Division of Applied Mathematics
School of Education, Culture & Communication
Mälardalen University
Box 883, 721 23 Västerås, Sweden

Seminar Report

Date: 18th October 2014

# PYTHON IMPLEMENTATION OF THE BLACK-SCHOLES MODEL AND CALCULATION OF PRICE AS FUNCTION OF TIME TO MATURITY AND SHOW IN A GRAPH HOW THE SOLUTION CONVERGE TO THE HOCKEY-STICK

MMA707- ANALYTICAL FINANCE I

TEACHER RESPONSIBLE: JAN RÖMAN

AUTHORS:

FYNN-AIKINS EKOW

ADOBAH-OTCHEY DANIEL

# Table of Contents

## 1. Introduction

This report is based on the usage of the Black Scholes Model to price derivatives particularly European Options using the Python Programming Language.

## 2. Theory on Black Scholes Model

### 2.1. Assumptions under the Black Scholes Model

The basic assumptions in the Black-Scholes world are:

- The underlying is a log normally distributed stochastic variable
- The volatility of the underlying is constant
- Interest rates are constant
- There are no transaction costs in any capital markets
- Borrowing and lending can be done at constant interest rate
- There is continuous trading in all instruments.

### 2.2. The No Dividend Assumption and the Standard Black Scholes PDE

The Black Scholes Model made another assumption in which the underlying derivate paid no dividend to the stocks shareholders. As is in this report, the price of the options $C(S,t)$ for call or $P(S,t)$ for put is based on the following parameters without dividend payments;

$k = exercise\ price$
$s_0 = price\ of\ underlying$
$T = time\ to\ maturity$
$\sigma = volatility$
$r = risk\ free\ rate$

The standard Black Scholes PDE is given by

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} = rf$$

*where*

$\frac{\partial f}{\partial t}$ *is the change in the option with respect to time,* $\Theta$

$\frac{\partial f}{\partial S}$ *is the change in the option with respect to price of underlying asset,* $\Delta$

$\frac{\partial^2 f}{\partial S^2}$ *is the change in the option with respect to volatility,* $\Gamma$

## 2.3. Formulae for Black Scholes European Call & Put

The Black Scholes Formulae, which is the solution to the Black Scholes PDE, used in the valuation of the option price, is given by;

$$C(S,t) = S\phi(d_1) - Ke^{-rT}\phi(d_2)$$

$$P(S,t) = Ke^{-rT}\phi(-d_2) - S\phi(-d_1)$$

*where*

$$d_1 = \frac{\ln(S/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

## 3. Applications and Results

### 3.1. Numerical Examples

[1] Using values obtained from Hull to calculate the call and put option prices for an option with the following parameters;

$k = 40$

$s_0 = 42$

$T = 0.5$

$\sigma = 0.1$

$r = 0.2$

$$d_1 = \frac{\ln(42/40) + (0.1 + 0.2^2/2) \times 0.5}{0.2\sqrt{0.5}} = 0.7693$$

$$d_2 = 0.7693 - 0.2\sqrt{0.5} = 0.6278$$

Therefore the value $C(S,t)$ of the call option is $4.76$

And the value $P(S,t)$ of the put option is 0.81

### 3.2 European Call Option Using Python

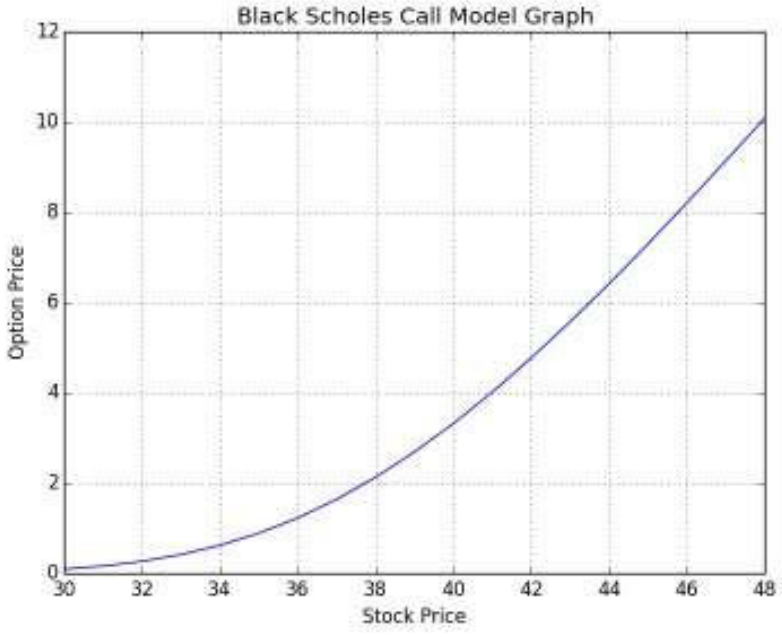
Using the same values from 3.1 we can use python to calculate $C(S,t)$ of the option as

```
>>> runfile('/Users/FADE/Documents/Python/Black Scholes Call.py', wdir=r'/Users/FADE/Documents/
Python')
Enter current stock price:
42
Enter strike price:
40
Enter risk-free rate:
10
Enter volatility rate:
20
Enter time to maturity in days:
183
S           stock price : 42
K           strike price: 40
r           risk-free rate: 0.1
sigma       volatility of the stock: 0.2
T           time to maturity: 0.501369863014
c_BS        : [ 4.64618156242
 ]
>>>
```

[1] [Options, Futures & Other Derivatives, 8th Edition], p.315-316, example 14.6

And then specifying a list of values for the underlying stock basing on the idea that the price of the stock could go up or down.

```
>>> runfile('/Users/FADE/Documents/Python/Black Scholes Call.py', wdir=r'/Users/FADE/Documents/Python')
Enter current stock price:
30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48
Enter strike price:
40
Enter risk-free rate:
10
Enter volatility rate:
20
Enter time to maturity in days:
183
S           stock price : (30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48)
K           strike price: 40
r           risk-free rate: 0.1
sigma       volatility of the stock: 0.2
T           time to maturity: 0.501369863014
c_BS        : [ 0.0922546932997
0.16077900727
0.264662974205
0.413922618839
0.618292218264
0.886301400987
1.22447408372
1.63677787222
2.12439244707
2.68579832636
3.31713133919
4.01271404675
4.76566563729
5.56850224741
6.41366250676
7.29392027914
8.20267171118
9.13410301674
10.0832576122
 ]
>>>
```



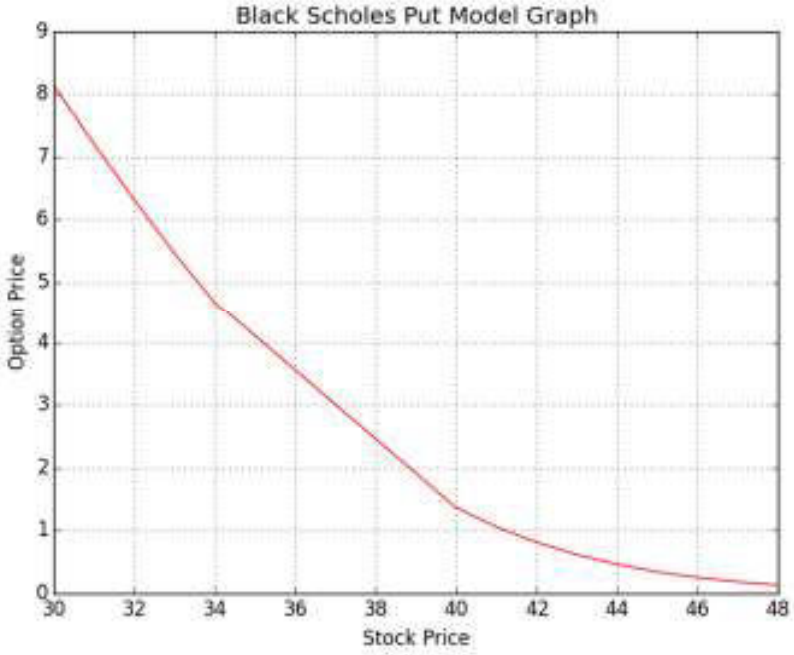
## 3.3 European Put Option Using Python

Using the same values from the 3.1 we can use python to calculate $P(S,t)$of the option as

```
>>> runfile('/Users/FADE/Documents/Python/Black Scholes Put.py', wdir=r'/Users/FADE/Documents/Python')
Enter current stock price:
42
Enter strike price:
40
Enter risk-free rate:
10
Enter volatility rate:
20
Enter time to maturity in days:
183
S           stock price : 42
K           strike price: 40
r           risk-free rate: 0.1
sigma       volatility of the stock: 0.2
T           time to maturity in days: 0.501369863014
p_BS        : [ 0.80963075828
 ]
>>>
```

And then specifying a list of values for the underlying basing on the idea that the price of the stock could go up or down.

```
>>> runfile('/Users/FADE/Documents/Python/Black Scholes Put.py', wdir=r'/Users/FADE/Documents/Python')
Enter current stock price:
30,31,32,33,34,40,41,42,43,44,45,46,47,48
Enter strike price:
40
Enter risk-free rate:
10
Enter volatility rate:
20
Enter time to maturity in days:
183
S           stock price : (30, 31, 32, 33, 34, 40, 41, 42, 43, 44, 45, 46, 47, 48)
K           strike price: 40
r           risk-free rate: 0.1
sigma       volatility of the stock: 0.2
T           time to maturity in days: 0.501369863014
p_BS        : [ 8.13621981429
7.20474412826
6.30862809519
5.45788773983
4.66225733925
1.36109646018
1.05667916774
0.80963075828
0.612467368402
0.457627627745
0.337885400124
0.246636832166
0.178068137732
0.127222733204
 ]
```




## 3.4 European Call Option Using Python with Varying Time Intervals

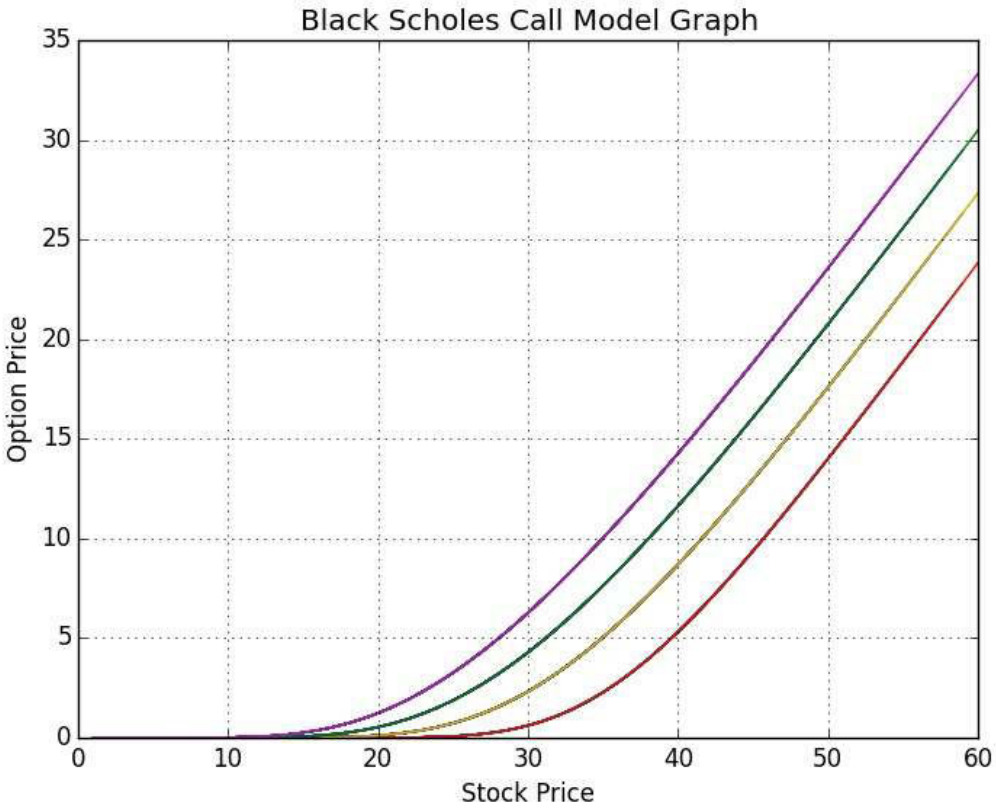
We illustrate an example in python below;

```
>>> runfile('/Users/FADE/Documents/Python/Black Scholes Call.py', wdir=r'/Users/FADE/Documents/Python')
Enter current stock price:
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38
,39,40,41,42,43,44,45,46,47,48,49,50,52,53,54,55,56,57,58,59,60
Enter strike price:
40
Enter risk-free rate:
10
Enter volatility rate:
20
Enter time to maturity in days:
5
S           stock price : (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22
, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48
, 49, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60)
K           strike price: 40
r           risk-free rate: 0.1
sigma       volatility of the stock: 0.2
T           time to maturity: 5
c_BS        : [ [1.7600380439681687e-73]
[1.7600380439681687e-73, 9.5745082264557047e-49]
[1.7600380439681687e-73, 9.5745082264557047e-49, 1.1309135333484187e-36]
[1.7600380439681687e-73, 9.5745082264557047e-49, 1.1309135333484187e-36, 3.5423696057440931e-29]
[1.7600380439681687e-73, 9.5745082264557047e-49, 1.1309135333484187e-36, 3.5423696057440931e-29, 5.67718
22702025543e-24]
```

.
.
.
.
.
.
.

.
.

```
[2.4000007295184882e-17, 1.8274704894263726e-11, 1.3202656068411581e-08, 7.8024839109807588e-07, 1.32310
0321290986e-05, 0.00010801308740633901, 0.0005501170375417632 4, 0.0020238687206065482, 0.005885553584928
7877, 0.014349798662292454, 0.030540815853076664, 0.058392051149904867, 0.10242374336780691, 0.167451425
41959346, 0.25828028943981018, 0.37942880274273239, 0.53490851522009253, 0.72807143162774723, 0.96152443
448508951, 1.2371026804972498, 1.5558901195225374, 1.9182743125782942, 2.3240235455152458, 2.77237601678
83835, 3.2621330233354744, 3.7917501953618746, 4.3594227253394724, 4.9631621042734295, 5.600863100792389
9, 6.27036062086089, 6.9694767138511136, 7.6960583969292244, 8.4480072050895778, 9.2233014836314737, 10.
020012460741651, 10.836315099726775, 11.670494655916606, 12.520949768936877, 13.38619281869093, 14.26484
8170958945, 15.155648841128201, 16.057432015207233, 16.969133787456705, 17.889783404171226, 18.818497243
176235, 19.754472707855673, 20.696982172162151, 21.645367078141483, 22.599032259069435, 23.5574405384277
34, 25.486597404428622, 26.456517386623165, 27.429514720097103, 28.405272353212592, 29.383505577904852,
30.363958858465161, 31.346402940326417, 32.330632220934412, 33.316462364356354]
 ]
>>>
```



# 4. Conclusions

## 4.1 Comments

For the Black Scholes model, the varying stock prices for an option lead to option prices that converge toward a hockey stick over time till the time to maturity.

## 4.2 Appendix

### 4.2.1 European Call Option for Black Scholes

```
import math
import matplotlib.pyplot as plt
import scipy.stats


def d1(S, K, r, sigma, T):
    return (math.log(S /float(K)) + (r + sigma**2 / 2) * T) / (sigma *
    math.sqrt(T))

def d2(S, K, r, sigma, T):
    return d1(S, K, r, sigma, T) - (sigma*math.sqrt(T))

def CallOption(S, K, r, sigma, T):
    return S * scipy.stats.norm.cdf(d1(S, K, r, sigma, T)) - K *
    math.exp(-r * T) * scipy.stats.norm.cdf(d2(S, K, r, sigma, T))

def plotData(S,K,r,sigma,T):
    list_S = []
    list_K = []
    list_plot_data_S = []
    list_plot_data_K = []

    if isinstance(S,tuple):
        list_S = list(S)
    elif isinstance(S,int):
        list_S.append(S)
    if isinstance(K,tuple):
        list_K = list(K)
    elif isinstance(K,int):
        list_K.append(K)


    for k in list_K:
        for s0 in list_S:



            list_plot_data_S.append(s0)
            list_plot_data_K.append(k)
            c_BS.append(CallOption(s0, k, r, sigma, T))
    plt.plot(list_plot_data_S,c_BS)



print "Enter current stock price:"
```

```
S = input()
print "Enter strike price:"
K= input()
print "Enter risk-free rate:"
r = float(input())/100
print "Enter volatility rate:"
sigma = float(input())/100
print "Enter time to maturity in days:"
T = float(input())/365
c_BS = []



print "S\tstock price :", S
print "K\tstrike price:", K
print "r\trisk-free rate:", r
print "sigma\tvolatility of the stock:", sigma
print "T\ttime to maturity in days:", T

plotData(S,K,r,sigma,T)
plt.xlabel("Stock Price")
plt.ylabel("Option Price")
plt.title("Black Scholes Euro Call Model Graph")
plt.grid(True)

print "c_BS\t: [",
for c in c_BS:
    print c
print " ]"


plt.show()
```

### 4.2.2 European Put Option for Black Scholes

```
import math
import matplotlib.pyplot as plt
import scipy.stats



def d1(S, K, r, sigma, T):
    return (math.log(S / float(K)) + (r + sigma**2 / 2) * T) / (sigma *
math.sqrt(T))

def d2(S, K, r, sigma, T):
    return d1(S, K, r, sigma, T) - (sigma*math.sqrt(T))

def PutOption(S, K, r, sigma, T):
    return (K * math.exp(-r * T) * scipy.stats.norm.cdf(-d2(S, K, r,
sigma, T))) - (S * scipy.stats.norm.cdf(-d1(S, K, r, sigma, T)))

def plotData(S,K,r,sigma,T):
    list_S = []
    list_K = []
```

```
    list_plot_data_S = []
    list_plot_data_K = []

    if isinstance(S,tuple):
        list_S = list(S)
    elif isinstance(S,int):
        list_S.append(S)
    if isinstance(K,tuple):
        list_K = list(K)
    elif isinstance(K,int):
        list_K.append(K)


    for k in list_K:
        for s in list_S:



            list_plot_data_S.append(s)
            list_plot_data_K.append(k)
            p_BS.append(PutOption(s, k, r, sigma, T))
    plt.plot(list_plot_data_S,p_BS,'-r')



print "Enter current stock price:"
S = input()
print "Enter strike price:"
K= input()
print "Enter risk-free rate:"
r = float(input())/100
print "Enter volatility rate:"
sigma = float(input())/100
print "Enter time to maturity in days:"
T = float(input())/365
p_BS = []


print "S\tstock price :", S
print "K\tstrike price:", K
print "r\trisk-free rate:", r
print "sigma\tvolatility of the stock:", sigma
print "T\ttime to maturity in days:", T

plotData(S,K,r,sigma,T)
plt.xlabel("Stock Price")
plt.ylabel("Option Price")
plt.title("Black Scholes Put Model Graph")
plt.grid(True)

print "p_BS\t: [",
for p in p_BS:
    print p
print " ]"


plt.show()
```

### 4.2.3 European Call Option for Black Scholes with Varying Time

```
import math
import matplotlib.pyplot as plt
import scipy.stats

def d1(S, K, r, sigma, T):
    return (math.log( S / float(K)) + (r + sigma**2/2) * T) / (sigma *
math.sqrt(T))

def d2(S, K, r, sigma, T):
    return d1(S, K, r, sigma, T) - (sigma*math.sqrt(T))

def CallOption(S, K, r, sigma, T):
    return S * scipy.stats.norm.cdf(d1(S, K, r, sigma, T)) - K *
math.exp(-r * T) * scipy.stats.norm.cdf(d2(S, K, r, sigma, T))

def plotData(S,K,r,sigma,T):
    list_S = []
    list_plot_data_S = []

    if isinstance(S,tuple):
        list_S = list(S)
    elif isinstance(S,int):
        list_S.append(S)


    print "c_BS\t: [",


    for x in range(1,T):
        c_BS = []
        list_plot_data_S = []
        for s in list_S:
            c_BS.append(CallOption(s, K, r, sigma, x))
            list_plot_data_S.append(s)
            plt.plot(list_plot_data_S,c_BS,'-')
            print c_BS
    print " ]"


print "Enter current stock price:"
S = input()
print "Enter strike price:"
K= input()
print "Enter risk-free rate:"
r = float(input())/100
print "Enter volatility rate:"
sigma = float(input())/100
print "Enter time to maturity in days:"
T = input()
c_BS = []


print "S\tstock price :", S
print "K\tstrike price:", K
print "r\trisk-free rate:", r
print "sigma\tvolatility of the stock:", sigma
```

```
print "T\ttime to maturity:", T

plotData(S,K,r,sigma,T)
plt.xlabel("Stock Price")
plt.ylabel("Option Price")
plt.title("Black Scholes Call Model Graph")
plt.grid(True)
plt.show()
```

### 4.2.4 European Put Option for Black Scholes with Varying Time

```
import math
import matplotlib.pyplot as plt
import scipy.stats



def d1(S, K, r, sigma, T):
    return (math.log(S / float(K)) + (r + sigma**2 / 2) * T) / (sigma *
math.sqrt(T))

def d2(S, K, r, sigma, T):
    return d1(S, K, r, sigma, T) - (sigma*math.sqrt(T))

def PutOption(S, K, r, sigma, T):
    return (K * math.exp(-r * T) * scipy.stats.norm.cdf(-d2(S, K, r,
sigma, T))) - (S * scipy.stats.norm.cdf(-d1(S, K, r, sigma, T)))

def plotData(S,K,r,sigma,T):
    list_S = []
    list_plot_data_S = []

    if isinstance(S,tuple):
        list_S = list(S)
    elif isinstance(S,int):
        list_S.append(S)


    print "p_BS\t: [",


    for x in range(1,T):
        p_BS = []
        list_plot_data_S = []
        for s in list_S:
            p_BS.append(PutOption(s, K, r, sigma, x))
            list_plot_data_S.append(s)
            plt.plot(list_plot_data_S,p_BS,'-')
            print p_BS
    print " ]"


print "Enter current stock price:"
```

```
S = input()
print "Enter strike price:"
K= input()
print "Enter risk-free rate:"
r = float(input())/100
print "Enter volatility rate:"
sigma = float(input())/100
print "Enter time to maturity in days:"
T = input()
p_BS = []


print "S\tstock price :", S
print "K\tstrike price:", K
print "r\trisk-free rate:", r
print "sigma\tvolatility of the stock:", sigma
print "T\ttime to maturity:", T

plotData(S,K,r,sigma,T)
plt.xlabel("Stock Price")
plt.ylabel("Option Price")
plt.title("Black Scholes Call Model Graph")
plt.grid(True)
plt.show()
```

## 5. References

[1] Röman, Jan, *Lecture Notes For Analytical Finance 1 (2014)*

[2] Hull J.C, *Options, Futures & Other Derivatives, 8th Edition, Prentice Hall Boston (1946), 315-316*