

Pricing Options Using Trinomial Trees

Paul Clifford Yan Wang Oleg Zaboronski

30.12.2009

1 Introduction

One of the first computational models used in the financial mathematics community was the *binomial tree model*. This model was popular for some time but in the last 15 years has become significantly outdated and is of little practical use. However it is still one of the first models students traditionally were taught. A more advanced model used for the project this semester, is the *trinomial tree model*. This improves upon the binomial model by allowing a stock price to move up, down *or* stay the same with certain probabilities, as shown in the diagram below.

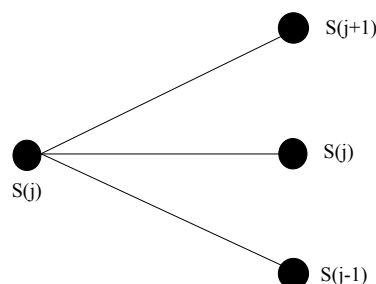
2 Project description.

The aim of the project is to apply the trinomial tree to the following problems:

- Pricing various European and American options
- Pricing barrier options
- Calculating the greeks

More precisely, the students are asked to do the following:

1. Study the trinomial tree and its parameters, p_u, p_d, p_m, u, d
2. Study the method to build the *trinomial tree* of share prices
3. Study the backward induction algorithms for option pricing on trees
4. Price various options such as European, American and barrier



5. Calculate the greeks using the tree

Each of these topics will be explained very clearly in the following sections. Students are encouraged to ask questions during the lab sessions about certain terminology that they do not understand such as barrier options, hedging greeks and things of this nature. Answers to questions listed below should contain analysis of numerical results produced by the simulation.

3 Trinomial Trees

Trinomial trees provide an effective method of numerical calculation of option prices within Black-Scholes share pricing model. Trinomial trees can be built in a similar way to the binomial tree. To create the jump sizes u and d and the transition probabilities p_u and p_d in a binomial model we aim to match these parameters to the first two moments of the distribution of our geometric Brownian motion. The same can be done for our trinomial tree for u, d, p_u, p_m, p_d .

We will use a trinomial tree model defined by

$$S(t + \Delta t) = \begin{cases} S(t)u & \text{with probability } p_u \\ S(t) & \text{with probability } 1 - p_u - p_d \\ S(t)d & \text{with probability } p_d \end{cases}$$

and we can match the first two moments of our models distribution according to the no arbitrage condition,

$$\mathbb{E}[S(t_{i+1})|S(t_i)] = e^{r\Delta t}S(t_i) \quad (1)$$

$$\text{Var}[S(t_{i+1})|S(t_i)] = \Delta t S(t_i)^2 \sigma^2 + \mathcal{O}(\Delta t) \quad (2)$$

where we have assumed that the volatility of the underlying asset σ is constant and the asset price follows geometric Brownian motion; r is the rate of risk-free investment. Condition (1) is a standard equilibrium or *no arbitrage* condition: it states that the average return from the asset should be equal to the risk free-return. It can be re-written in the following explicit form:

$$1 - p_u - p_d + p_u u + p_d d = e^{r\Delta t}.$$

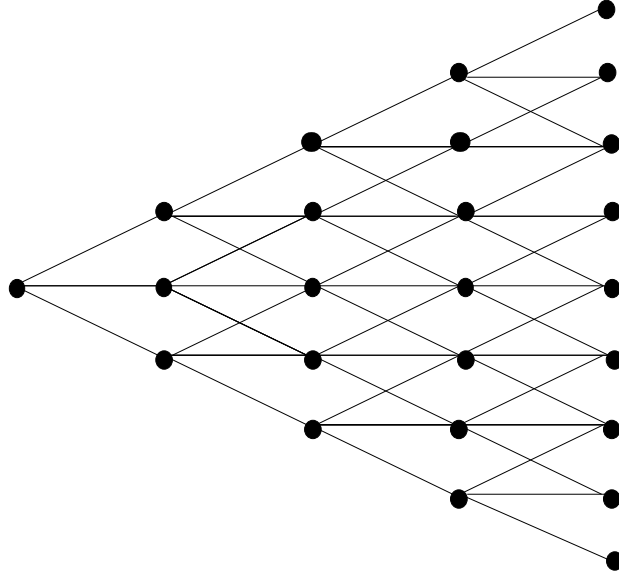
Conditions (1, 2) impose two constraints on 4 parameters of the tree. An extra constraint comes from the requirement that the size of the upward jump is the reciprocal of the size of the downward jump,

$$ud = 1 \quad (3)$$

While this condition is not always used for a trinomial tree construction, it greatly simplifies the complexity of the numerical scheme: it leads to a *recombining* tree, which has the number of nodes growing polynomially with the number of levels rather than exponentially.

Given the knowledge of jump sizes u, d and the transition probabilities p_u, p_d it is now possible to find the value of the underlying asset, S , for any sequence of price movements. A directed graph with nodes labeled by asset prices and edges connecting nodes separated by one time step and a single price jump is the desired *trinomial pricing tree*, see Fig. 3. Let us define the number of up, down and middle jumps as N_u, N_d, N_m , respectively, and so the value of the underlying share price at node j for time i is given by

$$S_{i,j} = u^{N_u} d^{N_d} S(t_0), \quad \text{where } N_u + N_d + N_m = N$$



3.1 Trinomial Tree Parameter Set

We have imposed three constraints (1, 2, 3) on four parameters u, d, p_u and p_m . As a result there exists a family of trinomial tree models. In the project we consider the following popular representative of the family: its jump sizes are

$$u = e^{\sigma\sqrt{2\Delta t}}, \quad d = e^{-\sigma\sqrt{2\Delta t}} \quad (4)$$

Its transition probabilities are given by

$$p_u = \left(\frac{e^{\frac{r\Delta t}{2}} - e^{-\sigma\sqrt{\frac{\Delta t}{2}}}}{e^{\sigma\sqrt{\frac{\Delta t}{2}}} - e^{-\sigma\sqrt{\frac{\Delta t}{2}}}} \right)^2 \quad (5)$$

$$p_d = \left(\frac{e^{\sigma\sqrt{\frac{\Delta t}{2}}} - e^{\frac{r\Delta t}{2}}}{e^{\sigma\sqrt{\frac{\Delta t}{2}}} - e^{-\sigma\sqrt{\frac{\Delta t}{2}}}} \right)^2 \quad (6)$$

$$p_m = 1 - p_u - p_d \quad (7)$$

Before we can use this model, we need to ensure that it is well-posed and satisfies all the necessary constraints imposed by recombination requirement and the underlying continuum limit.

Exercise 1: Derivation of Moments

1. Assuming that $\sigma\sqrt{\frac{\Delta t}{2}} > \frac{r\Delta t}{2}$, prove that $p_u + p_d < 1$.
2. Verify that $\mathbb{E}(S_{n+1}|S_n) = e^{r\Delta t}S_n$.
3. Verify that $\text{Var}(S_{n+1}|S_n) = \sigma^2\Delta t S_n^2 + O(\Delta t^{3/2})$.

4 Pricing Options Using Trinomial Trees

From the previous sections, it should be clear what we need in order to implement an option pricing algorithm using a trinomial tree. For pricing options on a trinomial tree we need to generate 3 separate quantities

- The transition probabilities of various share price movements. These are p_u, p_d , and p_m .
- The size of the moves up, down and middle. These are u, d and $m = 1$
- The payoff or *terminal condition* of our option at maturity i.e the end (or leaf) nodes!

So we have now explained that all we need to price an option using a trinomial tree are the following parameters

- p_u, p_d, p_m
- u, d, m
- Option type, i.e call, put

Once we have derived these parameters or have been given these parameters, then we can implement the option pricing algorithm explained in the next section.

4.1 European Option Pricing Using Trees

European call (put) option is a contract which gives its owner the right to buy (sell) an agreed asset (underlying asset) at the agreed price K (strike price) at the specified time T (maturity time).

The methodology when pricing options using a trinomial tree is exactly the same as when using a binomial tree. Once the share price tree is built, and the option payoffs at maturity time T are calculated:

$$C(S, T) = \max(S - K, 0) \text{ (Call option) ,} \quad (8)$$

$$C(S, T) = \max(K - S, 0) \text{ (Put option) .} \quad (9)$$

After that it remains to apply the following backward induction algorithm, where n represents the time position and j the space position:

$$C_{n,j} = e^{-r\Delta t} [p_u C_{n+1,j+1} + p_m C_{n+1,j} + p_d C_{n+1,j-1}] \quad (10)$$

The backward induction algorithm can be derived from the risk-neutrality principle and is the same for put and call options. When applied in the context of a trinomial tree (using the exact same methodology as the binomial tree), we can calculate the option value at interior nodes of the tree by considering it as a *weighting* of the option value at the future nodes, discounted by one time step. Thus we can calculate the option price at time n , C_n , as the option price of an up move $p_u C_{n+1}$ plus the option price of the middle move by $p_m C_{n+1}$ plus the option price of a down move by $p_d C_{n+1}$, discounted by one time step, $e^{-r\Delta t}$. So at any node on the tree our backward induction formula, (10), is applied to give us the option prices at any node in the tree. The name of the algorithm should now be clear since we only need to value the option at maturity, i.e the leaf nodes, and then work our way *backwards* through the tree calculating option values at all the nodes until we reach the root S_0, C_0 . The pseudo-code for an algorithm for pricing European options with N time steps is given below (Algorithm 1).

Note that N is the number of time steps. Therefore the number of tree levels is $N + 1$. Price neutrality principle leads to a relation between prices of put and call options known as 'put-call parity':

$$C_{call}(0) - C_{put}(0) = S(0) - e^{-rT} K. \quad (11)$$

Relation (11) is very useful for verifying correctness of your algorithms.

Algorithm 1 European Option Pricing Algorithm For Trees

```
1: Declare and initialize  $S(0)$ 
2: Calculate the jump sizes  $u, d$  and  $m$ 
3: Calculate the transition probabilities  $p_u, p_d$  and  $p_m$ 
4: Build the share price tree
5: Calculate the payoff of the option at maturity, i.e node  $N$ 
6: for (int  $i = N; i > 0; --i$ ) do
7:   Calculate option price at node  $i$  based on
8:    $C_{n,j} = e^{-r\Delta t} [p_u C_{n+1,j+1} + p_m C_{n+1,j} + p_d C_{n+1,j-1}]$ 
9: end for
10: Output option price
```

Exercises 2: European Option Pricing

1. Derive recursion (10) from the principle of price neutrality.
2. Derive relation (11) for European call and put options.
3. Price the European put and call options using the set of parameters given below. Produce tabulated output of your answers.
4. Check numerically that your answers satisfy put-call parity relation (11).

Use the following set of initial prices expressed in dollars,

$$S(0) = [\$40, \$50, \$60, \$70, \$80, \$90, \$100, \$110, \$120, \$130, \$140, \$150]$$

and the other fixed financial parameters are given by

$$\begin{aligned} K &= \$90 \\ \sigma &= 20 \text{ Percent per year}^{1/2} \text{ (annual volatility)} \\ r &= 5 \text{ Percent per year} \\ T &= \frac{1}{2} \text{ year} \\ \# \text{ of time steps } N &= 100 \end{aligned}$$

N. B. Make sure that all exponents used in your calculations are *dimensionless*.

5 Convergence of the trinomial tree to Black-Scholes option price formula.

Having priced the options numerically using the trinomial tree, one can compare the answers for European option prices against the predictions of Black-Scholes formula. By closed-form, we mean that we can calculate the price explicitly and without implementing a trinomial tree. There are various Black-Scholes calculators on the internet, but an even easier way is to make use of the Matlab command *blsprice*. An example of using it is given in the help section of matlab.

```
>> [Call, Put] = blsprice(100, 95, 0.1, 0.25, 0.5)
```

```
Call =
```

13.6953

Put =

6.3497

>>

As $N \rightarrow \infty$, the random walk on the trinomial tree converges to geometric Brownian motion, see the lectures where a similar correspondence was established between Black-Scholes and Cox-Ross-Rubinstein theory. Therefore, we expect our answers for option prices to converge to the corresponding answers given by Black-Scholes pricing formula. In the parameter set we gave you earlier, we asked you to use $N = 100$. The purpose of the present exercise is to investigate this convergence.

Exercise 3: Comparing Trinomial Tree to Black-Scholes theory. (Answer each of the questions below both for call and put options.)

1. For the array of initial share prices S , produce a table showing the difference between your trinomial tree answers and the exact Black-Scholes answers. You can use the matlab `blsprice` command for this.
2. Vary the number of time steps, N , in your trinomial tree and plot the convergence of your trinomial tree solution to that of the exact Black-Scholes solution.
3. What is the minimal value of N the relative error of the trinomial tree computation becomes less than 0.1% for the initial share price $S(0) = \$90$ and all option types?

6 American Option Pricing Using Trees

American call (put) option is a contract which gives it's owner the right to buy (sell) an agreed asset (underlying asset) at the agreed price K (strike price) at *or before* the specified time T (maturity time). Therefore, an American option can be exercised at *any* time τ , up until maturity T , where $0 < \tau \leq T$ where 0 stands for present time. The backward recursion (10) has to be modified accordingly. Fortunately, the modification can be still derived using the principle of price neutrality. The pay-off for American put and call options can be now defined at every node of the tree, not just the leafs:

$$\text{Call option payoff} = \max(S_{i,j} - K, 0) \quad \text{Put option payoff} = \max(K - S_{i,j}, 0). \quad (12)$$

The modified backward recursion takes the following form:

$$C_{n,j} = \max(\text{option payoff}, e^{-r\Delta t} [p_u C_{n+1,j+1} + p_m C_{n+1,j} + p_d C_{n+1,j-1}]) \quad (13)$$

In words, at each node of our tree we need to implement a comparison function between the payoff value of the option (the so-called 'intrinsic value') the the value resulting from the backward induction step. The corresponding pseudo-code is presented below (Algorithm 2).

Exercises 4: American Option Pricing

Students need to answer the following questions

1. Derive (13).

Algorithm 2 Option Pricing Algorithm For Trees

```
1: Declare and initialize  $S(0)$ 
2: Calculate the jump sizes  $u, d$  and  $m$ 
3: Calculate the transition probabilities  $p_u, p_m$  and  $p_d$ 
4: Build the share price tree
5: Calculate the payoff of the option at maturity, i.e node  $N$ 
6: for (int  $i = N; i > 0; --i$ ) do
7:   Calculate option price at node  $i$ , but use the formula,  $\max(\text{option payoff, tree price})$ 
8:    $C_{n,j} = \max(\text{option payoff}, e^{-r\Delta t} [p_u C_{n+1,j+1} + p_m C_{n+1,j} + p_d C_{n+1,j-1}])$ 
9: end for
10: Output option price
```

2. Compute both American call and put options for the exact same parameter set as was given in the European option pricing section using trinomial trees.
3. Produce tabulated output of your answers.
4. What can you say about American option prices versus European option prices ?
5. Does put-call parity (11) hold for American options? Explain your findings.

7 Pricing Double Barrier Knock-Out European Options

The trinomial tree method can be used to price options in the situations when closed form expressions for the price do not exist.

Barrier options are a type of exotic options, which allow an investor to bet on more complicated properties of the price movements of the underlying asset than just its final price.

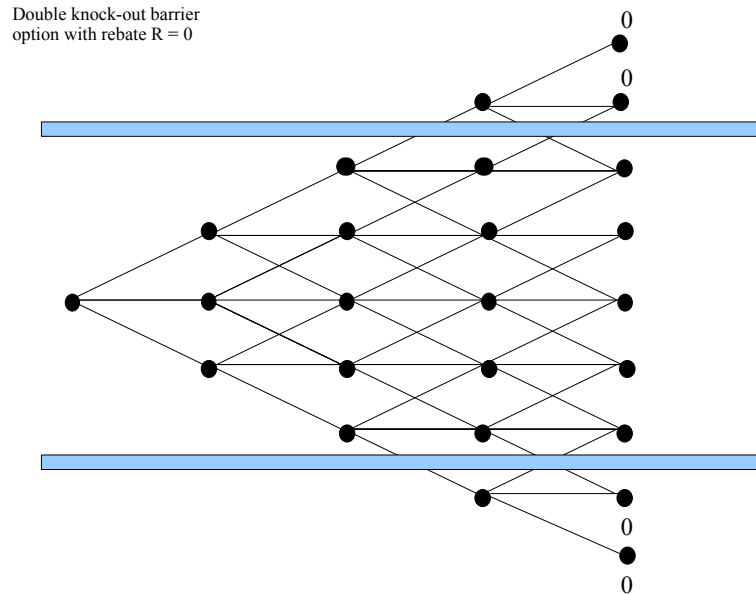
For example, an investor believes that an XXX PLC share which currently stands at \$80 will grow in a year to \$100. Basing on this belief the investor can buy a one year long European call option for this underlying asset with a strike price K below \$100. Assume that in addition, the investor has reasons to believe that during this year the share price will never exceed \$150 and will not fall below \$50. In this case, the investor can buy a cheaper *double barrier knock-out European call option* which has the same strike price, but becomes *null* and *void* if the price of the underlying asset strictly exceeds \$100 or falls strictly below \$50 at any time before or at maturity.

There are many other types of barrier options, see [http : //en.wikipedia.org/wiki/Barrier_option](http://en.wikipedia.org/wiki/Barrier_option) for more details, but general principle is the same for all 'exotic' options: an informed investor can save money by betting on some specific details of the stock price movement.

To price barrier options, we need to modify our algorithm by incorporating the barrier condition into the terminal option payoff at the maturity nodes and into the backward recursion step.

Exercise 5: Pricing Barrier Options

1. Derive modified terminal-pay off formula and the backward recursion for the European double knock out barrier option.
2. Price double barrier knock-out European call and put options based on upper and lower barriers of 130 and 60 respectively for the set of initial share prices $S(0)$ used to price 'vanilla' European options. Use the following parameters $K = 90, T = 0.5, r = 0.05, \sigma = 0.20$. Plot option prices as functions of the initial asset price.
3. Analyse the convergence of this method and compare it to the standard European option with no barriers.
4. Give a table of values for varying the number of time levels n to show the rates of convergence for both methods



8 Calculating the Greeks

The so called "greeks" are various hedging parameters that can be computed from the underlying option price. These are a necessary tool for any trader or structurer. The greeks are used to eliminate risk and they give instructions as to how the trader or structurer can disperse the risk they have taken on by selling an option. If we think of a call option (the analogous case applies for a put), as a function of various parameters $C(S, K, \sigma, r, T)$. So our call option price is a function that takes in 5 parameters and returns one parameter, our option price. So we can calculate partial derivatives of this option price with respect to various parameters. Each greek is given a different name depending on what derivative is taken. There are numerous greeks but we will concern ourselves with the two major ones - delta and gamma.

- delta $\Delta = \frac{\partial C}{\partial S}$
- gamma $\Gamma = \frac{\partial^2 C}{\partial S^2}$

So our delta represents the rate of change of our option price with respect to our underlying asset S . The gamma represents the rate of change of our delta with respect to the underlying. So we can see how these can be used to hedge some risk. If we know that our option price changes by a certain amount when our underlying share prices changes by a certain amount then we can use this extra information to hedge away some risk. For now, students should be concerned with the following question

How to evaluate and calculate greeks using a trinomial tree ?

If we simplify the reasoning of what a greek is, it should become clear what we need to do. A delta is the rate of change (derivative) of a variable, C , with respect to the underlying asset S . So we just need to discretize the partial derivative and we have our first greek, delta. To calculate various greeks, we can implement both a first order method and a second order method for calculating delta, $\frac{\partial C}{\partial S}$, and gamma $\frac{\partial^2 C}{\partial S^2}$. We just implement a numerical difference type discretization of these partial derivatives that are $\mathcal{O}(\Delta S)$ and $\mathcal{O}(\Delta S^2)$.

$$\begin{aligned}\frac{\partial C}{\partial S} &= \frac{C(S(0) + \Delta S) - C(S(0))}{\Delta S} = \frac{C(S(0) + \Delta S) - C(S(0) - \Delta S)}{2\Delta S} \\ \frac{\partial^2 C}{\partial S^2} &= \frac{C(S(0) + \Delta S) - 2C(S(0)) + C(S(0) - \Delta S)}{\Delta S^2}\end{aligned}$$

where $S(0)$ is the initial share price we want to calculate the option price for. ΔS is a small increment we create to evaluate the derivative $\frac{\partial C}{\partial S}$ numerically. Students can take this to be some tiny percentage of $S(0)$. We can now just create 2 separate trinomial trees, one with an initial share price of $S(0)$ and the other with an initial share price of $S(0) + \Delta S$ and apply the formula for delta! This is the first order scheme and is accurate to $\mathcal{O}(\Delta S)$, however if we implement a second order accurate discretization of $\frac{\partial C}{\partial S}$, we need to calculate option prices for 2 different initial share prices, $S(0) + \Delta S$ and $S(0) - \Delta S$. So we have a trade-off here, for a more accurate numerical discretization of the delta, we need to implement 3 separate trinomial trees - 2 for the second order discretization of the delta and one normal trinomial tree with initial share price $S(0)$ for the usual calculation of our option price.

The calculation of gamma applies the exact same logic we have just shown for delta, using separate trinomial trees and applying the formula $\frac{\partial^2 C}{\partial S^2} = \frac{C(S(0)+\Delta S)-2C(S(0))+C(S(0)-\Delta S)}{\Delta S^2}$

Exercise 6: Calculating the "Greeks"

After pricing the European and American options in the above section, students should know also

1. Calculate the delta and gamma of all the options priced in using the parameters of exercise 1(b)
2. Give a table showing your answers for these delta's and gamma's
3. Give one plot each for delta and gamma over the range of initial share prices

9 Working with market data.

Exercise 6: Trading AMD European options.

1. Download the file **options.20060321.csv** from the financial mathematics home page. Use it to extract the following data: AMD (Advanced Micro Devices) (close) share price on the 21st of March, 2006, the set of strike prices for European call and put options available on this day (rows 7703-7744 of the Excel sheet) and having AMD share as the underlying asset; the set of bid and ask prices for these options; expire date. The structure of the file is explained at

<http://www.historicaloptiondata.com/structure.aspx#structure>

2. Go to

<http://www.cboe.com/data/HistoricalVolatility06.aspx>

to find the value of the volatility of AMD stock in March 2006.

3. Federal reserve one-month interest rates in March 2006 stood at 4.55 per cent. Use all the data you've collected so far to calculate the trinomial tree option price for each of the strike prices available. Plot bid, ask and tree prices as functions of the strike price on the same semi-logarithmic plot.
4. Go to

<http://finance.yahoo.com/q/hp?s=AMD>

to find out the price of AMD share on the 21st of April 2006. Assume that an investor paid the ask price for each of 42 AMD European put, call options on the 21st of March. How much money will the investor gain by exercising all these options on the 21st of April? What would be the answer if our investor bought put options only? Plot the profits from exercising put and call options as functions of K on the same plot. Also, plot the volumes of each option type as functions of K on this plot. Has the market guessed the most profitable options correctly? On March the 21st, would you buy the options which turned out to be most profitable on April the 21st, basing on the trinomial tree result?

10 Structuring Your MATLAB Code

Due to the complexity of this project, we would recommend to students to write various functions in matlab "*m scripts*" to calculate various items needed for each part of the project. Students should avoid writing one long piece of matlab code that implements everything. It will be significantly easier to debug your matlab code if you write numerous small functions that calculate various things. For instance, an example piece of our main code for calculating a European option is given below. You can clearly see how easy it is to read and understand.

```
0001 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0002 %
0003 % Filename: trinomial_tree.m
0004 %
0005 % Code: This code calculates European option prices using
0006 %       a trinomial tree.
0007 %
0008 % Other Files Used: trans_prob.m, calc_ud.m, calc_pmd.m
0009 %                   backward_induction.m, visualization.m
0010 %
0011 % Author: Paul Clifford  Oleg Zaboronski
0012 %
0013 % Last Modified: Nov 12th
0014 %
0015 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0016
0017 function y = trinomial_tree( S, K, T, r, vol, n, op-type )
0018
0019 delta_T = T/(n-1);
0020
0021 [u,d]      = calculate_ud( n, delta_T, vol );
0022 [pu,pm,pd] = calculate_pupmpd( delta_T, vol );
0023
0024 x = build_price_tree();
0025 compute_payoff(x, K, op-type);
0026 answer = backward_evaluation();
```

Students should bare in mind that the example above is just one of the many codes needed for the project. You also have numerous other tasks so creating folder for each task and then inside each folder have all your m-scripts for that exercise. So for example, for this trinomial tree, we created a folder called "trinomial_tree" and that folder contains all our m-files **only for** the trinomial tree. When we write code for barrier options or for the Greeks we create separate folders for each of these exercises and write separate m scripts again for various calculations.

11 Presentation of the Project

Clear and concise explanation and presentation of results will be rewarded with high marks. There is no need to include numerous non-illustrative plots but it is always a good idea to emphasize (a) the financial reasoning for the plots and (b) the numerical or mathematical explanation of what is occurring in the plots. A project with few clearly explained

plots will receive a significantly higher mark than a project containing numerous plots with little or no clear explanation.

Students should also note that data tables are an excellent way to explain your answers and are preferable to graphs on a number of occasions. These are especially useful for convergence results and also pricing results.

12 Tabulated Output

Students should produce concise tabulated output to illustrate the performance of their trinomial tree algorithms, for example

Numerical	Exact	Error I	Error II	Evaluations	Level	Time
24.860025	24.773462	0.086563	-2.4468	8423	5	0.16402
24.760346	24.773462	0.013115	-4.33395	30182	6	0.10937
24.769341	24.773462	0.004120	-5.49168	112916	7	0.31250
24.773481	24.773462	1.8950e-05	-10.8736	394143	7	0.98040
24.773577	24.773462	2.1952e-05	-9.07785	1299782	7	2.97650
24.773487	24.773462	2.5404e-05	-10.5807	4130015	7	8.97656

13 Books, Papers and References

We encourage students to read some books/papers on trinomial trees. For example:

- John C. Hull "Options, Futures and Other Derivatives"
- Paul Wilmott "Paul Wilmott on Quantitative Finance"
- Espen Haug "The Complete Guide To Option Pricing Formulas"

There are also various papers available via Google online.

14 Rules, Regulations and Project Submission

This project is to be completed by individuals only and is not a group exercise. Plagiarism is taken extremely seriously and any student found to be guilty of plagiarism of fellow students will be severely punished.

1. Students must answer all questions.
2. Students are free to use Word, Latex or whatever project writing software they prefer.
3. The project must be handed up in either bound format or stapled format, no loosely packaged project will be accepted.
4. Students should code the assignment in MATLAB.
5. All graphs produced **must** be labeled displaying what the graph is, what parameters were used to produce the graph, colour-coded "key" attached to show clear understanding of what the graph represents.
6. Graphs produced not adhering to the rule above will not be marked!
7. Students can produce data tables to show answers if they wish. If so, data must be aligned, include column headings and be easy to read.