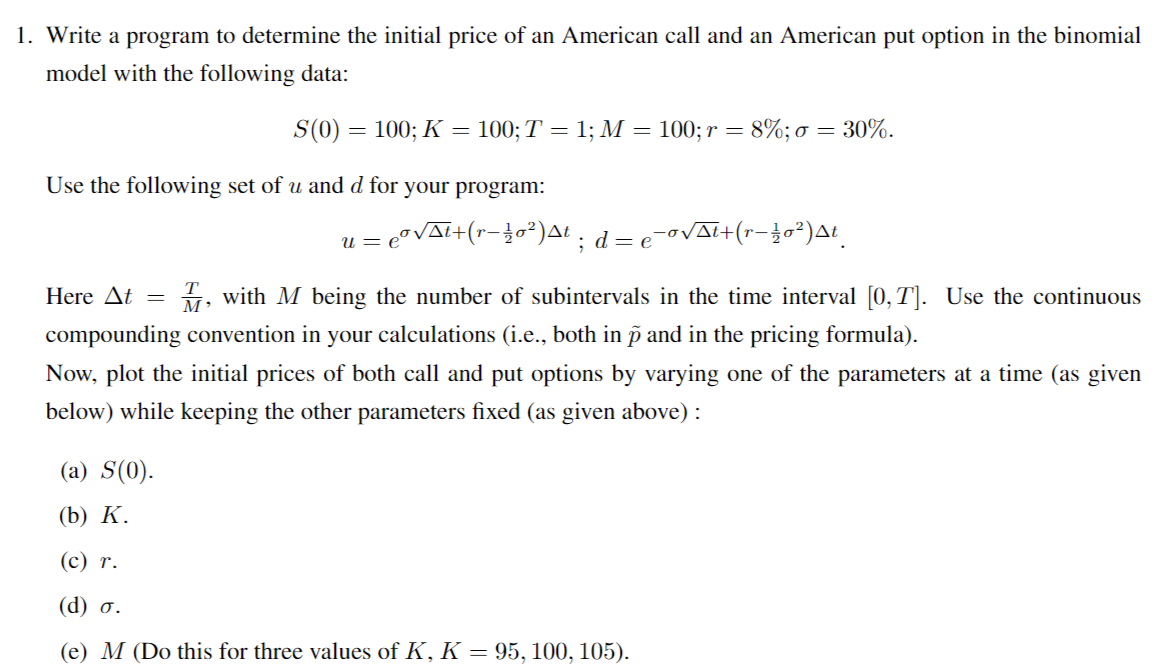
**Financial Engineering Lab MA – 374 Lab – 3**

**Name –** Rasesh Srivastava

**Roll Number –** 210123072

**Branch –** Mathematics and Computing

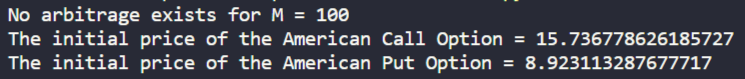
Question 1:



Using binomial model with continuous compounding convention:

The initial price of the American Call Option = 15.736778626185727

The initial price of the American Put Option = 8.923113287677717



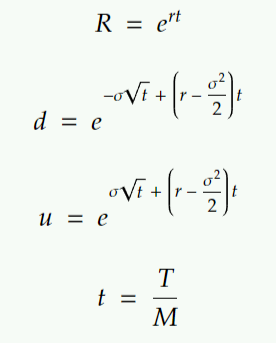
**No-arbitrage Condition:**

The code checks for the arbitrage possibilities using the following conditions necessary for the market to be arbitrage free -



That is, for no arbitrage opportunity to exist, following relations must hold true:

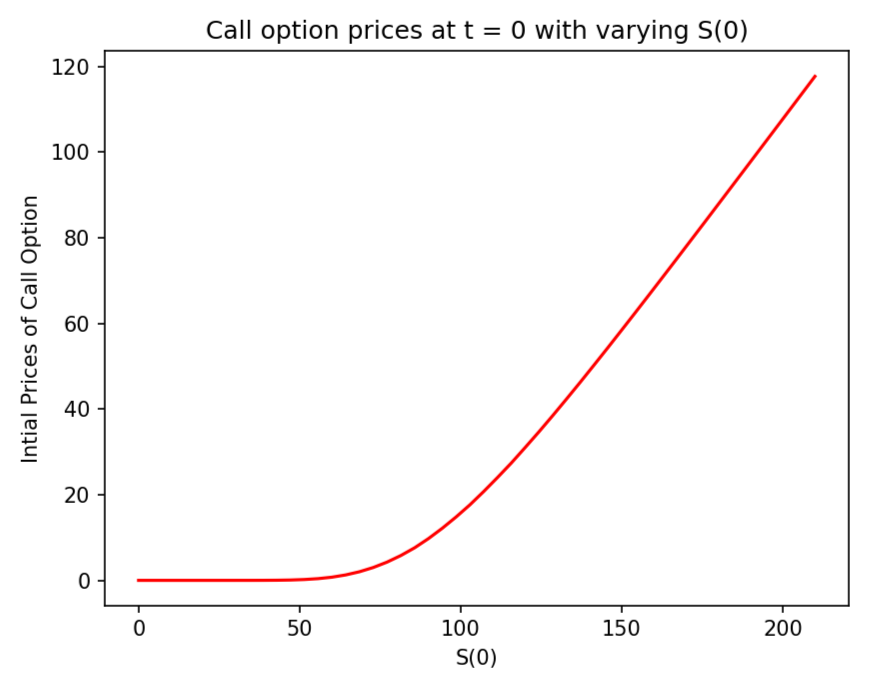
 where,

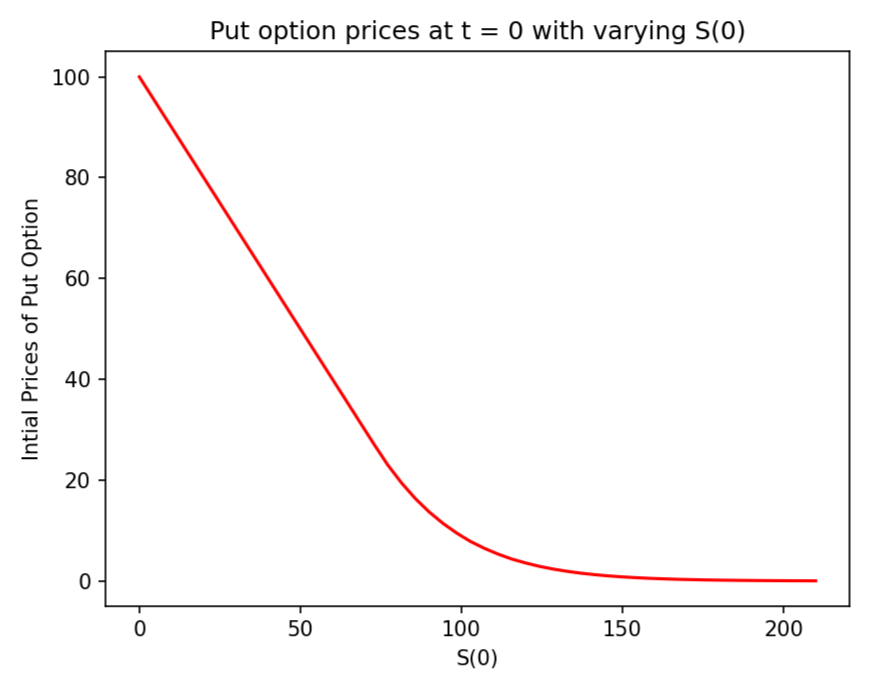


Since continuous compounding is followed and the final amount in continuous compounding is present value times ert, so R = ert.

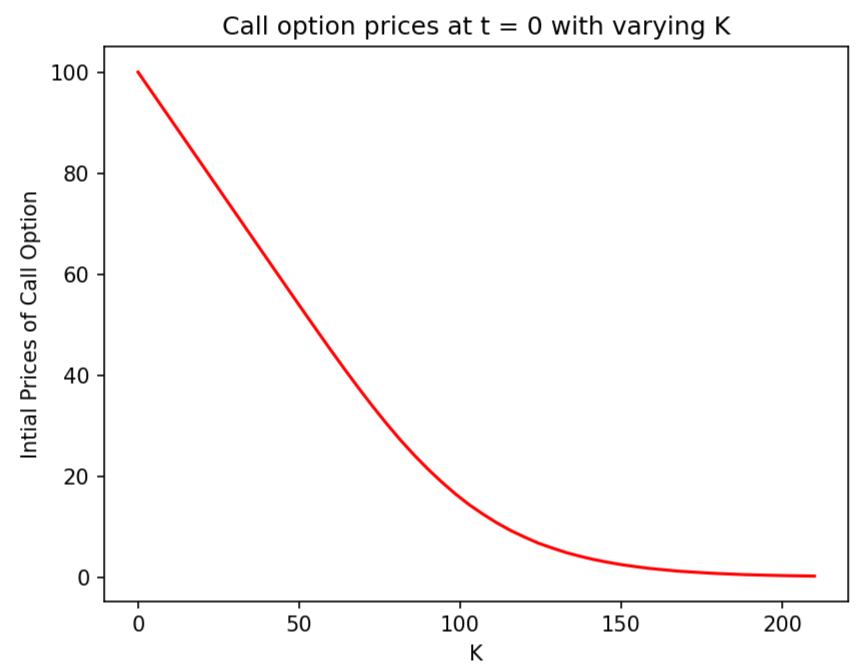
Plotting the initial prices of both call and put options by varying one of the parameters at a time while keeping the other parameters fixed:

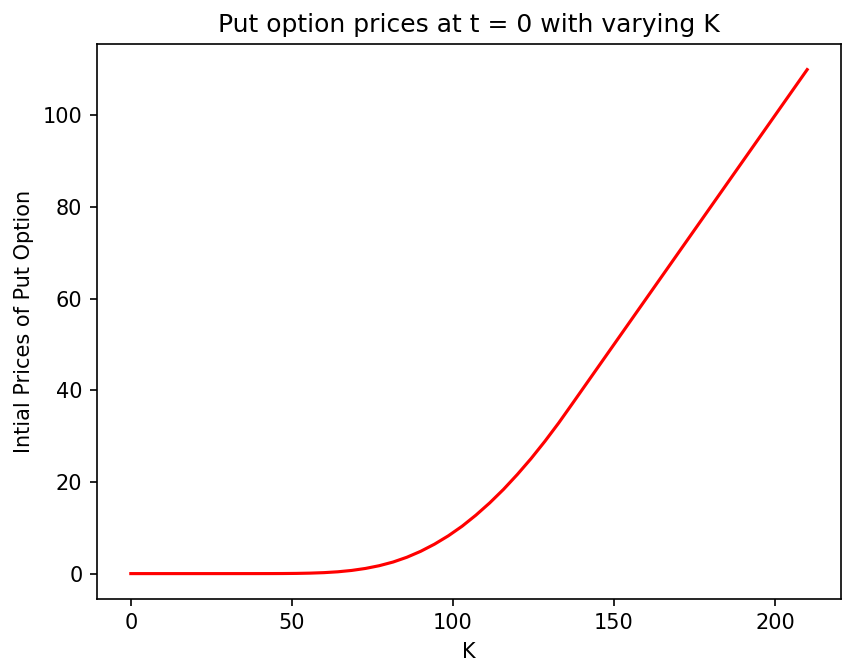
(a)



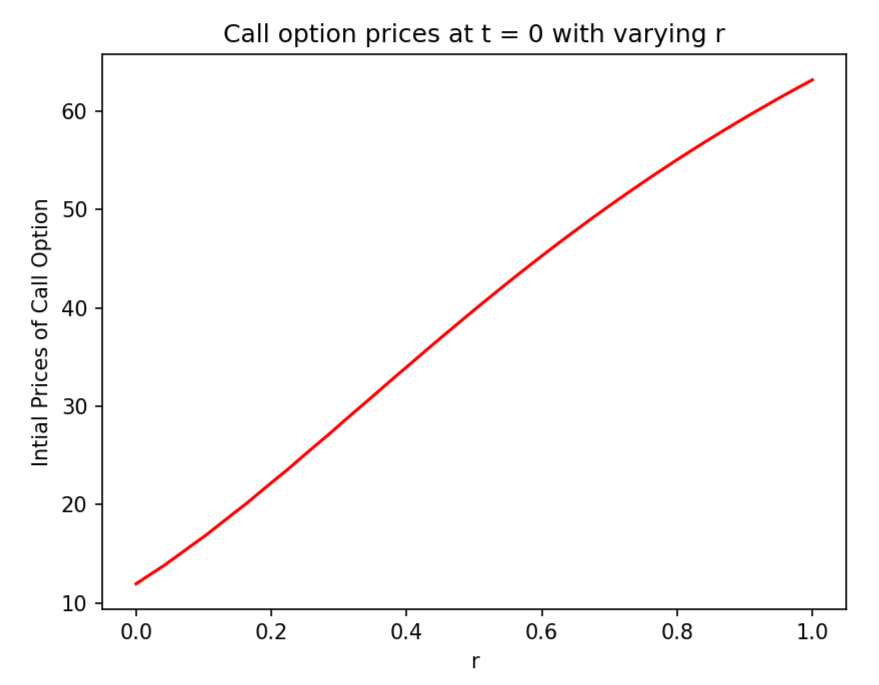


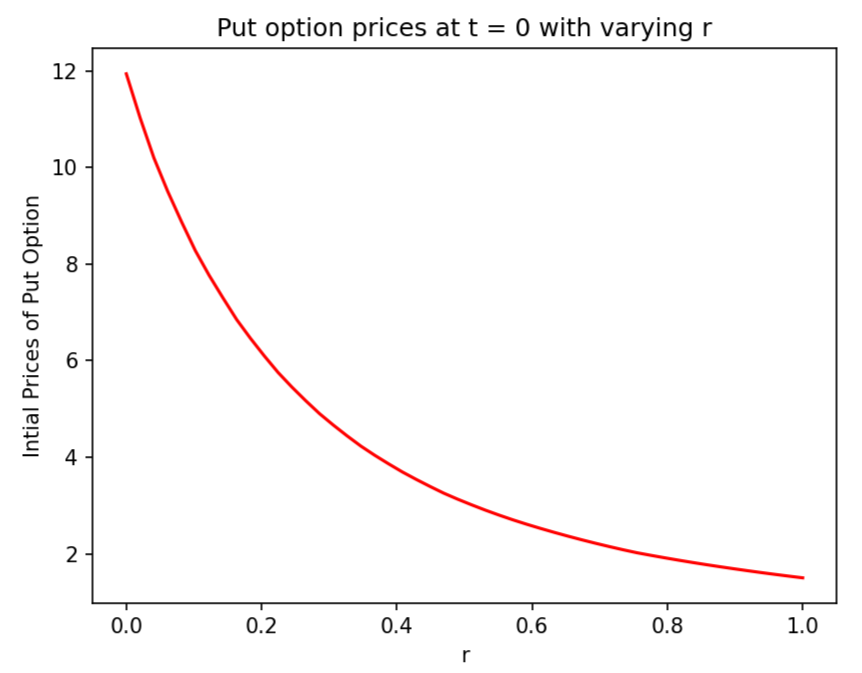
(b)



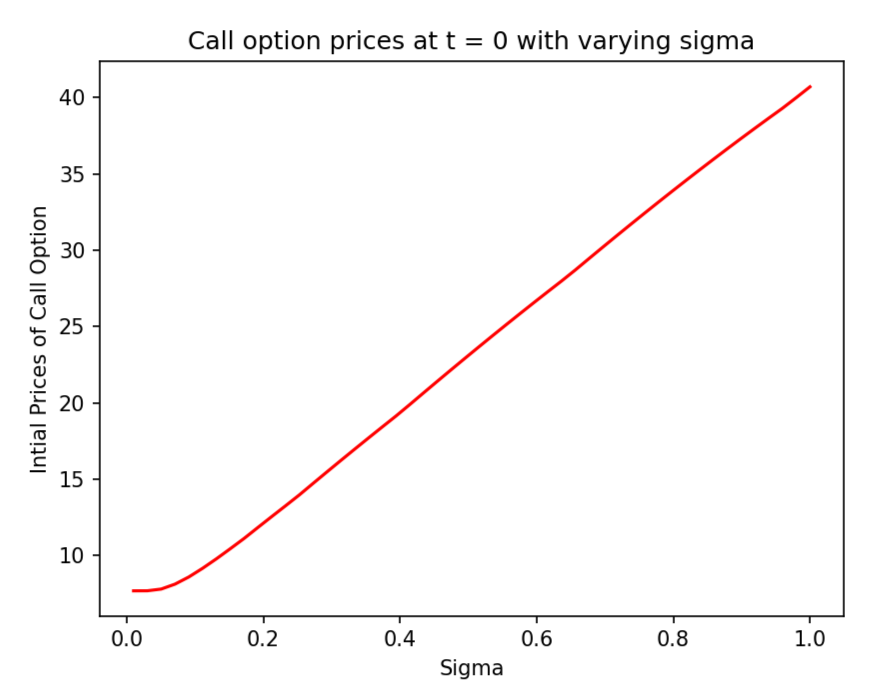


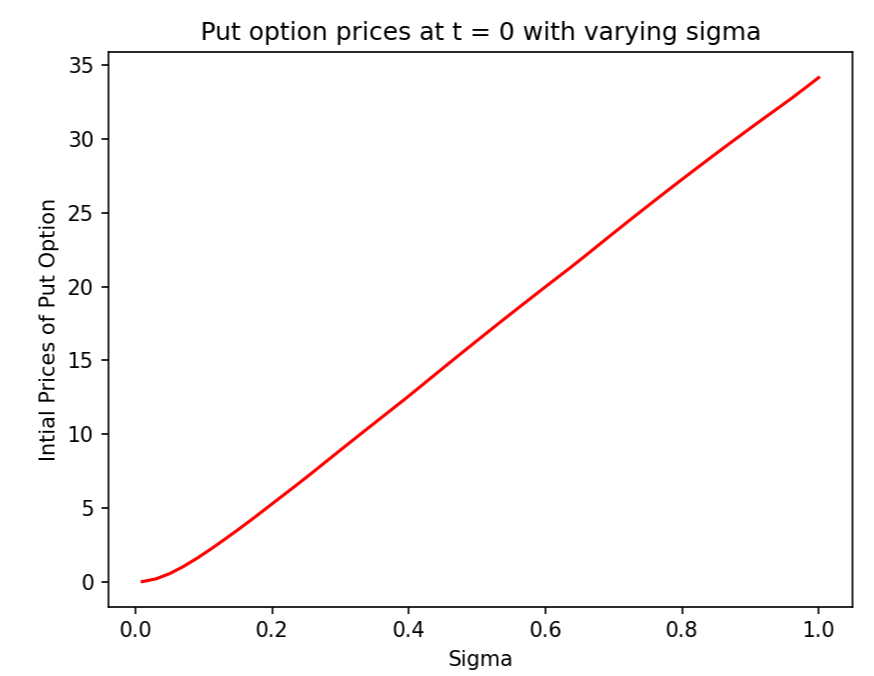
(c)



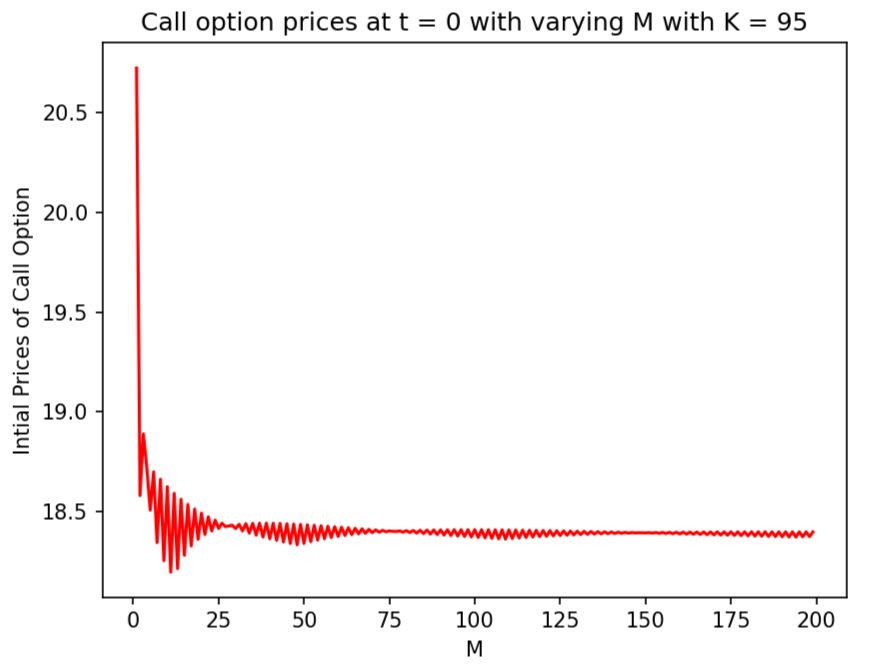


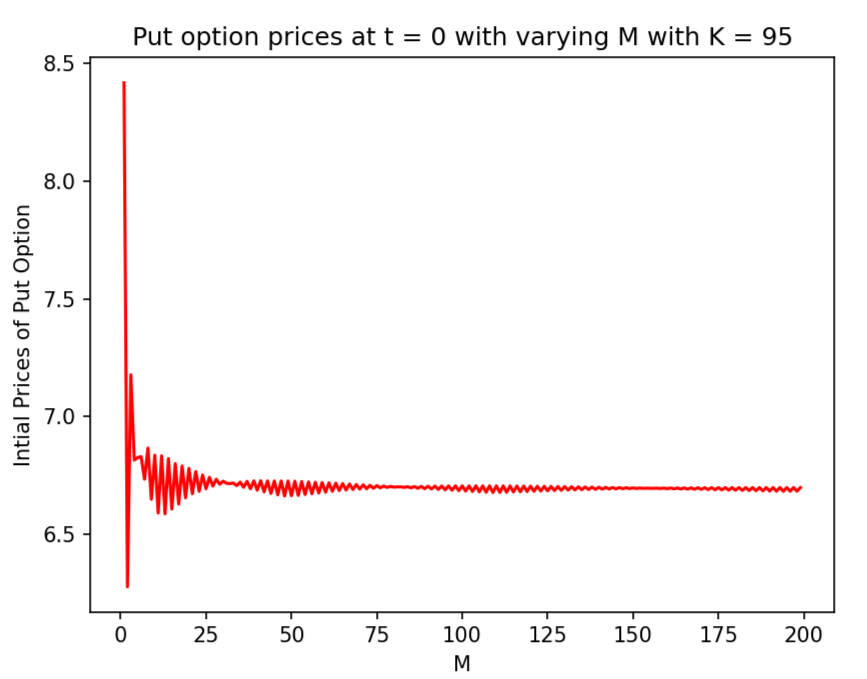
(d)

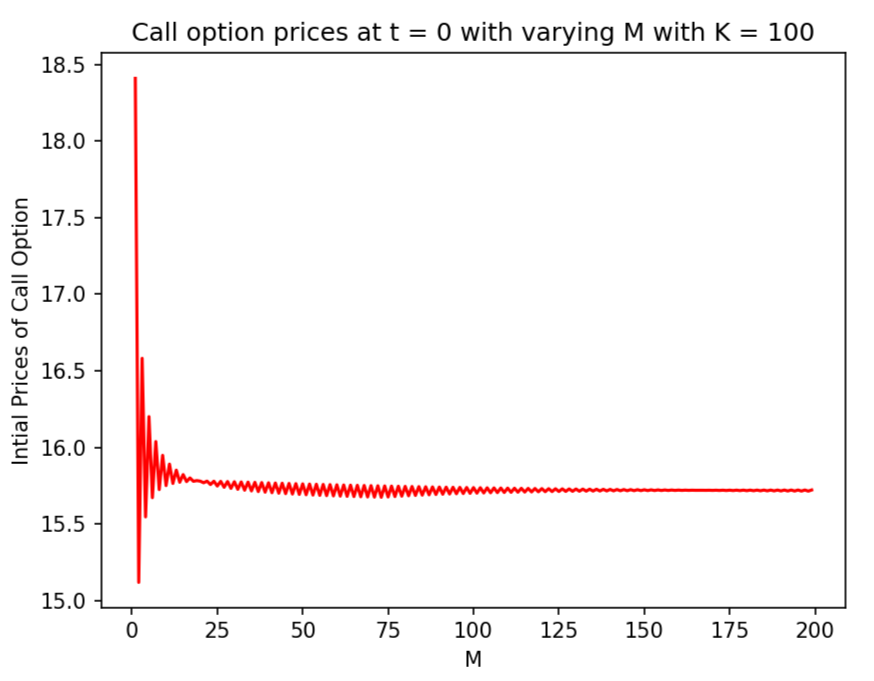


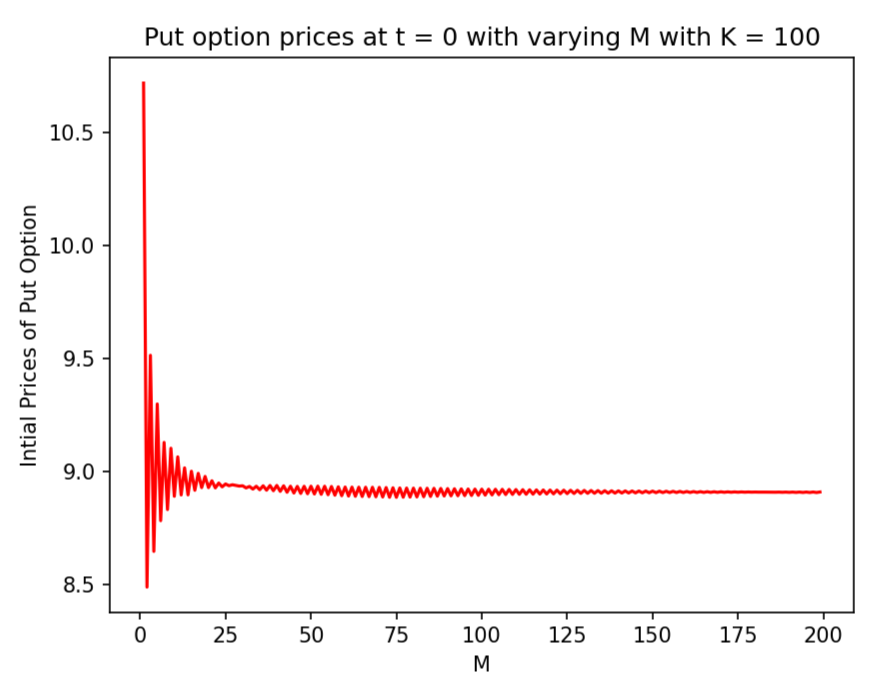


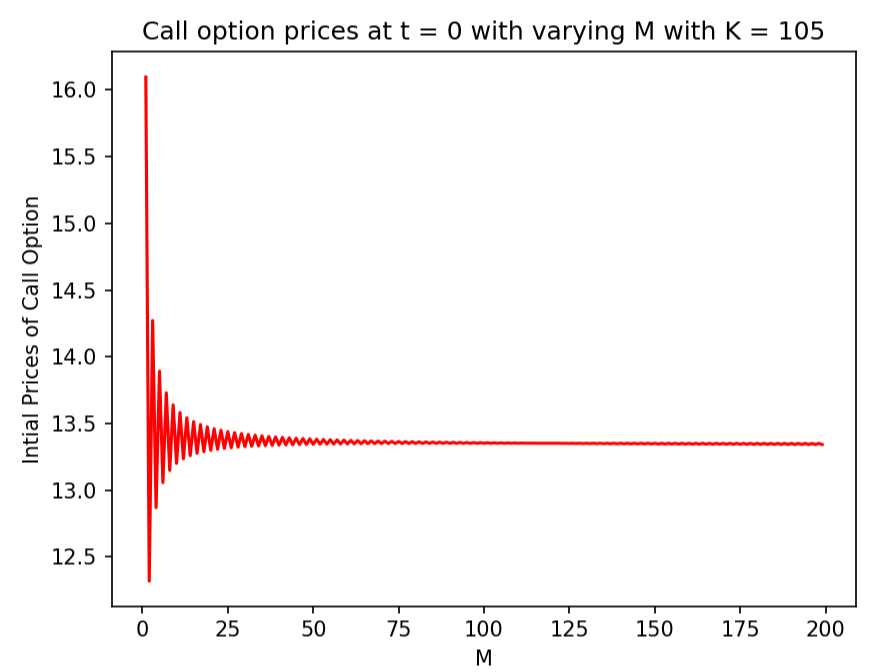
(e)

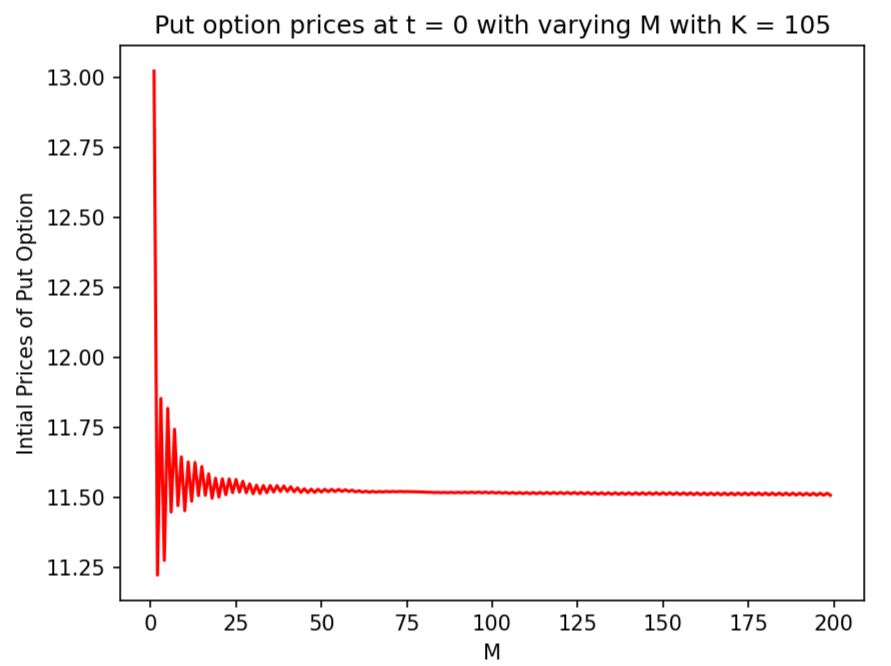








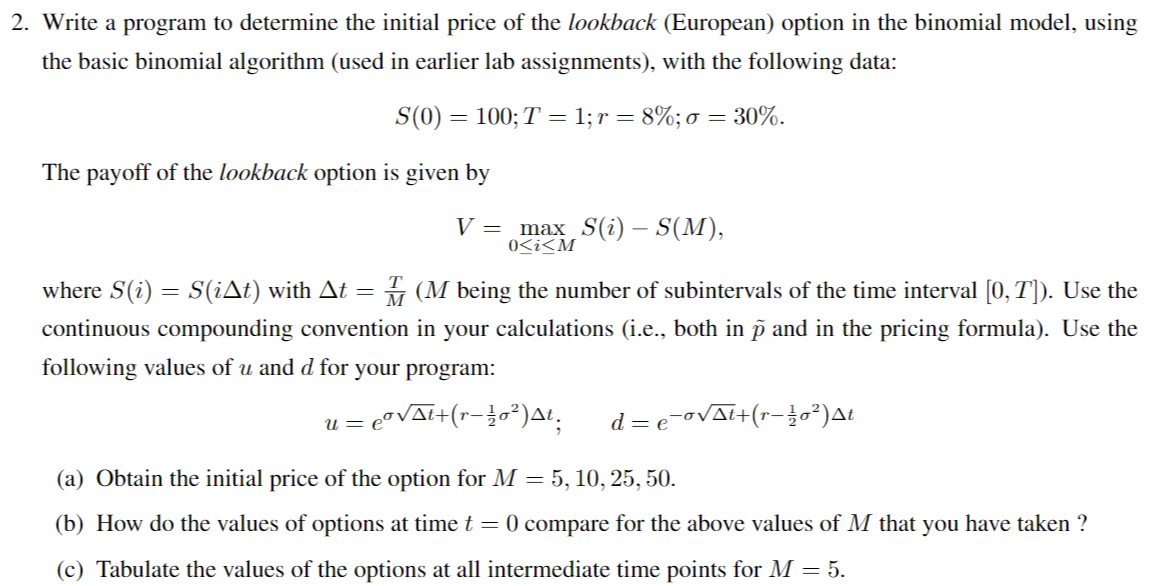




**Observations:**

* The price of the Call Option has positive dependence on S(0), and the price of the Put Option price has negative dependence on S(0).
* The price of the Put Option has positive dependence on K, and the price of the Call Option price has negative dependence on K.
* The price of the Call Option has positive dependence on r, and the price of the Put Option price has negative dependence on r.

Question 2:



(a)

The initial prices of the lookback (European) option in the binomial model, using the basic binomial algorithm for different values of M are as follows:

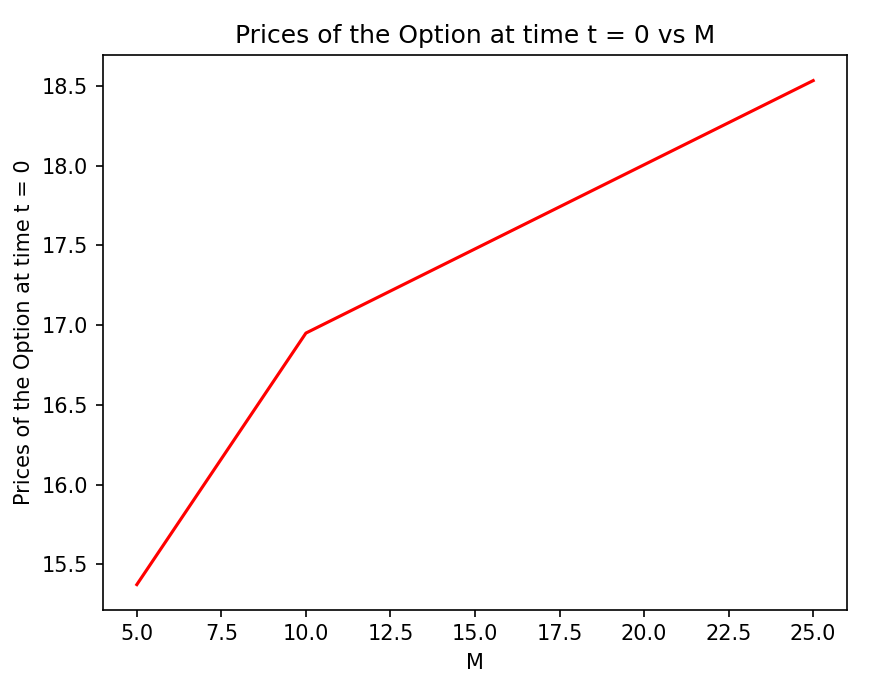
|  |  |  |  |
| --- | --- | --- | --- |
| **Serial Number** | **M** | **Initial Price of Lookback European Option** | **Execution Time in seconds** |
| 1 | 5 | 15.372952215663778 | 0.0011849403381347656 |
| 2 | 10 | 16.95034049177767 | 0.0050394535064697266 |
| 3 | 25 | 18.533781500094165 | 288.1385660171509 |
| 4 | 26 | 18.590003349885308 | 561.5937082767487 |
| 5 | 27 | 18.6418127619738 | 1192.9899275302887 |
| 6 | 50 | Could not compute | Could not compute |

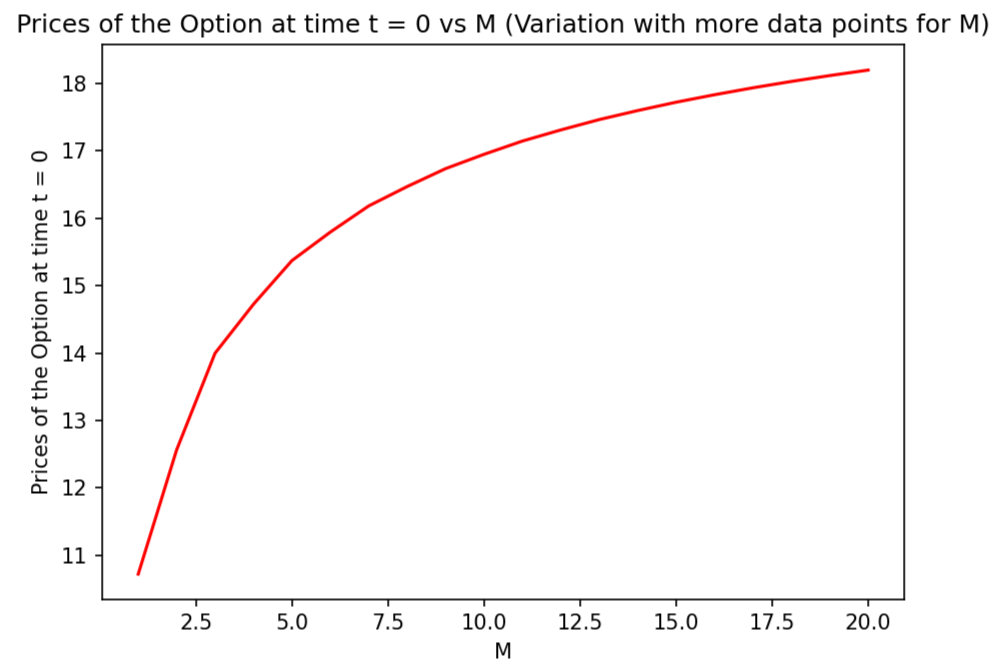
**Observations:**

For M = 50, the written algorithm is not able to calculate the option price in reasonably small time because the algorithm used has exponential time complexity. For M = 26 and 27, we can see that as we increase M by 1, the execution time almost doubles. So, computation for M = 50 is infeasible with this algorithm for finding the initial price of the European loopback option.

(b)

Comparison of the values of options at time t = 0 for the values of M that I have taken is done using the following plots:



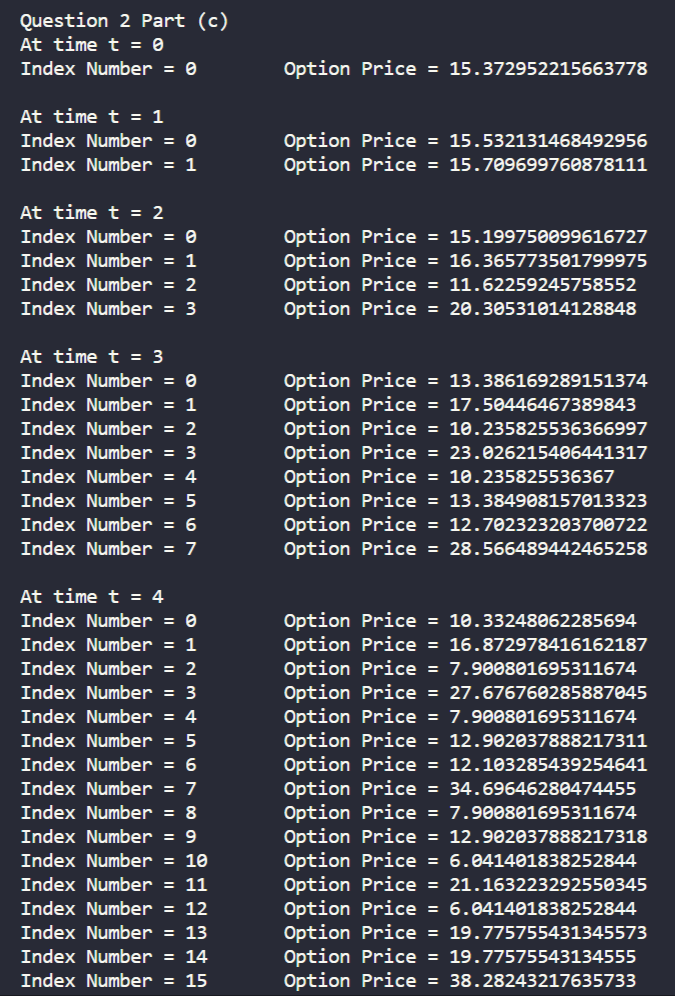


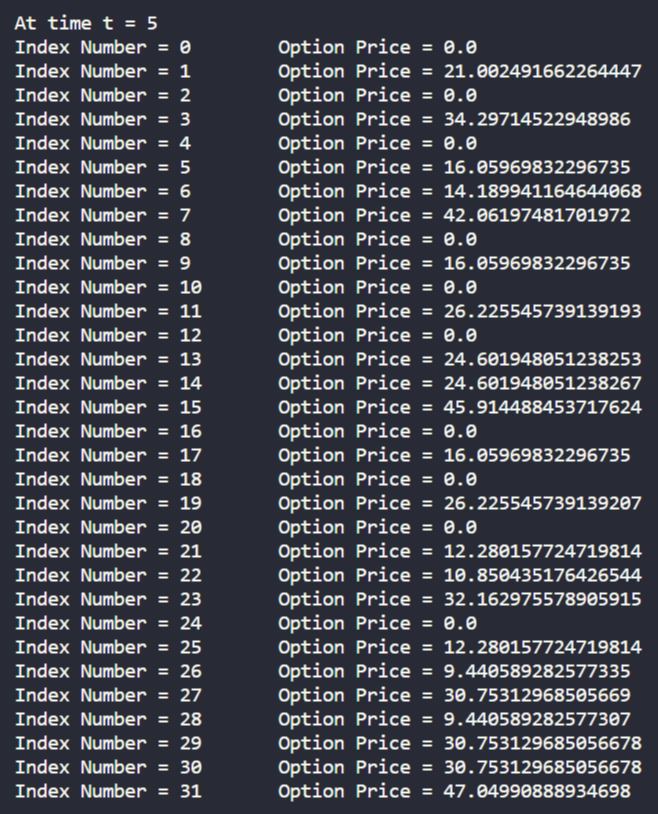
**Observations:**

As the magnitude of M rises, there is a corresponding increase in the initial option price, and it becomes apparent that the prices gradually approach convergence with further increments in M.

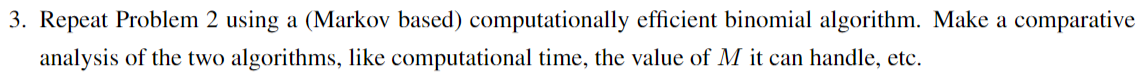
(c) Here, t depicts the time intervals with respect to M

The prices of the options at all intermediate time points for M = 5 are as follows:



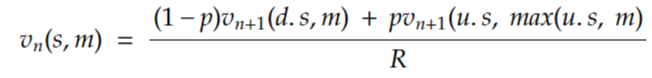


Question 3:



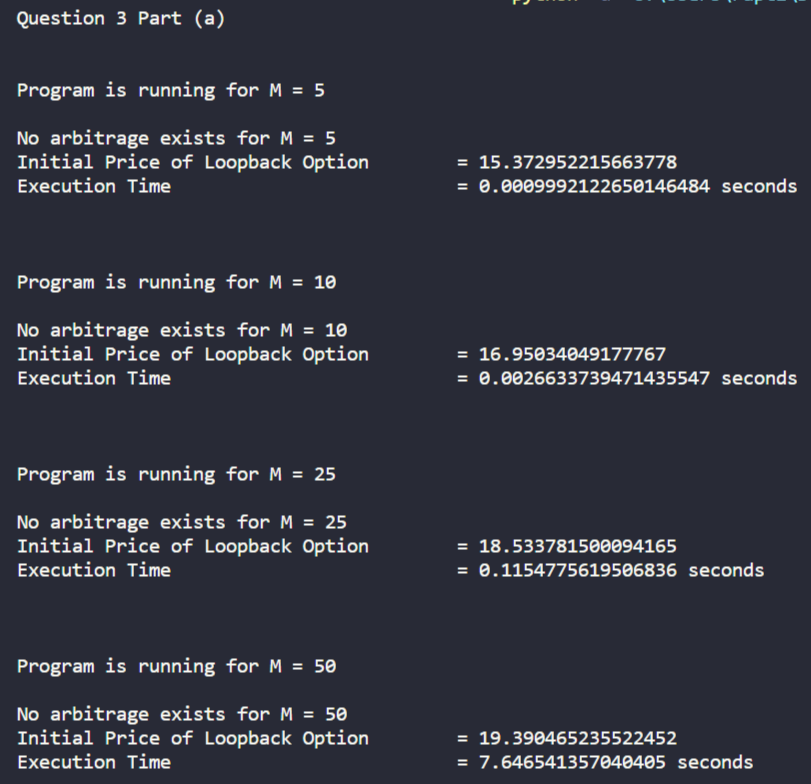
(a)

Using Markov property, we derive the following recurrence relation for finding out the initial price of the lookback (European) option:



The initial prices of the lookback (European) option in the binomial model, using a (Markov based) computationally efficient binomial algorithm for different values of M are as follows:

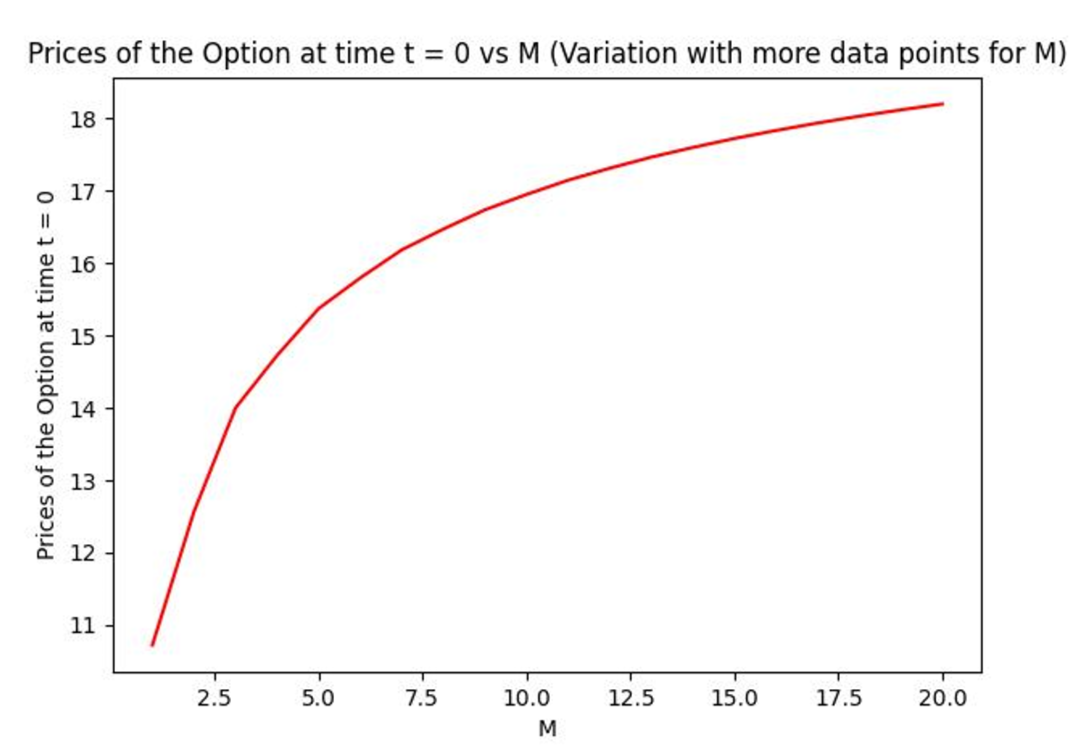
|  |  |  |  |
| --- | --- | --- | --- |
| **Serial Number** | **M** | **Initial Price of Lookback European Option** | **Execution Time in seconds** |
| 1 | 5 | 15.372952215663778 | 0.0009992122650146484 |
| 2 | 10 | 16.95034049177767 | 0.0026633739471435547 |
| 3 | 25 | 18.533781500094165 | 0.1154775619506836 |
| 4 | 50 | 19.390465235522452 | 7.646541357040405 |



(b)

Comparison of the values of options at time t = 0 for the values of M that I have taken is done using the following plots:





**Observations:**

As the magnitude of M rises, there is a corresponding increase in the initial option price, and it becomes apparent that the prices gradually approach convergence with further increments in M.

(c)

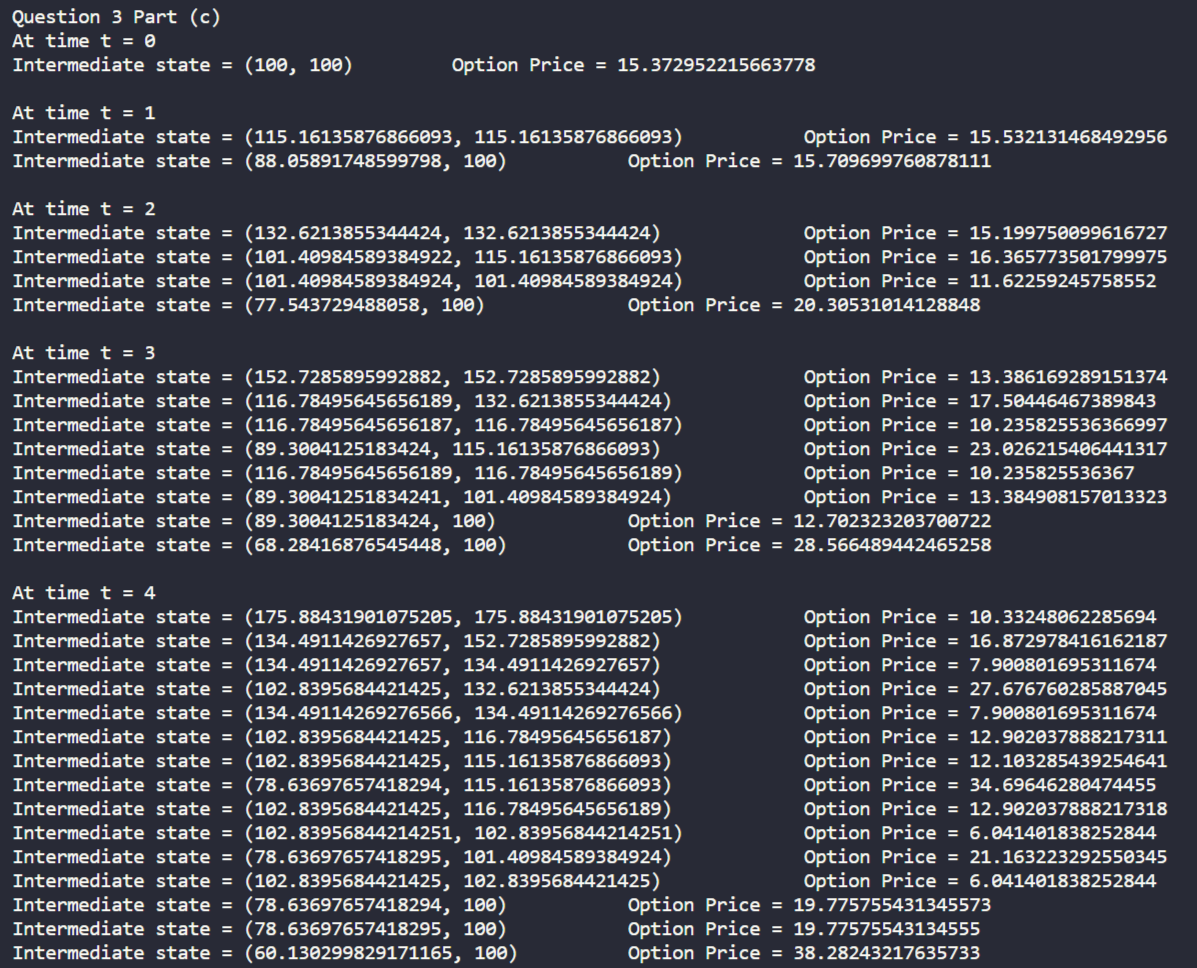
Here, t depicts the time intervals with respect to M

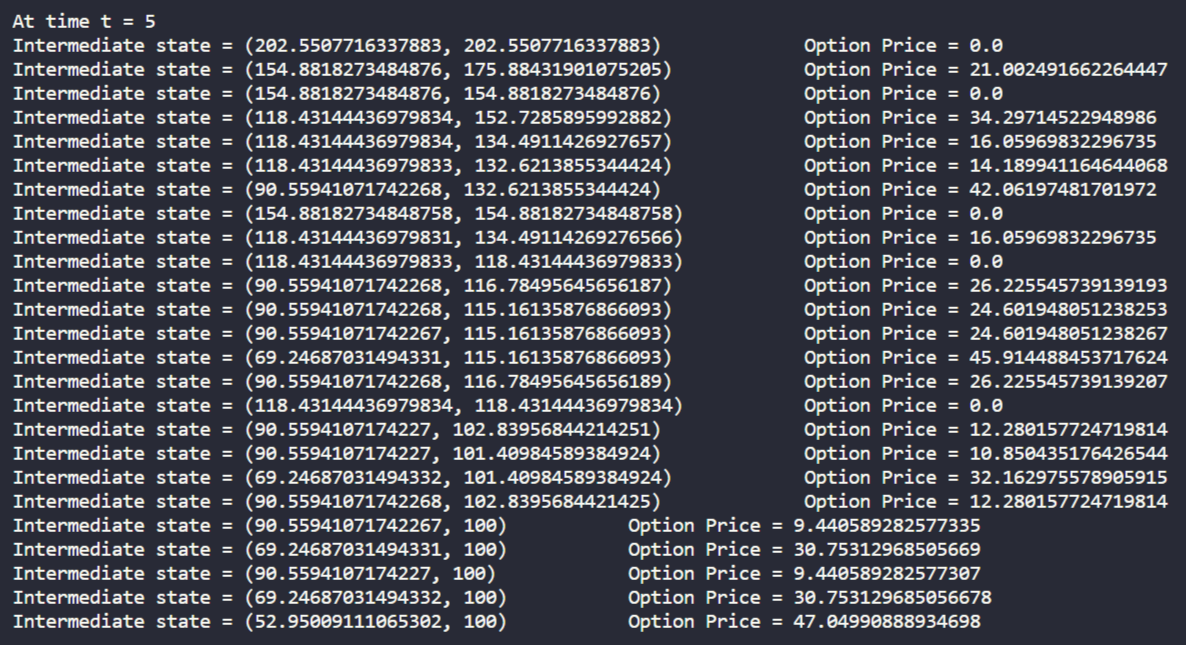
The prices of the options at all intermediate time points for M = 5 are as follows:

**Note:** Each state is defined by a tuple representing:

(i) the current stock price at that moment

(ii) the highest stock price encountered along the entire path up to the current state.





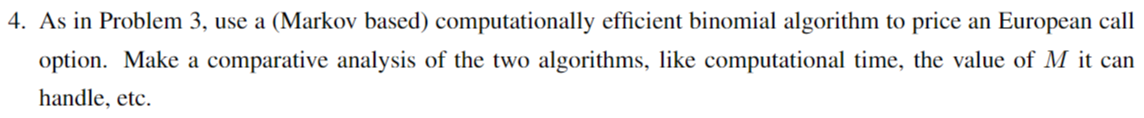
**Comparative Analysis of the two algorithms:**

* The unoptimized algorithm exhibits exponential space complexity, whereas the Markov-based optimized algorithm does not share this characteristic. This distinction arises from the incorporation of tabulation and memoization for temporary storage, leveraging dynamic programming principles.
* The unoptimized algorithm proves impractical as its time complexity can escalate dramatically for small values of M, such as 50. In contrast, the Markov-based algorithm demonstrates resilience, effectively handling such scenarios.
* The following table compares the execution time (in seconds) of the two algorithms for different values of M:

|  |  |  |
| --- | --- | --- |
| **M** | **Unoptimized Algorithm** | **Optimized Markov based Algorithm** |
| 5 | 0.0011849403381347656 | 0.0009992122650146484 |
| 10 | 0.0050394535064697266 | 0.0026633739471435547 |
| 25 | 288.1385660171509 | 0.1154775619506836 |
| 50 | Infeasible | 7.646541357040405 |

* In the unoptimized algorithm, both the time and space complexity are exponential. So, it can handle maximum M of around 30 on most processors/RAM since after that, the RAM wouldn’t have such huge computing power. After M = 25, the execution time almost doubles as M is increased by 1.
* In the optimized Markov based algorithm, we use memoization concept of dynamic programming which reduces both the space and time complexities of the unoptimized algorithm. So, this algorithm can handle larger values of M at least upto 50, as I have run the code till M = 50 for the previous questions.

Question 4:



(a)

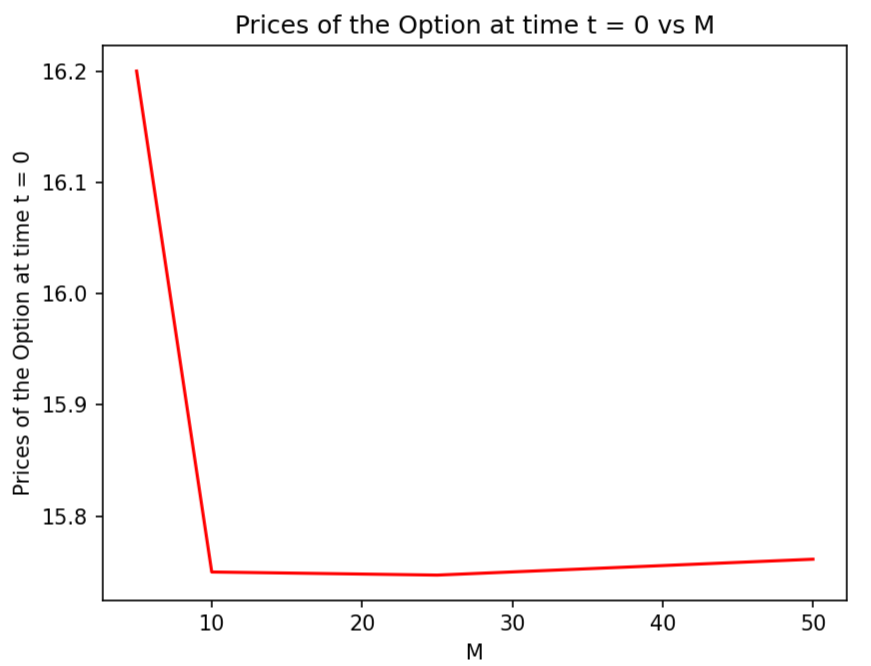
Using binomial model with continuous compounding convention and K=100:

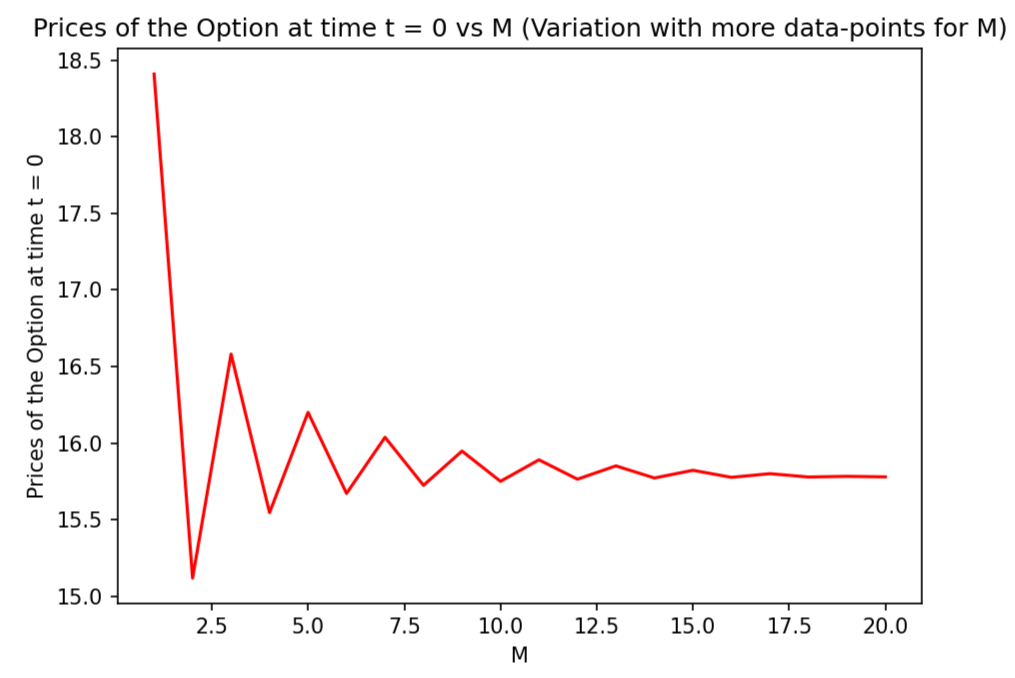
The prices of the European Call Option at time t = 0 for different values of M are as follows:

|  |  |  |
| --- | --- | --- |
| **M** | **European Call Option Price at t = 0** | **Execution Time in seconds** |
| 5 | 16.200135785709463 | 0.000099945068359375 |
| 10 | 15.749706920472502 | 0.000575362843355387 |
| 25 | 15.746918255600455 | 0.000826182123710216 |
| 50 | 15.761196879829429 | 0.001001596450805664 |

(b)

Comparison of the values of options at time t = 0 for the values of M that I have taken is done using the following plots:



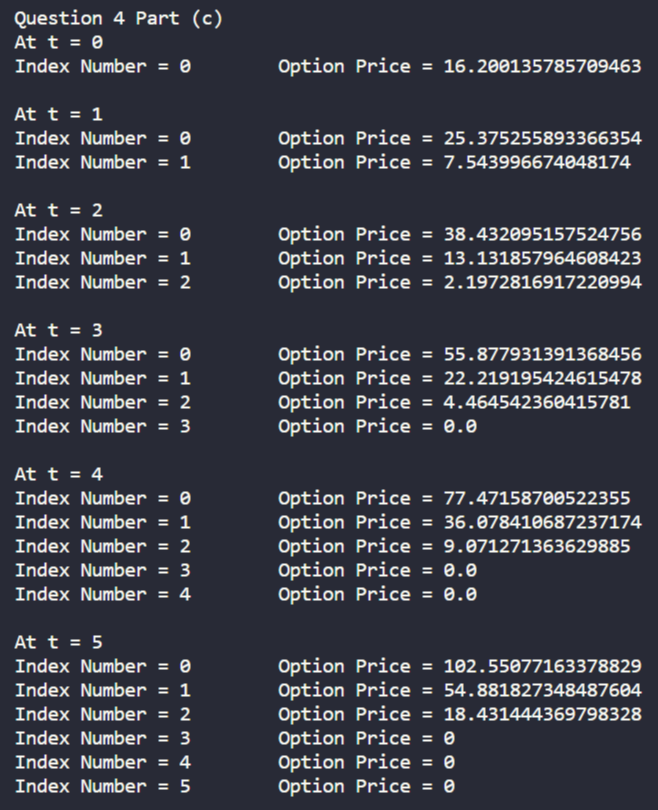


**Observations:**

The initial price of the European Call Option converges approximately around 15.75 as M increases.

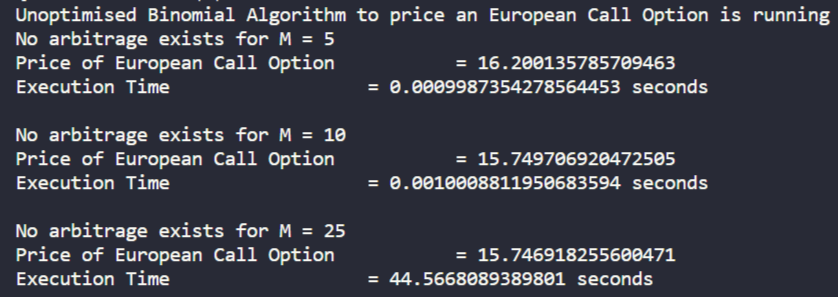
(c) Here, t depicts the time intervals with respect to M

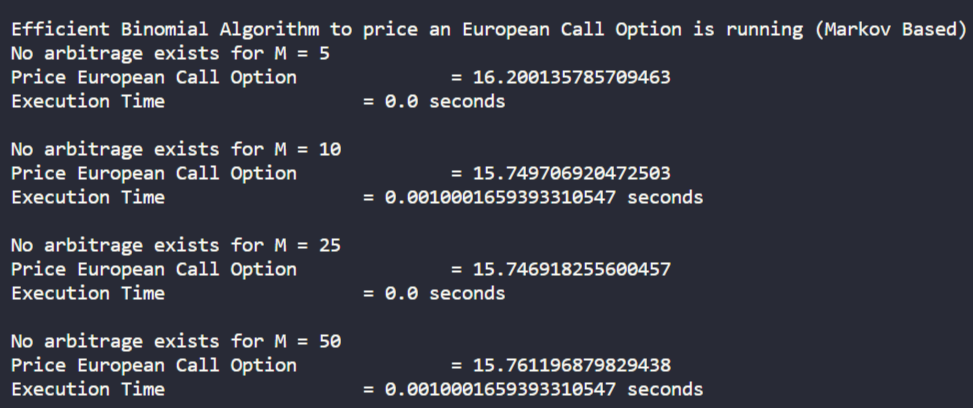
The prices of the options at all intermediate time points for M = 5 are as follows:



**Comparative Analysis of the algorithms:**

* The execution times of all the algorithms are compared below:







(Note: The execution times change when the code is run again and again and on different machines. This is just one screenshot of one instance of execution times)

* The unoptimized algorithm exhibits both exponential time and space complexity. On the other hand, the efficient algorithm demonstrates quadratic time and space complexity in terms of M, although the space complexity can be reduced to linear. However, considering the additional requirement to print intermediate information, I implemented an algorithm with quadratic complexity. The most efficient algorithm achieves nearly linear time complexity, especially when leveraging memoization for calculating nCr, and also maintains linear space complexity.
* The maximum values of M that these algorithms can handle are as follows:

1. Unoptimized Algorithm: Around 30, as explained in the previous question, question number 3, of this assignment.
2. Optimized and most Optimized Algorithms: They worked perfectly fine with M = 50 and M =100 also, which was the largest value of M mentioned in the Assignment 3 question paper. So, these algorithms can work correctly and in reasonably small time for fairly large values of M.