भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

MA 374: Financial Engineering Lab

Lab 03

AB Satyaprakash (180123062)

4th Feb 2021

Note: ✍🏻

1. ***For questions 2 and 3*** *(both algorithms), I have submitted codes in* ***python as well as C++****. I had initially written code in* ***C++****, as it is computationally more efficient (see q2 and q3 description), and later I also wrote them in* ***python****. The outputs are the same.*
2. ***Please compile C++ programs using g++****, i.e.* `g++ <filename>.cpp`
3. ***Please run python programs using python3****, i.e.* `python3 <filename>.py`

Question 1.

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:

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

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

- **The initial loopback option price for M = 5 is 9.119298985864685**
- **The initial loopback option price for M = 10 is 10.080582906831074**

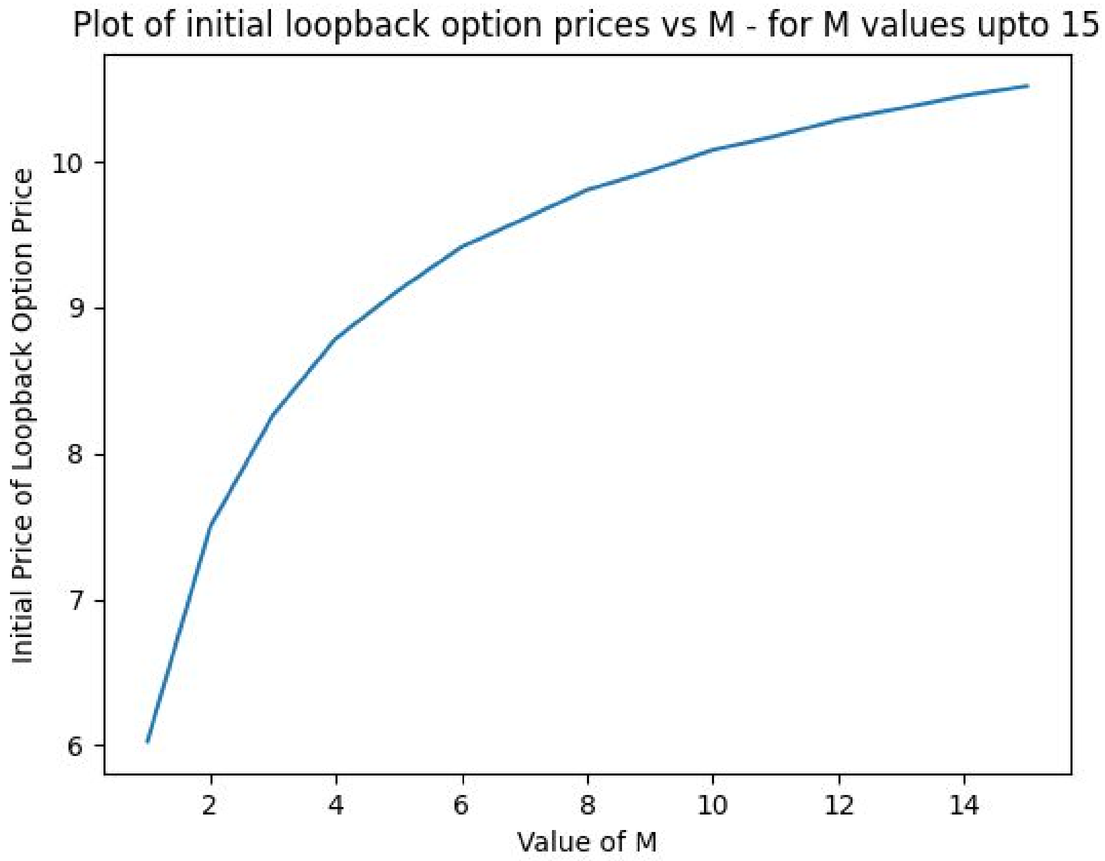
For the values of **M = 25 and M = 50**, the basic binomial model will scale in time complexity as it works in $O(2^M)$. And thus we can't computationally handle this in python. An appropriate message is shown in the terminal to the user:

```
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:

T = 0	T = 0.2	T = 0.4	T = 0.6	T = 0.8	T = 1.0
9.1193	9.028	8.5481	7.4168	5.5016	0.0
	9.5048	9.7991	9.9553	9.5714	11.1814
		7.1479	6.2019	4.6005	0.0
		12.1687	13.7129	15.6319	19.4527
			6.2019	4.6005	0.0
			8.3246	8.0036	9.3499
			7.1484	6.6808	6.3745
			17.5821	21.1881	25.3946
				4.6005	0.0
				8.0036	9.3499
				3.8469	0.0
				13.0714	16.2664
				3.8469	0.0
				10.6809	13.578
				10.6809	13.578
				25.0512	29.4826
					0.0
					9.3499
					0.0
					16.2664
					0.0
					7.8184
					5.3304
					21.235
					0.0
					7.8184
					2.9014
					18.8059
					2.9014
					18.8059
					18.8059
					32.1054

Question 2.

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.**

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.

```
The initial loopback option price for M = 5 is 9.11929898586469
The initial loopback option price for M = 10 is 10.08058290683101
The initial loopback option price for M = 25 is 11.003495335646338
The initial loopback option price for M = 50 is 11.510862222177268
```

Question 3.

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.

```
The initial european call option price for M = 5 is 12.163185946764592
The initial european call option price for M = 10 is 12.277327819222982
The initial european call option price for M = 25 is 12.136745963232974
The initial european call option price for M = 50 is 12.08536151007219
The initial european call option price for M = 100 is 12.12304707401248
The initial european call option price for M = 500 is 12.10864990170937
```

Extras...

An example of how computational-time has been obtained in python is shown below:

```
# Test computation time for M = 15:
-----------------------------------------------------
dt = T/M
u = exp(sig*sqrt(dt)+(r-sig*sig/2)*dt)
d = exp(-sig*sqrt(dt)+(r-sig*sig/2)*dt)
p = (exp(r*dt)-d)/(u-d)
q = 1-p
Map.clear()
start = time.time()
timeTempPrice = recursion(0, S0, S0, T, r, sig, M, u, d, p, q, dt, 0)
end = time.time()

print('Computational time for M = 15 is {}s'.format(end-start))
```

The following python packages need to be installed for all programs to work. (Use pip3 if on python version 3)

1. **Pandas**: `pip install pandas`
2. **Numpy:** `pip install numpy`
3. **Matplotlib:** `pip install matplotlib`
4. **Ipython:** `pip install ipython`