Computational Experiments with the CRR Binomial Options Pricing Model

(1) Siddharth Chakravarthy and (2) Anirudh Madhusudan

Division of Work

Tasks carried out by student (1)

Worked on the algorithmic and program formulation of question 1
Did several tests to improve runtime of problem 1
Black-Scholes Formula Function Block
Q3. All computations for q = 0
Q4. All computations for q = 0.04
Half of the duties for report writing
Printing - classroom submission

Tasks carried out by student (2)

Worked on the algorithmic and program formulation of question 1
Did several tests to improve runtime of problem 3 and 4
Black-Scholes Formula Tests & Plots
Q3. All computations for q = 0.04
Q4. All computations for q = 0.08
Half of the duties for report writing
Aggregate files - online submission

1. Cox-Ross-Rubinstein Binomial Option Pricing Model

The **Cox-Ross-Rubinstein Binomial** (CRR) Option Pricing Model was conceptualized by John Carrington Cox, Stephen Ross and Mark Edward Rubinstein. One of the important assumptions of this model is that the world is neutral toward risk and all investors involved are indifferent to risk. As a consequence of being in a risk-neutral world, the returns are expected to be equal to the risk-free interest rate. The CRR binomial model provides for a methodical approach to compute the option price at specified time, thereby creating a tree like structure of possible values at various nodes of the tree.

In order to experiment with the CRR Binomial Model, an exhaustive function that takes in multiple parameters has to be synthesized. The format for this function is given as:

CRRBinomial(Option, K, T, S0, σ, r, q, N, Exercise)

Option = C for calls and P for puts, K is the strike, T is the time to maturity, S0 is the initial stock price, σ is the volatility, r is the continuous compounding risk free interest rate, q is the continuous dividend yield, N is the number of time steps, Exercise = A for American options and E for European options.

1.1 Implementation

From the given arguments for the functional block, we compute the u, d and p values to aid the computation of the option prices.

American Option

An array having N+1 length is initialized to payoff values at the maturity. Similarly, another array having the same size is initialized to the stock price at maturity. These arrays are iteratively updated till the put option value at the initial time is computed.

European Option

The initial step for European Options is similar to that of American Options where an array having N+1 length is initialized to payoff values at the maturity. The expected payoff is updated for the prior steps until the option price value at the initial time is computed.

Reduction in Computational Time

During the preliminary analysis, it was found that using several arrays to create the tree increased the computational run-time. Therefore, the process was simplified by incorporating just two arrays that work toward creating the tree. Standard libraries such as numpy and scipy were used to expedite the run-time.

One of the main challenges in reducing the run-time was to optimize the algorithm such that the exercise boundary is obtained accurately with minimum number of steps. This was achieved by calculating the difference of option price and its intrinsic value with a stock-price step size of 1. When the difference shifts from lesser than 0.005 to greater than 0.005, the corresponding dollar range is chosen and the accurate exercise boundary is calculated within that range with even more granularity of the stock price with step size 0.01. For example if they exercise boundary is 91.34, the algorithm jumps from low to high stock price and it identifies the range of 91 to 92 where the difference shifts from lesser than, to greater than 0.005. Consequently, the search is carried out within 91-92 with stock price steps of 0.01 till 9.134 is achieved.

2. Black-Scholes Model v/s CRR Binomial Model for the European Call Option

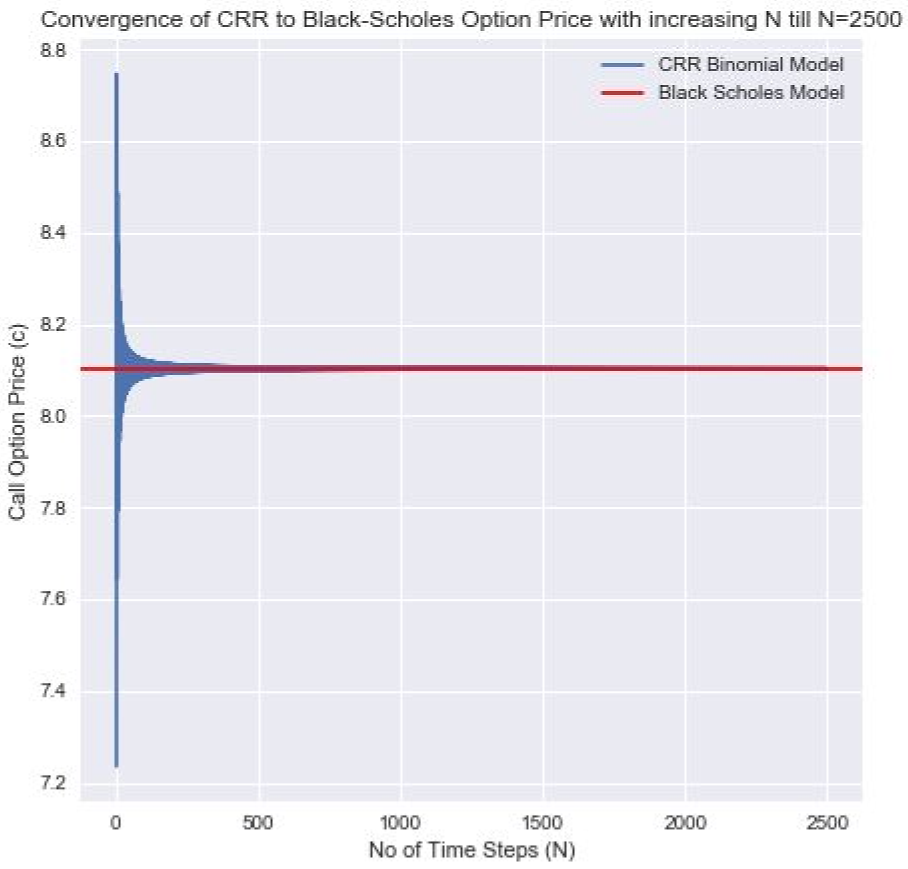
Black-Scholes model is used to calculate the european option price for calls and puts. Some of the assumptions of this model are that there are no dividends paid during the lifetime of the option and the underlying returns for the options follow a normal distribution.The model was developed by three eminent economists Fischer Black, Myron Scholes and Robert Merton.

For a 1-year european call option with K = 100, S0 = 100, r = 0.05, q = 0.04, σ = 0.2, the option price has been found to be 8.1026 dollars. Comparing the option prices obtained from the Black-Scholes Model with the CRR Binomial Model at a large value of 'N', a convergence has been observed. The convergence can be explained, since the CRR model assumes that the price movements have a binomial distribution, and this distribution approaches the underlying lognormal distribution of the Black-Scholes Model.

Two values of 'N', i.e. N=200 and N=2500 have been plotted. This is done in order to observe the graphs with more granularity for N=200, and have a holistic view for N=2500. Figures 1 and 2 show the aforementioned convergence.

Figure 1

Figure 2



3. Characterization of the American Put Option

3.1 Number of Steps Required (q=0 and q = 0.04)

Step sizes with intervals of 100 have been considered. For steps beyond 1000 until 5000, step size of 1000 has been considered. From the preliminary analysis that was conducted, it was observed that for large values of 'N', convergence has been observed in the value of the option price.

The objective of this implementation is to obtain the minimum number of time steps such that the option price error is less than 10^{-3} with respect to a high time step (N=5000) option price. For time period ranging from 1 to 12 months, these minimum time step values are computed. Table 1 illustrates the minimum time steps required to achieve the accuracy of 10^{-3} and the corresponding option prices for different time periods and for q=0 and q = 0.04.

For both q = 0 and q = 0.04 It has been observed that as the duration of the maturity increases, the corresponding number of time steps required to achieve the accuracy also increases. A similar comment can be made about the put price which increases as the time to maturity increases. Table 1 and Table 2 shows this behavior for q = 0 and q = 0.04 respectively.

Table 1 q=0

Month	Number of Time Steps	Put Price
1	400	2.126
2	500	2.911
3	500	3.478
4	600	3.934
5	600	4.319
6	600	4.654
7	600	4.952
8	700	5.220
9	700	5.464
10	700	5.688
11	700	5.896
12	700	6.089

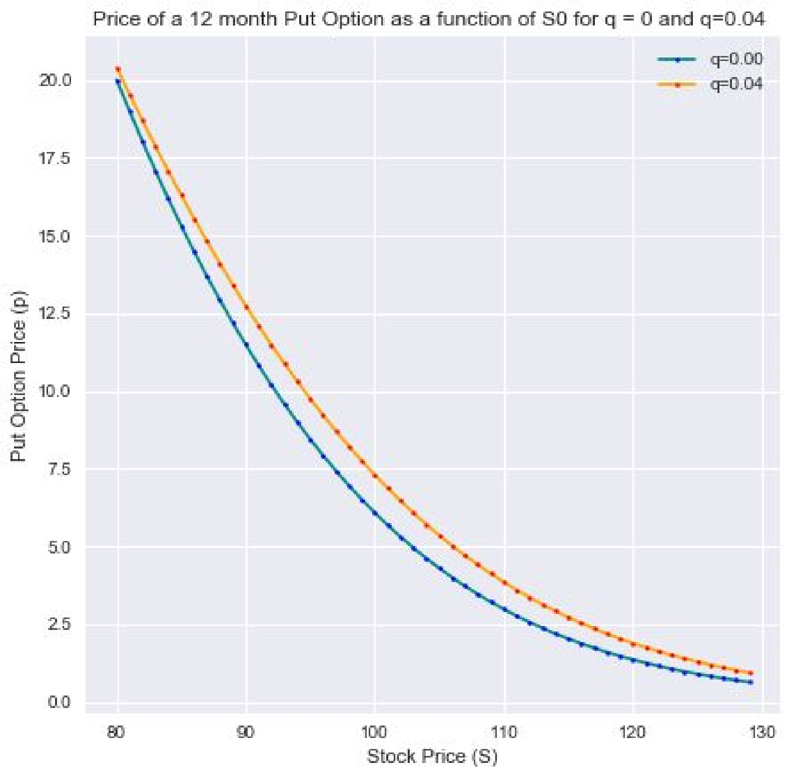
3.2 Put Price vs. S0 (q=0 and q = 0.04)

For a 12-month put option, the option price is calculated as a function of S0.To study the pattern, [80,130] are the range of S0 values under study. The trend shows a decreasing value of put option price with increasing value of stock price. Figure 3 shows the variation of put price as a function of S0 for both q = 0 and q = 0.04.

Table 2 q = 0.04

Month	Number of Time Steps	Put Price
1	500	2.257
2	600	3.163
3	700	3.843
4	800	4.407
5	900	4.895
6	900	5.330
7	900	5.724
8	1000	6.086
9	1000	6.421
10	1000	6.734
11	1000	7.027
12	1100	7.304

Figure 3



3.3 Critical Stock Price for Early Exercise (q=0 and q = 0.04)

With a step size of 0.01 of S0, the difference between the put price and its intrinsic value are computed. When the highest stock price for which the difference between the stock price and intrinsic value is lesser than 0.005, the S*(i) value is found. The optimization of the code is explained in the runtime improvement section. The critical stock price for different maturity periods are computed ranging from 1 to 12 months. A high time step value of N=3000 was chosen to achieve higher put price accuracy. Tables 3 and 4 represent the critical stock prices for q = 0 and q = 0.04 respectively. Figure 4 compares the critical stock price for q = 0 and q = 0.04.

Table 3: q = 0

Month	Critical Stock Price
1	91.30
2	88.91
3	87.34
4	86.17
5	85.23
6	84.44
7	83.77
8	83.18
9	82.66
10	82.20
11	81.77
12	81.38

Figure 4

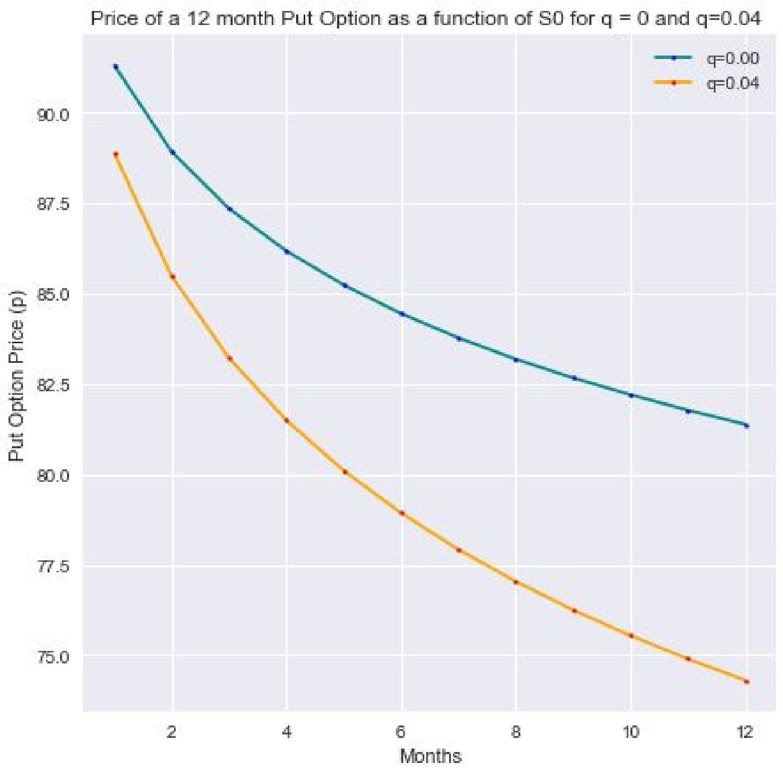


Table 4: q = 0.04

Month	**Critical Stock Price**
1	88.87
2	85.47
3	83.21
4	81.49
5	80.10
6	78.93
7	77.92
8	77.04
9	76.25
10	75.54
11	74.90
12	74.31

3.4 Continuous Dividend Yield and the Early Exercise Boundary (Puts)

There exists a direct relationship between the dividend D and the put price p. That is, as D increases, p also increases correspondingly. Also, an increase in the value of q implies that that the dividends increase. The results obtained in the analysis are in complete agreement with the aforementioned statement.

In order to understand the effect of yield on the early exercise boundary, it is suffice to understand how the yield affects the underlying stock price. The stock is expected to have higher put premium. As a result of the increased premium on the put, the reduction observed in the stock price will be larger before the put price and the intrinsic value become equal. As a consequence, the put's early exercise boundary should decrease.

4. Characterization of the American Call Option

4.1. Number of Steps Required (q=0.04 and q = 0.08)

Step Step sizes with intervals of 100 have been considered. For steps beyond 1000 until 5000, step size of 1000 has been considered. From the preliminary analysis that was conducted, it was observed that for large values of 'N', convergence has been observed in the value of the option Price. The objective of this implementation is to obtain the minimum number of time steps such that the error is less than 10^{-3} with respect to a *high time step(N=5000)* option price. For time period ranging from 1 to 12 months, this minimum time step value with a low error(high accuracy) is computed. The table below illustrates the number of minimum time steps required to achieve the accuracy of 10^{-3} and the corresponding option price for different time periods. It has been observed that as the duration of the maturity increases, the corresponding number of time steps required to achieve the accuracy also increases. A similar comment can be made about the call price which increases as the time to maturity increases. Table 5 and Table 6 shows this behavior for q = 0.04 and q = 0.08 respectively.

Table 5: q = 0.04

Month	Number of Time Steps	Call Price
1	600	2.335
2	700	3.314
3	900	4.066
4	1000	4.700
5	1100	5.257
6	1100	5.760

7	1200	6.220
8	1300	6.647
9	1300	7.046
10	1400	7.422
11	1400	7.778
12	1400	8.116

Table 6: q = 0.08

Month	**Number of Time Steps**	**Call Price**
1	400	2.186
2	500	3.022
3	600	3.635
4	600	4.132
5	700	4.556
6	700	4.927
7	700	5.259
8	700	5.559
9	800	5.833
10	800	6.087
11	800	6.321
12	800	6.540

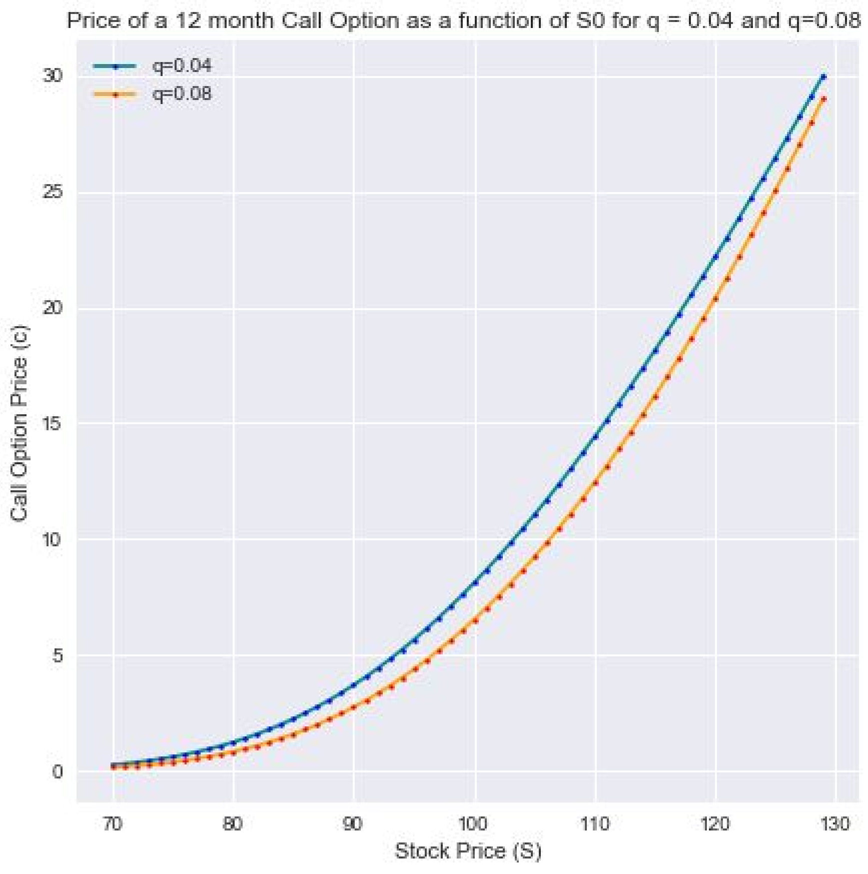
4.2 Call Price vs. S0 (q=0.04 and q = 0.08)

For a 12-month call option, the option price is calculated as a function of S0. To study the pattern, [80,130] are the range of S0 values under study. The trend shows a increasing value of call option price with increasing value of stock price. Figure 5 shows the variation of put price as a function of S0 for both q = 0.04 and q = 0.08.

4. 3 Critical Stock Price for Early Exercise S0 (q=0.04 and q = 0.08)

With a step size of 0.01 of S0, the difference between the call price and its intrinsic value are computed. When the lowest stock price for which the difference between the stock price and intrinsic value is lesser than 0.005, the S*(i) value is found. The optimization of the code is explained in the runtime improvement section. The critical stock price for different maturity periods are computed ranging from 1 to 12 months. A high time step value of N=3000 was chosen to achieve higher call price accuracy. Tables 7 and 8 represent the critical stock prices for q = 0.04 and q = 0.08 respectively. Figure 6 compares the critical stock price for q = 0.04 and q = 0.08.

Figure 5



4.4 Continuous Dividend Yield and the Early Exercise Boundary (Calls)

As discussed before, there exists a direct relationship between D, which is the value of the dividend and q, which is the continuous yield. According to Investopedia, with an increase in the value of the yield, the premium of the stock will reduce. Therefore, the early exercise boundary for call should decrease.

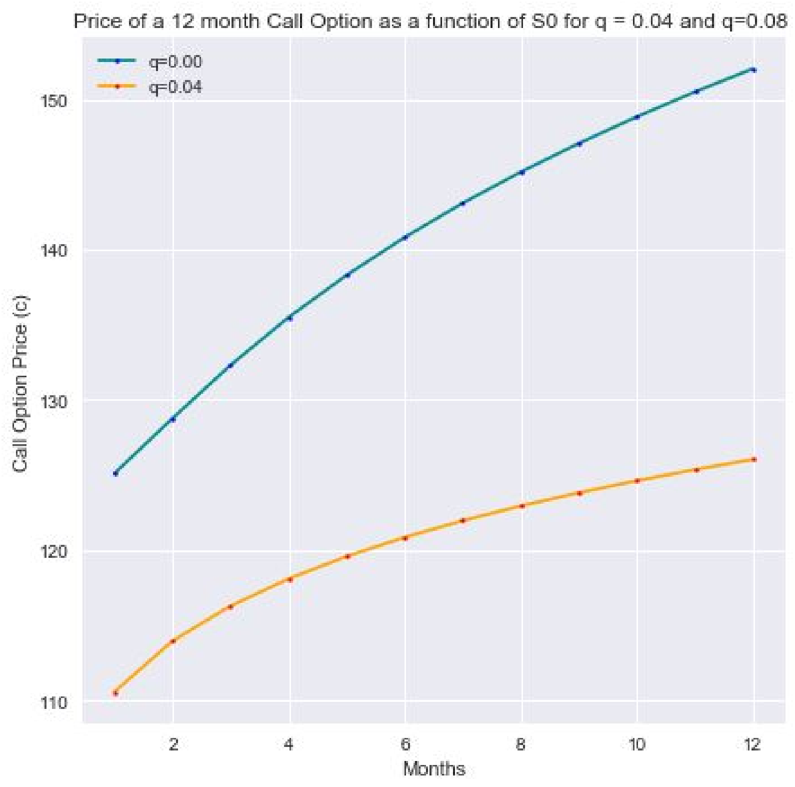
Table 7: q = 0.04

Month	Critical Stock Price
1	125.09
2	128.76
3	132.29
4	135.48
5	138.29
6	140.81
7	143.09
8	145.16
9	147.07
10	148.84
11	150.50
12	152.04

Table 8: q = 0.08

Month	Critical Stock Price
1	110.55
2	113.94
3	116.27
4	118.08
5	119.57
6	120.84
7	121.95
8	122.93
9	123.81
10	124.61
11	125.35
12	126.03

Figure 6



CRRBModel

April 19, 2018

0.0.1 1. Importing Libraries

```
In [1]:
        import numpy as np
        import matplotlib.pyplot as plt
        import time
        import seaborn as sns
        from scipy.stats import norm
```

0.0.2 2. Defining CRR Binomial Model Function

```
In [2]: def CRRBionomial(Option, K, T, S0, sigma, r, q, N, Exercise):

            start = time.clock()

            #Calculated Parameters
            delta=T/N
            s = np.zeros((N+1,2))
            u=np.e**(sigma*np.sqrt(delta))
            d=np.e**(-sigma*np.sqrt(delta))
            p=(np.e**((r-q)*delta)-d)/(u-d)

            if (Option=='P'):
                if (Exercise=='A'):

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

                        for i in range(N):
                            for j in range(N-i):
                                s[j][0] = S0*(u**(N-1-i-j))*(d**(j))
                                s[j][1] = max(K - max(s[j][0], 0), (np.e**(-r * delta))*(p*s[j]



                if Exercise== 'E':
```

```
            #Finding the stock price at node (i,j)
            for i in range(N+1):
                s[i][0] = S0*u**(N-i)*d**i
                s[i][1] = max(K-s[i][0],0)


            #Backward Induction
            for i in range(N):
                for j in range(N-i):
                    z2 = (1-p)*s[j+1][1]
                    x = np.e**(-r*delta)
                    z1 = p*s[j][1]
                    s[j][1] = x*(z1+z2)

    #CALL OPTION
    if Option == 'C':

        #American Option
        if Exercise== 'A':

            #Finding the stock price at node (i,j)
            for i in range(N+1):
                s[i][0] = S0*u**(N-i)*(d**i)
                s[i][1] = max(s[i][0]-K,0)


            for i in range(N):
                for j in range(N-i):
                    s[j][0] = S0*(u**(N-1-i-j))*(d**(j))
                    s[j][1] = max(max(s[j][0] - K, 0), (np.e**(-r * delta))*(p*s[j]


        #European Option
        if Exercise== 'E':

            #Finding the stock price at node (i,j)
            for i in range(N+1):
                s[i][0] = S0*u**(N-i)*(d**i)
                s[i][1] = max(s[i][0]-K,0)


            #Backward Induction
            for i in range(N):
                for j in range(N-i):
                    z2 = (1-p)*s[j+1][1]
                    x = np.e**(-r*delta)
                    z1 = p*s[j][1]
                    s[j][1] = x*(z1+z2)
price = s[0][1]
timetaken = time.clock() - start

return (price,timetaken)
```

0.0.3 3. Defining Black-Scholes Function

```
In [3]: def BlackScholesFormula (option, K, T, S0, sigma, r, q):

            d1 = (np.log(S0/K)+(r-q+0.5*sigma**2)*T)/(sigma*T**0.5)
            d2 = d1 - sigma*T**0.5

            if option == 'C':
                price = (S0*np.e**(-q*T)*norm.cdf(d1)) - K*np.e**(-r*T)*norm.cdf(d2)

            if option == 'P':
                price = -(S0*np.e**(-q*T)*norm.cdf(-d1)) + K*np.e**(-r*T)*norm.cdf(-d2)

            return price
```

0.0.4 4. Computing European Call Option Price using Black-Scholes Formula

1-year European call option, K = 100 , S0 = 100, r = 0.05, q = 0.04, = 0.02

```
In [4]: BSprice = BlackScholesFormula("C",100,1,100,0.2,0.05,0.04)
        print (BSprice)
```

```
8.10264353446
```

0.0.5 5. Computing European Call Option Price Using CRR Binomial Model for n = [1,2500]

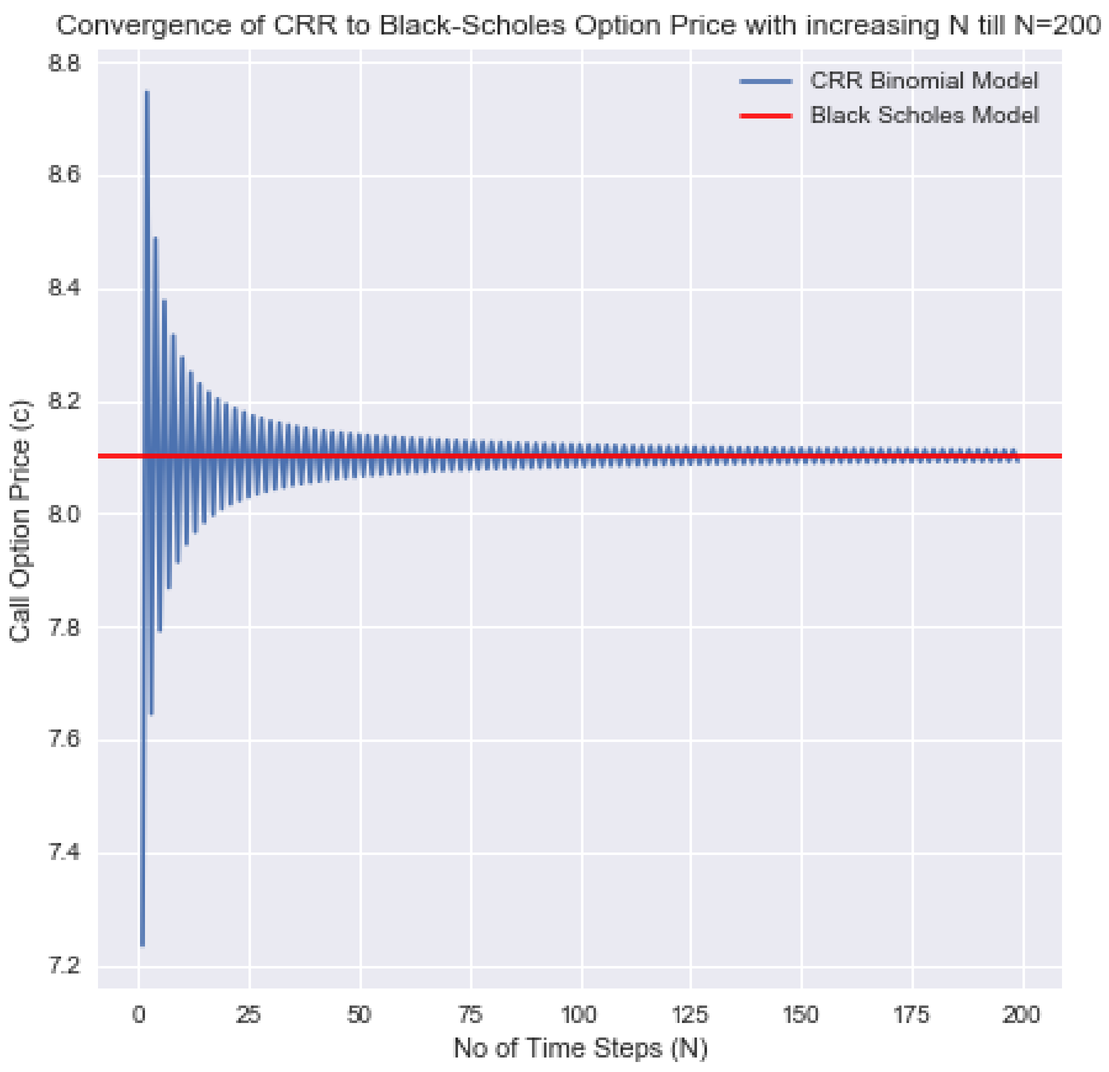
1-year European call option, K = 100 , S0 = 100, r = 0.05, q = 0.04, = 0.02

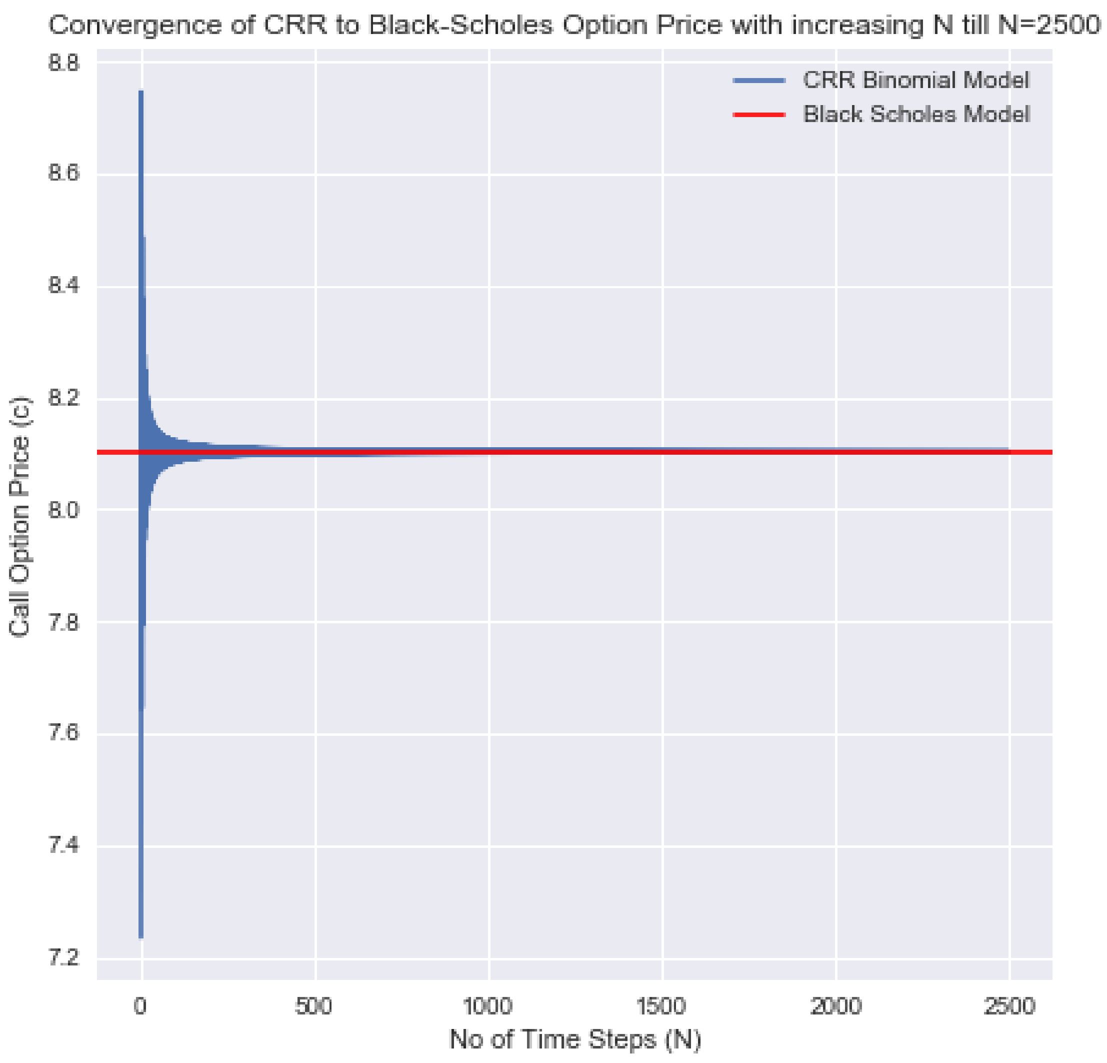
```
In [ ]: CRRPriceList = []

        for i in range(1,2500):
            CRRPrice,T = CRRBionomial("C", 100, 1, 100, 0.2, 0.05, 0.04, i, "E")
            CRRPriceList.append(CRRPrice)
```

0.0.6 6. Plotting the Convergence of CRR Model to the Black Scholes Model with increasing N

There are two graphs below, the first shows the convergence from N = 1 to 200, the second shows the same covergence from N = 1 to 2500.

```
In [52]: sns.set()
         plt.figure(figsize=(7,7))
         plt.plot(range(1,200),CRRPriceList[1:200], mfc='b',label="CRR Binomial Model")
         plt.axhline(y=BSprice,c='r',label="Black Scholes Model")
         plt.legend(loc="best")
         plt.ylabel("Call Option Price (c)")
         plt.xlabel("No of Time Steps (N)")
         plt.title("Convergence of CRR to Black-Scholes Option Price with increasing N till N=
         plt.show()


         sns.set()
         plt.figure(figsize=(7,7))
         plt.plot(range(1,2499),CRRPriceList[1:2500], mfc='b',label="CRR Binomial Model")
         plt.axhline(y=BSprice,c='r',label="Black Scholes Model")
         plt.legend(loc="best")
         plt.ylabel("Call Option Price (c)")
         plt.xlabel("No of Time Steps (N)")
         plt.title("Convergence of CRR to Black-Scholes Option Price with increasing N till N=
         plt.show()
```


Convergence of CRR to Black-Scholes Option Price with increasing N till N=200
CRR Binomial Model
Black Scholes Model
Call Option Price (c)
No of Time Steps (N)
8.8
8.6
8.4
8.2
8.0
7.8
7.6
7.4
7.2
0
25
50
75
100
125
150
175
200



0.0.7 7. No. of Steps to Achieve 10ˆ-3 Put Option Price Accuracy for CRR Bionomial Model | q = 0

```
In [54]: steps = [100,200,300,400,500,600,700,800,900,1000,2000,3000,4000,5000]
         for month in range(1,13):
             OptPrice = []
             for i in steps:
                 CRRPrice,T = CRRBionomial("P", 100, month/12, 100, 0.2, 0.05, 0, i, "A")
                 OptPrice.append(CRRPrice)

             OptPriceRev = OptPrice[::-1]
             stepRev = steps[::-1]
             for i in range(len(steps)+1):
                 if (abs(OptPriceRev[0]-OptPriceRev[i]) > 0.001):
```

```
                t = stepRev[i-1]
                CRRPrice,T = CRRBionomial("P", 100, month/12, 100, 0.2, 0.05, 0, t, "A")
                print ("The N value for ", month, "months is", (stepRev[i-1]),". The corre
                break
```

```
The N value for  1 months is 400 . The corresponding Option Price is:  2.12598815614
The N value for  2 months is 500 . The corresponding Option Price is:  2.91137673581
The N value for  3 months is 500 . The corresponding Option Price is:  3.47882597695
The N value for  4 months is 600 . The corresponding Option Price is:  3.93472235402
The N value for  5 months is 600 . The corresponding Option Price is:  4.31963127987
The N value for  6 months is 600 . The corresponding Option Price is:  4.65463550117
The N value for  7 months is 600 . The corresponding Option Price is:  4.95212836792
The N value for  8 months is 700 . The corresponding Option Price is:  5.22032186379
The N value for  9 months is 700 . The corresponding Option Price is:  5.46446125712
The N value for  10 months is 700 . The corresponding Option Price is:  5.68875026141
The N value for  11 months is 700 . The corresponding Option Price is:  5.89623285094
The N value for  12 months is 700 . The corresponding Option Price is:  6.08926078067
```

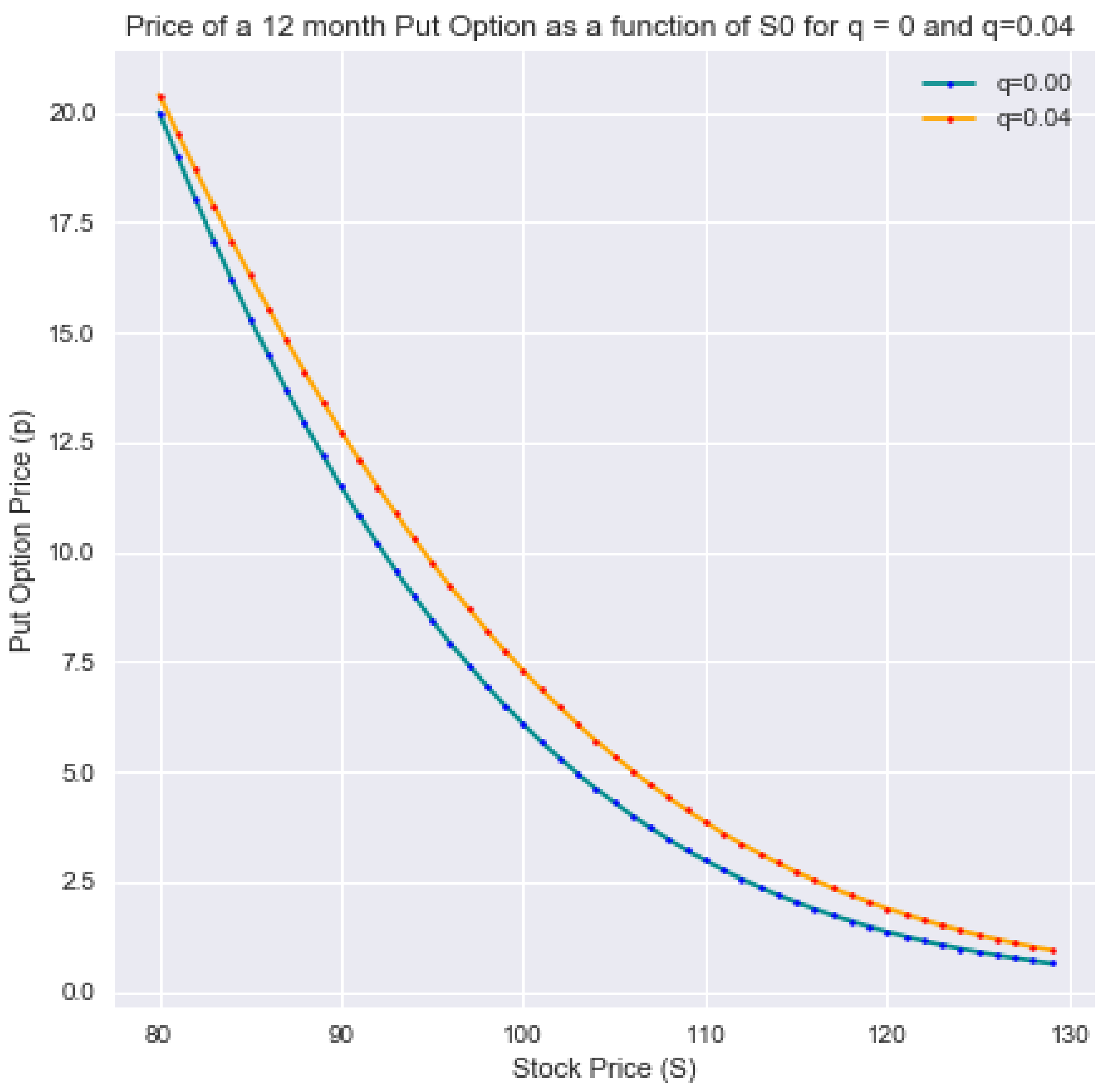
0.0.8 8. No. of Steps to Achieve 10^-3 Put Option Price Accuracy for CRR Bionomial Model | q = 0.04

```
In [59]: steps = [100,200,300,400,500,600,700,800,900,1000,1100,1200,1300,1400,3000,5000]
         Nvalues = []
         for month in range(1,13):
             OptPrice = []
             for i in steps:
                 CRRPrice,T = CRRBionomial("P", 100, month/12, 100, 0.2, 0.05, 0.04, i, "A")
                 OptPrice.append(CRRPrice)

             OptPriceRev = OptPrice[::-1]
             stepRev = steps[::-1]
             for i in range(len(steps)+1):
                 if (abs(OptPriceRev[0]-OptPriceRev[i]) > 0.001):
                     t = stepRev[i-1]
                     CRRPrice,T = CRRBionomial("P", 100, month/12, 100, 0.2, 0.05, 0.04, t, "A
                     print ("The N value for ", month, "months is", (stepRev[i-1]),". The corre
                     break
```

```
The N value for  1 months is 500 . The corresponding Option Price is:  2.25796069176
The N value for  2 months is 600 . The corresponding Option Price is:  3.16313414343
The N value for  3 months is 700 . The corresponding Option Price is:  3.84372216904
The N value for  4 months is 800 . The corresponding Option Price is:  4.40736410204
The N value for  5 months is 900 . The corresponding Option Price is:  4.89587969173
The N value for  6 months is 900 . The corresponding Option Price is:  5.33066868044
The N value for  7 months is 900 . The corresponding Option Price is:  5.72466807619
The N value for  8 months is 1000 . The corresponding Option Price is:  6.08637500923
The N value for  9 months is 1000 . The corresponding Option Price is:  6.42137637229
The N value for  10 months is 1000 . The corresponding Option Price is:  6.73404780388
The N value for  11 months is 1000 . The corresponding Option Price is:  7.02761145908
The N value for  12 months is 1100 . The corresponding Option Price is:  7.30469386127
```

0.1 9. Plotting Option Price of 12 month Put Option as a Function of S0 for q = 0 and q = 0.04

In [76]:
```
#q = 0.00 | Put Option
OptPrice1 = []
for i in range(80,130):
    #CRRBionomial(Option, K, T, S0, sigma, r, q, N, Exercise)
    CRRPrice,T = CRRBionomial("P", 100, 1,i, 0.2, 0.05, 0, 3000, "A")
    OptPrice1.append(CRRPrice)

#q = 0.04 | Put Option
OptPrice2 = []
for i in range(80,130):
    #CRRBionomial(Option, K, T, S0, sigma, r, q, N, Exercise)
    CRRPrice,T = CRRBionomial("P", 100, 1,i, 0.2, 0.05, 0.04, 3000, "A")
    OptPrice2.append(CRRPrice)

sns.set()
plt.figure(figsize=(7,7))
plt.plot(range(80,130),OptPrice1, marker='o',c='darkcyan',ms=2.5, mfc='b',label="q=0.
plt.plot(range(80,130),OptPrice2, marker='o',c='orange',ms=2.5, mfc='r',label="q=0.04
plt.legend(loc="best")
plt.ylabel("Put Option Price (p)")
plt.xlabel("Stock Price (S)")
plt.title("Price of a 12 month Put Option as a function of S0 for q = 0 and q=0.04 ")
plt.show()
```



0.2 10. Estimating the Critical Stock Price for Put Option at q = 0.00

```
In [69]: K = 100
         S_starter = []
         for t in range(1,13):
                 for s in range(60,100):
                     a, T = CRRBionomial("P", 100, t/12, s , 0.2, 0.05, 0, 3000, "A")
                     b = K-s
                     if (abs(b-a)>0.005):
                         S_starter.append(s-1)
                         break
         CritPrice1 = []
         for t in range(0,12):
                 for i in range(1,100):
                     S0 = S_starter[t] + i*0.01
                     a, T = CRRBionomial("P", 100, (t+1)/12, S0 , 0.2, 0.05, 0, 3000, "A")
                     b = K-S0
                     if (abs(b-a)>0.005):
                         print ("The Critical Stock Price for q=0 for month: ", t+1,"is :", S_s
                         CritPrice1.append(S_star)
                         break
                     if (abs(b-a)<0.005):
                         S_star = S0
```

```
The Critical Stock Price for q=0 for month:  1 is : 91.3
The Critical Stock Price for q=0 for month:  2 is : 88.91
The Critical Stock Price for q=0 for month:  3 is : 87.34
The Critical Stock Price for q=0 for month:  4 is : 86.17
The Critical Stock Price for q=0 for month:  5 is : 85.23
The Critical Stock Price for q=0 for month:  6 is : 84.44
The Critical Stock Price for q=0 for month:  7 is : 83.77
The Critical Stock Price for q=0 for month:  8 is : 83.18
The Critical Stock Price for q=0 for month:  9 is : 82.66
The Critical Stock Price for q=0 for month:  10 is : 82.2
The Critical Stock Price for q=0 for month:  11 is : 81.77
The Critical Stock Price for q=0 for month:  12 is : 81.38
```

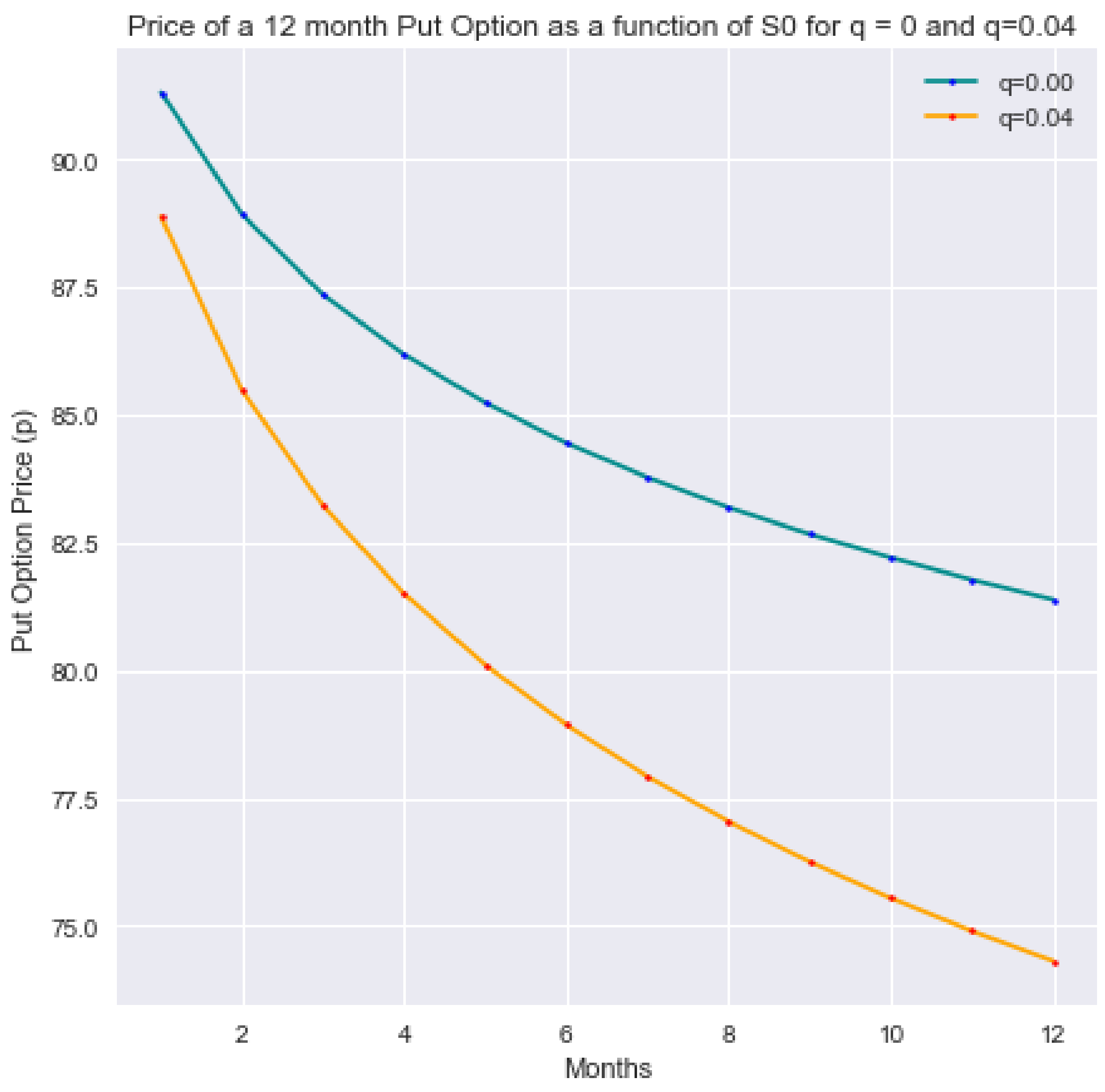
0.3 11. Estimating the Critical Stock Price for Put Option at q = 0.04

```
In [71]: K = 100
         S_starter1 = []
         for t in range(1,13):
                 for s in range(60,100):
                     a, T = CRRBionomial("P", 100, t/12, s , 0.2, 0.05, 0.04, 3000, "A")
                     b = K-s
                     if (abs(b-a)>0.005):
                         S_starter1.append(s-1)
                         break
         CritPrice2 = []
         for t in range(0,12):
                 for i in range(1,100):
                     S0 = S_starter1[t] + i*0.01
                     a, T = CRRBionomial("P", 100, (t+1)/12, S0 , 0.2, 0.05, 0.04, 3000, "A")
                     b = K-S0
                     if (abs(b-a)>0.005):
                         print ("The Critical Stock Price for q=0.04 for month: ", t+1,"is :",
                         CritPrice2.append(S_star)
                         break
                     if (abs(b-a)<0.005):
                         S_star = S0
```

```
The Critical Stock Price for q=0.04 for month:  1 is : 88.87
The Critical Stock Price for q=0.04 for month:  2 is : 85.47
The Critical Stock Price for q=0.04 for month:  3 is : 83.21
The Critical Stock Price for q=0.04 for month:  4 is : 81.49
The Critical Stock Price for q=0.04 for month:  5 is : 80.1
The Critical Stock Price for q=0.04 for month:  6 is : 78.93
The Critical Stock Price for q=0.04 for month:  7 is : 77.92
The Critical Stock Price for q=0.04 for month:  8 is : 77.04
The Critical Stock Price for q=0.04 for month:  9 is : 76.25
The Critical Stock Price for q=0.04 for month:  10 is : 75.54
The Critical Stock Price for q=0.04 for month:  11 is : 74.9
The Critical Stock Price for q=0.04 for month:  12 is : 74.31
```

0.4 12. Plotting the Critical Stock Prices for Put Option at q = 0 and q = 0.04 as a Function of Time

```
In [77]: sns.set()
         plt.figure(figsize=(7,7))
         plt.plot(range(1,13),CritPrice1, marker='o',c='darkcyan',ms=2.5, mfc='b',label="q=0.0
         plt.plot(range(1,13),CritPrice2, marker='o',c='orange',ms=2.5, mfc='r',label="q=0.04"
         plt.legend(loc="best")
         plt.ylabel("Put Option Price (p)")
         plt.xlabel("Months")
         plt.title("Price of a 12 month Put Option as a function of S0 for q = 0 and q=0.04 ")
         plt.show()
```



0.5 13. No. of Steps to Achieve 10ˆ-3 Call Option Price Accuracy for CRR Model | q = 0.04

```
In [85]: steps = [500,600,700,800,900,1000,1100,1200,1300,1400,1500,1600,1700,1800,1900,2000,3
         for month in range(1,13):
             OptPrice = []
             for i in steps:
                 CRRPrice,T = CRRBionomial("C", 100, month/12, 100, 0.2, 0.05, 0.04, i, "A")
                 OptPrice.append(CRRPrice)

             OptPriceRev = OptPrice[::-1]
             stepRev = steps[::-1]
             for i in range(len(steps)+1):
```

```
            if (abs(OptPriceRev[0]-OptPriceRev[i]) > 0.001):
                t = stepRev[i-1]
                CRRPrice,T = CRRBionomial("C", 100, month/12, 100, 0.2, 0.05, 0.04, t, "A
                print ("The N value for ", month, "months is", (stepRev[i-1]),". The corre
                break
```

```
The N value for  1 months is 600 . The corresponding Option Price is:  2.33514846077
The N value for  2 months is 700 . The corresponding Option Price is:  3.31435124277
The N value for  3 months is 900 . The corresponding Option Price is:  4.06697130743
The N value for  4 months is 1000 . The corresponding Option Price is:  4.70075777352
The N value for  5 months is 1100 . The corresponding Option Price is:  5.25794534715
The N value for  6 months is 1100 . The corresponding Option Price is:  5.76012662503
The N value for  7 months is 1200 . The corresponding Option Price is:  6.22046630996
The N value for  8 months is 1300 . The corresponding Option Price is:  6.6473174013
The N value for  9 months is 1300 . The corresponding Option Price is:  7.04649168852
The N value for  10 months is 1400 . The corresponding Option Price is:  7.42249590082
The N value for  11 months is 1400 . The corresponding Option Price is:  7.77840992862
The N value for  12 months is 1400 . The corresponding Option Price is:  8.11687479705
```

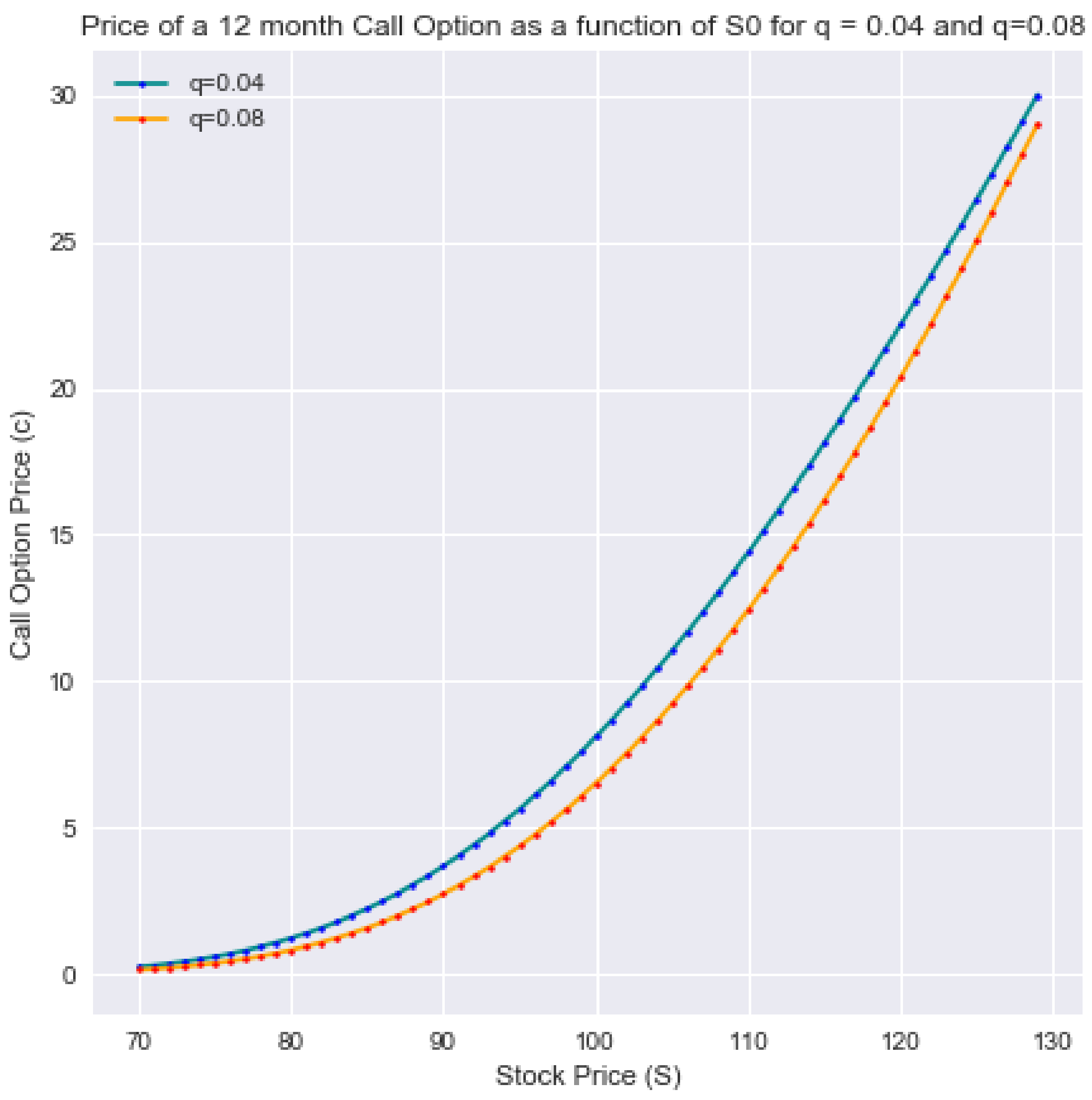
0.6 14. No. of Steps to Achieve 10ˆ-3 Call Option Price Accuracy for CRR Model | q = 0.08

```
In [88]: steps = [100,200,300,400,500,600,700,800,900,1000,1100,1200,1300,1400,3000,5000]
         Nvalues = []
         for month in range(1,13):
             OptPrice = []
             for i in steps:
                 CRRPrice,T = CRRBionomial("C", 100, month/12, 100, 0.2, 0.05, 0.08, i, "A")
                 OptPrice.append(CRRPrice)

             OptPriceRev = OptPrice[::-1]
             stepRev = steps[::-1]
             for i in range(len(steps)+1):
                 if (abs(OptPriceRev[0]-OptPriceRev[i]) > 0.001):
                     t = stepRev[i-1]
                     CRRPrice,T = CRRBionomial("C", 100, month/12, 100, 0.2, 0.05, 0.08, t, "A
                     print ("The N value for ", month, "months is", (stepRev[i-1]),". The corre
                     break
```

```
The N value for  1 months is 400 . The corresponding Option Price is:  2.18601657724
The N value for  2 months is 500 . The corresponding Option Price is:  3.02210818504
The N value for  3 months is 600 . The corresponding Option Price is:  3.63529420481
The N value for  4 months is 600 . The corresponding Option Price is:  4.13277444432
The N value for  5 months is 700 . The corresponding Option Price is:  4.55657310147
The N value for  6 months is 700 . The corresponding Option Price is:  4.92776512311
The N value for  7 months is 700 . The corresponding Option Price is:  5.25924767627
The N value for  8 months is 700 . The corresponding Option Price is:  5.5593326373
The N value for  9 months is 800 . The corresponding Option Price is:  5.83394701699
The N value for  10 months is 800 . The corresponding Option Price is:  6.08701272419
The N value for  11 months is 800 . The corresponding Option Price is:  6.32185859612
The N value for  12 months is 800 . The corresponding Option Price is:  6.54095776056
```

0.7 15. Plotting Option Price of 1 yr Call Option as a Function of S0 for q = 0.04 & q = 0.08

```
In [93]: #q = 0.04 | Call Option
         OptPrice3 = []
         for i in range(70,130):
             #CRRBionomial(Option, K, T, S0, sigma, r, q, N, Exercise)
             CRRPrice,T = CRRBionomial("C", 100, 1,i, 0.2, 0.05, 0.04, 3000, "A")
             OptPrice3.append(CRRPrice)

         #q = 0.08 | Call Option
         OptPrice4 = []
         for i in range(70,130):
             #CRRBionomial(Option, K, T, S0, sigma, r, q, N, Exercise)
             CRRPrice,T = CRRBionomial("C", 100, 1,i, 0.2, 0.05, 0.08, 3000, "A")
             OptPrice4.append(CRRPrice)

         sns.set()
         plt.figure(figsize=(7,7))
         plt.plot(range(70,130),OptPrice3, marker='o',c='darkcyan',ms=2.5, mfc='b',label="q=0.
         plt.plot(range(70,130),OptPrice4, marker='o',c='orange',ms=2.5, mfc='r',label="q=0.08
         plt.legend(loc="best")
         plt.ylabel("Call Option Price (c)")
         plt.xlabel("Stock Price (S)")
         plt.title("Price of a 12 month Call Option as a function of S0 for q = 0.04 and q=0.0
         plt.show()
```



0.8 16. Estimating the Critical Stock Price for Call Option at q = 0.04

```
In [90]: K = 100
         S_starter = []
         for t in range(1,13):
                 for s in range(170,100):
                     a, T = CRRBionomial("C", 100, t/12, s , 0.2, 0.05, 0.04, 3000, "A")
                     b = -(K-s)
                     if (abs(b-a)>0.005):
                         S_starter.append(s+1)
                         break
         CritPrice1 = []
         for t in range(0,12):
                 for i in range(1,100):
                     S0 = S_starter[t] - i*0.01
                     a, T = CRRBionomial("C", 100, (t+1)/12, S0 , 0.2, 0.05, 0.04, 3000, "A")
                     b = -(K-S0)
                     if (abs(b-a)>0.005):
                         print ("The Critical Stock Price for q=0.04 for month: ", t+1,"is :",
                         CritPrice1.append(S_star)
                         break
                     if (abs(b-a)<0.005):
                         S_star = S0
```

```
The Critical Stock Price for q=0.04 for month:  1 is : 125.09
The Critical Stock Price for q=0.04 for month:  2 is : 128.76
The Critical Stock Price for q=0.04 for month:  3 is : 132.29
The Critical Stock Price for q=0.04 for month:  4 is : 135.48
The Critical Stock Price for q=0.04 for month:  5 is : 138.29
The Critical Stock Price for q=0.04 for month:  6 is : 140.81
The Critical Stock Price for q=0.04 for month:  7 is : 143.09
The Critical Stock Price for q=0.04 for month:  8 is : 145.16
The Critical Stock Price for q=0.04 for month:  9 is : 147.07
The Critical Stock Price for q=0.04 for month:  10 is : 148.84
The Critical Stock Price for q=0.04 for month:  11 is : 150.50
The Critical Stock Price for q=0.04 for month:  12 is : 152.04
```

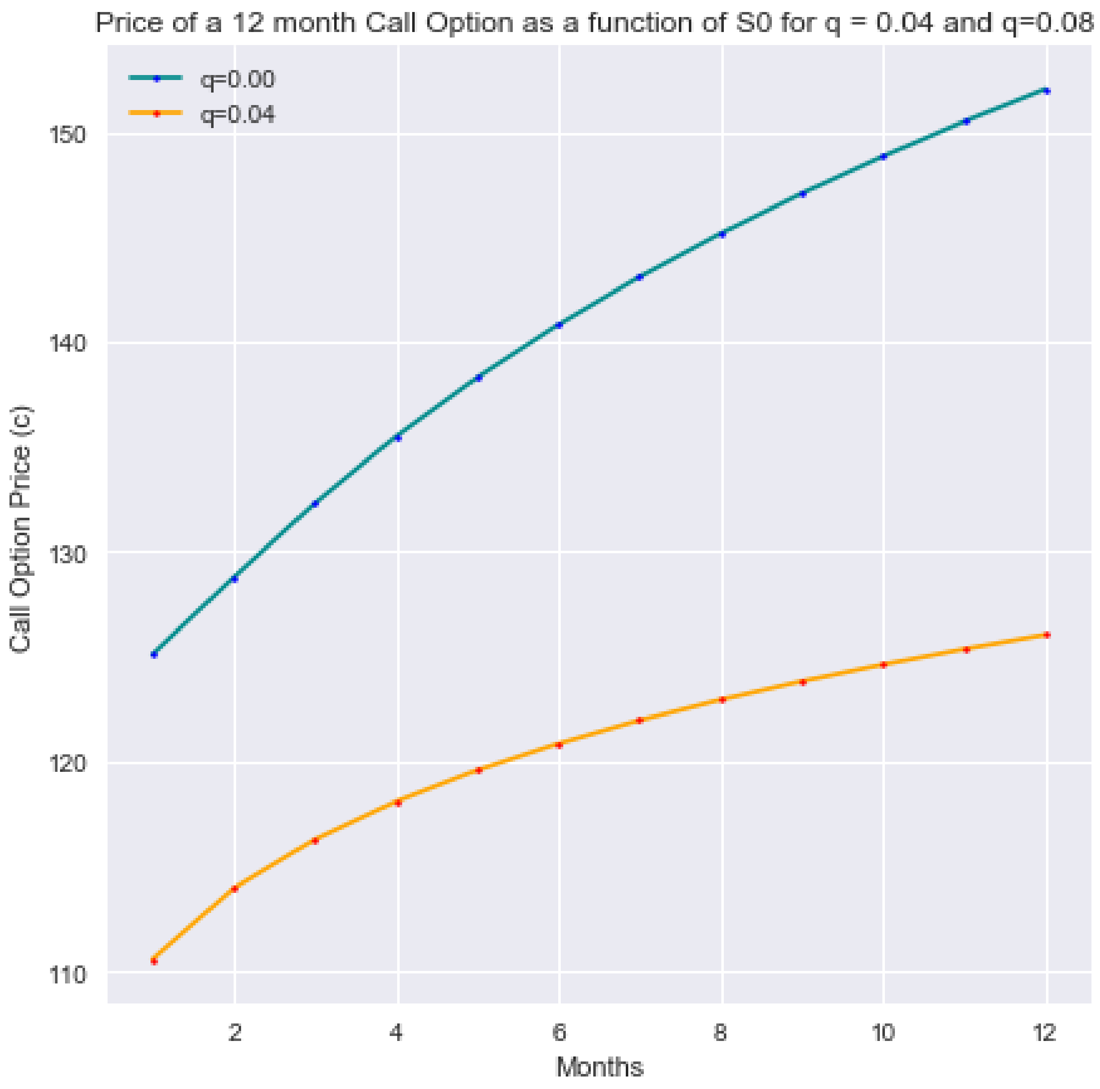
0.9 17. Estimating the Critical Stock Price for Call Option at q = 0.08

```
In [91]: K = 100
         S_starter = []
         for t in range(1,13):
                 for s in range(170,100):
                     a, T = CRRBionomial("C", 100, t/12, s , 0.2, 0.05, 0.08, 3000, "A")
                     b = -(K-s)
                     if (abs(b-a)>0.005):
                         S_starter.append(s+1)
                         break
         CritPrice2 = []
         for t in range(0,12):
                 for i in range(1,100):
                     S0 = S_starter[t] - i*0.01
                     a, T = CRRBionomial("C", 100, (t+1)/12, S0 , 0.2, 0.05, 0.08, 3000, "A")
                     b = -(K-S0)
                     if (abs(b-a)>0.005):
                         print ("The Critical Stock Price for q=0.08 for month: ", t+1,"is :",
                         CritPrice2.append(S_star)
                         break
                     if (abs(b-a)<0.005):
                         S_star = S0
```

```
The Critical Stock Price for q=0.08 for month:  1 is : 110.55
The Critical Stock Price for q=0.08 for month:  2 is : 113.94
The Critical Stock Price for q=0.08 for month:  3 is : 116.27
The Critical Stock Price for q=0.08 for month:  4 is : 118.08
The Critical Stock Price for q=0.08 for month:  5 is : 119.57
The Critical Stock Price for q=0.08 for month:  6 is : 120.84
The Critical Stock Price for q=0.08 for month:  7 is : 121.95
The Critical Stock Price for q=0.08 for month:  8 is : 122.93
The Critical Stock Price for q=0.08 for month:  9 is : 123.81
The Critical Stock Price for q=0.08 for month:  10 is : 124.61
The Critical Stock Price for q=0.08 for month:  11 is : 125.35
The Critical Stock Price for q=0.08 for month:  12 is : 126.03
```

1 18. Plotting the Critical Stock Prices for Call Option at q = 0.04 and q = 0.08 as a Function of Time

```
In [95]: sns.set()
         plt.figure(figsize=(7,7))
         plt.plot(range(1,13),CritPrice1, marker='o',c='darkcyan',ms=2.5, mfc='b',label="q=0.0
         plt.plot(range(1,13),CritPrice2, marker='o',c='orange',ms=2.5, mfc='r',label="q=0.04")
         plt.legend(loc="best")
         plt.ylabel("Call Option Price (c)")
         plt.xlabel("Months")
         plt.title("Price of a 12 month Call Option as a function of S0 for q = 0.04 and q=0.0
         plt.show()
```


Price of a 12 month Call Option as a function of S0 for q = 0.04 and q=0.08
q=0.00
q=0.04
Call Option Price (c)
Months
110
120
130
140
150
2
4
6
8
10
12