भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

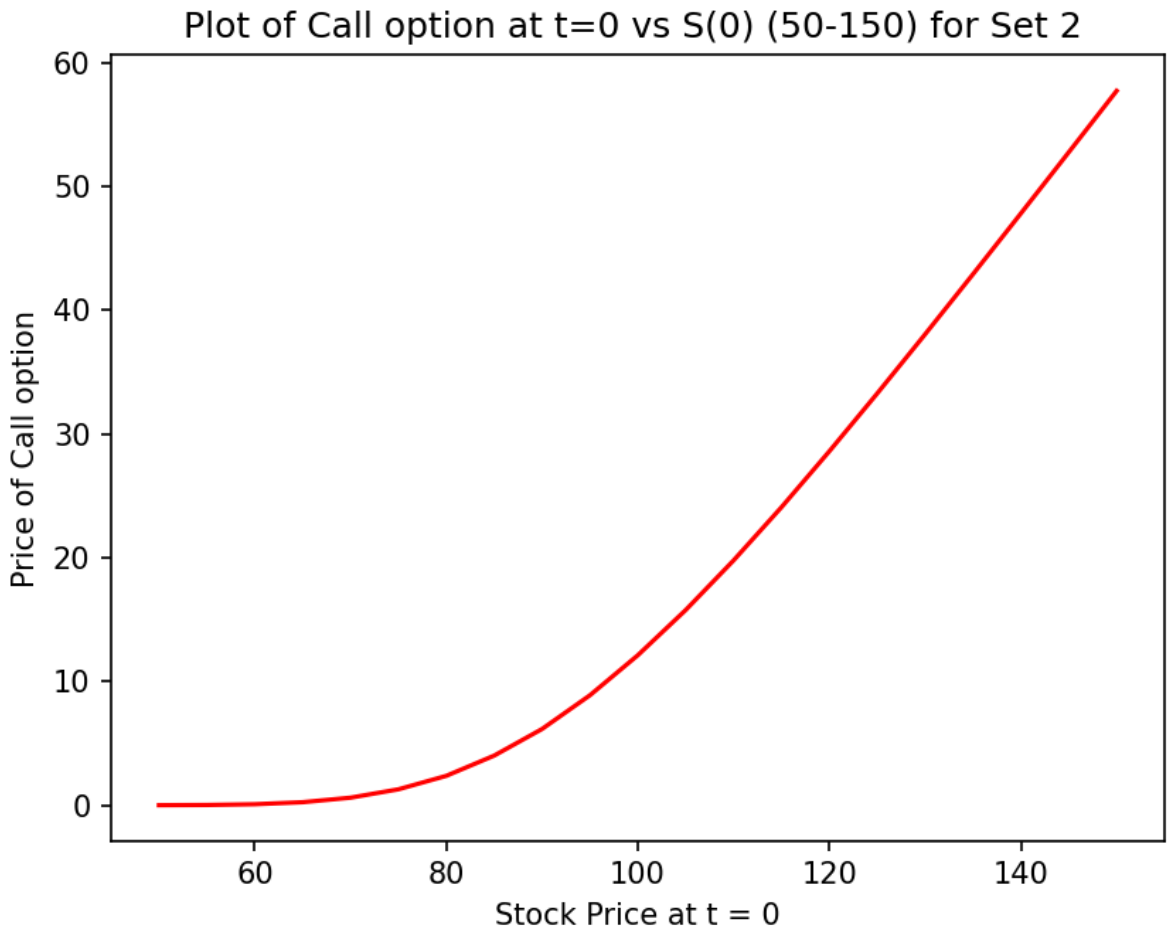
MA 374: Financial Engineering Lab

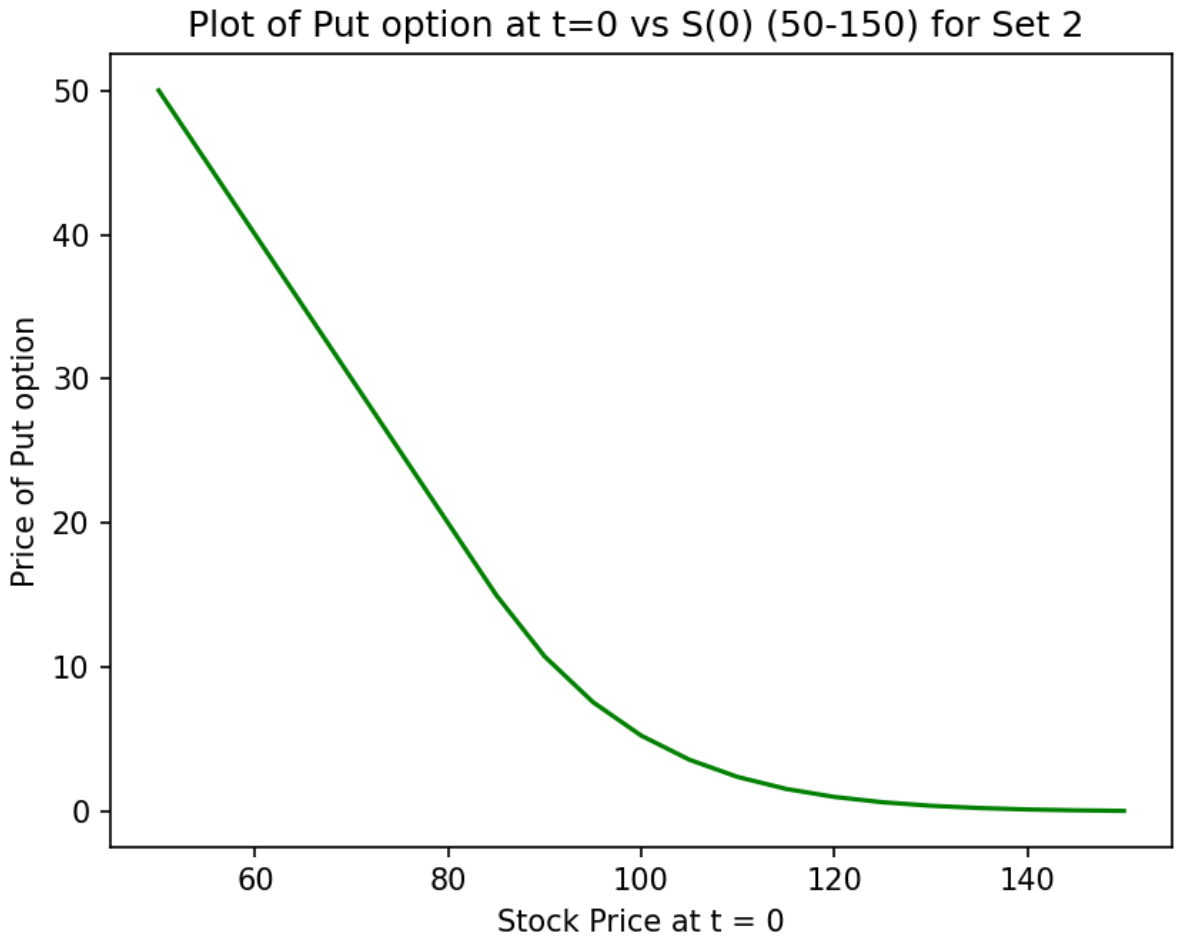
Lab 03

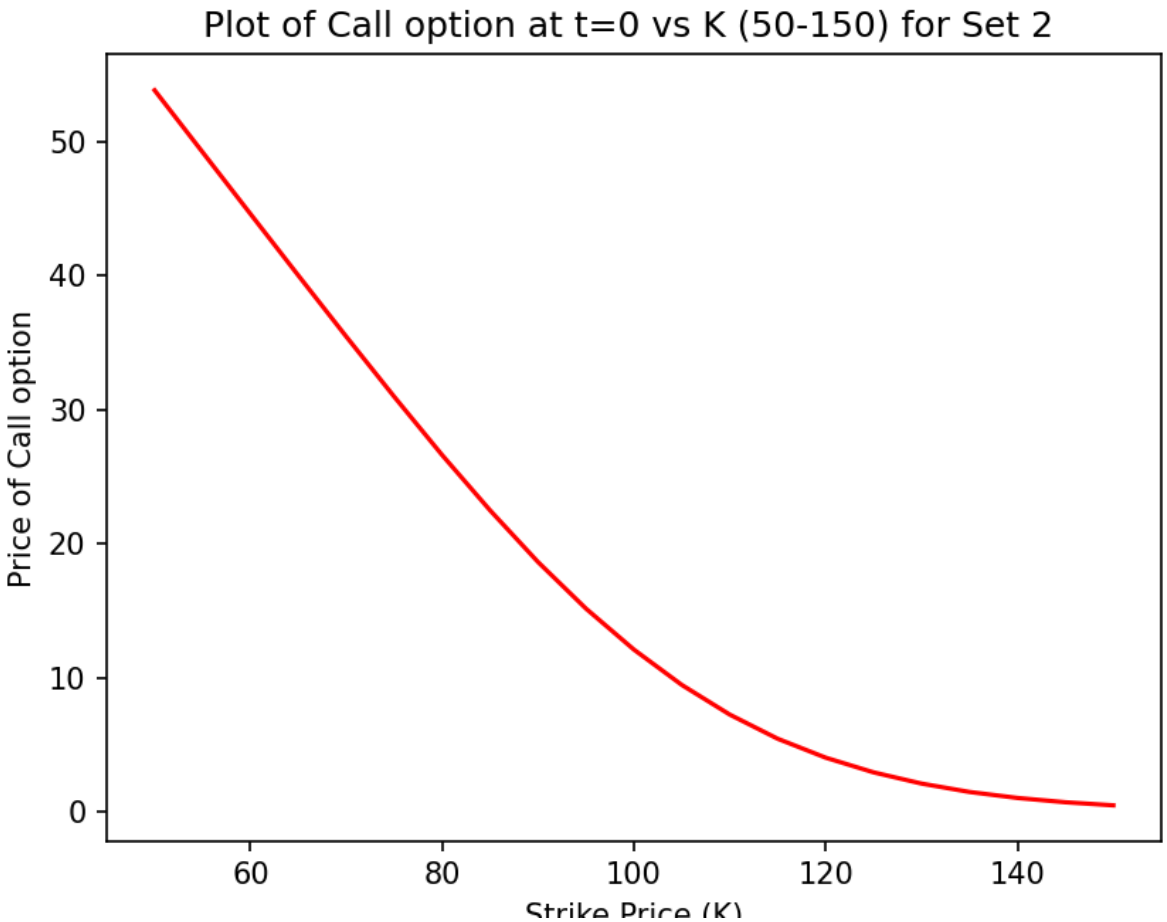
Jwalit Devalia (200123026)

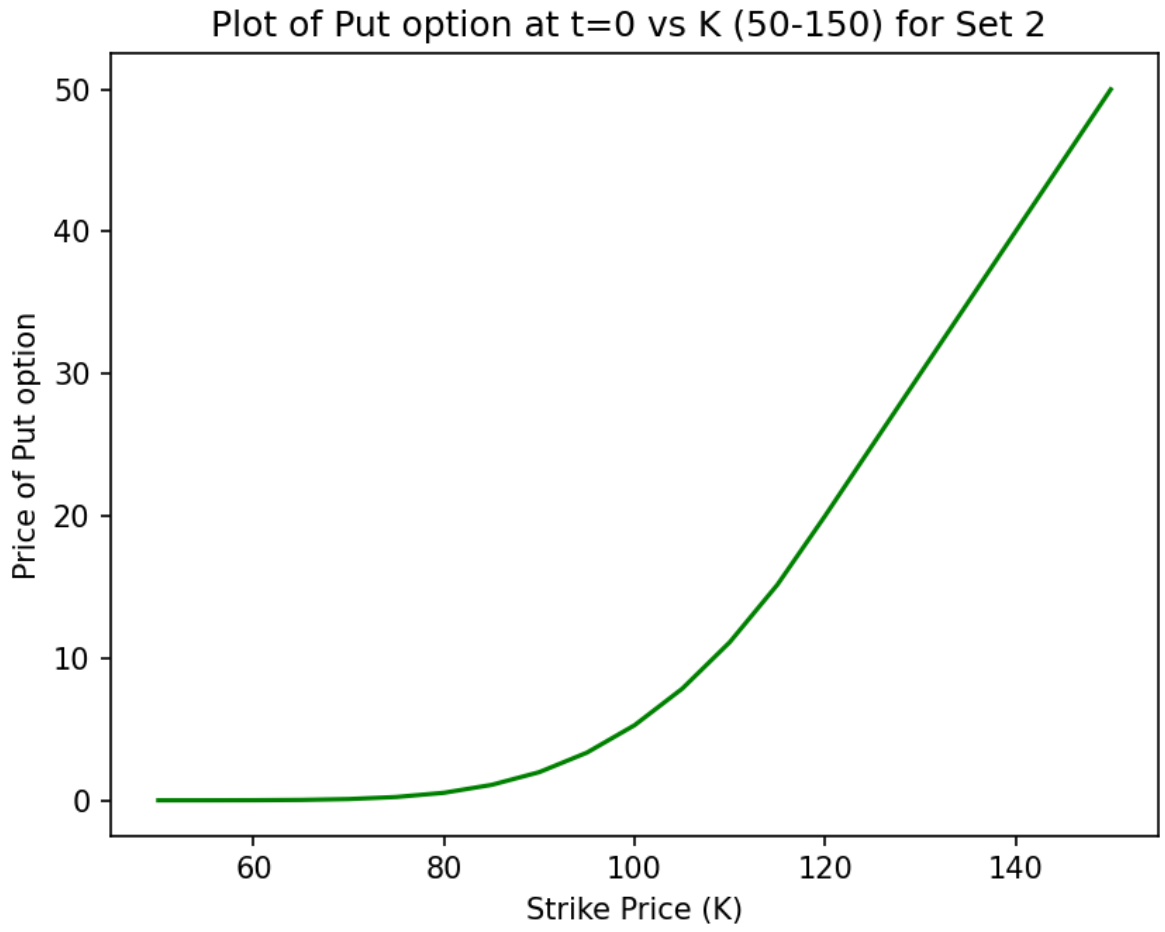
Question 1.

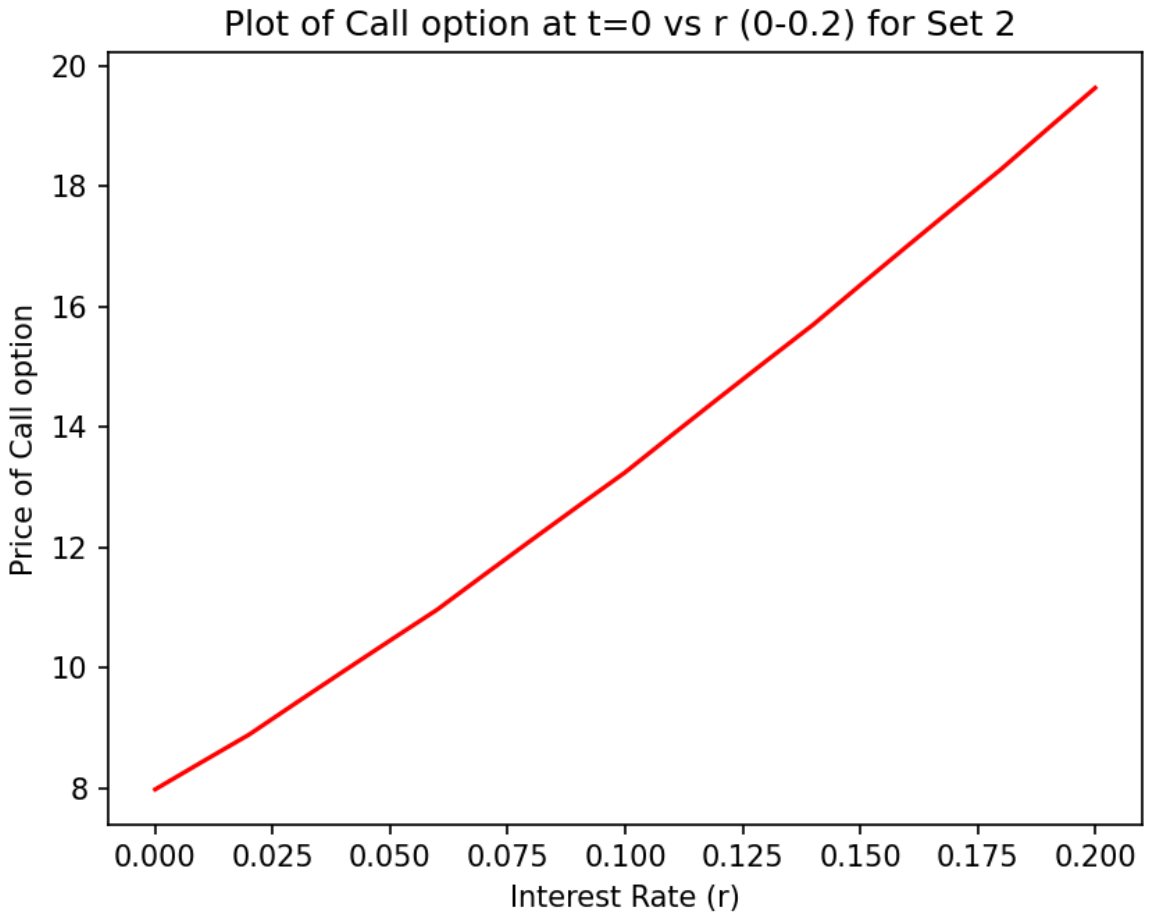
```
PS D:\sem-6\FElab> python -u  d:\sem-6\FElab\lab3\q1.
The Call price for set 2 is: 12.12304707401244
The Put price for set 2 is: 5.279837145989147
```

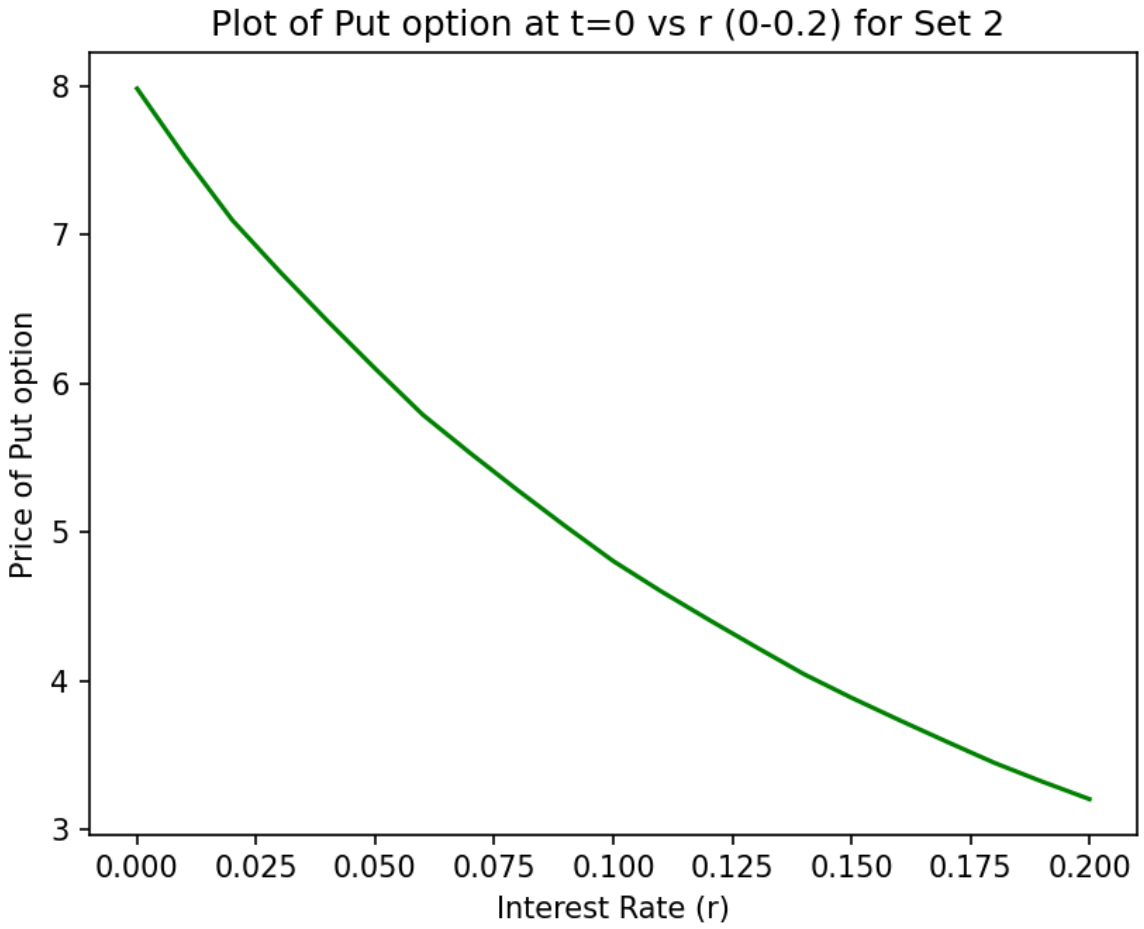


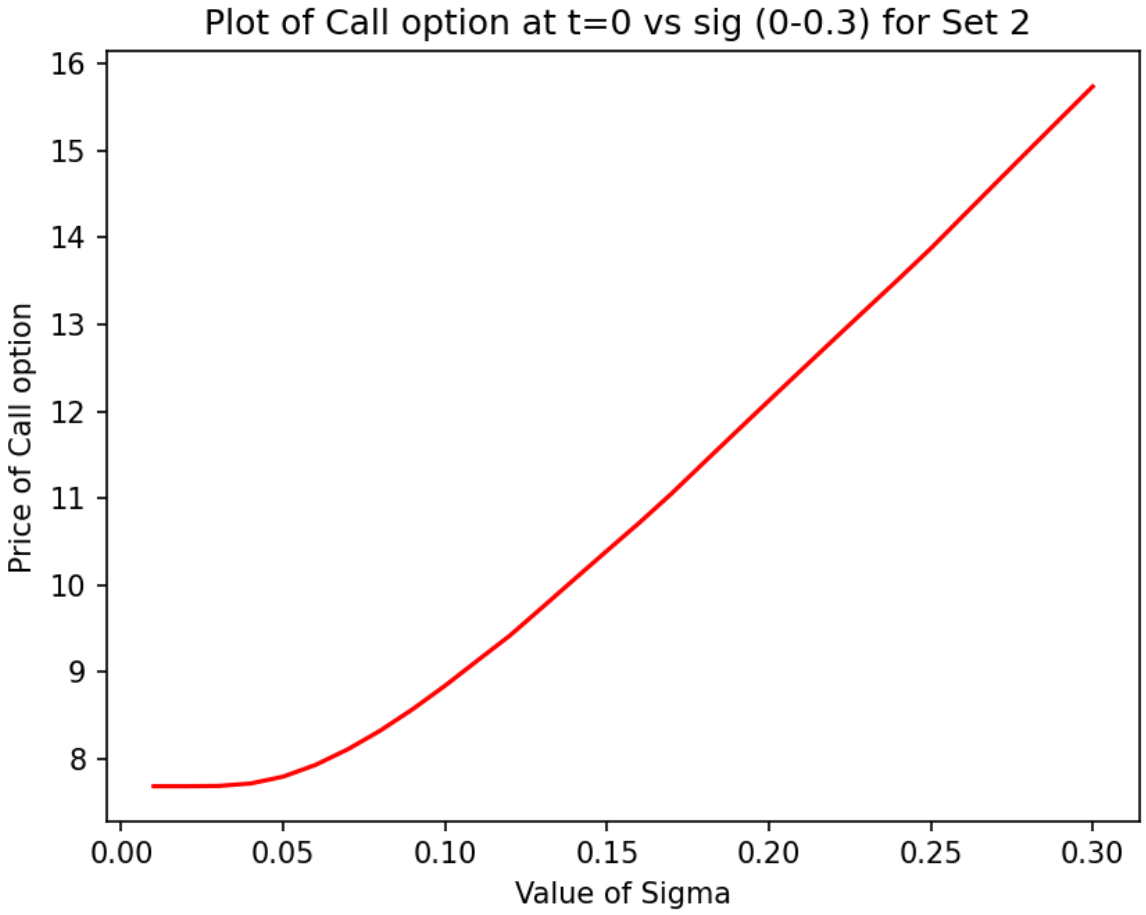

Plot of Put option at t=0 vs S(0) (50-150) for Set 2
Price of Put option
Stock Price at t = 0
50
40
30
20
10
0
60
80
100
120
140

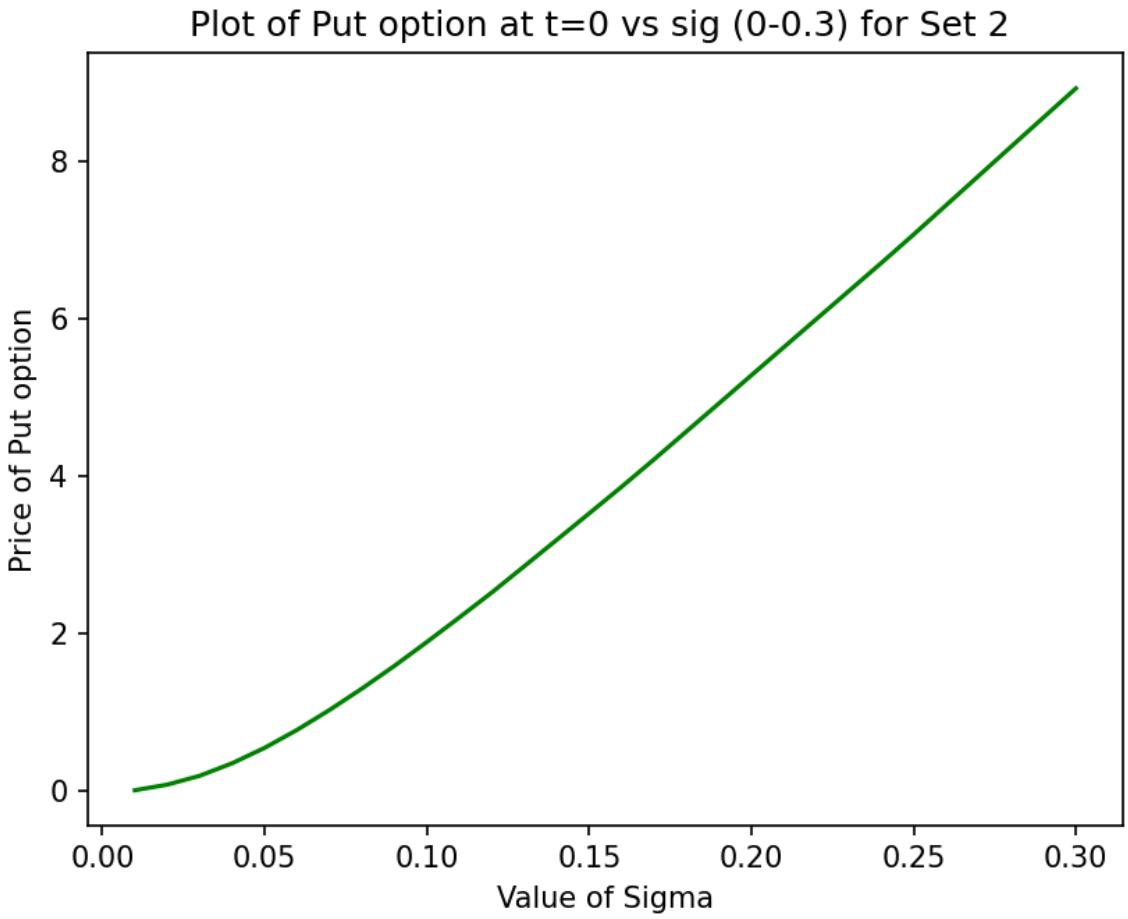

Plot of Call option at t=0 vs K (50-150) for Set 2
Price of Call option
50
40
30
20
10
0
60
80
100
120
140
Strike Price (K)

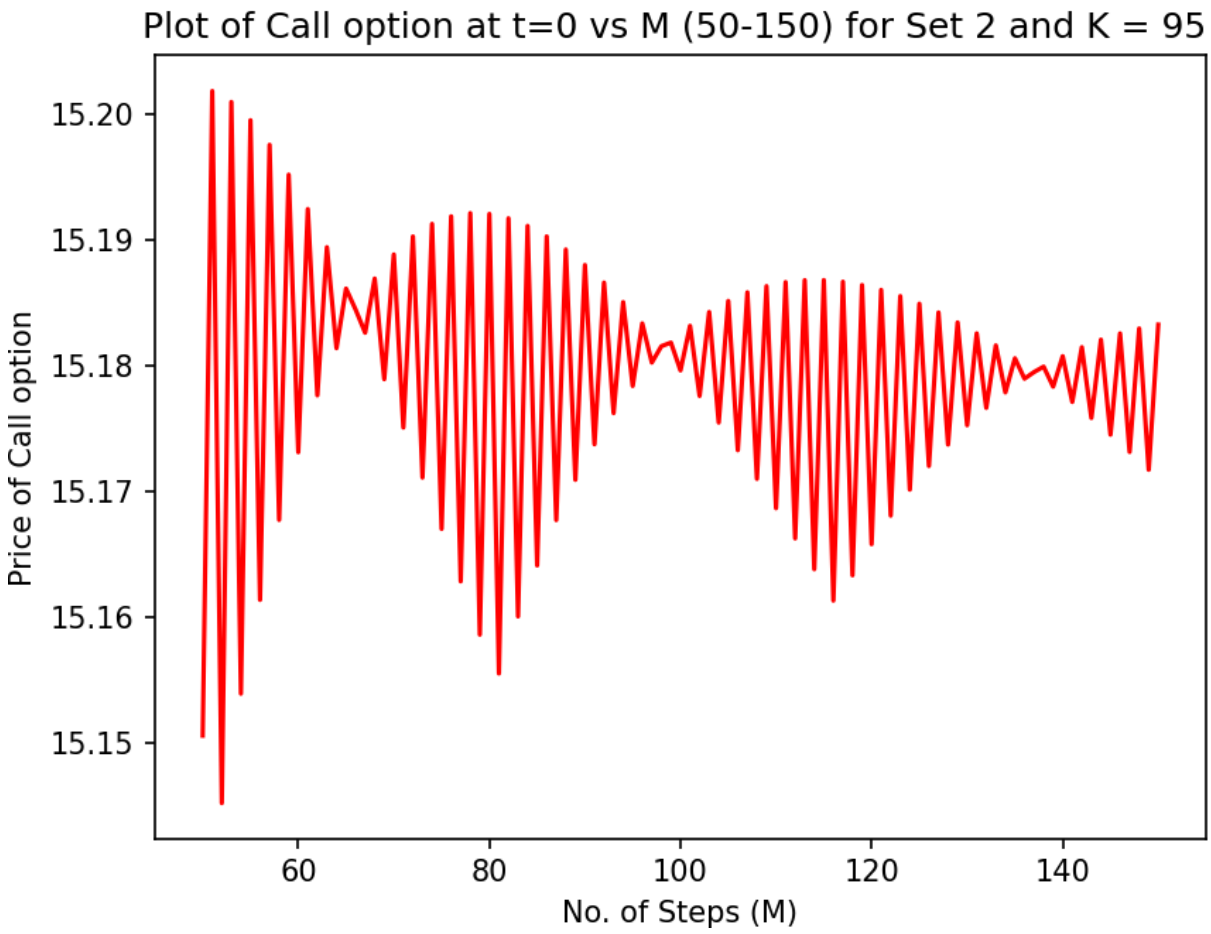

Plot of Put option at t=0 vs K (50-150) for Set 2
Price of Put option
Strike Price (K)
0
10
20
30
40
50
60
80
100
120
140

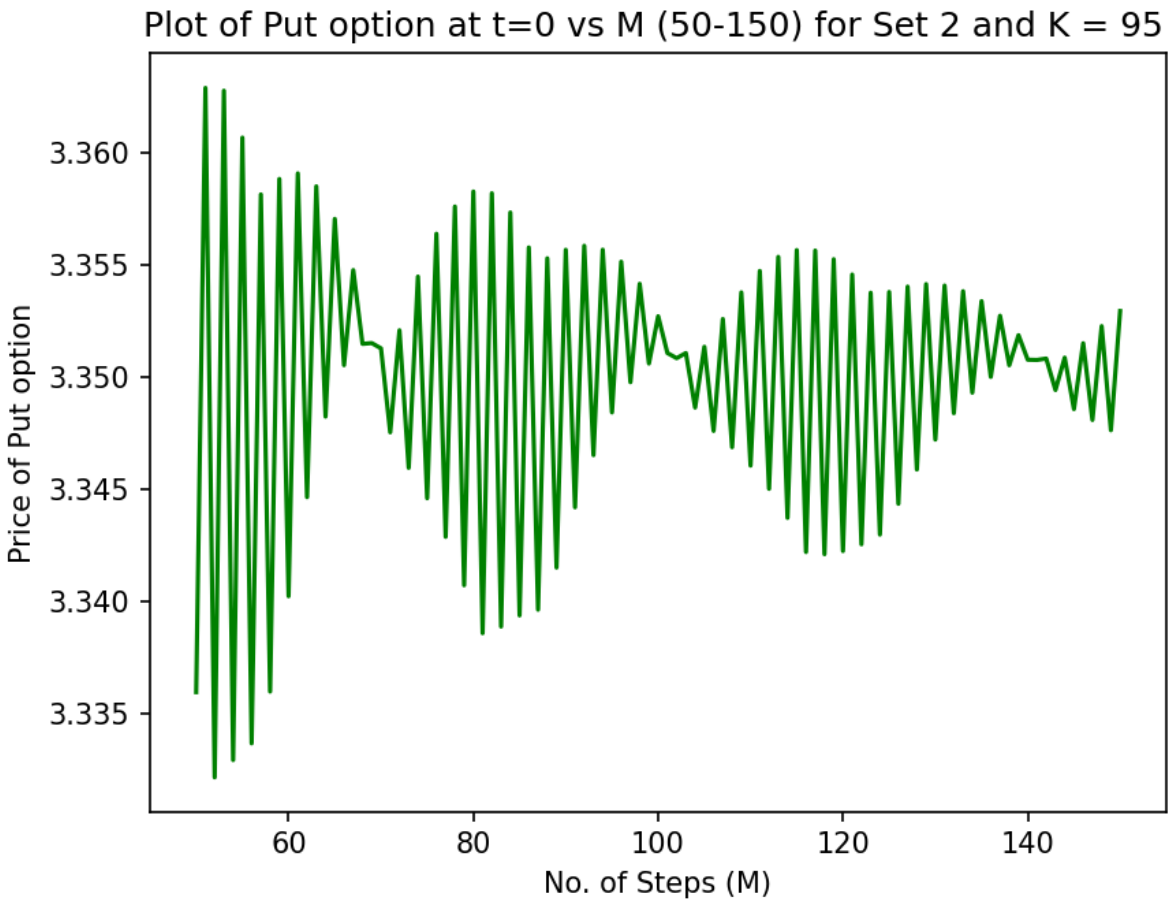

Plot of Call option at t=0 vs r (0-0.2) for Set 2
Price of Call option
20
18
16
14
12
10
8
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
Interest Rate (r)

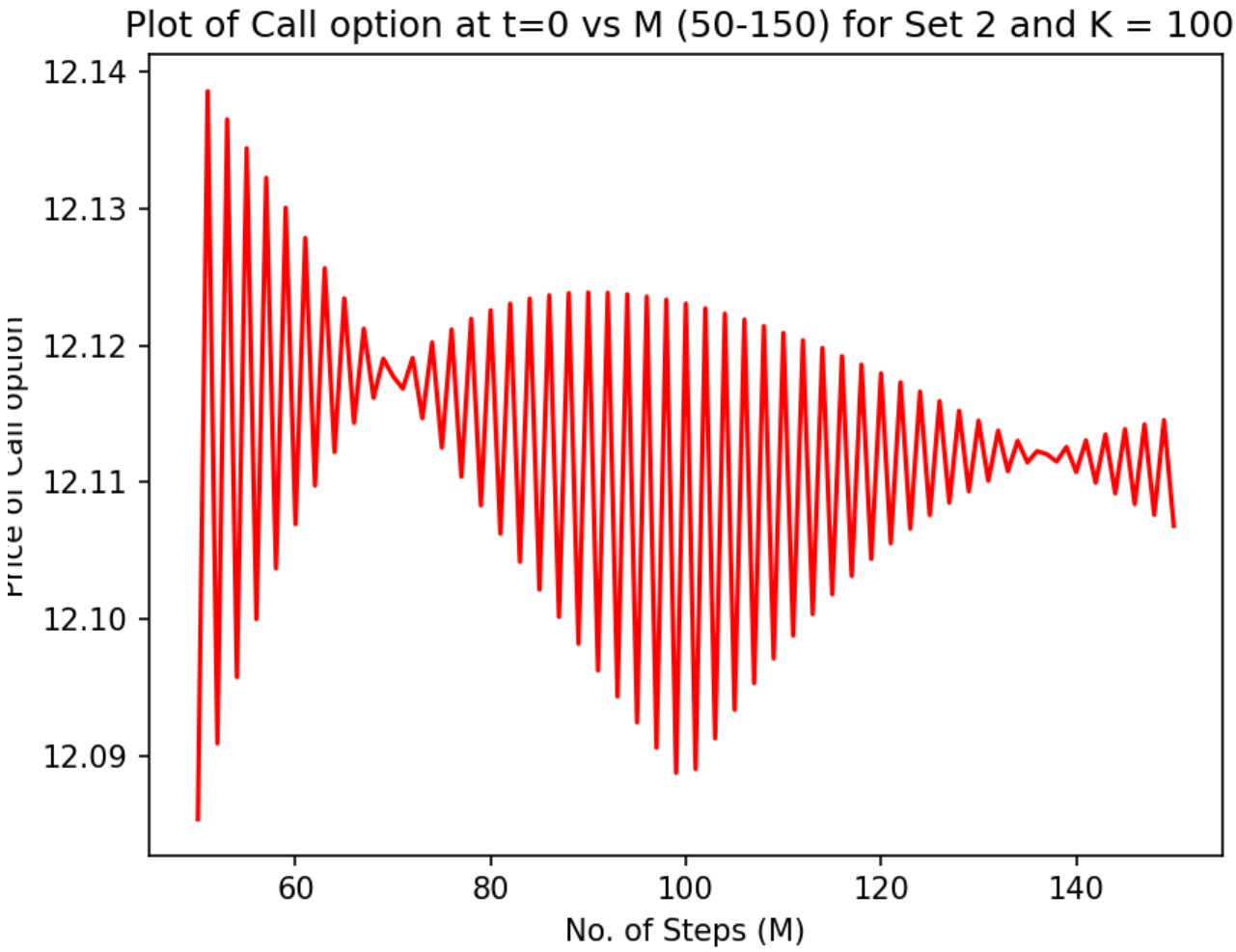

Plot of Put option at t=0 vs r (0-0.2) for Set 2
Price of Put option
8
7
6
5
4
3
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
Interest Rate (r)

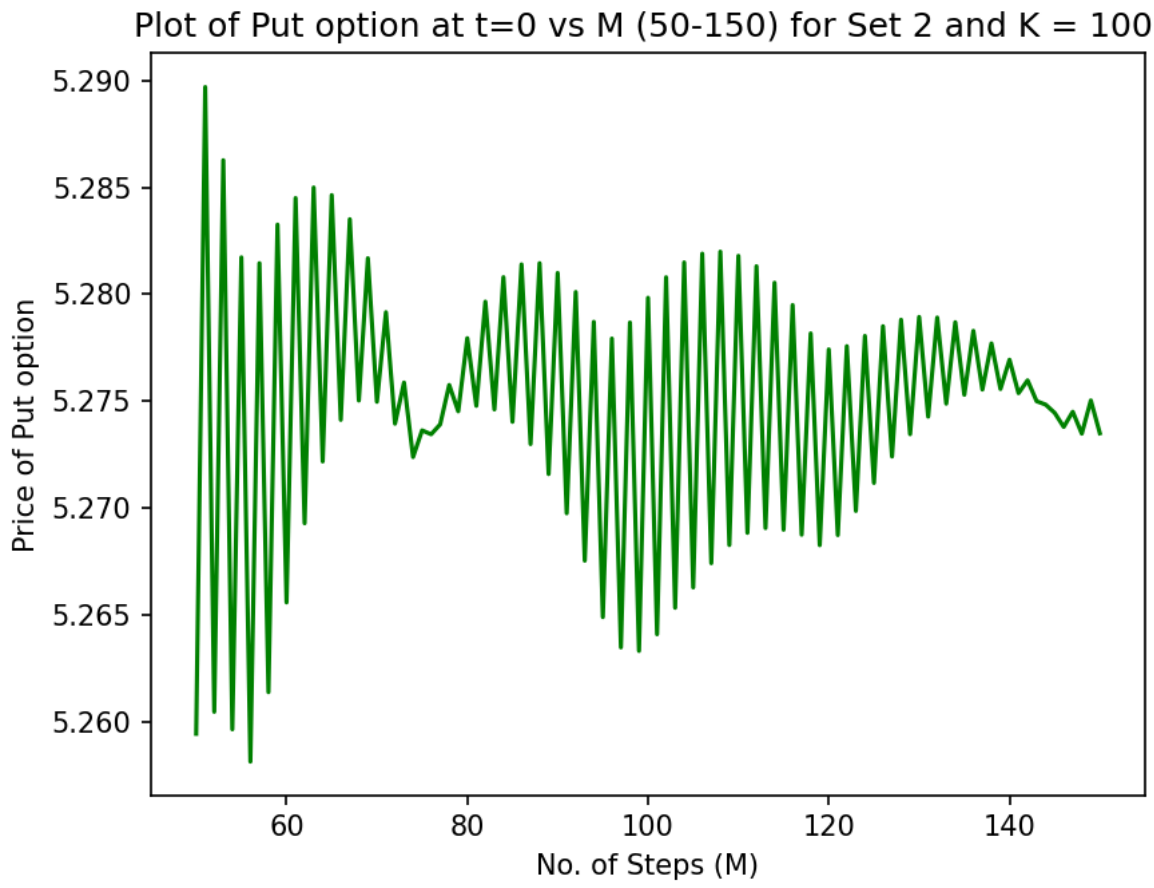

Plot of Call option at t=0 vs sig (0-0.3) for Set 2
Price of Call option
16
15
14
13
12
11
10
9
8
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Value of Sigma

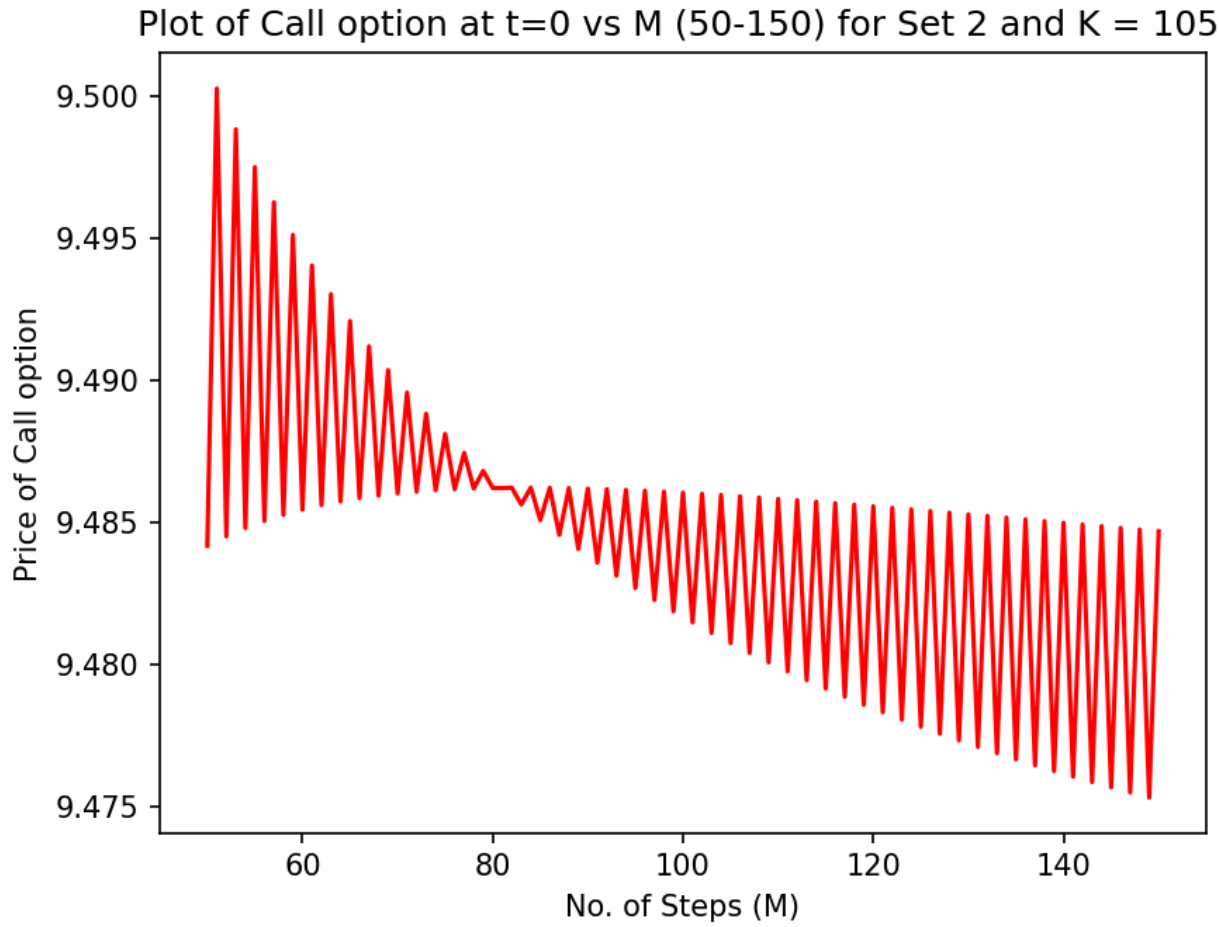

Plot of Put option at t=0 vs sig (0-0.3) for Set 2
Price of Put option
0
2
4
6
8
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Value of Sigma

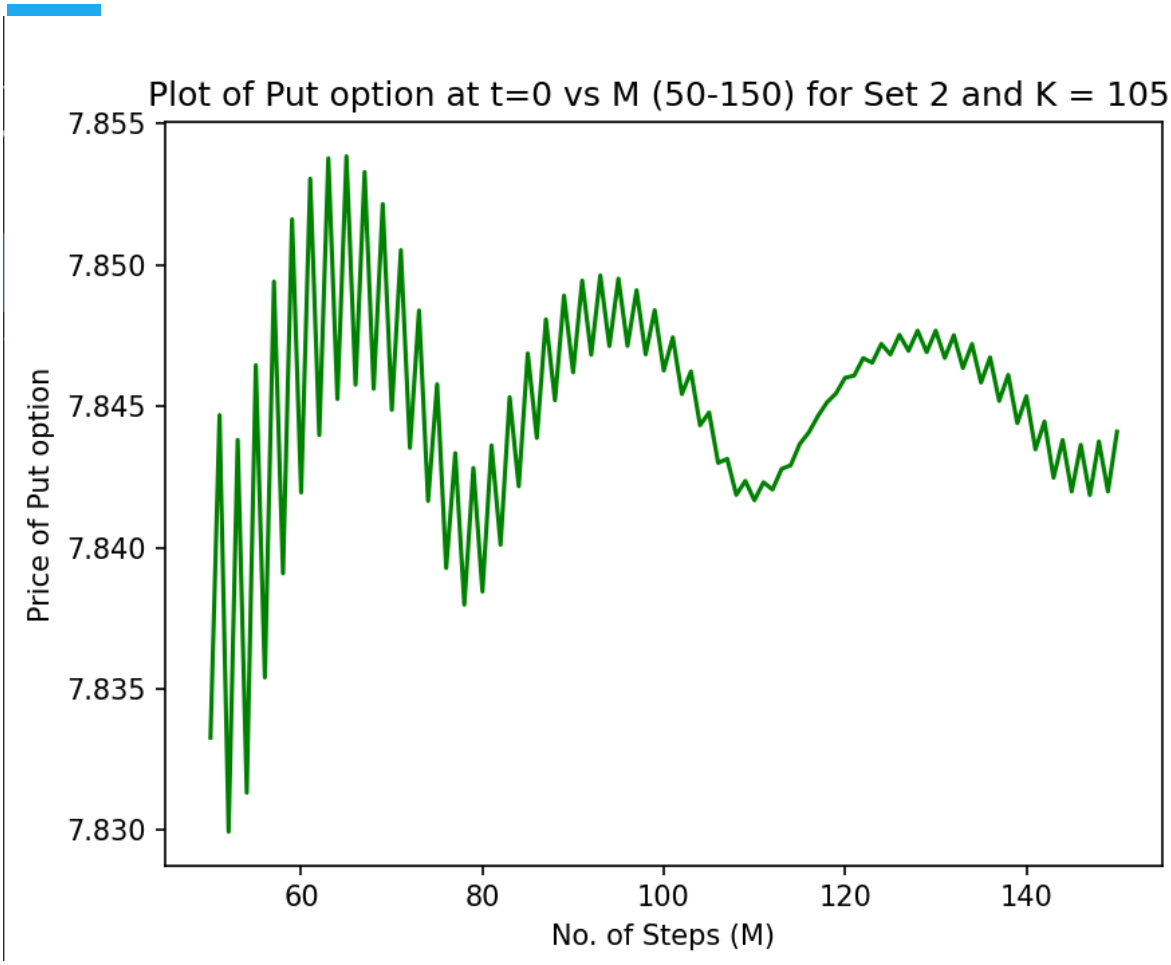

Plot of Call option at t=0 vs M (50-150) for Set 2 and K = 95
Price of Call option
15.20
15.19
15.18
15.17
15.16
15.15
60
80
100
120
140
No. of Steps (M)


Plot of Put option at t=0 vs M (50-150) for Set 2 and K = 95
Price of Put option
3.360
3.355
3.350
3.345
3.340
3.335
60
80
100
120
140
No. of Steps (M)


Plot of Call option at t=0 vs M (50-150) for Set 2 and K = 100
Price of Call Option
12.14
12.13
12.12
12.11
12.10
12.09
60
80
100
120
140
No. of Steps (M)


Plot of Put option at t=0 vs M (50-150) for Set 2 and K = 100
Price of Put option
5.290
5.285
5.280
5.275
5.270
5.265
5.260
60
80
100
120
140
No. of Steps (M)


Plot of Call option at t=0 vs M (50-150) for Set 2 and K = 105
9.500
9.495
9.490
9.485
9.480
9.475
Price of Call option
60
80
100
120
140
No. of Steps (M)


Plot of Put option at t=0 vs M (50-150) for Set 2 and K = 105
Price of Put option
No. of Steps (M)
7.855
7.850
7.845
7.840
7.835
7.830
60
80
100
120
140

Question 2.

Data given to determine the initial price of a ***loopback*** (European) option using the binomial algorithm are:

$S(0) = 100, T = 1, r = 8\%, \sigma = 20\%$

Also given u and d for this question:

$$u = e^{\sigma\sqrt{\Delta t}+\left(r-\frac{1}{2}\sigma^2\right)\Delta t}, d = e^{-\sigma\sqrt{\Delta t}+\left(r-\frac{1}{2}\sigma^2\right)\Delta t}$$

The payoff for the ***loopback*** option is given by:

$$V = \max_{0\leq i\leq M}(S(i)) - S(M)$$

(a) Using the basic binomial algorithm, we obtain the initial option price for different values of M as follows:

```
----------------------  sub-part(a)  ----------------------


*********  Executing for M = 5  *********

No arbitrage exists for 5
Initial Price of Loopback Option        = 9.119298985864683
Execution Time                  = 0.0 sec



*********  Executing for M = 10  *********

No arbitrage exists for 10
Initial Price of Loopback Option        = 10.080582906831
Execution Time                  = 0.0036995410919189453 sec



*********  Executing for M = 25  *********

No arbitrage exists for 25
Initial Price of Loopback Option        = 11.00349533564633
Execution Time                  = 242.44394779205322 sec
```

For **M = 50**, the basic binomial model will scalein time complexity as it works in $O(2^M)$. And thus we can't computationally handle this in python. An appropriate message is shownin the terminal to the user:

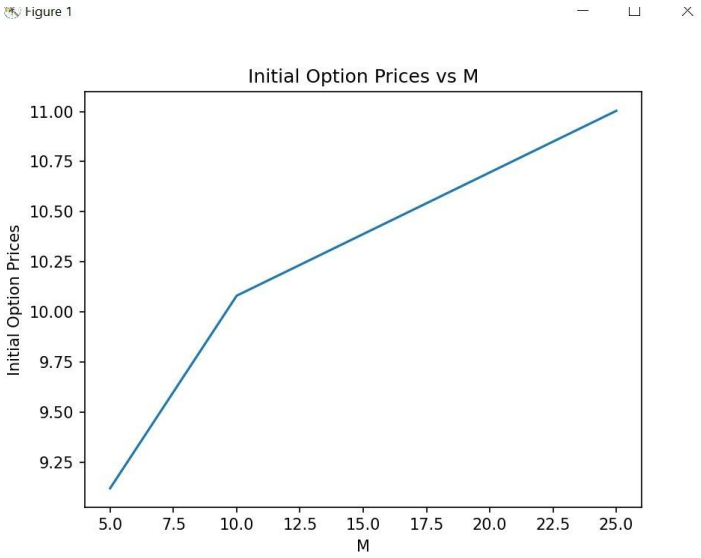
```
Due to complexity constraints, the initial value of the
loopback option price cannot be calculated using the basic
binomial algorithm for the case M = 25 and 50
```

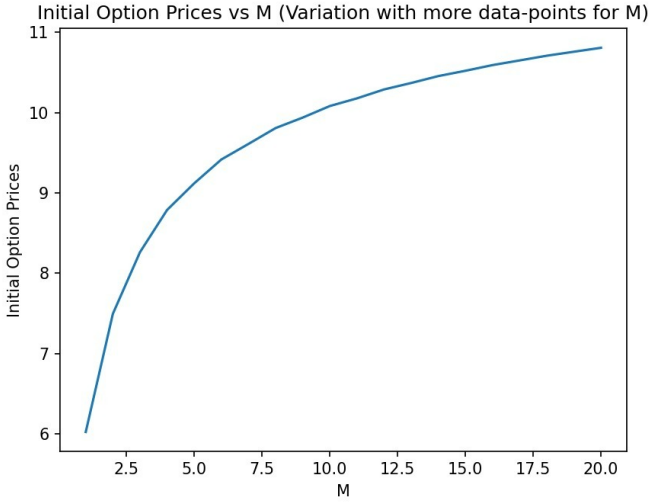
This issue will be addressed using a Markov based, computationally efficient binomial algorithm, in question 2.

(b) The following conclusions can be drawn from the comparison of initial loopback option prices:

- From the graph below, ***it is seen that the initial values for the***

loopback option tend to converge.





- Also, for the initial values of M (the values have been observed as far as 15), ***an increasing pattern of the initial option value with M is observed***.

(c) The option values at all intermediate time points for M = 5 are shown in the table below:

```
----------------------  sub-part(c)  ----------------------
At t = 0
Index no = 0    Price = 9.119298985864683

At t = 1
Index no = 0    Price = 9.027951165547751
Index no = 1    Price = 9.504839866450853

At t = 2
Index no = 0    Price = 8.548076183576441
Index no = 1    Price = 9.799118753547026
Index no = 2    Price = 7.147915756774744
Index no = 3    Price = 12.168664659721792

At t = 3
Index no = 0    Price = 7.416771005131011
Index no = 1    Price = 9.955271272957816
Index no = 2    Price = 6.201916453882752
Index no = 3    Price = 13.712862965988533
Index no = 4    Price = 6.201916453882752
Index no = 5    Price = 8.32461466963314
Index no = 6    Price = 7.14841820819012
Index no = 7    Price = 17.582062714095418

At t = 4
Index no = 0    Price = 5.501638813873981
Index no = 1    Price = 9.571391531700229
Index no = 2    Price = 4.600479677676438
Index no = 3    Price = 15.631851880479827
Index no = 4    Price = 4.600479677676438
Index no = 5    Price = 8.003613780975444
Index no = 6    Price = 6.6808429992566465
Index no = 7    Price = 21.18808934534565
Index no = 8    Price = 4.600479677676438
Index no = 9    Price = 8.003613780975444
Index no = 10   Price = 3.8469288844156075
Index no = 11   Price = 13.071380970928788
Index no = 12   Price = 3.8469288844156075
Index no = 13   Price = 10.68090442602997
Index no = 14   Price = 10.68090442602997
Index no = 15   Price = 25.051229457037028
```

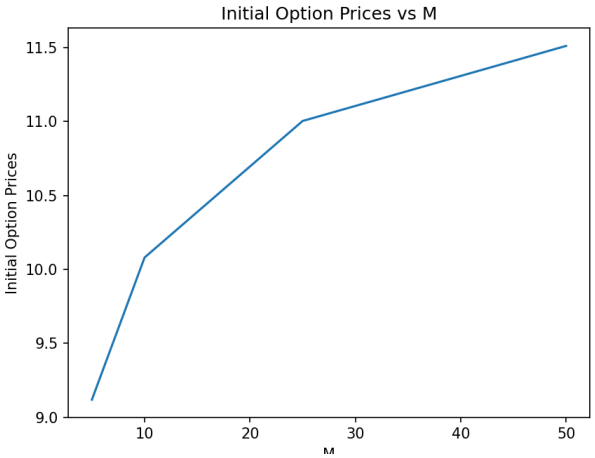
```
Index no = 14   Price = 10.68090442602997
Index no = 15   Price = 25.051229457037028

At t = 5
Index no = 0    Price = 0.0
Index no = 1    Price = 11.1814131177845O1
Index no = 2    Price = 0.0
Index no = 3    Price = 19.452691543130413
Index no = 4    Price = 0.0
Index no = 5    Price = 9.349916553291678
Index no = 6    Price = 6.374517470614265
Index no = 7    Price = 25.39456347506497
Index no = 8    Price = 0.0
Index no = 9    Price = 9.349916553291678
Index no = 10   Price = 0.0
Index no = 11   Price = 16.266373556657385
Index no = 12   Price = 0.0
Index no = 13   Price = 13.578002496522686
Index no = 14   Price = 13.578002496522686
Index no = 15   Price = 29.48259712227059
Index no = 16   Price = 0.0
Index no = 17   Price = 9.349916553291678
Index no = 18   Price = 0.0
Index no = 19   Price = 16.266373556657385
Index no = 20   Price = 0.0
Index no = 21   Price = 7.81841602958671 44
Index no = 22   Price = 5.330382286201839
Index no = 23   Price = 21.234976911949744
Index no = 24   Price = 0.0
Index no = 25   Price = 7.81841602958671 44
Index no = 26   Price = 2.9013504971397026
Index no = 27   Price = 18.805945122887607
Index no = 28   Price = 2.9013504971397026
Index no = 29   Price = 18.805945122887607
Index no = 30   Price = 18.805945122887607
Index no = 31   Price = 32.10539403853048
```

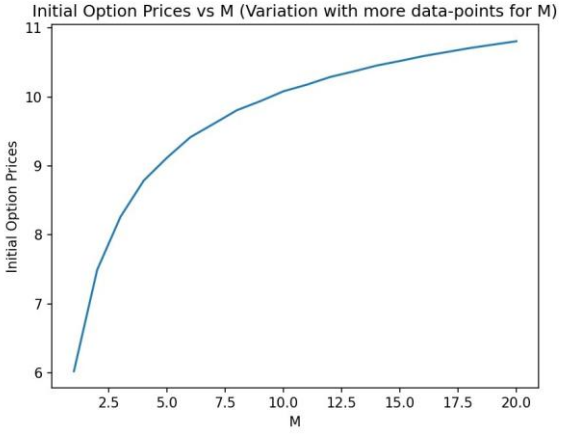
Question 3

Problem 1 is repeated using a Markov based computationally efficient algorithm. **In this case, we make use of dynamic programming, and we use a map (in C++) or a dictionary (in python), to store the payoffs and keep a track of the max Stock price, even as we explore all the paths in our binomial model using a recursive function.**

a)

```
*********  Executing for M = 5  *********

No arbitrage exists for M = 5
Initial Price of Loopback Option        = 9.119298985864683
Execution Time                          = 0.0009968280792236328 sec


*********  Executing for M = 10  *********

No arbitrage exists for M = 10
Initial Price of Loopback Option        = 10.080582906831
Execution Time                          = 0.0010673999786376953 sec


*********  Executing for M = 25  *********

No arbitrage exists for M = 25
Initial Price of Loopback Option        = 11.00349533564633
Execution Time                          = 0.059099674224853516 sec


*********  Executing for M = 50  *********

No arbitrage exists for M = 50
Initial Price of Loopback Option        = 11.510862222177286
Execution Time                          = 3.8091089725494385 sec
```


Initial Option Prices vs M
Initial Option Prices
11.5
11.0
10.5
10.0
9.5
9.0
10
20
30
40
50
M


Initial Option Prices vs M (Variation with more data-points for M)
Initial Option Prices
11
10
9
8
7
6
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
M

```
----------------------  sub-part(c)  ----------------------
At t = 0
Intermediate state = (100, 100)         Price = 9.119298985864683

At t = 1
Intermediate state = (110.676651999383, 110.676651999383)               Price = 9.027951165547751
Intermediate state = (92.54800352077254, 100)           Price = 9.504839866450853

At t = 2
Intermediate state = (122.49321297792528, 122.49321297792528)           Price = 8.548076183576441
Intermediate state = (102.42903178906215, 110.676651999383)             Price = 9.799118753547026
Intermediate state = (102.42903178906214, 102.42903178906214)           Price = 7.147915756774744
Intermediate state = (85.65132955680926, 100)           Price = 12.168664659721792

At t = 3
Intermediate state = (135.57138705044142, 135.57138705044142)           Price = 7.416771005131011
Intermediate state = (113.3650230595177, 122.49321297792528)            Price = 9.955271272957816
Intermediate state = (113.3650230595177, 113.3650230595177)             Price = 6.201916453882752
Intermediate state = (94.79602394643446, 110.676651999383)              Price = 13.712862965988533
Intermediate state = (113.36502305951768, 113.36502305951768)           Price = 6.201916453882752
Intermediate state = (94.79602394643445, 102.42903178906214)            Price = 8.32461466963314
Intermediate state = (94.79602394643445, 100)           Price = 7.14841820819012
Intermediate state = (79.26859549382432, 100)           Price = 17.582062714095418

At t = 4
Intermediate state = (150.04587225655362, 150.04587225655362)           Price = 5.501638813873981
Intermediate state = (125.46861206060268, 135.57138705044142)           Price = 9.571391531700229
Intermediate state = (125.46861206060268, 125.46861206060268)           Price = 4.600479677676438
Intermediate state = (104.91706553244704, 122.49321297792528)           Price = 15.631851880479827
Intermediate state = (104.91706553244704, 113.3650230595177)            Price = 8.003613780975444
Intermediate state = (104.91706553244704, 110.676651999383)             Price = 6.6808429992566465
```

```
At t = 4
Intermediate state = (150.04587225655362, 150.04587225655362)          Price = 5.501638813873981
Intermediate state = (125.46861206060268, 135.57138705044142)          Price = 9.571391531700229
Intermediate state = (125.46861206060268, 125.46861206060268)          Price = 4.600479677676438
Intermediate state = (104.91706553244704, 122.49321297792528)          Price = 15.631851880479827
Intermediate state = (104.91706553244704, 113.3650230595177)           Price = 8.003613780975444
Intermediate state = (104.91706553244704, 110.676651999383)            Price = 6.6808429992566465
Intermediate state = (87.73182757949854, 110.676651999383)             Price = 21.18808934534565
Intermediate state = (125.46861206060267, 125.46861206060267)          Price = 4.600479677676438
Intermediate state = (104.91706553244703, 113.36502305951768)          Price = 8.003613780975444
Intermediate state = (104.91706553244701, 104.91706553244701)          Price = 3.8469288844156075
Intermediate state = (87.73182757949853, 102.42903178906214)           Price = 13.071380970928788
Intermediate state = (87.73182757949853, 100)          Price = 10.68090442602997
Intermediate state = (73.36150254849147, 100)          Price = 25.051229457037028

At t = 5
Intermediate state = (166.06574787682462, 166.06574787682462)          Price = 0.0
Intermediate state = (138.86445913876912, 150.04587225655362)          Price = 11.181413117784501
Intermediate state = (138.8644591387691, 138.8644591387691)            Price = 0.0
Intermediate state = (116.118695507311, 135.57138705044142)            Price = 19.452691543130413
Intermediate state = (116.118695507311, 125.46861206060268)            Price = 9.349916553291678
Intermediate state = (116.11869550731102, 122.49321297792528)          Price = 6.374517470614265
Intermediate state = (97.09864950286031, 122.49321297792528)           Price = 25.39456347506497
Intermediate state = (116.11869550731102, 116.11869550731102)          Price = 0.0
Intermediate state = (97.09864950286031, 113.3650230595177)            Price = 16.266373556657385
Intermediate state = (97.09864950286031, 110.676651999383)             Price = 13.578002496522686
Intermediate state = (81.1940548771124, 110.676651999383)              Price = 29.48259712227059
Intermediate state = (116.11869550731099, 125.46861206060267)          Price = 9.349916553291678
Intermediate state = (116.11869550731099, 116.11869550731099)          Price = 0.0
Intermediate state = (97.0986495028603, 113.36502305951768)            Price = 16.266373556657385
```

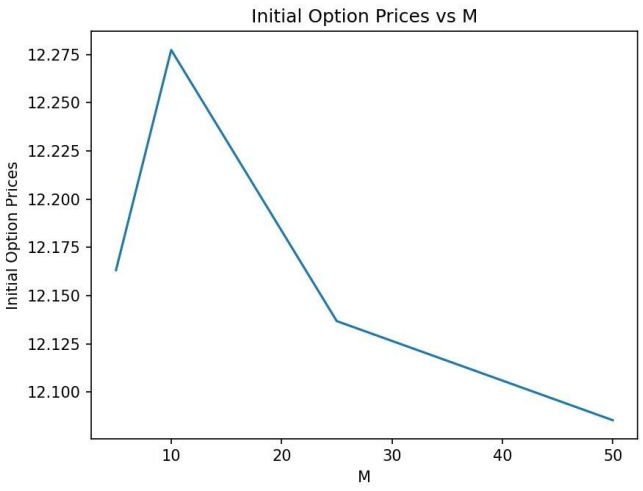
```
At t = 5
Intermediate state = (166.06574787682462, 166.06574787682462)            Price = 0.0
Intermediate state = (138.86445913876912, 150.04587225655362)            Price = 11.181413117784501
Intermediate state = (138.8644591387691, 138.8644591387691)              Price = 0.0
Intermediate state = (116.118695507311, 135.57138705044142)              Price = 19.452691543130413
Intermediate state = (116.118695507311, 125.46861206060268)              Price = 9.349916553291678
Intermediate state = (116.11869550731102, 122.49321297792528)            Price = 6.374517470614265
Intermediate state = (97.09864950286031, 122.49321297792528)             Price = 25.39456347506497
Intermediate state = (116.11869550731102, 116.11869550731102)            Price = 0.0
Intermediate state = (97.09864950286031, 113.3650230595177)              Price = 16.266373556657385
Intermediate state = (97.09864950286031, 110.676651999383)               Price = 13.578002496522686
Intermediate state = (81.1940548771124, 110.676651999383)                Price = 29.48259712227059
Intermediate state = (116.11869550731099, 125.46861206060267)            Price = 9.349916553291678
Intermediate state = (116.11869550731099, 116.11869550731099)            Price = 0.0
Intermediate state = (97.0986495028603, 113.36502305951768)              Price = 16.266373556657385
Intermediate state = (116.11869550731097, 116.11869550731097)            Price = 0.0
Intermediate state = (97.0986495028603, 104.91706553244701)              Price = 7.8184160295867144
Intermediate state = (97.0986495028603, 102.42903178906214)              Price = 5.330382286201839
Intermediate state = (81.19405487711239, 102.42903178906214)             Price = 21.234976911949744
Intermediate state = (97.0986495028603, 100)            Price = 2.9013504971397026
Intermediate state = (81.19405487711239, 100)           Price = 18.805945122887607
Intermediate state = (67.89460596146952, 100)           Price = 32.10539403853048
```

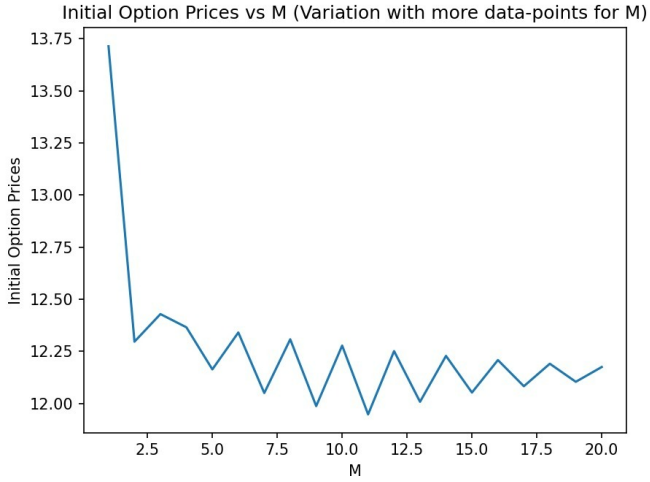
The 2 algorithms, i.e, Basic Binomial and Efficient Binomial (Markov Based) can be compared as follows:

1. **Time complexity:**
    - The basic binomial algorithm has a time complexity of the order $O(2^M)$
    - The Markov based has **polynomial** time complexity.
2. **Permissible values of M:**
    - The basic binomial algorithm can take values of **M up to 20** (**or up to 25 in C++)**, after which it becomes computationally inefficient for calculations.
    - The Markov based algorithm can handle **M values up to 50** or more, because of its computational efficiency.
3. **Computational time (measured for M = 15 in python):**
    - The basic binomial algorithm takes around **0.7-0.8 seconds** to run once, given all the input values with M took 15.
    - The Markov based algorithm takes around **0.004 seconds** to run once given all the input values and M took 15.

Question 4

```
####################  Unoptimised Binomial Algorithm executing  ####################
No arbitrage exists for M = 5
European Call Option            = 12.16318594676458
Execution Time                  = 0.0 sec

No arbitrage exists for M = 10
European Call Option            = 12.27732781922299
Execution Time                  = 0.0029909610748291016 sec

No arbitrage exists for M = 25
European Call Option            = 12.136745963232949
Execution Time                  = 86.05734705924988 sec



####################  Efficient Binomial Algorithm executing (Markov Based)  ####################
No arbitrage exists for M = 5
European Call Option            = 12.163185946764584
Execution Time                  = 0.002786397933959961 sec

No arbitrage exists for M = 10
European Call Option            = 12.277327819222982
Execution Time                  = 0.0 sec

No arbitrage exists for M = 25
European Call Option            = 12.136745963232947
Execution Time                  = 0.00024819374084472656 sec

No arbitrage exists for M = 50
European Call Option            = 12.0853615100722
```






```
-----------------------  sub-part(c)  ----------------
At t = 0
Index no = 0    Price = 12.163185946764584

At t = 1
Index no = 0    Price = 18.65868251160212
Index no = 1    Price = 6.0592900974208455

At t = 2
Index no = 0    Price = 27.525444303544514
Index no = 1    Price = 10.392778619897372
Index no = 2    Price = 1.9207528986659217

At t = 3
Index no = 0    Price = 38.72072884252166
Index no = 1    Price = 17.21677529537563
Index no = 2    Price = 3.9032313677700126
Index no = 3    Price = 0.0

At t = 4
Index no = 0    Price = 51.633140251025104
Index no = 1    Price = 27.055880055074176
Index no = 2    Price = 7.9318974975518906
Index no = 3    Price = 0.0
Index no = 4    Price = 0.0

At t = 5
Index no = 0    Price = 66.06574787682459
Index no = 1    Price = 38.86445913876909
Index no = 2    Price = 16.118695507311017
Index no = 3    Price = 0
```

Similar to problems 1 and 2 we compute the initial option price of the European Call Option with 2 different algorithms - Basic Binomial and

Markov-based efficient algorithm.

- **In the case of Basic Binomial, we use recursion to take explore all possible paths for the Stock price using the Binomial model.**
- **In the case of the Markov-based algorithm, we use dynamic programming as in the case of Question 2 and make changes only to the payoff and the key for the map/dictionary which now is {n, count of ups}.**

The 2 algorithms, i.e, Basic Binomial and Efficient Binomial (Markov Based) can be compared as follows:

1. **Time complexity:**
    - The basic binomial algorithm has a time complexity of the order $O(2^M)$.
    - The Markov based has a time complexity of the order $O(m^2)$.
2. **Permissible values of M:**
    - The basic binomial algorithm can take values of **M up to 20 (or up to 25 in C++)**, after which it becomes computationally inefficient for calculations.
    - The Markov based algorithm can handle **M values up to 1000 in C++ (or up to 500 in python after which the max recursion depth is exceeded)**, because of its computational efficiency.
3. **Computational time (measured for M = 15 in python):**
    - The Basic Binomial algorithm takes around **0.85 seconds** to run once given all the input values and M took 15.

- The Markov based algorithm takes around **0.20 seconds** to run once given all the input values and M took 15.