(20/09) MScFE 630 Computational Finance (C20-S1)
Timezone Group 5 – E

Submission 2: Price a Vanilla European Call Option

Contributing Members:

Surbhi Gupta (gupta.surbhi2305@gmail.com)

Suraj Hegde (suraj1997pisces@gmail.com)

Rajeshwar Singh (singhrajeshwar.icfai@gmail.com)

Pradhyumn Singh (shekhawat2407@gmail.com)

ABSTRACT

The submission contains two files

1) The ipynb file that contains the code
2) The pdf file that contains the explanation, and the steps involved

We have added both files in the compressed folder.

INTRODUCTION

While the Black-Scholes model assumes constant volatility of the underlying , the Heston model proposed a model that proposed a stochastic volatility model. It was the first of its kind to result in a closed form characteristic function.
Originally similar to Black-Scholes, the Heston model assumed constant risk-free interest rates. Although there have been modifications to consider stochastic interest rates, we explore the Fourier approximation assuming the constant risk-free rate. The Heston model has the following dynamics

$$dS_t = \mu S dt + \sqrt{v_t} S dW^1_t$$

Where S_t is the asset value, W^1_t is a standard Brownian Motion, μ is the rate of return of the asset and the volatility term v_t follows a CIR or a square root process.
By applying Ito's formula to the volatility process, we can write it as

$$dv_t = \kappa(\theta - v_t)\, dt + \sigma\sqrt{v_t}\, dW^2_t$$

Where

- W^2_t is a standard Brownian motion and has a correlation ρ with W^1_t.
- θ is the long variance or the long run average price variance. This means that as t tends to infinity, the expected value of the volatility v_t tends to θ
- The rate at which v_t reverts to θ is given by κ
- σ here is the variance of v_t . It can be thought of as the volatility of volatility

If $2\kappa\theta > \sigma^2$, then v_t is strictly positive. This is also known as the Feller condition
We define the characteristic function of $s_t = \log(S_t)$ proposed by Albrecher et al,

$$\phi_{St} = \exp(C(\tau;u) + D(\tau;u)v_t + iu\log(S_t))$$

We then calculate the price of an European call option as

$$c = S_0\mathbb{Q}^S[S_T > K] - e^{-rT}K\mathbb{Q}[S_T > K]$$

$$= S_0\left(\frac{1}{2} + \frac{1}{\pi}\int_0^\infty \frac{Im\left[e^{-it\ln K}\varphi_{M_2}(t)\right]}{t}dt\right)$$

$$-e^{-rT}K\left(\frac{1}{2} + \frac{1}{\pi}\int_0^\infty \frac{Im[e^{-it\ln K}\varphi_{M_1}(t)]}{t}dt\right)$$

$$\varphi_{M_1}(t) = \phi_{S_T}(t)$$

$$\varphi_{M_2}(t) = \frac{\phi_{S_T}(u-i)}{\phi_{S_T}(-i)}$$

We estimate the first integral term as “first_integral” and the second integral term as “second_integral” in the code.

CODE WALKTHROUGH

1. Pricing a Vanilla Call Option using Fourier Technique

We start with importing the libraries and initialising the required variables .

```
In [209]: # Group Assignment 2
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt
import math

# Stock and variable information
S0 = 100
v0 = 0.06
kappa = 9
theta = 0.06
r = 0.08
sigma = 0.3
rho = -0.4
gamma = 0.75

# Strike info
K = 100
T = 1
k_log = np.log(K)

t_max = 30
N = 100
```

We define the stock value (S0), risk-free interest rate (r), time period (T), strike price (K) and the various variables required for the Heston model. We also define the number of intervals (N) and upper bound for integration for the Fourier pricing technique (t_max).
We now define all the helper functions used for calculation of Call Price using the Heston model characteristic function.

In [136]:
```
a = sigma**2/2

# Functions for use in the Heston model calculation for Call price

def b(u):
    return kappa - rho * sigma * 1j * u

def c(u):
    return -(u ** 2 + u * 1j) / 2

def d(u):
    return np.sqrt(b(u) ** 2 - 4 * a *c(u))

def xminus(u):
    return (b(u) - d(u)) / (2 * a)

def xplus(u):
    return (b(u) + d(u)) / (2 * a)

def g(u):
    return xminus(u) / xplus(u)
```

In [137]:
```
def C(u):
    val = T * xminus(u) - np.log((1 - g(u) * np.exp(-T * d(u))) / (1 - g(u))) / a
    return r * T * 1j * u + theta * kappa * val

def D(u):
    val1 = 1 - np.exp(-T * d(u))
    val2 = 1 - g(u) * np.exp(-T * d(u))
    return (val1/val2) * xminus(u)

def log_char(u):
    return np.exp(C(u) + D(u) * v0 + 1j * u * np.log(S0))

def adj_char(u):
    return log_char(u - 1j) / log_char(-1j)
```

The function log_char is φ_{M1} and adj_char function is φ_{M2} as mentioned in the previous section

We define the terms “delta_t” and “t_n” to vectorize our code to estimate the integral.

The integral terms are then estimated using the Riemann sum by using the number of terms N and the upper bound of integral t_max.

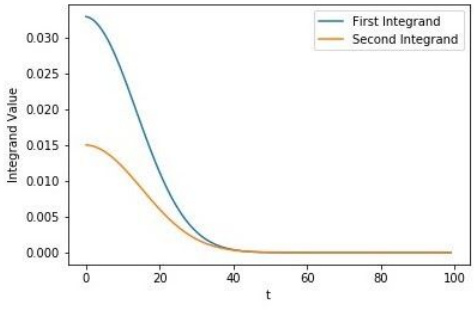
In [138]:
```
# Variables to vectorize the code - creating time samples
delta_t = t_max/N
from_1_to_N = np.linspace(1,N,N)
t_n = (from_1_to_N-1/2)*delta_t

# Approximate integral estimates and Fourier call price estimate
first_integral = sum((((np.exp(-1j * t_n * k_log) * adj_char(t_n)).imag) / t_n) * delta_t)
second_integral = sum((((np.exp(-1j * t_n * k_log) * log_char(t_n)).imag) / t_n) * delta_t)
fourier_cp = S0 * (1/2 + first_integral/np.pi) - K * np.exp(-r * T) * (1/2 + second_integral/np.pi)
print("Fourier Call Price : {}".format(fourier_cp))
```

```
Fourier Call Price : 13.734895692109077
```

Plot the graph of the two integrands vs t

In [45]:
```
plt.plot(((((np.exp(-1j*t_n*k_log)*adj_char(t_n)).imag)/t_n)*delta_t),label = "First Integrand")
plt.plot(((((np.exp(-1j*t_n*k_log)*log_char(t_n)).imag)/t_n)*delta_t),label = "Second Integrand")
plt.xlabel("t")
plt.ylabel("Integrand Value")
plt.legend()
plt.show()
```



The Fourier call price is approximated with a value of 13.7348956 .

2. Simulate a Share Price Path

Using $\sigma = 0.3$ and $\gamma = 0.75$ and the formula as given in the problem statement, we simulate the share price path.

t

In [140]:
```
# Time periods of monthly intervals, 1/12
def find_terminal_price(S0,r,Z,gamma,sigma):
    for i in range(0,12):
        sigma = sigma * S0 ** (gamma - 1)
        S0 = S0 * np.exp((r - sigma**2/2)*(1/12) + sigma*np.sqrt(1/12)*Z)
    return S0
```

Since we are performing monthly simulations for a period of one year, dt is taken as 1/12. Sigma is calculated using the closed - form CEV formula. Without actually having to store all the Share price values in an array , we can find the terminal price by updating S0 and sigma in every iteration.
Z will be our standard normal normal variable ~ N(0,1) . We will be assigning values to Z in our Monte Carlo simulation.

3. Monte Carlo Estimates and calculation of standard deviation and price of vanilla call option

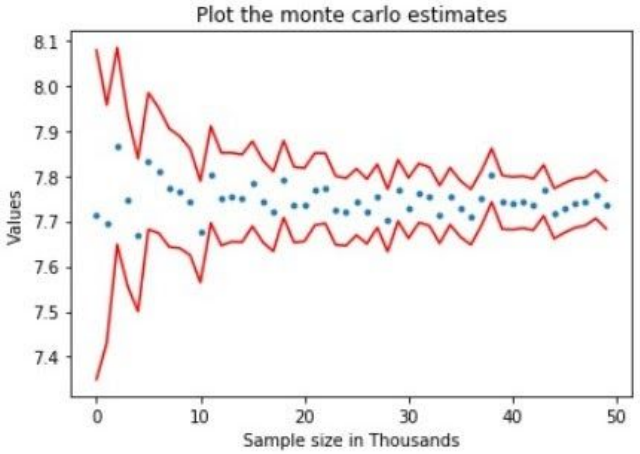
We calculate the discounted value of the call option in the function “discounted_payoff” Performing Monte Carlo simulations for sample sizes 1000,2000, 3000 all the way till 50000, we store the estimated values and the standard deviations in the variables “mcall_est” and “mcall_std” respectively. The discounted payoff in the Monte Carlo simulation is called with the terminal value calculated in the previous function and then stored in the mcall_est and mcall_std variables.

In [215]:
```
# Calculate Discounted Payoff
def discounted_payoff(S,K,r,T):
    return np.maximum(S-K,0) * np.exp(-r * T)
```

In [216]:
```
# Monteo Carlo simulation
mcall_est = [None]*50
mcall_std = [None]*50
for i in range(1,51):
    Z = norm.rvs(size = 1000*i)
    term_val = find_terminal_price(S0, r, Z, gamma, sigma)
    mcall_val = discounted_payoff(term_val,K,r,T)
    mcall_est[i-1] = np.mean(mcall_val)
    mcall_std[i-1] = np.std(mcall_val) / np.sqrt(i*1000)
```

We can now plot these values with three standard deviation error bounds.

In [148]:
```
# Plotting the graphs
plt.title("Plot the monte carlo estimates")
plt.plot(mcall_est,'.')
plt.plot(mcall_est + 3 * np.array(mcall_std),'r')
plt.plot(mcall_est - 3 * np.array(mcall_std),'r')
plt.xlabel('Sample size in Thousands')
plt.ylabel('Values')
plt.show()
```



The graph plots the Sample size against the values of the estimates. The three standard deviation error bounds are in red, while the estimated values are plotted as points in blue.

ANALYSIS

The Heston model has many advantages

- Asset price does not follow a log normal distribution. This assumption makes it closer to real life market evolution
- Volatility is mean reverting
- Quasi-closed form solution for European options
- Cheaper computation in comparison to Monte Carlo simulation

The disadvantages are that the calibration of the model parameters is tricky and difficult. The results are highly sensitive to the model parameters considered, like theta, sigma, v0.

The Black Scholes model is nested in the Heston model. Since we consider $\theta = v_0$ in our code, we can achieve the same Black - Scholes price by taking the sigma as $\sqrt{v_0}$

In [127]:
```
sigma_bs = np.sqrt(v0)
d1 = (np.log(S0/K) + (r + sigma_bs ** 2/2) * T) / (sigma_bs * np.sqrt(T))
d2 = d1 - sigma_bs * np.sqrt(T)
bs_call = S0 * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
print("Black Scholes Price (sigma = sqrt(v0)) : {}".format(bs_call))
print("Heston Model Fourier Price : {}".format(fourier_cp))
print("Error : {}".format(fourier_cp - bs_call))
```

```
Black Scholes Price (sigma = sqrt(v0)) : 13.709833079020868
Heston Model Fourier Price : 13.734895692109077
Error : 0.025062613088209673
```

We have calculated the analytical call price using these parameters and compared it with our values from the Heston model. The price is almost the same with an error of 0.025

The Heston model is quite sensitive to the model parameters. We try to compare the difference by changing each of these parameters. The original values of (v0.kappa,theta,r,sigma,rho,gamma) taken were (0.06,9,0.06,0.08,0.3,-0.4,0.75) in the calculation. The parameters have been labelled respectively in the following graphs

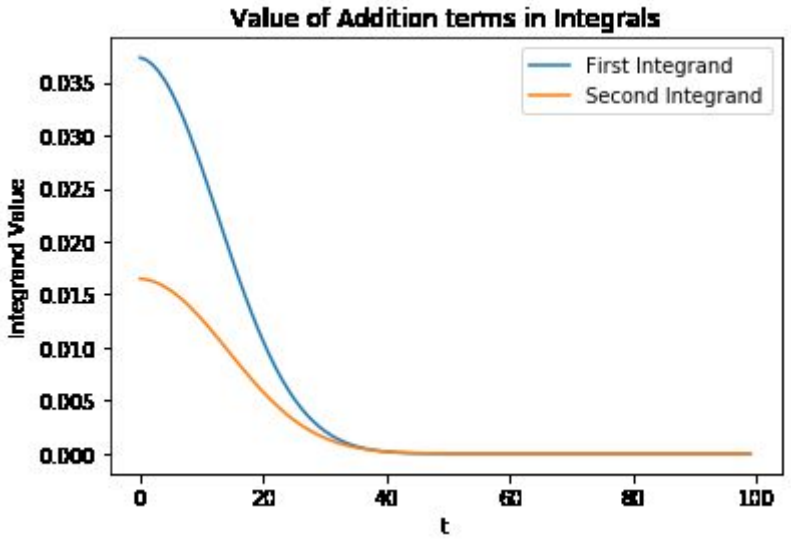


Figure 1 : Graph with Heston model parameters (0.07,10,0.07,0.08,0.4,-0.4,0.65)

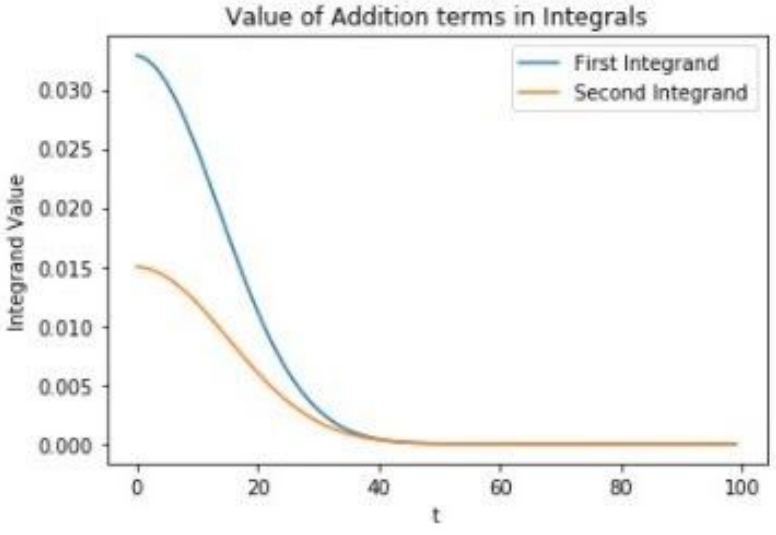


Figure 2 : Graph with Heston model parameters (0.06,9,0.06,0.08,0.3,-0.4,0.75)

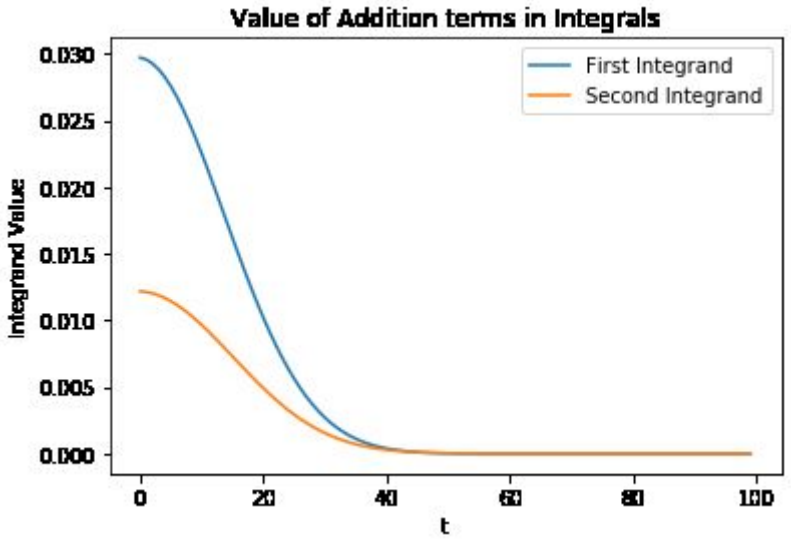


Figure 3 : Graph with Heston model parameters (0.05,8,0.06,0.07,0.2,-0.4,0.85)

We attempted to analyse the difference in values of the addition terms of the integrands by changing the parameters by a really small factor. The shape of the graphs remain the same but as shown even the smallest change brings about a considerable relative change in the integrand values.

BIBLIOGRAPHY

1. https://en.wikipedia.org/wiki/Heston_model
2. Hestonn, S. L. (1993). “A Closed-form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options”
3. Hsu,Y L Lin T and Lee, C (2008). “Constant Elasticity of Variance (cev) Option Pricing Model: Integration and Detailed Derivation”
4. Interest Rate Models - With Smile, Inflation and Credit , Damian Brigo and Fabio Merecurio
5. Module 5 Notes