report

May 14, 2019

1 Report TP 9, Dimitris Proios

In this TP we compared two options evaluation models. The black sholes and the binomial model under steady parameters both converge to the same value. This can be prooved mathematically since the asymtotic of binomial distribution on which Binomial model is based on approaches the lognormal distribution used in Black Scholes.

1) Below we we see the implemenations of both Black Scholes. At the end we calculate the value of the cal; option

[366]:

```
import numpy as np
import scipy.stats as si

def Black_Scholes_Algo(S, K, T, r, sigma):
            #S: spot price
            #K: strike price
            #T: time to maturity
            #r: interest rate
            #sigma: volatility of underlying asset
            d1 = (np.log(S / K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.
 ↪sqrt(T))
            d2 = (np.log(S / K) + (r - 0.5 * sigma ** 2) * T) / (sigma * np.
 ↪sqrt(T))

            call = (S * si.norm.cdf(d1, 0.0, 1.0) - K * np.exp(
            -r * T) * si.norm.cdf(d2, 0.0, 1.0))
            return call
# current stock price
S0 = 100; S = S0
# initial time

t0 = 0
# time until option experiance,
call_maturiy_T = 1;
t = call_maturiy_T
#steady; risk-free-rate
r=0.05
```

```
# K = Options striking pruce
K = 120
volatility_sigma = 0.2; s = volatility_sigma
C = Black_Scholes_Algo(S,K,t,r,s)
print("EX 1.1: the value of this call at t = 0 IS c = ", C)
```

```
EX 1.1: the value of this call at t = 0 IS c =  3.247477416560816
```

```
Not the that the value takes into account the following parameters:
    """
    N = number of binomial iterations
     / tree deth
    S0 = initial stock price
    u = factor change of upstate
    /volatility
    r = risk free interest rate per annum
    K = strike price"""
```

2) Below we we see the implemenations of both algorithms.

[367]:
```
# binomial tree to determine the call option value.
def binomial_model(N, S0, s, r, K, t_maturity):
    t=t_maturity/N
    u = np.e**(s*np.sqrt(t))
    d = np.e**(-s*np.sqrt(t))
    p = (np.e**(r*t) - d ) / (u-d);
    stock = np.zeros([N + 1, N + 1])

    stock[0][0] = S0

    for i in range(1, N + 1):
        for j in range(0, i + 1):
            stock[j, i] = S0 * (u ** j) * (d ** (i-j))

    option = np.zeros([N + 1, N + 1])
    stock_max = stock[:,N]
    for j in range(N+1):
        option[:, j] = np.maximum(
            np.zeros(N+1),
            (stock_max[j] - K)
        )

    for i in range(N-1, -1, -1):
        for j in range(0,i+1):
            option[i,j] = (np.e**(-r * t) )* (p * option[i+1,j+1] + (1 - p) *
 ↪option[i+1,j])
```

```python
    return option[0][0]


# current stock price
S0 = 100
# initial time
t0 = 0
# time until option experiance,
call_maturiy_T = 1;
#steady; risk-free-rate
r=0.05
# K = Options striking pruce
K = 120
s = 0.2
max_tree_depth = 300
# binomial_model(max_tree_depth, S0, s, r, K, call_maturiy_T)
tree_depths = [ x for x in range(1, max_tree_depth)]
models = [
         binomial_model(N, S0, s, r, K, call_maturiy_T)
         for N in tree_depths
]
```

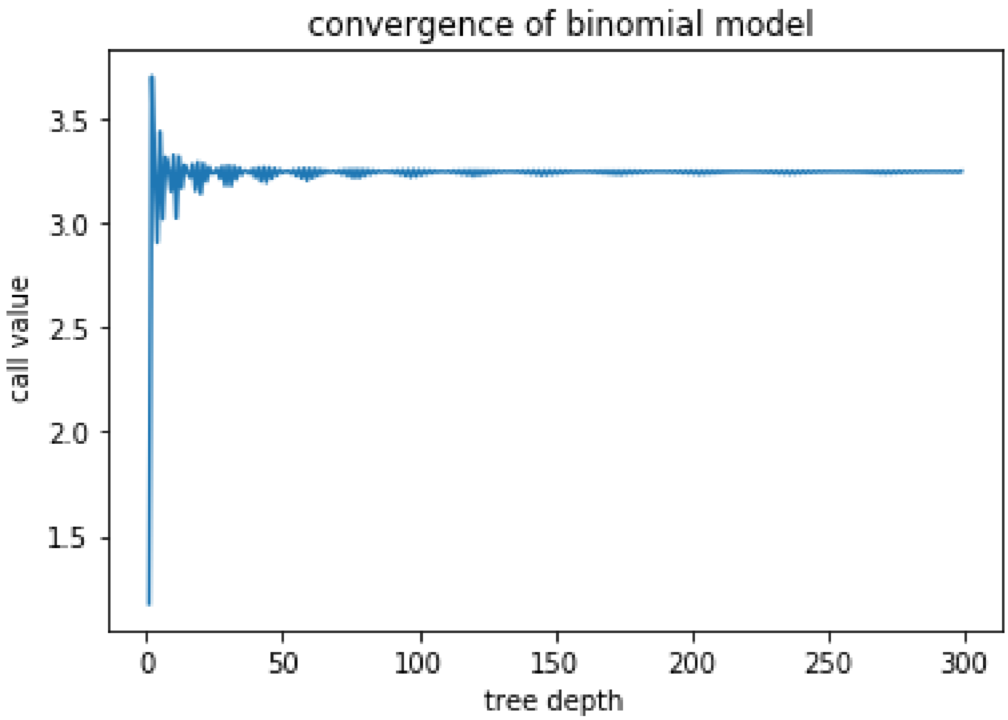
3) How deep should be the tree in order to get a reasonable approximation?

According to theory the binomial model the binomial converges on the Black–Scholes formula value as the number of time steps increases. The following convergence can be seen below too in the experiment, we ccan see that after depth 50 we don't have significant fluctuations.

4) Plot the evolution of the estimated value of the call option as a function of the binomial tree depth.

[368]:
```python
import matplotlib.pyplot as plt
plt.plot( tree_depths, models)
plt.xlabel("tree depth")
plt.ylabel(" call value")
plt.title(" convergence of binomial model")
```

[368]:
```
Text(0.5, 1.0, ' convergence of binomial model')
```


convergence of binomial model
3.5
3.0
2.5
2.0
1.5
call value
0
50
100
150
200
250
300
tree depth