Applied Computational Finance

Lesson 1 - Pricing with Trees

Giovanni Della Lunga

Laurea Magistrale in Quantitative Finance

Bologna - April-May 2021

Outline

Course Contents and Learning Outcomes

- This series of lectures focuses on a number of topics that have one point in common.
- They are all related to the need to manage the problem of early exercise.
- In doing this we will present some numerical techniques of general use in the field of computational finance.
- In the first part we are going to deal with finite difference tools which, in many cases, provide an extremely efficient numerical solution from the computational point of view.
- But we will soon find a big problem to address. The main weakness of these procedures is that they do not generalize very easily to the case of multiple risk factors.

Course Contents and Learning Outcomes

- We are therefore faced with a dilemma.
- On the one hand we have procedures such as the traditional Monte Carlo method by means of which we can easily manage the problem of dimensionality but which do not apply to the case in which an early exercise is allowed.
- On the other hand, we have finite difference methods which simply deal with the case of anticipated exercise but which cannot easily be extended to the case of multiple dimensions.
- In the second part of the course we will solve this dilemma by introducing the solution currently used: the so-called Least Squares Montecarlo Method (Longstaff and Schwartz, 2001).

Course Contents and Learning Outcomes

This is the general structure of this course:

- Pricing with Trees
- Introduction to Finite Difference Methods for Solving Partial Derivative Differential Equations
- The use of Finite Difference in More than One Dimension: the Heston Model example
- Improving Montecarlo Efficiency: Variance Reduction Techniques
- Introduction to the Longstaff-Schwartz Montecarlo Method

Teaching tools: Jupyter Notebook

- In these lessons we'll use the Python Programming Language;
- The Python world developed the IPython notebook system.
- Notebooks allow you to write text, but you insert code blocks as "cells" into the notebook.
- A notebook is interactive, so you can execute the code in the cell directly!
- Recently the Notebook idea took a much enhanced vision and scope, to explicitly allow languages other than Python to run inside the cells.
- Thus the Jupyter Notebook was born, a project initially aimed at Julia, Python and R (Ju-Pyt-e-R). But in reality many other languages are supported in Jupyter.

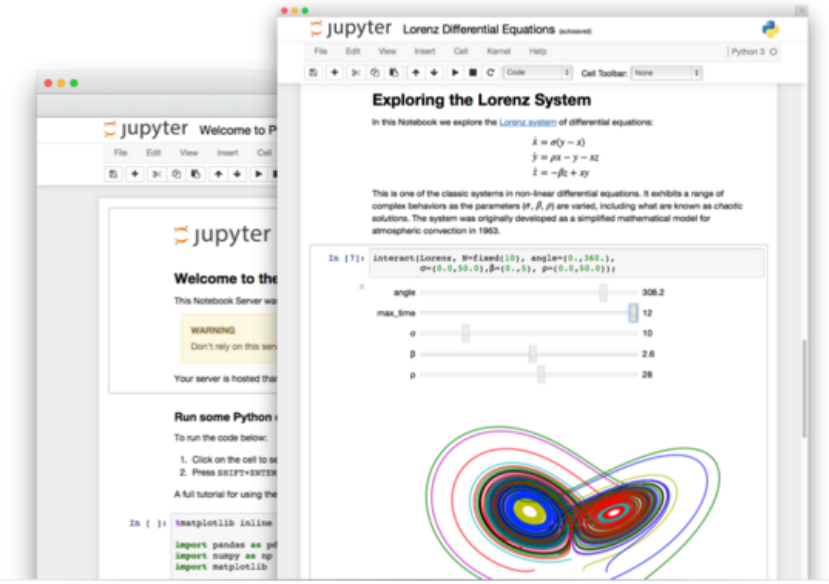
Teaching tools: Jupyter Notebook

Jupyter Notebook

The Jupyter Notebook is a web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more.

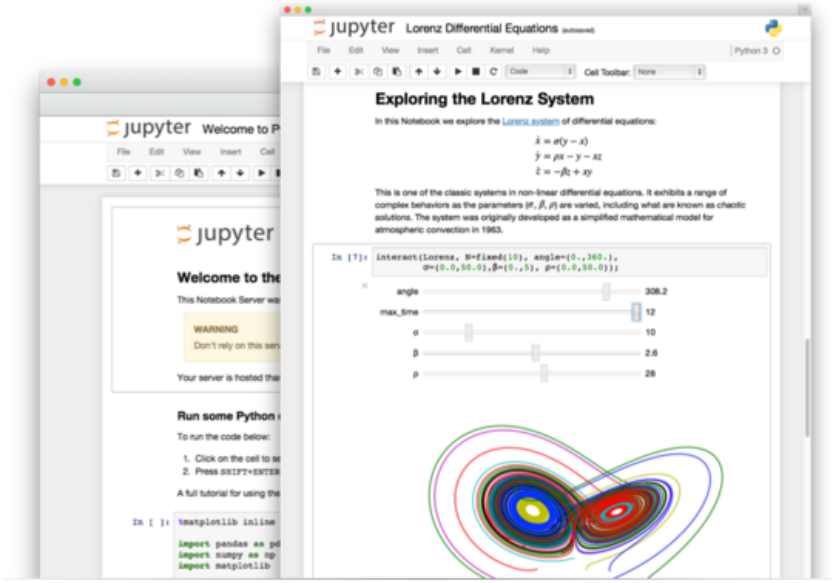
Teaching tools: Jupyter Notebook

- Jupyter was designed to enable sharing of notebooks with other people.
- The idea is that you can write some code, mix some text with the code, and publish this as a notebook.
- In the notebook they can see the code as well as the actual results of running the code.



Teaching tools: Jupyter Notebook

- This is a nice way of sharing little experimental snippets, but also to publish more detailed reports with explanations and full code sets.
- Of course, a variety of web services allows you to post just code snippets (e.g. gist).
- What makes Jupyter different is that the service will actually render the code output.



Teaching tools: Google Colab

- Colaboratory, or "Colab" for short, is a product from Google Research.
- Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

https://colab.research.google.com/notebooks/intro.ipynb?hl=en

Teaching tools: Google Colab

- More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs.
- Colab notebooks are stored in *Google Drive*, or can be loaded from *GitHub*. Colab notebooks can be shared just as you would with Google Docs or Sheets.

https://colab.research.google.com/notebooks/intro.ipynb?hl=en

GitHub Repository for this Course

- GitHub is a provider of Internet hosting for software development and version control using Git. It offers the distributed version control and source code management (SCM) functionality of Git, plus its own features.
- It provides access control and several collaboration features such as bug tracking, feature requests, task management, continuous integration for every project.
- You can find all the teaching materials (notebook, slides, code, etc...) at this address **https://github.com/polyhedron-gdl** in the repository **applied-computational-finance/2021**.

Assessment Methods

- The final exam consists in the development of a python procedure for the solution of a pricing problem.
- The procedure must be described in a notebook that will be discussed with me at the time of the exam.
- A good knowledge of Python Language and a Beginner Level Knowledge of Jupyter Notebooks are both fundamental for the Assessment, so don't exitate to ask me more about these items...

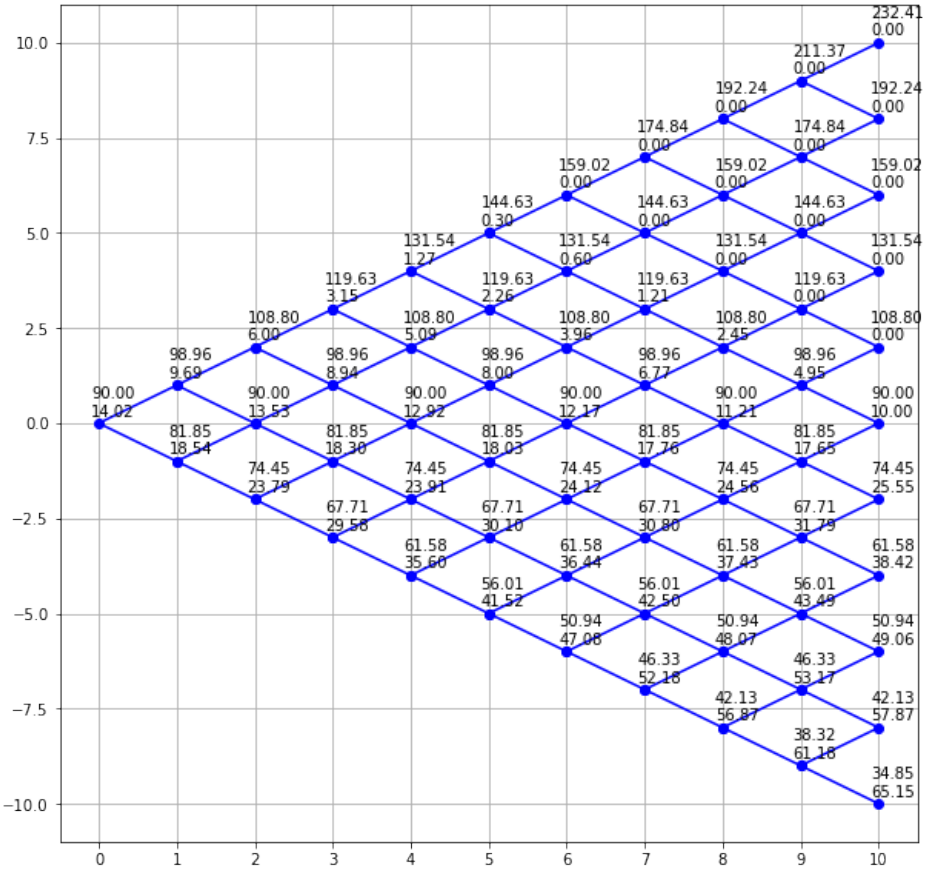

LET'S
START

Outline

Binomial Pricing Model

- The binomial pricing model traces the evolution of the option's key underlying variables in discrete-time.
- This is done by means of a binomial lattice (Tree), for a number of time steps between the valuation and expiration dates.
- Each node in the lattice represents a possible price of the underlying at a given point in time.
- Valuation is performed iteratively, starting at each of the final nodes (those that may be reached at the time of expiration), and then working backwards through the tree towards the first node (valuation date).
- The value computed at each stage is the value of the option at that point in time.

Binomial Pricing Model



Option valuation using this method is a three-step process:

- Price tree generation,
- Calculation of option value at each final node,
- Sequential calculation of the option value at each preceding node.

Create the Binomial Price Tree

- The tree of prices is produced by working forward from valuation date to expiration.
- At each step, it is assumed that the underlying instrument will move up (with probability p) or down (with probability $q = 1 - p$) by a specific factor u or d per step of the tree (where, by definition, $u \geq 1$ and $0 < d \leq 1$)
- So, if S is the current price, then in the next period the price will either be $S_{up} = S \cdot u$ or $S_{down} = S \cdot d$.
- The up and down factors are calculated using the underlying volatility, σ , and the time duration of a step, Δt, measured in years (using the day count convention of the underlying instrument).

Create the Binomial Price Tree

- With respect to the definition of p, u and d some important assumptions about the behavior of the stochastic process of the underlying stock have to be done. In the following we'll follow the framework set up by Cox, Ross and Rubinstein.
- Let's assume that the stochastic process is continuous as $n \to \infty$. The parameters p, u and d must be chosen in a way to determine the right values of the stock expected return and variance at the end of each interval Δt.

Create the Binomial Price Tree

- For each step, the expected future stock price is

$$E_\pi[S_{t+\Delta t}] = pS_t u + (1-p)S_t d \Rightarrow \frac{E_\pi[S_{t+\Delta t}]}{S_t} = pu + (1-p)d \quad (1)$$

and, conditional to S_t

$$E_\pi\left[\frac{S_{t+\Delta t}}{S_t}|S_t\right] = pu + (1-p)d \quad (2)$$

The variance of this return is given by

$$\begin{aligned} Var\left[\frac{S_{t+\Delta t}}{S_t}|S_t\right] &= E_\pi\left[\left(\frac{S_{t+\Delta t}}{S_t}\right)^2|S_t\right] - \left(E_\pi\left[\frac{S_{t+\Delta t}}{S_t}|S_t\right]\right)^2 \\ &= [pu^2 + (1-p)d^2] - [pu + (1-p)d]^2 \end{aligned}$$

Create the Binomial Price Tree

- The Cox, Ross and Rubinstein binomial tree model assume that in the limit $n \to \infty$, the discrete process must converge to a geometric brownian motion for the underlying stock price.
- So the idea is to find values for u and d in order to have

$$Var\ [X_t|S_t] = \sigma^2 \Delta t$$

where $X_t = \frac{S_{t+\Delta t}}{S_t}$

- We add an additional assumption in order to obtain a tree that recombines, so that the effect of a down movement followed by an up movement is the same as the effect of an up movement, followed by a down movement.

$$u = \frac{1}{d} \tag{3}$$

Create the Binomial Price Tree

- With a bit of trivial algebra we find

$$\begin{aligned} Var\ [X_t|S_t] &= pu^2 + (1-p)d^2 - p^2u^2 - (1-p)^2d^2 - 2p(1-p)ud \\ &= pu^2(1-p) + p(1-p)d^2 - 2p(1-p)ud \\ &= p(1-p)(u^2 + d^2 - 2ud) \\ &= p(1-p)(u-d)^2 \end{aligned} \tag{4}$$

- Now it's time to say something more about the probability p
- For this we will resort to the hypothesis of absence of arbitrage...

Create the Binomial Price Tree

- For our purposes it's enough to consider only one period.
- Let's say that at time $t + \Delta t$ an option on S can assume only two possibile values: either C_u or C_d
- If we have a portfolio with a long position of α shares of the stock and a short position in the option, the value of our portfolio in the two states will be:

$$\alpha \cdot S_t \cdot u - C_u$$

$$\alpha \cdot S_t \cdot d - C_d$$

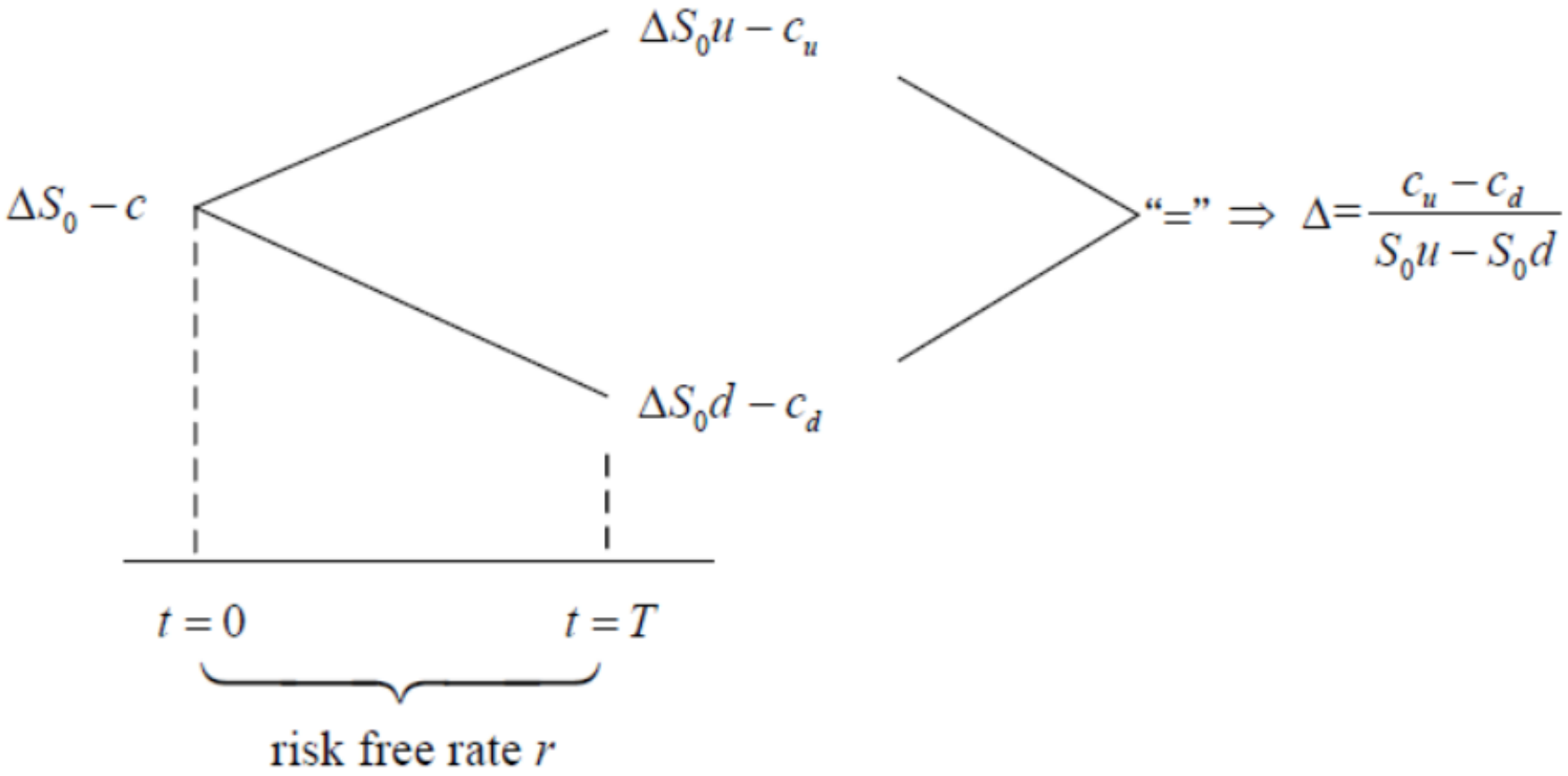
Create the Binomial Price Tree

- It is easy to check that if we choose α as

$$\alpha = \frac{C_u - C_d}{(u-d)S_t} \sim \frac{\partial C}{\partial S} \tag{5}$$

  our portfolio will have the same value in both states so it will be a **risk free** portfolio.

Create the Binomial Price Tree



Create the Binomial Price Tree

- To preclude the existence of arbitrage the return of our portfolio must be the risk free rate, so in Δt we have

$$P_{t+\Delta t} = e^{r\Delta t} P_t$$

- If c_t is the option value at time t we can write

$$\frac{P_{t+\Delta t}}{P_t} = \frac{S_t \alpha \cdot u - c_u}{S_t \alpha - c_t} = e^{rT} \tag{6}$$

from this we can obtain the value for c_t:

$$\begin{aligned} c_t &= \frac{S_t \cdot \alpha \cdot e^{r\Delta t} + c_u - u \cdot S_t \cdot \alpha}{e^{r\Delta t}} \\ &= \frac{S_t \cdot \alpha \cdot (e^{r\Delta t} - u) + c_u}{e^{r\Delta t}} \end{aligned}$$

Create the Binomial Price Tree

- Using the previous value for α

$$
\begin{aligned}
c_t &= \frac{\frac{c_u - c_d}{u-d}\left(e^{r\Delta t} - u\right) + c_u}{e^{r\Delta t}} \\
&= \frac{c_u e^{r\Delta t} - c_u u - c_d e^{r\Delta t} + c_d u + c_u u - c_u d}{e^{r\Delta t}(u - d} \\
&= e^{-r\Delta t}\frac{c_u(e^{r\Delta t} - d) + c_d(u - e^{r\Delta t})}{(u-d)} \\
&= e^{-r\Delta t}\left[c_u \frac{e^{r\Delta t} - d}{u-d} + c_d \frac{u - e^{r\Delta t}}{u-d}\right] \\
&= e^{-r\Delta t}\Big[p \cdot c_u + (1-p)\cdot c_d\Big] \qquad (7)
\end{aligned}
$$

Create the Binomial Price Tree

- In the previous equation

$$p \equiv \frac{e^{rT} - d}{u - d} \qquad 1 - p \equiv \frac{u - e^{rT}}{u - d} \tag{8}$$

  if we assume $d \leq r \leq d$, p is always greater than zero and less than one showing the basic properties of a probability measure.
- This is the so called **risk neutral probability**
- Now we can compute the variance

$$Var\ [X_t|S_t] = p(1-p)(u-d)^2 = \sigma^2 \Delta t \tag{9}$$

Create the Binomial Price Tree

Using the value for p, we obtain

$$\begin{aligned} Var\ [X_t|S_t] &= p(1-p)(u-d)^2 \\ &= \frac{e^{r\Delta t}-d}{u-d}\left[\frac{u-e^{r\Delta t}}{u-d}\right](u-d)^2 \\ &= (e^{r\Delta t}-d)(u-e^{r\Delta t}) \\ &= e^{r\Delta t}(u+d)-ud-e^{2r\Delta t} \end{aligned} \tag{10}$$

from the last row, keeping the lower order terms we have

$$\begin{aligned} Var\ [X_t|S_t] &= e^{r\Delta t}(u+d)-ud-e^{2r\Delta t} \\ &\sim (1+r\Delta t)(u+d)-1-1-2r\Delta t \\ &= (1+r\Delta t)(u+d-2) \end{aligned} \tag{11}$$

Create the Binomial Price Tree

If we choose

$$u = e^{\sigma\sqrt{\Delta t}}, \qquad d = \frac{1}{u} = e^{-\sigma\sqrt{\Delta t}} \tag{12}$$

up to the first order in Δt we have

$$u \sim 1 + \sigma\sqrt{\Delta t} + \frac{1}{2}\sigma^2\Delta t, \qquad d \sim 1 - \sigma\sqrt{\Delta t} + \frac{1}{2}\sigma^2\Delta t \tag{13}$$

Create the Binomial Price Tree

and finally

$$
\begin{aligned}
Var\ [X_t|S_t] &= (1+r\Delta t)(u+d-2) \\
&\sim (1+r\Delta t)\left(1+\sigma\sqrt{\Delta t}+\frac{1}{2}\sigma^2\Delta t+1-\sigma\sqrt{\Delta t}+\frac{1}{2}\sigma^2\Delta t-2\right) \\
&= (1+r\Delta t)\left(\sigma^2\Delta t\right) \\
&\sim \sigma^2\Delta t+O\left((\Delta t)^2\right) \qquad (14)
\end{aligned}
$$

Create the Binomial Price Tree

- As we have already said, the CRR method ensures that the tree is recombinant, i.e. if the underlying asset moves up and then down (u, d), the price will be the same as if it had moved down and then up (d, u) — here the two paths merge or recombine.
- This property reduces the number of tree nodes, and thus accelerates the computation of the option price.
- This property also allows that the value of the underlying asset at each node can be calculated directly via formula, the node-value will be:

$$S_n = S_0 \cdot u^{N_u - N_d} = S_0 \cdot u^{N_u} \cdot d^{N_d}$$

Where N_u is the number of up ticks and N_d is the number of down ticks.

Find option value at each final node

At each final node of the tree — i.e. at expiration of the option—the option value is simply its intrinsic, or exercise, value:

$$Max[(S_n - K), 0], \quad \text{for a call option}$$

$$Max[(K - S_n), 0], \quad \text{for a put option}$$

Where K is the strike price and S_n is the spot price of the underlying asset at the $n - th$ period.

Find option value at earlier nodes

- Once the above step is complete, the option value is then found for each node, starting at the penultimate time step, and working back to the first node of the tree (the valuation date) where the calculated result is the value of the option.
- The "binomial value" is found at each node, using the risk neutrality assumption. Under this assumption,the expected value is calculated using the option values from the later two nodes (Option up and Option down) weighted by their respective risk neutral probabilities — p of an up move in the underlying, and $(1-p)$ of a down move.

Find option value at earlier nodes

- The expected value is then discounted at r, the risk free rate corresponding to the life of the option.
- We finally obtain the iteration formula:

$$c_{t-\Delta t,i} = e^{-r\Delta t}\left[p \cdot c_{t,i} + (1-p) \cdot c_{t,i+1}\right] \tag{15}$$

- If exercise is permitted at the node, then the model takes the greater of binomial and exercise value at the node.

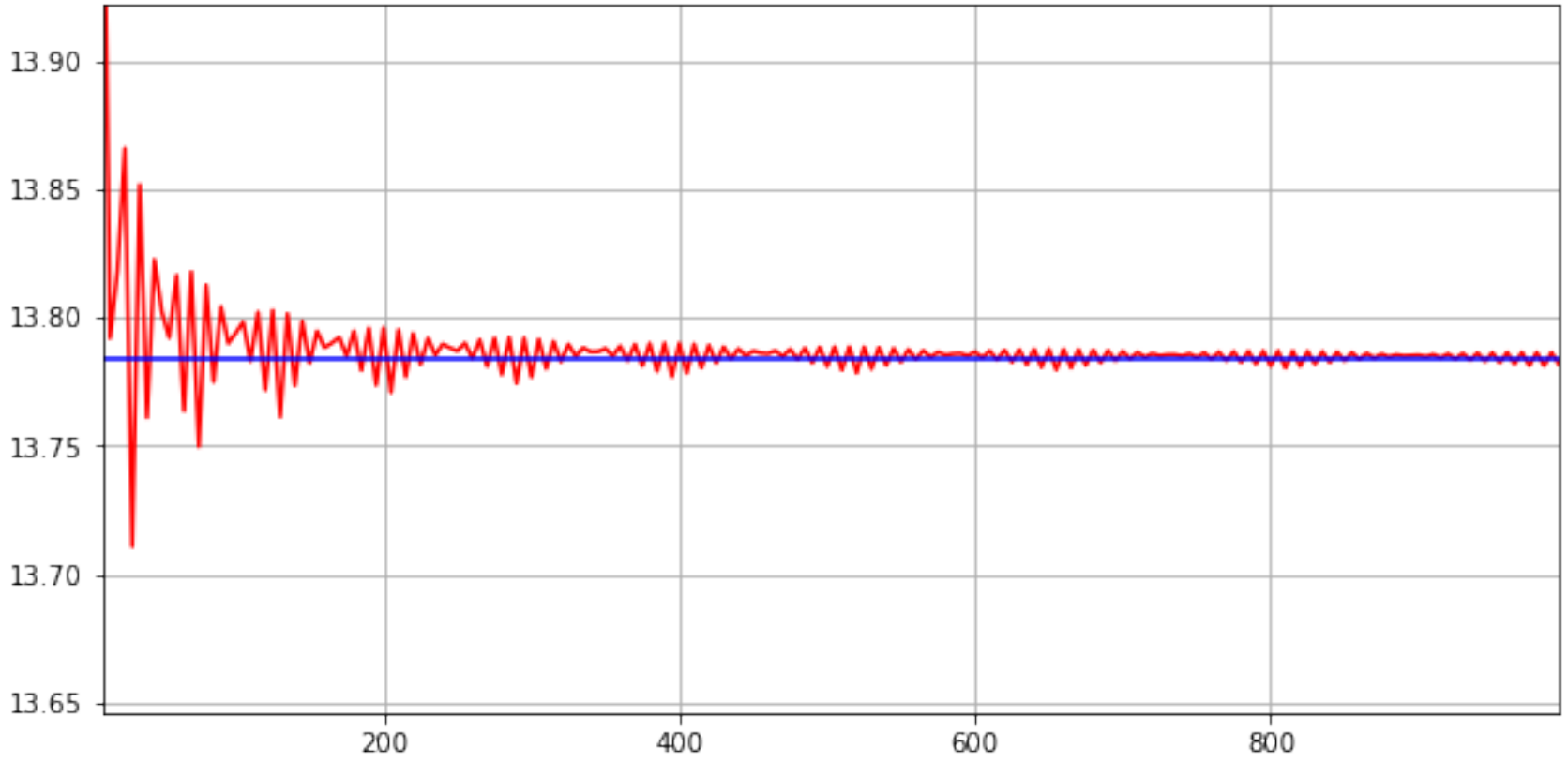
$$c_{t-\Delta t,i} = \max\left[e^{-r\Delta t}\left(p \cdot c_{t,i} + (1-p) \cdot c_{t,i+1}\right), S_t - K\right] \tag{16}$$

for a call option, and

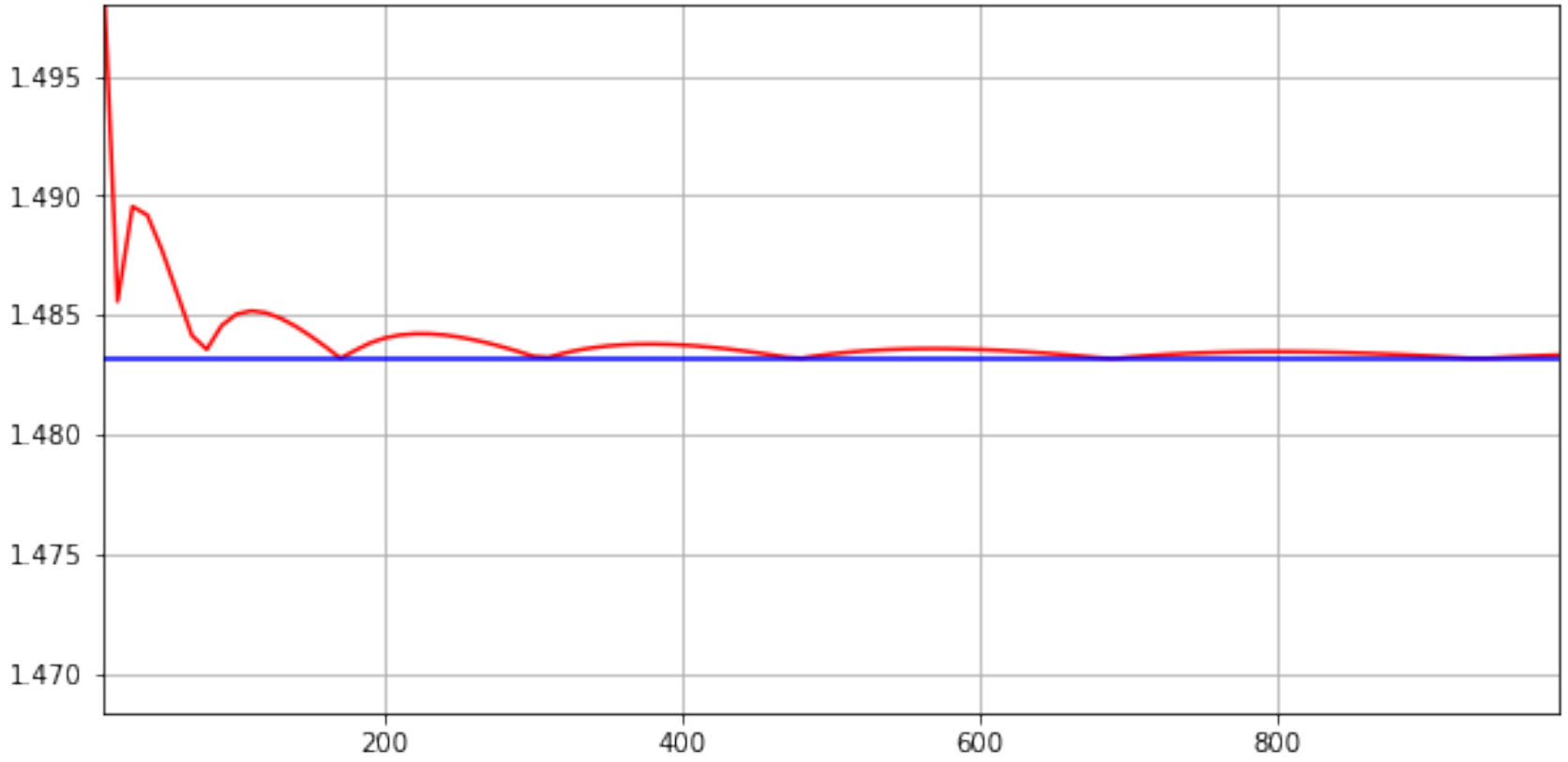
$$c_{t-\Delta t,i} = \max\left[e^{-r\Delta t}\left(p \cdot c_{t,i} + (1-p) \cdot c_{t,i+1}\right), K - S_t\right] \tag{17}$$

for a put option.

Convergency Rate



Convergency Rate



CRR Closed Formula for European Option

- Actualy in the case of non early exercise (european exercise) we don't need to build the tree in order to price the option;
- It easy to find that European call options are priced at the final time step simply as

$$\text{Call} = \exp\left(-rT\right) \sum_{i=0}^{n} \binom{n}{i} p^i \left(1-p\right)^{n-i} \max\left(Su^i d^{n-i} - K, 0\right)$$

- This formula represents the expected value of the option at the final time step, discounted by the risk-free rate.
- Since it is in closed form, it is very easy to implement.

Including Dividends

- Consider a stock paying a known dividend yield at rate q.
- The total return from dividends and capital gains in a risk-neutral world is r.
- The dividends provide a return of q.
- Capital gains must therefore provide a return of $r - q$.
- If the stock starts at S_0, its expected value after one time step of length Δt must be $S_0 e^{(r-q)\Delta t}$...

Including Dividends

- ... this means

$$p \cdot S_0 \cdot u + (1-p) \cdot S_0 \cdot d = S_0 e^{(r-q)\Delta t}$$

so that

$$p = \frac{e^{(r-q)\Delta t} - d}{u - d} \tag{18}$$

- As in the case of options on non-dividend-paying stocks, we match volatility by setting $u = e^{\sigma\sqrt{\Delta t}}$ and $d = 1/u$. This means that we can use the same equations as before except that we sobstitute r with $r - q$.

Trinomial Tree

- Under the trinomial method, the underlying stock price is modeled as a recombining tree, where, at each node the price has three possible paths: an up, down and stable or middle path.
- These values are found by multiplying the value at the current node by the appropriate factor u, d or m where

$$u = e^{\sigma\sqrt{2\Delta t}},\ d = e^{-\sigma\sqrt{2\Delta t}},\ m = 1$$

and the corresponding probability are:

$$p_u = \left(\frac{e^{(r-q)\Delta t/2} - e^{-\sigma\sqrt{\Delta t/2}}}{e^{\sigma\sqrt{\Delta t/2}} - e^{-\sigma\sqrt{\Delta t/2}}}\right)^2,\ p_d = \left(\frac{e^{\sigma\sqrt{\Delta t/2}} - e^{(r-q)\Delta t/2}}{e^{\sigma\sqrt{\Delta t/2}} - e^{-\sigma\sqrt{\Delta t/2}}}\right)^2$$

and $p_m = 1 - (p_u + p_d)$

Trinomial Tree

- As with the binomial model, these factors and probabilities are specified so as to ensure that the price of the underlying evolves as a martingale, while the moments are matched to those of the log-normal distribution;
- Note that for p_u , p_d and p_m to be in the interval $(0, 1)$ the following condition on Δt has to be satisfied:

$$\Delta t < 2\frac{\sigma^2}{(r-q)^2}.$$

Convergency Rate