FE545 Final Exam

Due Date: May 5th (Sunday). Tree-based methods can be used for obtaining option prices, which are especially popular for pricing American options. Binomial, trinomial, and random tree methods can be used to price many options, including plain vanilla options, and also exotic options such as barrier options, digital options, Asian options, and others. In this assignment, you are required to price American option using both trinomial and random tree methods through which you are to apply some of the design patterns in C++ programming to solve the problem. The problem has multiple parts, and each part has different points assigned.

Pricing a derivative security entails calculating the expected discounted value of its payoff. This reduces, in principle, to a problem of numerical integration; but in practice this calculation is often difficult for high-dimensional pricing problems. The binomial options pricing model (BOPM) approach has been widely used since it is able to handle a variety of conditions for which other models cannot easily be applied. This is largely because the BOPM is based on the description of an underlying instrument over a period of time rather than a single point. As a consequence, it is used to value American options that are exercisable at any time in a given interval. The trinomial option pricing model, proposed by Phelim Boyle in 1986, is considered to be more accurate than the binomial model, and will compute the same results, but in fewer steps.

Broadie and Glasserman (1997) proposed the method of the simulated random tree to price American options, which can derive the upper and lower bounds for American options. This combination makes it possible to measure and control errors as the computational effort increases. The main drawback of the random tree method is that its computational requirements grow exponentially in the number of exercise dates m, so the method is applicable only when m is small. Nevertheless, for problems with small m it is very effective, and it also serves to illustrate a theme of managing scores of high and low bias.

The typical simulation approach to European option pricing is to use simulation to estimate the expectation

$$C = E[e^{-rT}(S_T - K)^+] \tag{1}$$

under the risk neutral measure. As usual, r denotes the riskless rate of interest, T the option maturity, K the strike price, and S_r the terminal stock price. The American option pricing problem is to find

$$C = \max_{\tau}[e^{-r\tau}(S_\tau - K)^+] \tag{2}$$

over all stopping times $\tau \leq T$.

We focus on a discrete time approximation to this problem where we restrict the exercise opportunities to lie in the finite set of times $0 = t_o < t_1 < ... < t_d = T$. The analogous procedure for an American option would be to simulate a path of asset prices, say, $S_0, S_1, ..., S_T$, at corresponding times $0 = t_0 < t_1 < ... < t_d = T$; then compute a discounted option value corresponding to this path, and finally average the results over many simulated paths. The main question is how to compute a discounted option value corresponding to the asset price path.

Broadie and Glasserman (1997) developed a stochastic random tree method to estimate the lower and upper bound of the American option value. Let $\tilde{h}_i$ denote the payoff function for exercise at t_i, which we now allow to depend on i. Let $\tilde{V}_i(x)$ denote the value of the option at t_i given $X_i = x_i$ (the option has not exercised). We are ultimately interested in $\tilde{V}_0(X_0)$. This value is determined recursively as follows:

$$\tilde{V}_m(x) = \tilde{h}_m(x) \quad (3)$$

$$\tilde{V}_{i-1}(x) = max\{\tilde{h}_{i-1}(x), E[D_{i-1,i}(X_i)\tilde{V}_i(X_i)|X_{i-1} = x]\}, \quad (4)$$

$i = 1, ..., m$ and we have introduced the notation $D_{i-1,i}(X_i)$ for the discount factor from t_{i-1} to t_i. It states that the option value at expiration is give by the payoff function $\tilde{h}_m$; and at time $(i-1)$th exercise date the option value is the maximum of the immediate exercise value and the expected present value of continuing.

As its name suggests, the random tree method is based on simulating a tree of paths of the underlying Markov chain $X_0, X_1, ..., X_m$. Fix a branching parameter $b \geq 2$. From the initial state X_0, simulate b independent successor states $X_1^1, ..., X_1^b$ all having the law of X_1. From each X_1^i, simulate b independent successors $X_2^{i1}, ..., X_2^{ib}$ from the conditional law of X_2 given $X_1 = X_1^i$. From each $X_2^{i_1 i_2}$, generate b successors $X_3^{i_1 i_2 1}, ..., X_3^{i_1 i_2 b}$, and so on. We denote a generic node in the tree at time step i by $X_i^{j_1 j_2 \cdots j_i}$. The superscript indicates that this node is reached by following the j_1-th branch out of X_0, the j_2th branch out the next node, and so on.

At all terminal nodes, set the estimator equal to the payoff at that node:

$$\hat{v}_m^{j_1 \cdot \cdot j_m} = h_m(X_m^{j_1 \cdots j_m}). \tag{5}$$

At node $j_1 j_2 ... j_i$ at time step i, and for each $k = 1, ..., b$, set

$$\hat{v}_{ik}^{j_1 j_2 \cdots j_i} = \begin{cases} h_i(X_i^{j_1 j_2 \cdots j_i}) & \text{if } \frac{1}{b}\sum_{j=1}^{b} \hat{v}_{i+1}^{j_1 j_2 \cdots j_i j} \leq h_i(X_i^{j_1 j_2 \cdots j_i}); \\ \hat{v}_{i+1}^{j_1 j_2 \cdots j_i k} & \text{otherwise} \end{cases} \tag{6}$$

Then set

$$\hat{v}_i^{j_1 \cdot \cdot j_i} = \frac{1}{b} \sum_{k=1}^{b} \hat{v}_{i,k}^{j_1 \cdots j_i} \tag{7}$$

Working backward, we then set

$$\hat{V}_i^{j_1..j_i} = max\left\{h_i(X_i^{j_1...j_i}), \frac{1}{b}\sum_{k=1}^{b}\hat{V}_{i,k}^{j_1...j_i}\right\} \tag{8}$$

The high estimator of the option price at the current time and state is $\hat{v}_0$.

The low estimator is defined as follows. At all terminal nodes, set the estimator equal to the payoff at that node:

$$\hat{v}_m^{j_1j_2...j_m} = h_m(X_m^{j_1j_2...j_m})$$

At node $j_1j_2...j_i$ at time step i, and for each $k = 1, ..., b$, set

$$\hat{v}_{ik}^{j_1j_2...j_i} = \begin{cases} h_i(X_i^{j_1j_2...j_i}) & \text{if} \frac{1}{b-1}\sum_{j=1;j\neq k}^{b}\hat{v}_{i+1}^{j_1j_2...j_ij} \leq h_i(X_i^{j_1j_2...j_i}); \\ \hat{v}_{i+1}^{j_1j_2...j_ik} & \text{otherwise} \end{cases} \tag{9}$$

We then set

$$\hat{v}_i^{j_1j_2...j_i} = \frac{1}{b}\sum_{k=1}^{b}\hat{v}_{ik}^{j_1j_2...j_i}. \tag{10}$$

The low estimator of the option price at the current time and state is $\hat{v}_0$.

Assume a random tree for pricing American put option is given in Figure (1). Please use the random tree in Figure (1) to calculate the high and low estimate of the American call option price. Please show your steps for both the high estimator and

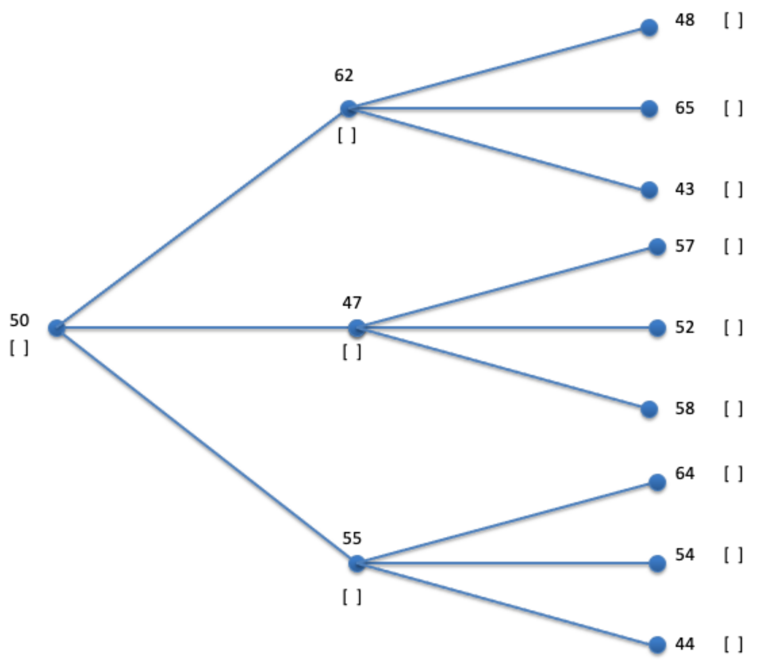

Figure 1: High Estimator

low estimator.

Now we begin by pricing a standard American call option on a single asset which pays continuous dividends and whose price is governed by a geometric Brownian motion process. We assume that the risk neutralized price of the underlying asset, S_t, satisfies the stochastic differential equation

$$dS_t = S_t[(r - \delta)dt + \sigma dz_t], \tag{11}$$

where z_t is a standard Brownian motion process. In Eq. (12),

r is the riskless interest rate, δ is the dividend rate, and $\sigma > 0$ is the volatility parameter. Under the risk neutral measure, $ln(S_i/S_{i-1})$ is normally distributed with mean $(r-\delta-\sigma^2/2)(t_i - t_{i-1})$ and variance $\sigma^2(t_i - t_{i-1})$. Given S_{i-1}, S_i can be simulated using

$$S_i = S_{i-1}e^{(r-\delta-\sigma^2/2)(t_i-t_{i-1})+\sigma\sqrt{t_i-t_{i-1}}Z}, \tag{12}$$

where Z is a standard normal random variable. The parameters were chosen so the early exercise opportunity would have significant value.

Note: the stochastic tree simulation method can be summarized in the follow steps:

1. For each simulation run, build random tree with b branches at each time step m with S_0 according to Eq. (12);

2. Estimate high and low estimators of the option using Eq. (6) and Eq. (9) for each simulated tree;

3. Take average of all the n simulation runs as the final estimators.

Please design some new classes and reuse some of the classes using the trinomial tree project for pricing American Call and Put options with the following parameters:

- $T = 1$

- $S_0 : 50$
- $K = 50$
- $r = 0.05$
- $\delta = 0.08$
- $\sigma = 0.3$
- Four exercise opportunities at times $m = \{0, T/3, 2T/3, T\}$
- The number of tree branches for random tree method: $b = 3$

For your submission, you need submit your results (output of your program) in one PDF file and clearly explain your results for all the parts of the assignment. You will also need to provide C++ source code for the follow parts in your solutions in separate projects (Note: you need to submit source code in a zip file for each project of these four parts of the questions. Provide an Readme.txt file to document how to build the projects. Please also note the four parts will be evaluated separately.):

I) Please price the American Call and Put options using the above parameters using the trinomial tree method using the four exercising times. In this part, you will reuse the *TrinomialTree* class from your homework assignment for both American Call and Put options. [**20 points**] a). Please use the *TrinomialTree* class from your homework assignment. The purpose of this step is to make sure you have a baseline solution that you can compare against for the following parts of the assignments.

b). Please reuse the classes provided in the *random_tree_project.zip* file for this part of the assignment.

II) Please create *RandomHighTree* and *RandomLowTree* classes. And use *GetGBMNextPrice*(...) function in *Random.h* provided in this assignment to build random trees. These two classes should have *void BuildTree*() and *double GetThePrice*() member functions. [**30 points**]
Please Note:

a). These two classes should have different algorithms to build and update *theTree* member and get the option prices. Please reuse the classes provided in the *random_tree_project.zip* file for this assignment.

b). Please run 100 times to get averages of the high and low estimators of the American Call and Put option prices and compare them with the results from the *TrinomialTree*

III) Please create *PayOffFactory* class using the singleton pattern to register "TTCall" (Trinomial Tree American Call Option), "TTPut" (Trinomial Tree American Put Option), “RTCallH” (Random Tree American Call Option High Estimator), “RTCallL” (Random Tree American Call Option Low Estimator), “RTPutH” (Random Tree American Put Option High Estimator), and “RTPutL” (Random Tree American Put Option Low Estimator) classes. Generate American option prices using the parameter specified above. [**30 points**] Please Note:

a). Please use the $PayOffConstructible$ and $PayOffFactory$ (sample of these classes are provided along with this assignment in the $factory\_arglist\_solution.zip$ file) to manage the class registration and object creation for all the tree option pricing classes. It means that you need to register the 6 classes for the 6 IDs specified in the problem exactly e.g. ‘TTCall”, “TTPut”, “RTCallH”, “RTCallL”, ‘RTPutH”, and ‘RTPutL”.

b). Please run *ArgumentList* class to pass the parameters for all the class registration with the factory class. The implementation of *ArgumentList* class is provided in ArgList.h and ArgList.cpp respectively.

IV) Please use Observable and Observer pattern to develop a pricing report class which is an observer of all the tree based option pricers. In this part, you are required to use the QuantLib *Observable* and *Observer* abstract classes as the bases for this assignment. Please refer to the breakout exercise from Lecture 13 as an example. [**20 points**] Please Note:

a). Please create new class called *TreeObservable* using the QuantLib *Observable* class as the base of the $SimpleTrinomialTree$, $RandomHighTree$, $RandomHighTree$ classes. Please also create a new class called $OptionPricingReportWriter$ which implements the abstract class *Observer* from QuantLib.

b). In the main function, you need to create an instance of the $OptionPricingReportWriter$ object. You need to register all the tree option pricing objects with the $OptionPricingReportWriter$ object when each of these pricing objects were created by the object factory. It means that you need to register the 6 option pricing objects with the 6 option IDs specified in the problem e.g. “TTCall”, “TTPut”, “RTCallH”, “RTCallL”, ‘RTPutH”, and ‘RTPutL”.

c). When each of the option objects is called, the $update()$ method of the $OptionPricingReportWriter$ object will be called, and it will write to the console the option pricer ID and the option value.

Homework Honor Policy: You are allowed to discuss the problems between yourselves, but once you begin writing up your solution, you must do so independently, and cannot show one another any parts of your written solutions.