Financial Engineering Lab MA – 374 Lab – 1

Name – Rasesh Srivastava

Roll Number – 210123072

Branch – Mathematics and Computing

Write a program, using the binomial pricing algorithm, to determine the price of an European call and an European put option (in the binomial model framework) with the following data :

$$S(0) = 100; K = 105; T = 5; r = 0.05; \sigma = 0.4.$$

Take $u = e^{\sigma\sqrt{\Delta t}+\left(r-\frac{1}{2}\sigma^2\right)\Delta t}$ and $d = e^{-\sigma\sqrt{\Delta t}+\left(r-\frac{1}{2}\sigma^2\right)\Delta t}$, where $\Delta t = \frac{T}{M}$, with M being the number of subintervals in the time interval $[0, T]$. Use the continuous compounding convention in your calculations (i.e., both in $\tilde{p}$ and in the pricing formula).

Question 1:

1. Run your program for $M = 1, 5, 10, 20, 50, 100, 200, 400$ to get the initial option prices and tabulate them. What do you observe? How large can M be?

M = number of subintervals in the time interval [0, T]

The initial option prices for the European Call Option and European Put Option (in the binomial model framework) are as follows:

M	Price of European Call Option	Price of European Put Option
1	43.69045	25.46453
5	41.35488	23.12896
10	41.59075	23.36483
20	41.46340	23.23749
50	41.22778	23.00186
100	41.19156	22.96564
200	41.25225	23.02634
400	41.23138	23.00546

Binomial Pricing Algorithm:

$$\Delta t = \frac{T}{M}$$

(Using the continuous compounding convention)

At time t = t_i (= i * Δt), there are i + 1 possible option prices, that is,

$$S_n^i = d^{i-n}u^n S_0\,,\ 0 \leqslant n \leqslant i$$

Since continuous compounding convention is used, the probability p of an upward return in price is:

$$p = \frac{e^{r\Delta t} - d}{u - d}$$

At time t = T, we are calculating the price of the options using the payoff functions of the call and put options respectively.

$$C_n^M = max\left(S_n^M - K,\ 0\right),\ 0 \leqslant n \leqslant M$$
$$P_n^M = max\left(K - S_n^M,\ 0\right),\ 0 \leqslant n \leqslant M$$

Where C_n^M / P_n^M is the n th possible price of the call / put option respectively for the M th interval.

Initial Call Option Price = C_0^0 and Initial Put Option Price = P_0^0, these 2 are the required values.
So, we continuously apply Backward Induction to find out the option prices at t = 0 by using following relations:

$$C_n^i = (1-p)C_{n+1}^{i+1} + pC_n^{i+1}\,,\ 0 \leqslant n \leqslant i \text{ and } 0 \leqslant i \leqslant M-1$$

$$P_n^i = (1-p)P_{n+1}^{i+1} + pP_n^{i+1}\,,\ 0 \leqslant n \leqslant i \text{ and } 0 \leqslant i \leqslant M-1$$

and hence, we get the required initial option prices information.

Observations:

1) The option prices for both European call and put options converge towards the theoretical Black-Scholes option prices as we increase M. This means that the computed option prices become more accurate as we use smaller time steps.
2) Initially, with a low number of subintervals M, we observe some fluctuations or instability in the computed option prices. As M increases, these fluctuations decrease, and the option prices converge.
3) The computational time required to run the program increases as we raise the value of M. This is because the algorithm must calculate option prices

for more time steps. At some point, the increase in computational time becomes significant.

4) There is a point (around M = 50) where increasing M further doesn't significantly improve the accuracy of the option prices. The improvement becomes marginal, and the computational cost increases.

The smallest possible value of M is 1 (since it is the number of subintervals in the time interval [0, T]).
For M = 1, No-Arbitrage Condition is satisfied (as checked by the code), that is, there is no arbitrage for M = 1, so, there is no lower bound on the value of M.

As we keep on increasing M, the No-Arbitrage conditions, that is,

$$d < R < u$$ where,

$$R = e^{rt}$$

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

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

$$t = \frac{T}{M}$$

are always satisfied for all M >= 1 (as checked from the code by taking very large values of M),
That is, for all the possible values of M, the No-Arbitrage conditions are satisfied, so there is no upper bound on the value of M. M can be infinite, since the market is arbitrage free for all the possible values of M.

So, from the no-arbitrage conditions, there is no upper bound on the value of M, that is, the value of M can be as large as possible. (The value of M can be infinite also).

However, when we use M=50, we get fairly accurate answers. As we increase M after M =50, the change in the option prices is very small. So, we can get sufficiently accurate answers by using M = 50.

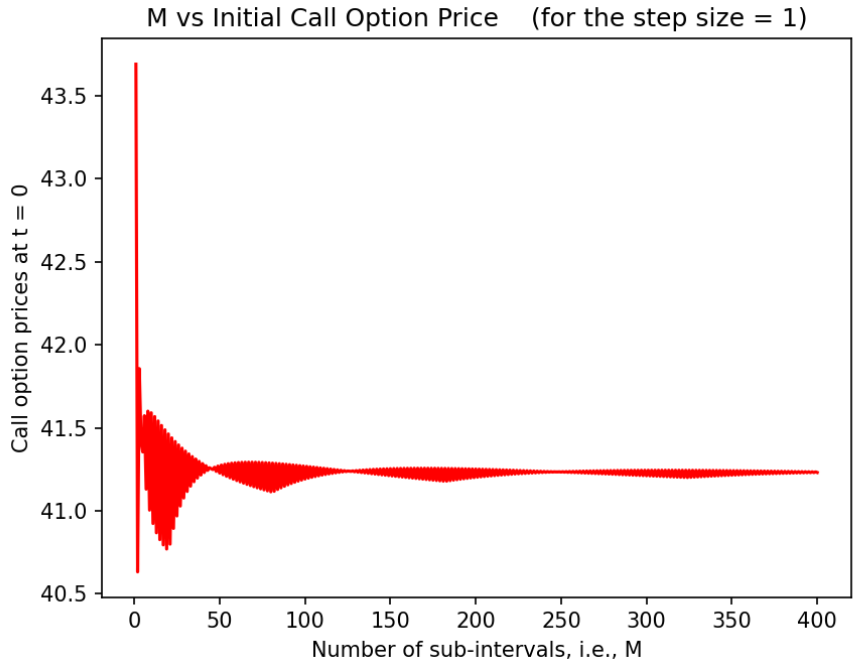
Also, in Python, the maximum size of a NumPy 2D array is 10^9. The size of our option prices matrix is (M+1) * (M+1). So, the maximum value of M possible in Python is roughly around $\sqrt{10^9}$, which is roughly equal to 31,000 (since there

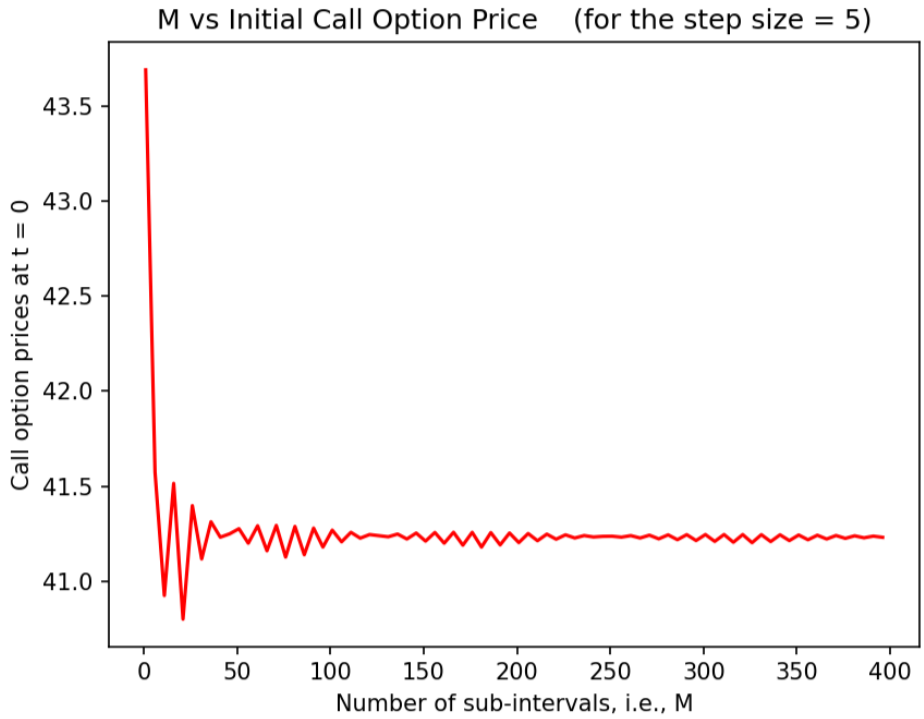
are some overhead charges also for array creation in NumPy in Python). So, the maximum possible value of M in Python is 31,000 approximately if the maximum size of a 2D array in Python is 10^9.

Question 2:

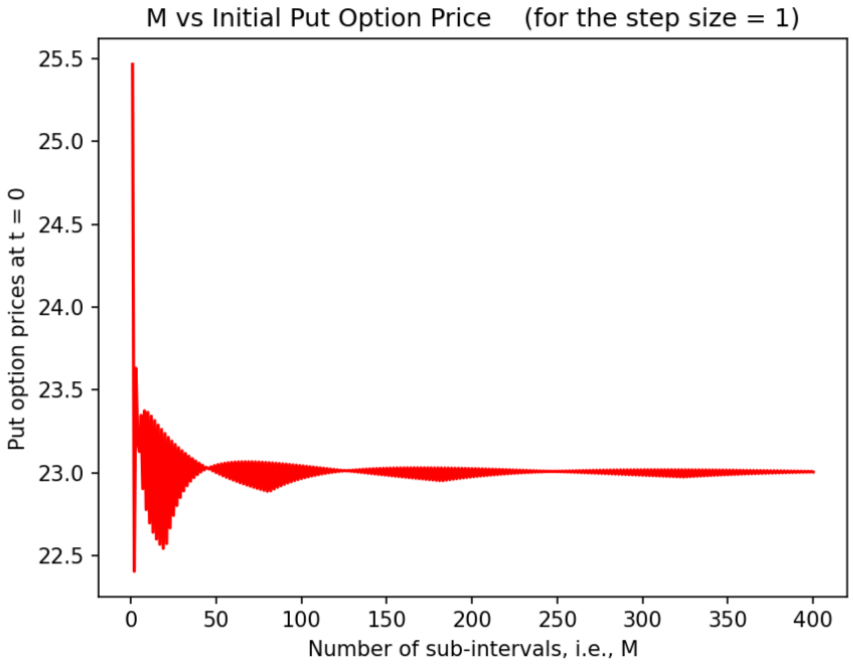
2. How do the values of options at time $t = 0$ compare for various values of M? Compute and plot graphs (of the initial option prices) varying M in steps of 1 and in steps of 5. What do you observe about the convergence of option prices?

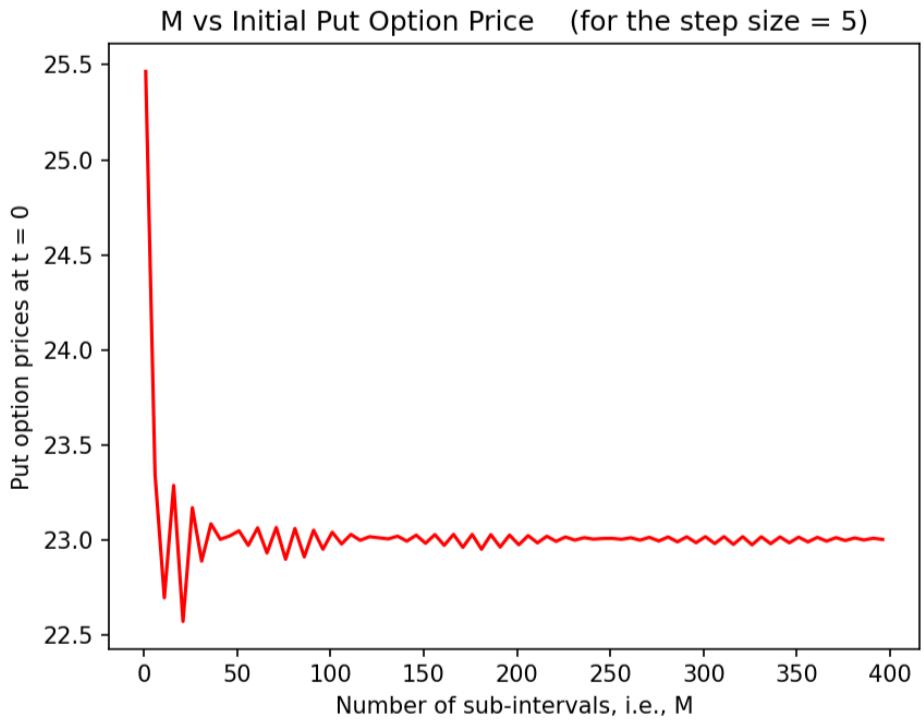
The plots of the initial option prices for the Call Option are:





The plots of the initial option prices for the Put Option are:





Observations:

- From the plots of the initial option prices for the Call Option, we observe that the initial Call Option price converges to 41.23 approximately.
- From the plots of the initial option prices for the Put Option, we observe that the initial Put Option price converges to 23.01 approximately.
- The convergence of the initial option prices is quite fast since not too many iterations are required to approximately attain the converged value.
- The convergence of the plots of the initial option prices is faster when the step value is 5.
- The deviations of the plots of the initial option prices from the convergence value are higher when the value of M is small.
- The convergence is not flawless because the values tend to oscillate around the specified convergence values mentioned earlier in the observations, even when the number of sub-intervals is increased beyond 400. However, this oscillation is typical for numerical algorithms of this nature and provides an accurate approximation of the required value, as fluctuations tend to occur at the 4th or 5th decimal place onwards, so these fluctuations do not severely affect the convergence values.

Question 3:

3. Tabulate the values of the options at $t = 0, 0.50, 1, 1.50, 3, 4.5$ for the case $M = 20$.

Note that your program should check for the no-arbitrage condition of the model before proceeding to compute the prices.

The required values of the call options at the given time stamps in the question for M = 20 are as follows:

Time Stamps → / **Options ↓**	**t = 0**	**t = 0.50**	**t = 1**	**t = 1.50**	**t = 3**	**t = 4.50**
1	41.46	77.09	136.67	231.47	912.43	3095.24
2	-	38.06	72.39	130.63	580.30	2041.04
3	-	16.75	34.58	67.56	357.73	1334.39
4	-	-	14.47	31.01	209.13	860.70
5	-	-	5.14	12.19	111.82	543.19
6	-	-	-	3.94	51.89	330.35
7	-	-	-	0.99	19.50	187.68
8	-	-	-	-	5.42	92.04
9	-	-	-	-	0.97	32.54
10	-	-	-	-	0.08	5.71
11	-	-	-	-	0.00	0.00
12	-	-	-	-	0.00	0.00
13	-	-	-	-	0.00	0.00
14	-	-	-	-	-	0.00
15	-	-	-	-	-	0.00
16	-	-	-	-	-	0.00
17	-	-	-	-	-	0.00
18	-	-	-	-	-	0.00
19	-	-	-	-	-	0.00

Observations:

1) At t = 0.25*j, where j = 0, 1, 2, 3...., we will get j + 1 different values of the options, since j + 1 different option prices are available according to the Binomial Tree Model Pricing Framework.

The required values of the put options at the given time stamps in the question for M = 20 are as follows:

Time Stamps → / **Options ↓**	**t = 0**	**t = 0.50**	**t = 1**	**t = 1.50**	**t = 3**	**t = 4.50**
1	23.24	13.97	6.66	2.21	0.00	0.00

2	-	23.40	13.58	6.02	0.00	0.00
3	-	34.56	23.51	13.08	0.07	0.00
4	-	-	35.39	23.55	0.70	0.00
5	-	-	47.50	36.25	3.43	0.00
6	-	-	-	49.12	10.56	0.00
7	-	-	-	60.34	23.12	0.00
8	-	-	-	-	39.17	0.00
9	-	-	-	-	54.91	4.61
10	-	-	-	-	67.56	20.75
11	-	-	-	-	76.56	43.84
12	-	-	-	-	82.64	63.15
13	-	-	-	-	86.72	76.09
14	-	-	-	-	-	84.77
15	-	-	-	-	-	90.58
16	-	-	-	-	-	94.48
17	-	-	-	-	-	97.09
18	-	-	-	-	-	98.85
19	-	-	-	-	-	100.02

Observations:

1) At t = 0.25*j, where j = 0, 1, 2, 3...., we will get j + 1 different values of the options, since j + 1 different option prices are available according to the Binomial Tree Model Pricing Framework.

No-arbitrage Condition:

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

$$d < e^{rt} < u$$

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

$$d < R < u$$ where,

$$R = e^{rt}$$

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

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

$$t = \frac{T}{M}$$