CSE 390 : Computational Finance– Homework II

Dr. Ritwik Banerjee

Submission Deadline: Nov 09, 2018 (Friday), 11:59 pm

This programming assignment is about applying your programming skills to some option-pricing problems in the binomial model we studied in class.

- You may use one (or more) of the following: **Java**, **C**, **C++**, **Python**, **MATLAB**, **R**.
- You may use any "data science" library that is more-or-less standard in today's programming world, as long as any code directly dealing with the financial model and computation is your original work.

Background Synopsis: *Binomial Pricing Model*

As we have seen, this model can be used to compute the *fair* price of an option on an underlying asset. The inputs required by this model are

i) The (continuously compounded) risk-free rate of interest r.

ii) The time T to expiration (a.k.a. maturity) of the option. If needed, you may assume a default value of 1 year.

iii) The number n of time periods into which T is divided. The periods are of identical length, i.e., $\Delta t = T/n$.

iv) The volatility σ of the option price.

Based on these inputs, the three model parameters u, d, and p can be calculated.
In the binomial tree model, S_0 (the option price at time $T = 0$) is the root node, and a *price path* in the tree represents a possible sequence of option prices $S_0, S_1, \ldots, S_n$. As discussed in class, we can look at this binomial tree as a discrete-time stochastic process, defined as

$$S_{k+1} = \begin{cases} uS_k & \text{with probability } p \\ (1/u)S_k & \text{with probability } 1-p \end{cases}$$

where S_k denotes the option price at kth time period $(0 \leq k \leq n-1)$[1].

Problem I : *American Put Option* [20 points]

The problem here is to write a program to compute the value (in the absence of arbitrage) of an American put option on a stock. Remember that the payoff of a put option with strike price K, when exercised at time period k is $\max\{K - S_k, 0\}$. The inputs to your program will be provided as a tab-separated plain-text file, where each line has the following format:

r `<TAB>` T `<TAB>` n `<TAB>` σ `<TAB>` S_0 `<TAB>` K

That is, your program will calculate the price of an option once *per line* of input from this file.

[1] Here, we are doing away with the parameter d to avoid having a stochastic process with multiple variables.

Suggested approach: First, calculate the two parameters u and p. Next, define a recursive function to calculate the option price at a vertex v of the binomial tree[2]. Finally, once you have the recursion set up, invoking the function at the root node will give the price of the option.

Problem II : *Asian Call Option* [20 points]

A very different kind of option is the *Asian* option, which is one of the many types of options that are called "exotic options" (named such due to features that make it more complex to analyze). Here, the payoff is determined by the average underlying price over the entire period of time. This is very different from European and American options in the sense that the payoff is non-Markov. Or, in terms of the binomial tree model, the payoff is *path-dependent*. With strike price K at time k, it is given by $\max\{\mu_k - K, 0\}$, where μ_k is the 'running average' stock price:

$$\mu_k = \frac{S_0 + S_1 + \ldots + S_k}{k+1}.$$

Suggested approach: Only one aspect of your solution to the previous problem needs to be changed here, and that is the vertex representation. You will need to incorporate the running average stock price on the path from the root node all the way to the vertex itself. Once you have the proper vertex representation, define a recursive function to calculate the option price at a vertex v. The final answer (i.e., the current price of the option) will be given by invoking the function at the root node of the binomial tree. Note that initially at the root node, $\mu_0 = S_0$.

Problem III : *Time complexity* [10 points]

Run each program with $n = 10$ to see how long it takes. Based on this analysis, extrapolate to provide an estimate for the time it will take for your program to run with $n = 500$. How does the time increase as a function of n?

Note: This problem does not require additional code. Simply write the three following pieces of information in a plain text file named `complexity.txt`:

(a) Time taken (in milliseconds, up to 3 decimal places) for $n = 10$.

(b) Time taken (in milliseconds, zero decimal places) for $n = 500$.

(c) Time complexity as a function $T(n)$ of n. This will be a mathematical function, for example, $T(n) = \Theta(n^2 \log n)$.

What is the expectation from a working submission?

Submit a single `.zip` file containing the following:

→ One plain text `README.txt` file that explains how to run your code (including providing the input tab-separated file). Note: at no point should the grader have to change anything in your code. For example, the input file should not be hard-coded in your program.

→ A single folder containing all your code (which could be just one file . . . although I hope it is not) and the `complexity.txt` file.

→ There is no need to submit any sample data. Your code will be tested with a tab-separated file (each line being an individual test-case) provided as input.

→ The assignment is worth 50 points, and due by **11:59 pm, Nov 09 (Friday)**.

[2] Hint: If you know the price at a vertex, then does the depth uniquely identify the vertex? Think of this to come up with a representation for a vertex in the tree.