Black and Scholes formula

BS model assumes the distribution of stock as lognormal. In particular, it writes

$$\ln \frac{S(T)}{S(0)} \sim \mathcal{N}((r - \frac{1}{2}\sigma^2)T, \sigma^2 T)$$

with respect to risk neutral measure. In the above, the parameters stand for

- $S(0)$: The initial stock price
- $S(T)$: The stock price at T
- r: interest rate
- σ: volatility

The call and put price with maturity T and K will be known as C_0 and P_0 given as below:

$$C_0 = \mathbb{E}[e^{-rT}(S(T) - K)^+] = S_0 \Phi(d_1) - Ke^{-rT}\Phi(d_2),$$

and

$$P_0 = \mathbb{E}[e^{-rT}(S(T) - K)^-] = Ke^{-rT}\Phi(-d_2) - S_0\Phi(-d_1),$$

where d_i are given as

$$d_1 = \frac{(r + \frac{1}{2}\sigma^2)T - \ln \frac{K}{S_0}}{\sigma\sqrt{T}},$$

and

$$d_2 = \frac{(r - \frac{1}{2}\sigma^2)T - \ln \frac{K}{S_0}}{\sigma\sqrt{T}},$$

ex

Verify put-call parity:

$$C_0 - P_0 = S(0) - e^{-rT}K.$$

Black and Scholes formula for European call and put is coded and an example will be demonstrated

In [1]:

```
#BS formula is given here
import numpy as np
import scipy.stats as ss
import time
import math
```

In [2]:

```
def d1f(St, K, t, T, r, sigma):
    ''' Black-Scholes-Merton d1 function.
        Parameters see e.g. BSM_call_value function. '''
    d1 = (math.log(St / K) + (r + 0.5 * sigma ** 2)
          * (T - t)) / (sigma * math.sqrt(T - t))
    return d1
```

In [3]:

```
def BSM_call_value(St, K, t, T, r, sigma):
    ''' Calculates Black-Scholes-Merton European call option value.

    Parameters
    ==========
    St : float
        stock/index level at time t
    K : float
        strike price
    t : float
        valuation date
    T : float
        date of maturity/time-to-maturity if t = 0; T > t
    r : float
        constant, risk-less short rate
    sigma : float
        volatility

    Returns
    =======
    call_value : float
        European call present value at t
    '''
    d1 = d1f(St, K, t, T, r, sigma)
    d2 = d1 - sigma * math.sqrt(T - t)
    call_value = St * ss.norm.cdf(d1) - math.exp(-r * (T - t)) * K * ss.norm.cdf
(d2)
    return call_value
```

Ex.

Find BS Call price for the given parameters below

In [4]:

```
#An example is given here
S0 = 100.0
K = 110.0
r=0.0475
sigma = 0.20
T = 1.
Otype='C' #Call
```

In [5]:

```
#demonstration for call evaluation
BSM_call_value(S0, K, 0, T, r, sigma)
```

Out[5]:

5.943273183452838

In [6]:

```
def BSM_put_value(St, K, t, T, r, sigma):
    ''' Calculates Black-Scholes-Merton European put option value.

    Parameters
    ==========
    St : float
        stock/index level at time t
    K : float
        strike price
    t : float
        valuation date
    T : float
        date of maturity/time-to-maturity if t = 0; T > t
    r : float
        constant, risk-less short rate
    sigma : float
        volatility

    Returns
    =======
    put_value : float
        European put present value at t
    '''
    put_value = BSM_call_value(St, K, t, T, r, sigma) \
        - St + math.exp(-r * (T - t)) * K
    return put_value
```

Ex Find Put value with the same parameters above

In [7]:

```
#demonstration for call evaluation
BSM_put_value(S0, K, 0, T, r, sigma)
```

Out[7]:

```
10.840425228041752
```

Next, we shall write a file to include the above functions for later use.

In [8]:

```
%reset -f
from BSM_option_valuation import *
S0 = 100.0
K = 110.0
r=0.0475
sigma = 0.20
t = 0.
T = 1.
icall = BSM_call_value(S0, K, 0, T, r, sigma)
iput = BSM_put_value(S0, K, 0, T, r, sigma)
print('call is ' + repr(icall) + ' and put is ' + repr(iput))
```

```
call is 5.943273183452838 and put is 10.840425228041752
```