PSTAT 170 Project

December 2, 2021

PSTAT 170 PROJECT

AIDAN BAKER

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
```

[2]:
```
#QUESTION 1

def european_call(s0, k, r, sig, T, n):
    call = []
    for i in range(n):
        s = [s0]
        s.append(s[-1] * np.exp(((r - 0.5 * sig ** 2) * T) + (sig * np.sqrt(T)␣
↪* np.random.normal(0, 1))))
        call_value = max(s[-1] - k, 0)
        call.append(call_value)
    call_price = np.mean(call) * np.exp(-r * T)
    return call_price

print('n=100:',european_call(100, 100 , .02, .2, 1, 100))
print('n=1000:',european_call(100, 100 , .02, .2, 1, 1000))
print('n=10000:',european_call(100, 100 , .02, .2, 1, 10000))
print('n=100000:',european_call(100, 100 , .02, .2, 1, 100000))
```

```
n=100: 7.284402649984895
n=1000: 8.849616240396482
n=10000: 8.620972831626153
n=100000: 8.888327992706287
```

[3]:
```
#QUESTION 2


def asian_call_option(s0, k, r, T, sig, m, n):
    delta_t = T / m
    call = []
    for i in range(0, n):
        s = [s0]
```

```
        for j in range(0, m):
            s.append(s[-1] * np.exp((r - 0.5 * sig ** 2) * delta_t + (sig * np.
 ↪sqrt(delta_t) * np.random.normal(0, 1))))
        avg = np.mean(s)
        call.append(max((avg - k), 0))
    call_value = np.mean(call) * np.exp(-r * T)
    return call_value

print('n=100:', asian_call_option(100,100, 0.02, 1, 0.2, 10000, 100))
print('n=1000:', asian_call_option(100,100, 0.02, 1, 0.2, 10000, 1000))
print('n=10000:', asian_call_option(100,100, 0.02, 1, 0.2, 10000, 10000))
print('n=100000:', asian_call_option(100,100, 0.02, 1, 0.2, 10000, 100000))
```

```
n=100: 5.301618471218216
n=1000: 4.802494023974515
n=10000: 4.9320877178662474
n=100000: 5.025398453584617
```

[30]:
```
#QUESTION 3
def u_and_o(s0, k, r, T, sig, m, n, barrier):
    delta_t = T / m
    call = []
    for i in range(0, n):
        s = [s0]
        for j in range(0, m):
            s.append(s[-1] * np.exp((r - 0.5 * sig ** 2) * delta_t + (sig * np.
 ↪sqrt(delta_t) * np.random.normal(0, 1))))
            if s[-1] > barrier:
                s[-1] = 0
        call.append(max((s[-1] - k), 0))
    call_value = np.mean(call) * np.exp(-r * T)
    return call_value

print('n=100: ',u_and_o(100,100, 0.02, 1, 0.2, 10000, 100,120))
print('n=1000: ',u_and_o(100,100, 0.02, 1, 0.2, 10000, 1000,120))
print('n=10000: ',u_and_o(100,100, 0.02, 1, 0.2, 10000, 10000,120))
print('n=100000: ',u_and_o(100,100, 0.02, 1, 0.2, 10000, 100000,120))
```

```
n=100:  1.1862212025973262
n=1000:  1.092724453463964
n=10000:  1.132964070964681
n=100000:  1.176826038436539
```