 = 16*stock_path**4 - 56*stock_path**2 + 16
        term = k
        if term == 2:
            X = np.stack([f_term1, f_term2], axis = 1)
        elif term == 3:
            X= np.stack([f_term1, f_term2,f_term3], axis = 1)
        elif term == 4:
            X = np.stack([f_term1, f_term2, f_term3, f_term4], axis =

        A = np.dot(X.T,X)
        b = np.dot(X.T,payoffs)
        Y = np.dot(np.linalg.inv(A), b)
        cont_val = np.dot(X, Y)
        exercise_value = np.maximum(0, 1 - stock_matrix[:, i])
        cash_flows[:, i] = np.maximum(cont_val, exercise_value)
        index[:, i] = np.where(exercise_value > cont_val, 1, 0)
    exercise_index = []
    for i in range(len(index)):
        exercise_index.append(np.argmax(index[i]))
    index = np.zeros((N, n))
    for i in range(N):
        index[i, exercise_index[i]] = 1
    sum_payoffs = 0
    discount_factor = np.tile(np.exp(-r*delta_t*np.arange(1, n+1, 1)),
    premium = np.mean(np.sum(index*cash_flows*K*np.exp(discount_factor
    return premium
```

In [172]:
```
print("Premium at t = 0.25 and k = 2", lsmc_hermite(2, t1, stock_matri
print("Premium at t = 0.25 and k = 3", lsmc_hermite(3, t1, stock_matri
print("Premium at t = 0.25 and k = 4", lsmc_hermite(4, t1, stock_matri
```

```
Premium at t = 0.25 and k = 2 7.13777243709695
Premium at t = 0.25 and k = 3 7.128829616710593
Premium at t = 0.25 and k = 4 7.107477407656418
```

In [173]:
```
print("Premium at t = 1 and k = 2", lsmc_hermite(2, t2, stock_matrix_2
print("Premium at t = 1 and k = 3", lsmc_hermite(3, t2, stock_matrix_2
print("Premium at t = 1 and k = 4", lsmc_hermite(4, t2, stock_matrix_2
```

```
Premium at t = 1 and k = 2 12.408053588344476
Premium at t = 1 and k = 3 12.211311695344534
Premium at t = 1 and k = 4 12.213386669746786
```

In [174]:
```
print("Premium at t = 3 and k = 2", lsmc_hermite(2, t2, stock_matrix_3
print("Premium at t = 3 and k = 3", lsmc_hermite(3, t2, stock_matrix_3
print("Premium at t = 3 and k = 4", lsmc_hermite(4, t2, stock_matrix_3
```

```
Premium at t = 3 and k = 2 19.22787006186154
Premium at t = 3 and k = 3 18.484032421633753
Premium at t = 3 and k = 4 18.26582892695256
```

c)

In [175]:
```
# Monomials Polynomials
def lsmc_monomials(k, time, stock_matrix):
    index = np.zeros((N, n))
    index[:, n-1] = np.ones(N)
    cash_flows = np.zeros((N, n))
    cash_flows[:, n-1] = np.maximum(0, 1 - stock_matrix[:, n-1])
    delta_t = time/n
    disc_factor = np.exp(-r * delta_t)
    for i in reversed(range(n-1)):
        payoffs = disc_factor * cash_flows[:, i+1]
        stock_path = stock_matrix[:, i]
        f_term1 = stock_path
        f_term2 = stock_path**2
        f_term3 = stock_path**3
        f_term4 = stock_path**4
        term = k
        if term == 2:
            X = np.stack([f_term1, f_term2], axis = 1)
        elif term == 3:
            X= np.stack([f_term1, f_term2,f_term3], axis = 1)
        elif term == 4:
            X = np.stack([f_term1, f_term2, f_term3, f_term4], axis =

        A = np.dot(X.T,X)
        b = np.dot(X.T,payoffs)
        Y = np.dot(np.linalg.inv(A), b)
        cont_val = np.dot(X, Y)
        exercise_value = np.maximum(0, 1 - stock_matrix[:, i])
        cash_flows[:, i] = np.maximum(cont_val, exercise_value)
        index[:, i] = np.where(exercise_value > cont_val, 1, 0)
    exercise_index = []
    for i in range(len(index)):
        exercise_index.append(np.argmax(index[i]))
    index = np.zeros((N, n))
    for i in range(N):
        index[i, exercise_index[i]] = 1
    sum_payoffs = 0
    discount_factor = np.tile(np.exp(-r*delta_t*np.arange(1, n+1, 1)),
    premium = np.mean(np.sum(index*cash_flows*K*np.exp(discount_factor
    return premium
```

In [176]:
```
print("Premium at t = 0.25 and k = 2", lsmc_monomials(2, t1, stock_mat
print("Premium at t = 0.25 and k = 3", lsmc_monomials(3, t1, stock_mat
print("Premium at t = 0.25 and k = 4", lsmc_monomials(4, t1, stock_mat
```

```
Premium at t = 0.25 and k = 2 7.1416376734103135
Premium at t = 0.25 and k = 3 7.1041692635580445
Premium at t = 0.25 and k = 4 7.197070021242284
```

In [177]:
```
print("Premium at t = 1 and k = 2", lsmc_monomials(2, t2, stock_matrix
print("Premium at t = 1 and k = 3", lsmc_monomials(3, t2, stock_matrix
print("Premium at t = 1 and k = 4", lsmc_monomials(4, t2, stock_matrix
```

```
Premium at t = 1 and k = 2 12.465618294199826
Premium at t = 1 and k = 3 12.23424887664641
Premium at t = 1 and k = 4 12.190800124715155
```

In [178]:
```
print("Premium at t = 3 and k = 2", lsmc_monomials(2, t2, stock_matrix
print("Premium at t = 3 and k = 3", lsmc_monomials(3, t2, stock_matrix
print("Premium at t = 3 and k = 4", lsmc_monomials(4, t2, stock_matrix
```

```
Premium at t = 3 and k = 2 19.37539341503627
Premium at t = 3 and k = 3 18.762091717241145
Premium at t = 3 and k = 4 18.493720178345917
```

d)

We find that the values are increasing in time and remain relatively the same for the different polynomial values taken.

In [ ]:

In [ ]:

In [ ]: