IE523: Financial Computing
Fall, 2016
Programming Assignment 9: Pricing an American-Option via Trinomial Trees
(Tentative) Due Date: 10 November 2016

1 Trinomial Trees and Lattices

Pretty much everything parallels the Binomial Tree from a conceptual sense. If the stock price at discrete time step k is denoted by S_k. In this case, the process S_k is specified as

$$S_{k+1} = \begin{cases} uS_k & \text{with probability } p_u, \\ \frac{S_k}{u} & \text{with probability } p_d, \\ S_k & \text{with probability } 1 - p_u - p_d. \end{cases}$$

Just as with the binomial tree whose leaves can be combined to form the binomial lattice, you can get a *trinomial lattice* from a trinomial tree. To ensure the martingale property

$$E\{S_{k+1} \mid S_k\} = R \times S_k,$$

where $R = e^{\sigma\sqrt{\frac{T}{n}}}$, as with the Binomial Tree. We need to assign[1]

$$\begin{aligned} u &= e^{\sigma\sqrt{2(T/n)}} (= R^{\sqrt{2}}) \\ p_u &= \left(\frac{\sqrt{R} - \frac{1}{\sqrt{u}}}{\sqrt{u} - \frac{1}{\sqrt{u}}} \right)^2 \\ p_d &= \left(\frac{\sqrt{u} - \sqrt{R}}{\sqrt{u} - \frac{1}{\sqrt{u}}} \right)^2 \end{aligned}$$

Just as with the binomial lattice. you can use (k, i) to denote an arbitrary vertex on the trinomial lattice. The vertex (k, i) is connected to $\{(k + 1, i + 1), (k + 1, i), (k+1, i-1)\}$ with the appropriate probabilities along the appropriate arcs. If $V(k, i)$ is value of the option at (k, i), for the european call option we have the recursion

$$V(k, i) = \begin{cases} \max\{0, S_0 \times u^i - K\} & \text{if } (k, i) \text{ is a terminal node} \\ \frac{p_u \times V(k+1,i+1) + (1-p_u-p_d) \times V(k+1,i) + p_d \times V(k+1,i-1)}{R} & \text{otherwise} \end{cases}$$

The option price is $V(0, 0)$.

You are going to do a little more than this for this programming assignment.

[1] You can verify this at your leisure.

2 Part 1: Pricing an American-Option using a Trinomial Model by Recursion

Using my C++ program american_option_pricing_by_binomial_model.cpp on Compass as reference, write a C++ program that takes as command-line input the values of T, N, r, σ, S_0 and K, and presents the value of an *American-Option* using a *trinomial* model.

Input: The command-line input is the same as what I have for the C++ code that priced the European option in lesson 6 of my notes.

Output: A sample output that I expect from your code is shown in figure 1

```
wirelessprvnat-172-17-179-161:Debug sreenivas$ time ./American\ Option\ Trinomial\ Model 0.5 20 0.05 0.3 50 40
Recursive Trinomial American Option Pricing
Expiration Time (Years) = 0.5
Number of Divisions = 20
Risk Free Interest Rate = 0.05
Volatility (%age of stock value) = 30
Initial Stock Price = 50
Strike Price = 40
--------------------------------------
R = 1.00125
Up Factor = 1.06938
Uptick Probability = 0.250935
Downtick Probability = 0.249067
Notick Probability = 0.499998
--------------------------------------
Trinomial Price of an American Call Option = 11.5491
Trinomial Price of an American Put Option = 0.570673

real    2m27.566s
user    2m24.358s
sys     0m0.781s
wirelessprvnat-172-17-179-161:Debug sreenivas$
```

Figure 1: Sample output (**Note**: 20 divisions will take some time to run on your computer. Be patient. The next part of this assignment will be able to handle a large number of divisions, as you will see.).

3 Part 2: Pricing an American-Option using a Trinomial Model by Dynamic Programming

Using my C++ program american_option_pricing_by_dynamic_programming.cpp on Compass as reference, write a C++ program that takes as command-line input the values of T, N, r, σ, S_0 and K, and presents the value of an *American-Option* using a *trinomial* model.

Input: The command-line input is the same as what I have for the C++ code that priced the European option in lesson 6 of my notes.

Output: A sample output that I expect from your code is shown in figure 2

```
MacBook-Air:Debug sreenivas$ time ./American\ Option\ via\ Trinomial\ Dynamic\ P
rogramming 1 200 0.05 0.3 50 60
American Option Pricing by Trinomial-Model-Inspired Dynamic Programming
Expiration Time (Years) = 1
Number of Divisions = 200
Risk Free Interest Rate = 0.05
Volatility (%age of stock value) = 30
Initial Stock Price = 50
Strike Price = 60
--------------------------------------
R = 1.00025
Up Factor = 1.03045
Uptick Probability = 0.250417
Downtick Probability = 0.249583
Notick Probability = 0.5
--------------------------------------
Price of an American Call Option = 3.44963
Price of an American Put Option = 11.3385
--------------------------------------

real    0m0.259s
user    0m0.246s
sys     0m0.008s
MacBook-Air:Debug sreenivas$
```

Figure 2: Sample output (**Note**: A large number of Divisions should not be a problem when you use Dynamic Programming.)