Binomial Model

Fabio Cannizzo

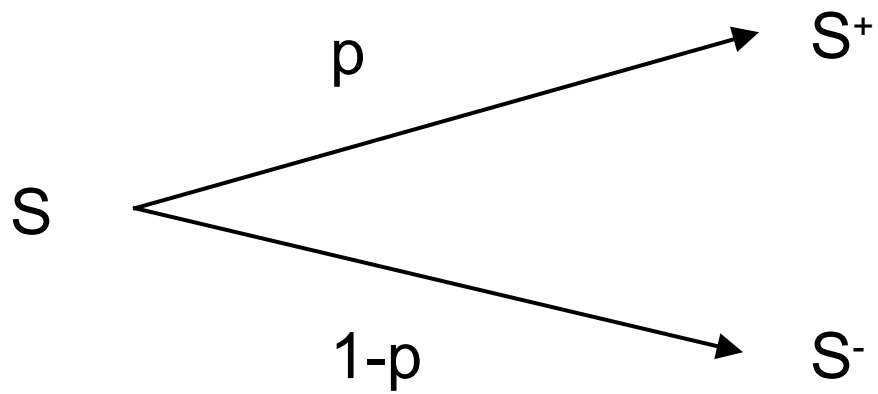
Model Description

- Let's consider our first stochastic model of asset prices, the binomial model.
- It assumes that time is divided up into discrete periods. At every step the price of the stock can go up or down to some specified prices, but cannot stay the same. It cannot do anything else.

Model Description

- What determines if the stock goes up or down?
- It is random. At every time step, the stock price goes up with probability *p* or down with probability *1-p*.
- The up/down movement at each step is independent on other time steps

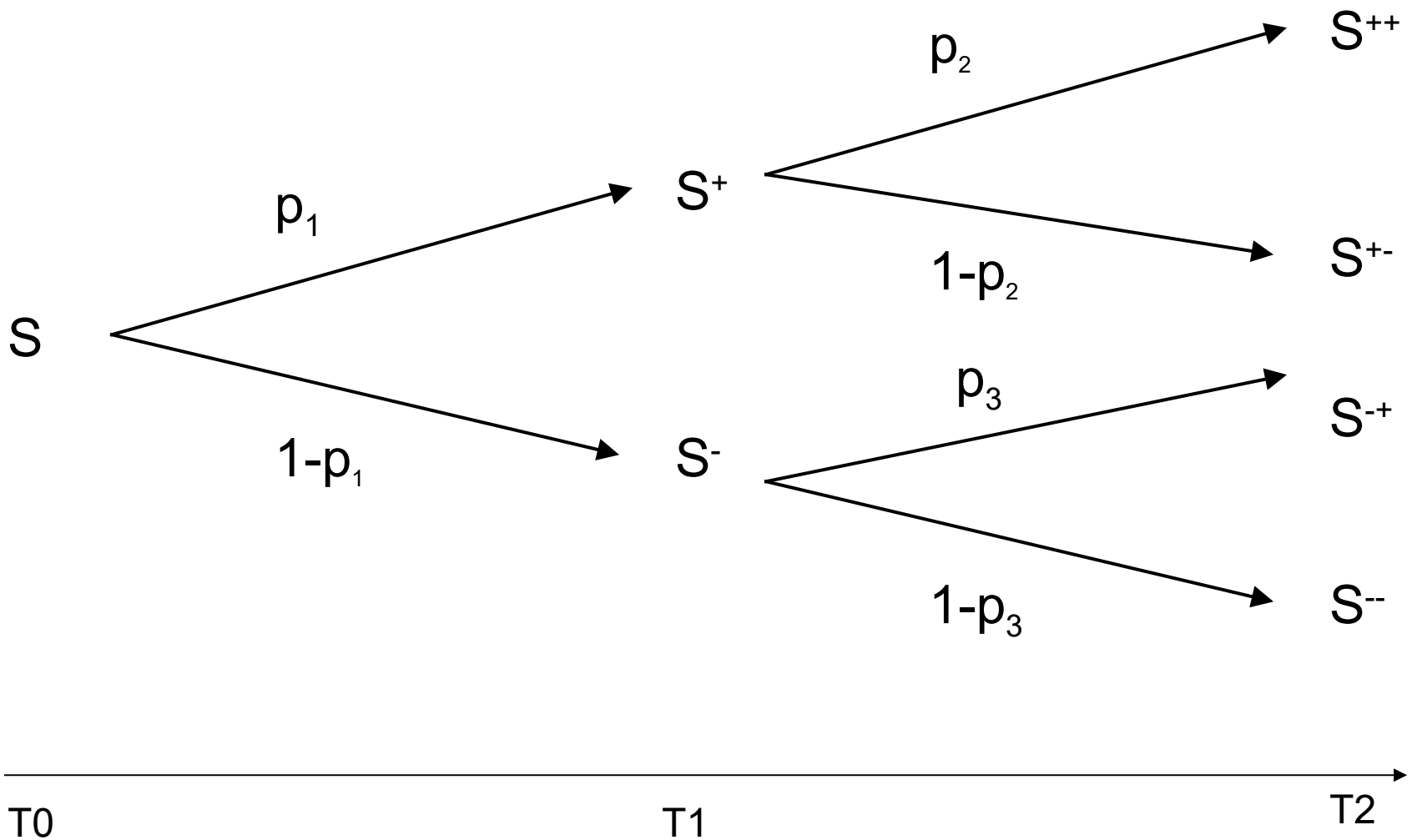
Model Description



Model Description

- Putting together multiple time steps we obtain a tree.
- Time steps could have different length, probabilities could be different at every point of the grid and up and down ticks could be also different at every point of the grid

Model Description



Model Rationale

- The model itself may seem too simplistic. In reality it is extremely flexible and powerful and can capture many features of real world prices
- If you think about it, this is not that absurd. The time step could be as small as half second and the up or down increment could be 1 tick.

Model Tractability

- The number of nodes grows with power of two. In the N^{th} step we would have 2^N nodes to deal with, which becomes very soon un-tractable, both with respect to computing cost and memory requirements.
- We need to design the up and down movement so that the tree is **recombining**, i.e. that 1 up + 1 down = 1 down + 1 up

Constant Returns

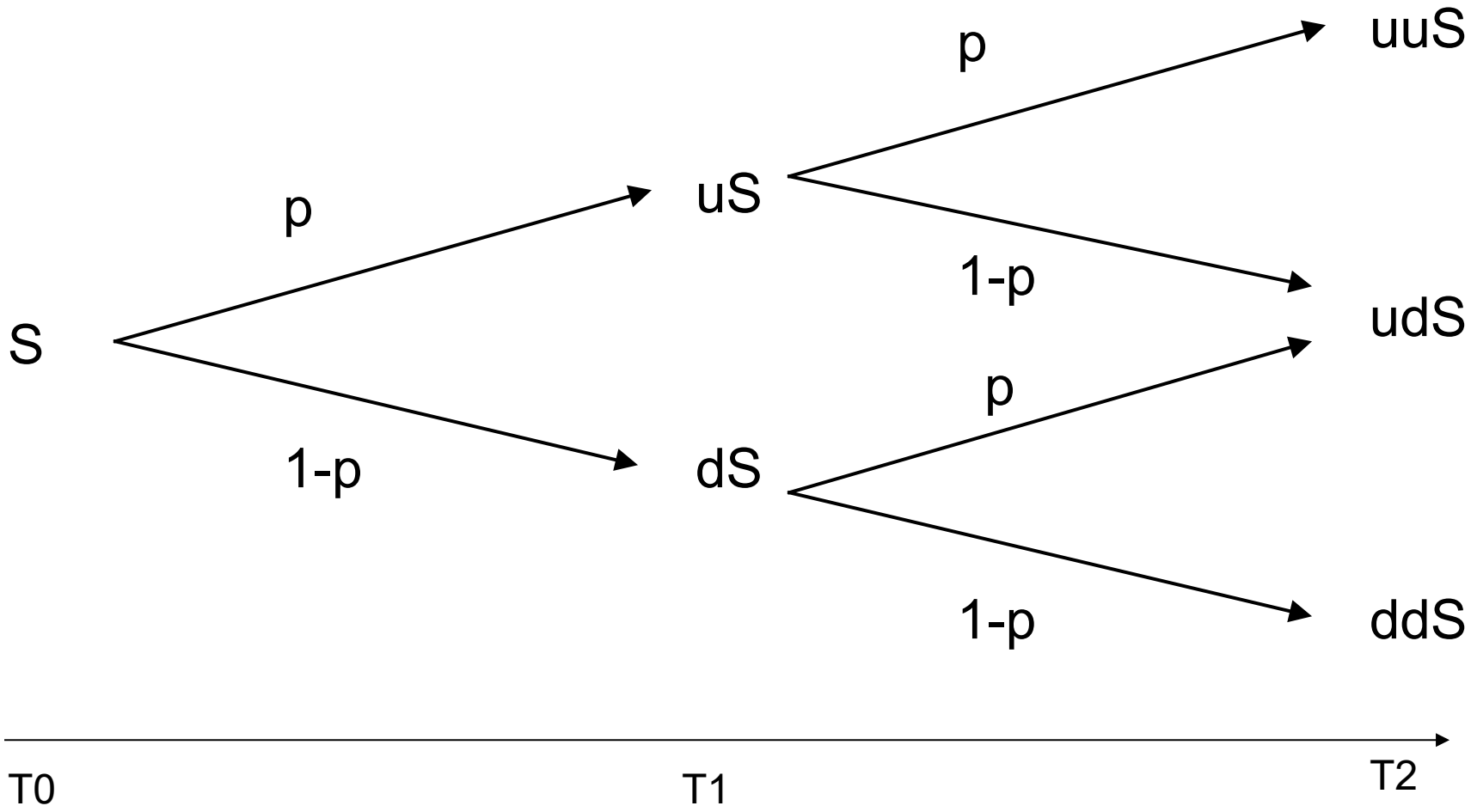
- There are different ways to achieve that. For instance, we could impose that returns are constant in time

$$\begin{cases} S^{+} = uS \\ S^{-} = dS \end{cases} \Rightarrow \begin{cases} S^{+-} = udS \\ S^{-+} = duS \end{cases} \Rightarrow S^{+-} = S^{-+}$$

More Assumptions

- The number of nodes at time steps N is now N+1, i.e. the size of the tree grows linearly, and is tractable
- To simplify our discussions, we also assume that time steps have identical size, and that probabilities are identical across the entire tree

Binomial Model in Price

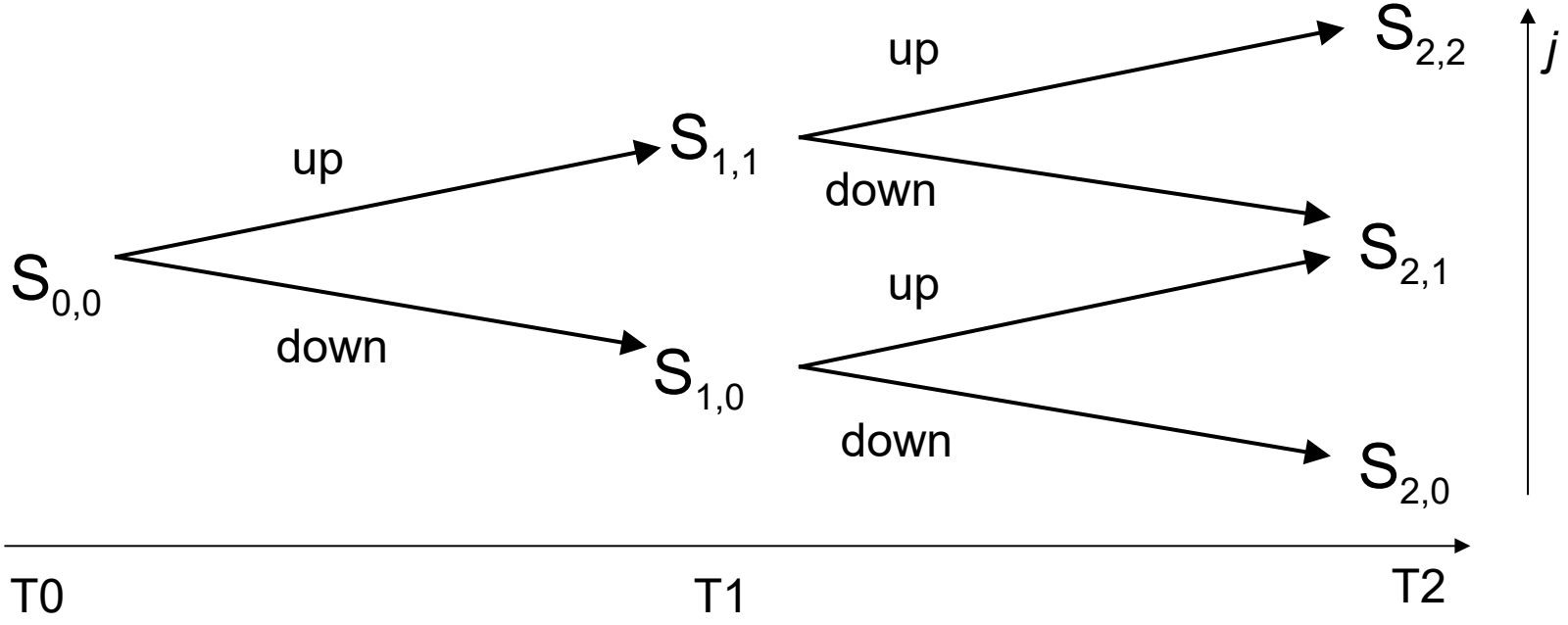


Assumptions

- What assumption have we made so far?
  - the price behaves randomly (it is almost certain that market have a very low degree of predictability, so no big deal)
  - That there is a true model out there, i.e. that this particular random mechanism really is how prices move
  - That we know its parameters, and that these are constant in time
- That's lot of assumptions! Later, we will relax some.

Notations

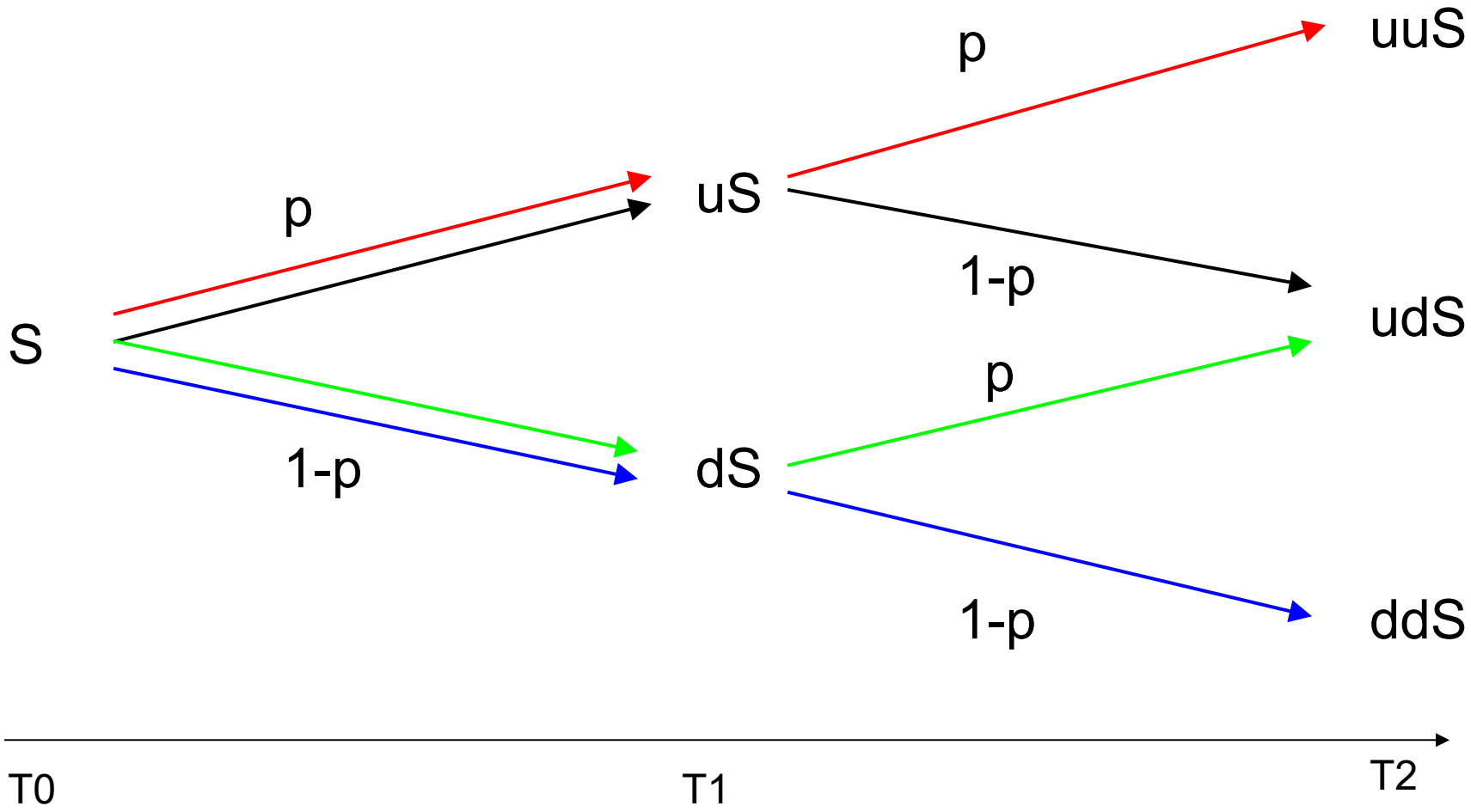
- Let's identify each state in a price tree with the two indices *t* and *j*, where *t* is time step index and *j* is the price level index defined as the number of "up" movement necessary to reach that state



State Probabilities

- Suppose N=2, we have 3 final states.
- What is the probability to be in each of the 3 states?
- There 4 possible paths: uu, ud, du, dd
- The two intermediate paths lead to the same final state, because the tree is recombining.
- Because movements at every time step are independent, the probability of a path is the product of the probabilities of each time step

State Probabilities



State Probabilities

- The probability associated with each path is:
  - $P(uu)=p^2$
  - $P(ud)=P(du)=p(1-p)$
  - $P(dd)=(1-p)^2$
- The probability of the states is:
  - $P(S_{2,2})=P(uu)=p^2$
  - $P(S_{2,1})=P(du)+P(ud)=2p(1-p)$
  - $P(S_{2,0})=P(dd)=(1-p)^2$

State Probabilities

- More generally, given N steps, there are N+1 final prices (i.e. states of the world).
- The probability of any possible path reaching the generic state $u^j d^{N-j} S$ is $p^j (1-p)^{N-j}$.
- To know the total probability to reach that node, we need to compute how many paths can reach that node. I.e. in how many ways can I pick an *up* movement *j* times out of *N* movements? This is the binomial coefficient (*N,j*)

$$P\left(S_{N,j} = u^j d^{N-j} S_{0,0}\right) = \binom{N}{j} p^j (1-p)^{N-j} = \frac{N!}{j!(N-j)!} p^j (1-p)^{N-j}$$

Single Step Price Moments

- At every time step we can compute **expectation**, **second moment** and **variance** for price returns

$$E\left[\frac{S_{t+1}}{S_t}|S_t\right]=pu+(1-p)d$$

$$E\left[\left(\frac{S_{t+1}}{S_t}\right)^2|S_t\right]=pu^2+(1-p)d^2$$

$$Var\left[\frac{S_{t+1}}{S_t}|S_t\right]=\left[pu^2+(1-p)d^2\right]-\left[pu+(1-p)d\right]^2$$

$$=p(1-p)(u-d)^2$$

Multi Step Price Moments

- The **expectation** of the price return over multiple steps can be obtained via induction

$$E\left[\frac{S_{t+2}}{S_t}|S_t\right]=E\left[\frac{S_{t+2}}{S_{t+1}}\frac{S_{t+1}}{S_t}|S_t\right]=E\left[\frac{S_{t+2}}{S_{t+1}}|S_t\right]E\left[\frac{S_{t+1}}{S_t}|S_t\right]=$$

$$E\left[E\left[\frac{S_{t+2}}{S_{t+1}}|S_{t+1}\right]|S_t\right]\left[pu+(1-p)d\right]=\left[pu+(1-p)d\right]^2 \qquad \text{2 steps}$$

$$E\left[\frac{S_{t+3}}{S_t}|S_t\right]=E\left[\frac{S_{t+3}}{S_{t+2}}\frac{S_{t+2}}{S_t}|S_t\right]=E\left[\frac{S_{t+3}}{S_{t+2}}|S_t\right]E\left[\frac{S_{t+2}}{S_t}|S_t\right]=$$

$$¿\,E\left[E\left[\frac{S_{t+3}}{S_{t+2}}|S_{t+2}\right]|S_t\right]\left[pu+(1-p)d\right]^2=\left[pu+(1-p)d\right]^3 \qquad \text{3 steps}$$

$$E\left[\frac{S_{t+N}}{S_t}|S_t\right]=\left[pu+(1-p)d\right]^N \qquad \text{N steps}$$

Multi Step Price Moments

- With analogous reasoning, the **second moment**, and the **variance** of the price over multiple steps are

$$E\left[\left(\frac{S_{t+N}}{S_t}\right)^2 | S_t\right] = \left[pu^2 + (1-p)d^2\right]^N$$

$$Var\left[\frac{S_{t+N}}{S_t} | S_t\right] = \left[pu^2 + (1-p)d^2\right]^N - \left[pu + (1-p)d\right]^{2N}$$

Multi Step Price Moments

- An alternative way… since we know the probabilities of each node, we could also compute the expectation simply using the definition:

$$E\left[\frac{S_{t+N}}{S_t}\middle|S_t\right]=\sum_{j=0}^{N}\frac{S_{N,j}}{S_t}P\left(S_{N,j}\right)=$$

$$¿\sum_{j=0}^{N}\binom{N}{j}p^j\left(1-p\right)^{N-j}u^j d^{N-j}$$

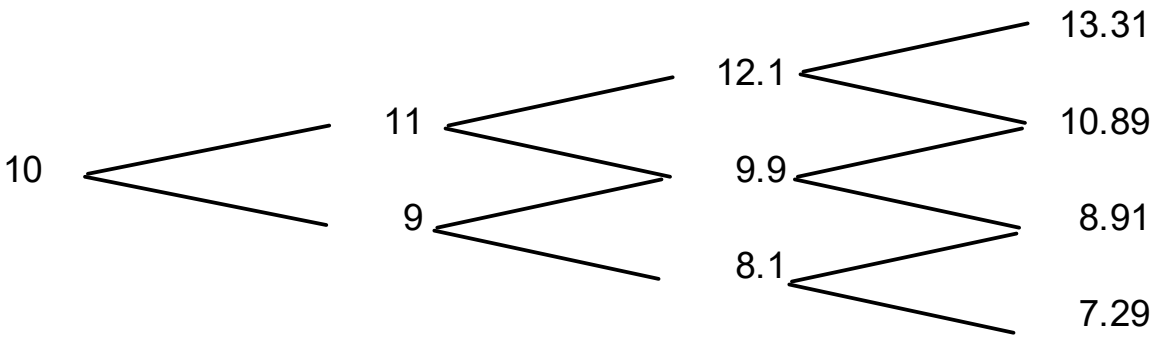
Price Binomial Model Example

- Let's consider a share with initial price of 10. Every month the stock can have a positive return of 10% with probability 60% or a negative return of 10% with probability 40%. If we consider an horizon of 3 months, we can construct the corresponding tree and we can also compute expectation and variance of the price.

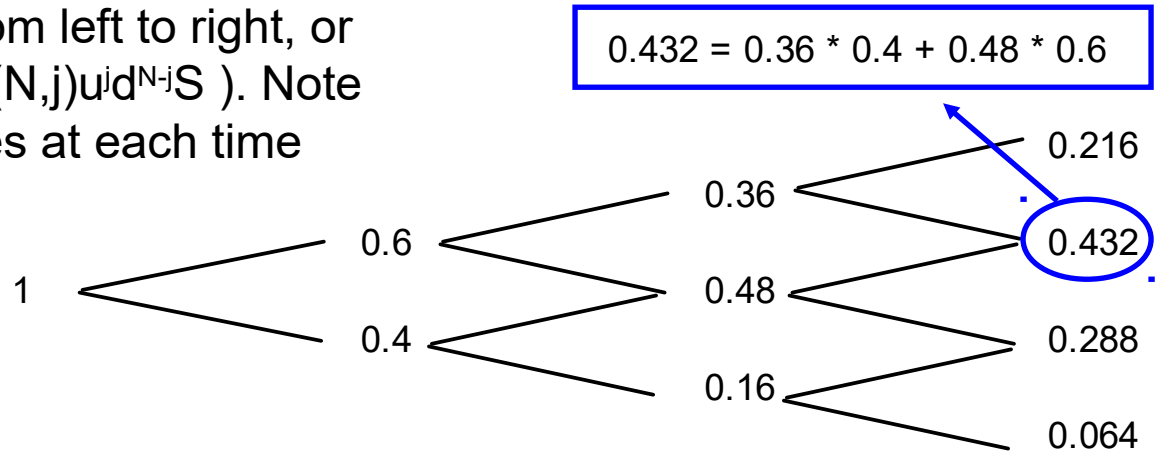
Price Binomial Model Example

Prices (can compute them propagating on the tree from left to right, or directly using the formula $u^j d^{N-j} S_0$)

u	1.1
d	0.9
p	0.6
1-p	0.4



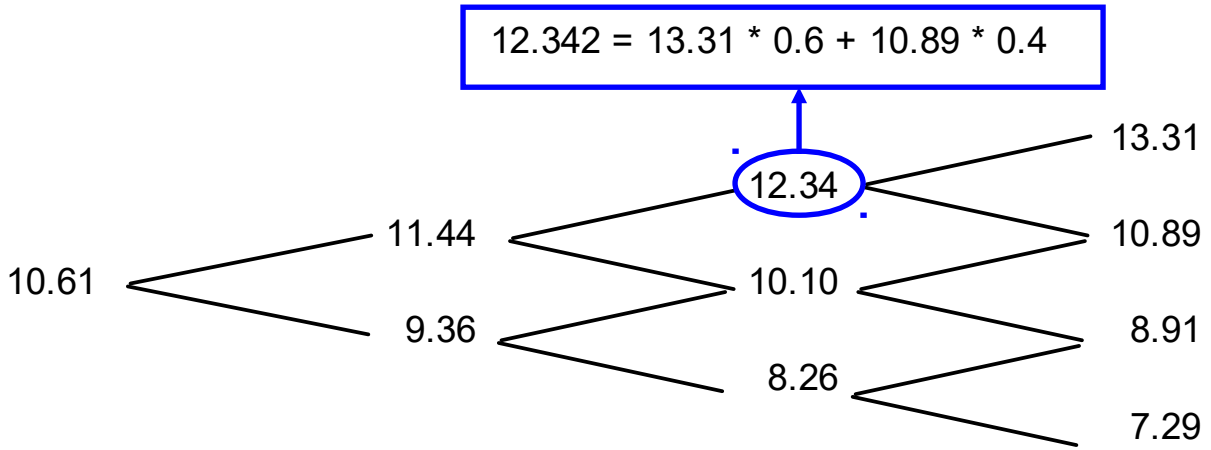
Node Probabilities (can compute them propagating on the tree from left to right, or directly using the formula $(N,j)u^j d^{N-j} S$). Note that the sum of probabilities at each time step is 1



Price Binomial Model Example

Three equivalent ways to compute the **price expectation**:

1) via backward recursive expectations on the tree from right to left



2) using the formula

$$E\left[S_3|S_0\right]=S_0\,E\left[\frac{S_3}{S_0}|S_0\right]=S_0\left[pu+(1-p)d\right]^3=10\left[0.6\cdot 1.1+0.4\cdot 0.9\right]^3=10.61$$

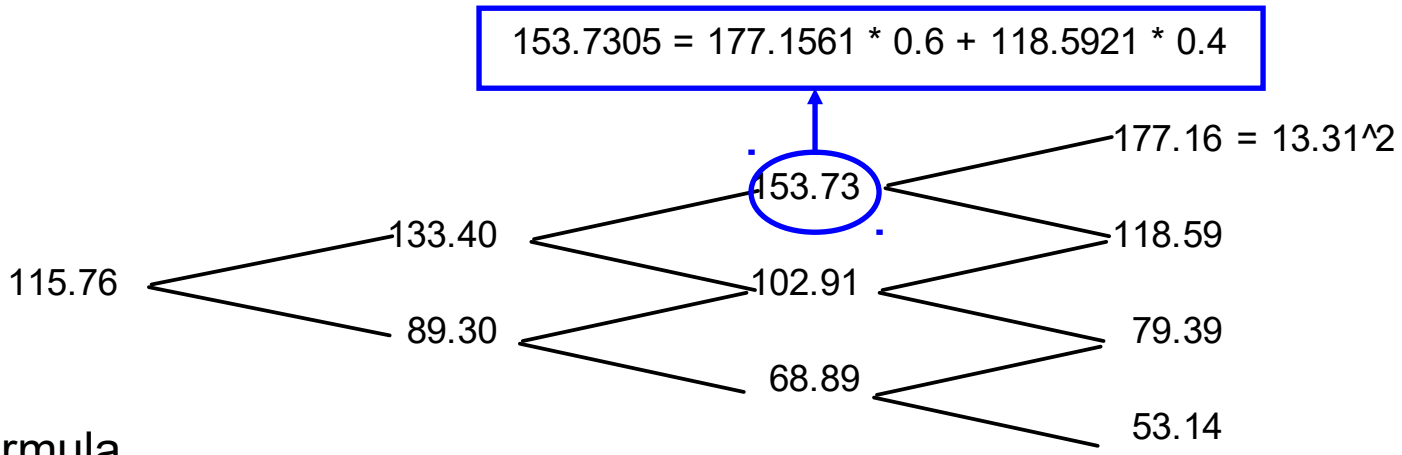
3) using state probabilities

$$E\left[S_3|S_0\right]=\sum_{j=0}^{3} S_{3,j}\,P\left(S_{3,j}\right)=7.29\cdot 0.063+8.91\cdot 0.288+10.89\cdot 0.432+13.31\cdot 0.216=10.61$$

Price Binomial Model Example

Three equivalent ways to compute the **price second moment**:

1) via backward recursive expectations on the tree



2) using the formula

$$E\left[S_3^2|S_0\right]=S_0^2\left[pu^2+(1-p)d^2\right]^3=100\left[0.6\cdot 1.1^2+0.4\cdot 0.9^2\right]^3=115.76$$

3) using state probabilities

$$E\left[S_3^2|S_0\right]=\sum_{j=0}^{3} S_{3,j}^2 P\left(S_{3,j}\right)=115.76$$

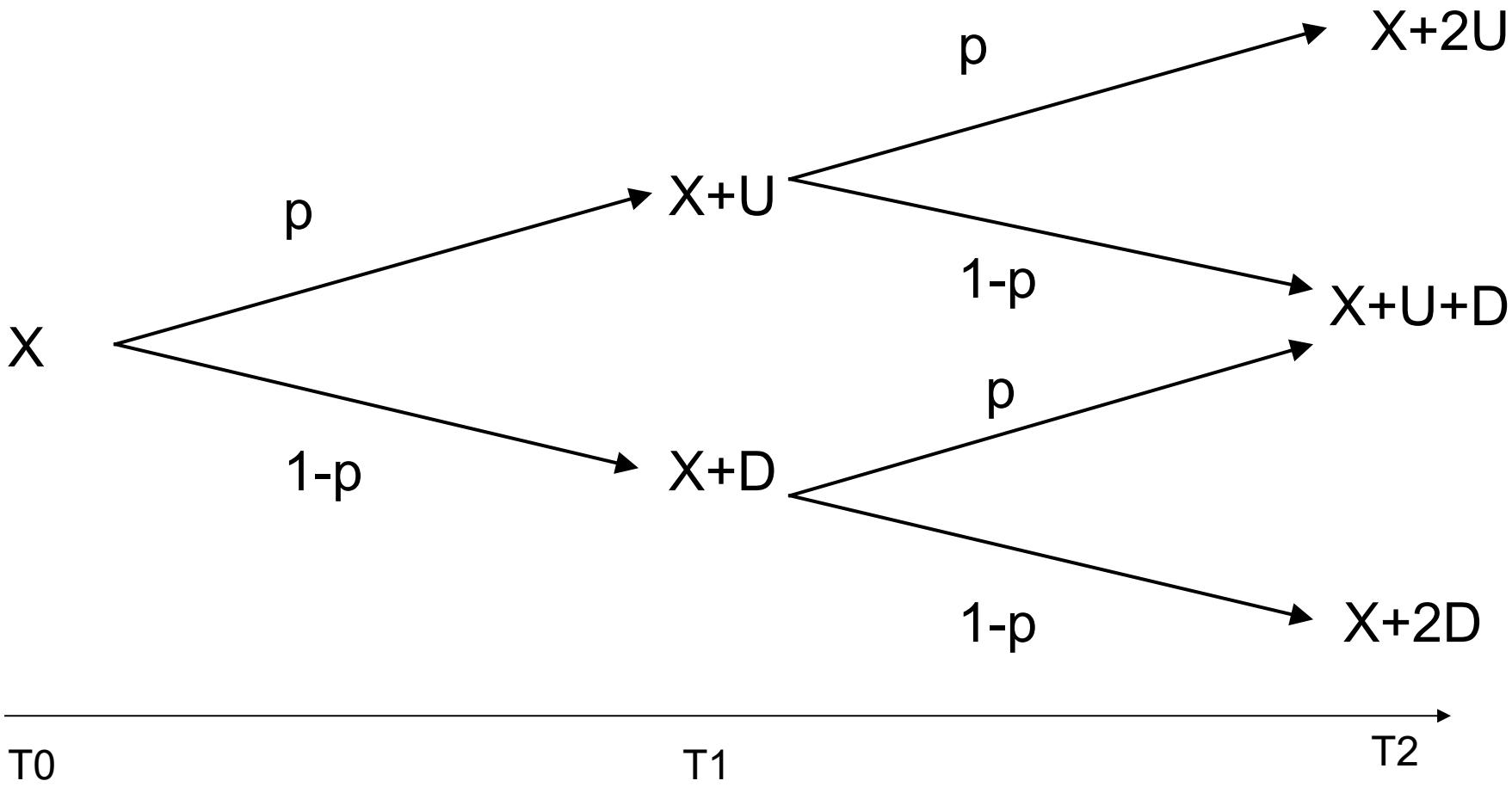
Choice of State Variables

- We can construct the tree either in the variable S or in the variable $X=\ln S$
- The variables which is immediately available from the tree is the chosen "**state variable**"
- For a choice of state variable to be good, it must be possible to reconstruct the derived variables necessary for pricing in a unique way
- E.g. if the state variable is $X=\ln S$, then S can be uniquely **reconstructed** at every node as $S=\exp X$

Increments or Returns?

- If our tree is constructed on the state variable **X=ln S**, if the stock price falls below 1 then X becomes negative.
- Our framework does not allow to capture that: $d^n X_0$ will never be negative
- Instead of using returns, we can use increments
- As before we need to make sure the tree is recombining

Binomial Model in Log-Price



Constant Increments Recombining

- Setting constant increments in time, makes the tree recombining

$$\begin{cases} X^{+} = X + U \\ X^{-} = X - D \end{cases} \Rightarrow \begin{cases} X^{+-} = X + U - D \\ X^{-+} = X - D + U \end{cases} \Rightarrow X^{+-} = X^{-+}$$

Single Step Log-Return Moments

- Properties of the process are simpler if computed for **log-returns**

$$E\left[X_{t+1}-X_t \mid X_t\right]=pU+\left(1-p\right)D$$

$$E\left[\left(X_{t+1}-X_t\right)^2 \mid X_t\right]=pU^2+\left(1-p\right)D^2$$

$$\begin{aligned} Var\left[X_{t+1}-X_t \mid X_t\right] &= pU^2+\left(1-p\right)D^2-\left[pU+\left(1-p\right)D\right]^2 \\ &= p\left(1-p\right)\left(U-D\right)^2 \end{aligned}$$

Multi Step Log-Return Moments

- The **expectation** of log-returns over multiple steps is

$$
\begin{aligned}
E\left[X_{t+N}-X_t|X_t\right] &= E\left[\left(\sum_{i=t}^{t+N-1} X_{i+1}-X_i\right)|X_t\right] \\
&= \sum_{i=t}^{t+N-1} E\left[X_{i+1}-X_i|X_t\right] \\
&= \sum_{i=t}^{t+N-1} E\left[E\left[X_{i+1}-X_i|X_i\right]|X_t\right] \\
&= \sum_{i=t}^{t+N-1} E\left[pU+(1-p)D|X_t\right] \\
&= N\left[pU+(1-p)D\right]
\end{aligned}
$$

Multi Step Log-Returns Moments

- The **variance** of the log-return over multiple steps is

$$Var\left[X_{t+N} - X_t \middle| X_t\right] = Var\left[\sum_{i=t}^{t+N-1} X_{i+1} - X_i \middle| X_t\right]$$
$$= \sum_{i=t}^{t+N-1} Var\left[X_{i+1} - X_i \middle| X_t\right]$$
$$= Np(1-p)(U-D)^2$$

- I.e. mean and variance are just their one-step ones multiplied by the number of steps

Log-Return Binominal Model Example

- If we know U,D and p, we can compute expectation and variance
- Every trading day a share can have a positive log-return of 1% with probability 60% or a negative log-return of 1% with probability 40%.
  - Daily expected return:
    $E[X_{t+1}|X_t]=0.6*0.01+0.4*(-0.01)=0.002$
  - Annualized expected return (256 trading days)
    $E[X_{t+256}|X_t]=256*E[X_{t+1}|X_t]=0.512=51.2\%$
  - Daily variance
    $Var[X_{t+1}|X_t]=0.6*0.4*(0.01-(-0.01))^2=0.000096$
  - Annualized variance (256 trading days)
    $Var[X_{t+256}|X_t]=256*Var[X_{t+1}|X_t]=0.024576$
  - Annualized standard deviation (**volatility**)
    $StDev[X_{t+256}|X_t]=15.67\%$

Historical Estimation

- From historical time series, we can compute expectation and variance
- We have the time series of daily prices S_t, t=1,N
- We take the log-returns: $X_t = \ln S_t / S_{t-1}$
- Xt is a random variable
- Assuming that the return at time t_i is independent from the return at another time t_j, expected daily returns and daily variances can be trivially estimated from this time series using the formula for the sample mean and the sample variance

Parameters Calibration

- Given the estimated annualized expectation μ and variance σ estimated for the log returns of an asset, for a given number of time steps N, we want to compute the parameters *u*, *d* and *p* which match mean and variance of the process at some time horizon T
- We have two equations in 3 unknowns, therefore there is one degree of freedom we can play with. For example, we can choose d=1/u (or D=-U), as proposed in Cox, Ross, Rubenstein 1979

Parameters Calibration (log-returns)

$$D = -U$$

$$N\left[pU + (1-p)D\right] = E\left[X_{t+N} - X_t\right] = \mu T$$

$$Np(1-p)\left(U-D\right)^2 = Var\left[X_{t+N} - X_t\right] = \sigma^2 T$$

System of equations (note that we are matching the properties of the log-price process)

$$U = \sigma\sqrt{\Delta t}\sqrt{1 + \frac{\mu^2}{\sigma^2}\Delta t} \underset{\Delta t \to 0}{\approx} \sigma\sqrt{\Delta t}$$

$$p = \frac{1}{2} + \frac{\mu}{2\sigma}\sqrt{\Delta t}\frac{1}{\sqrt{1 + \frac{\mu^2}{\sigma^2}\Delta t}} \underset{\Delta t \to 0}{\approx} \frac{1}{2} + \frac{\mu}{2\sigma}\sqrt{\Delta t}$$

Solution

where $\Delta t = \frac{T}{N}$

Parameters Calibration Example

- We want to construct a tree for a stock with annualized expected log return 8% and volatility 40%, for an horizon of 2 years with 200 time steps

$\Delta t = 2/200 = 0.01$ $\quad U \approx 0.4 * 0.01^{0.5} = 0.04$

$\mu = 0.08$ $\quad \Rightarrow \quad D \approx -0.04$

$\sigma = 0.4$ $\quad p \approx 0.5(1 + 0.08/0.4 * 0.01^{0.5}) = 0.51$

Parameters Calibration (returns)

$$d = 1/u$$

$$\left[pu + (1-p)d\right]^N = E\left[\frac{S_{t+N}}{S_t}\right] = M_1$$

$$\left[pu^2 + (1-p)d^2\right]^N = E\left[\left(\frac{S_{t+N}}{S_t}\right)^2\right] = M_2$$

System of equations (note that in this case we are matching the first two moments of the price process at time T, indicated with M1 and M2).

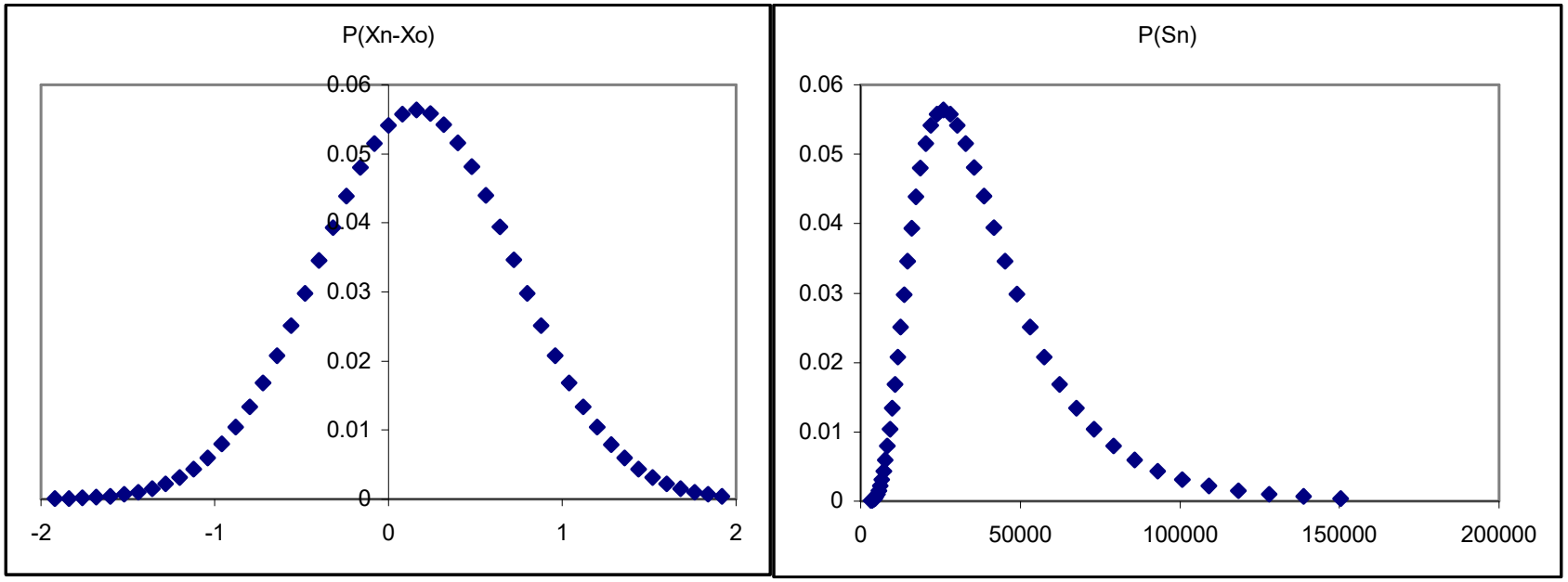
$$pu + (1-p)d = \sqrt[N]{M_1} = \chi_1 \qquad p = \frac{\chi_1 - d}{u - d}$$

$$pu^2 + (1-p)d^2 = \sqrt[N]{M_2} = \chi_2 \qquad d = \frac{\chi_2 + 1 - \sqrt{(\chi_2 + 1)^2 - 4\chi_1^2}}{2\chi_1}$$

Distribution Example

- Let's have a look at the distribution we achieve after 2 years

So	10	dt	0.01	p	0.509998
N	200	sqrt dt	0.1	U	0.040008
T	2			D	-0.04001
sigma	40%	sqrt(1+…)	1.0002		
mu	8%				



Log-Return Distribution

- The distribution looks normal, is it a coincidence?
- If we keep fixed T and increase N, the binomial distribution for time T tend to the Normal distribution $N(\mu T, \sigma^2 T)$ as a consequence of the **central limit theorem** (de Moivre–Laplace theorem)
- This hold for any time step, because we can have an infinite number of intermediate steps in any time interval

Limit for $\Delta t \to 0$

- When $\Delta t \to 0$ a binomial tree of log-returns approximates a continuous process with Gaussian increments, constant volatility and constant drift. This is an **Arithmetic Brownian Motion**.

$$dX_t = \mu\, dt + \sigma\, dW_t$$

- Therefore we conclude that the tree of prices will approximate a **Geometric Brownian Motion**

$$\frac{dS_t}{S_t} = \left(\mu + \frac{\sigma^2}{2}\right) dt + \sigma\, dW_t$$

Intuition

- A binomial pricing model is just an approximation to an Arithmetic Brownian Motion (incremental type) or to a Geometric Brownian Motion (multiplicative type)
- It has the same ability to describe dynamics of asset prices, therefore the same strengths and weaknesses

Geometric Brownian Motion

- Features
  - Constant drift (can account for a bias)
  - Constant volatility (usually so, but it can change)
  - No jumps (sometime jumps may happen)
  - Gaussian log-returns (not too bad for medium or long horizon)
  - Independent returns (does history affect future?)

Calibration to GBM or ABM

- Usually we start from some continuous process, and we want to construct a tree which approximate it

$$\frac{dS_t}{S_t} = m\,dt + \sigma\,dW_t, \qquad E\left[\frac{S_T}{S_0}\right] = e^{mT}, \qquad \mathrm{Var}\left[\frac{S_T}{S_0}\right] = e^{2mT}\left(e^{\sigma^2 T} - 1\right)$$

$$X_t = \ln S_t$$

$$dX_t = \left(m - \frac{\sigma^2}{2}\right)dt + \sigma\,dW_t, \quad E[X_T - X_0] = \left(m - \frac{\sigma^2}{2}\right)T, \quad \mathrm{Var}[X_T - X_0] = \sigma^2 T$$

- Depending on which state variable we are using, we can match either
  - expectation and variance of the GBM driving the price
  - expectation and variance or the ABM driving the log of the price
- Note that while in continuous time these two things are equivalent, in discrete time they are not, especially for small N
- And remember we still have one degree of freedom!

Parameters Calibration (returns)

- Academics have been looking for simplifications of these formulas for GBM

let $\chi_1 = e^{m\Delta_t}, \quad \chi_2 = e^{\left(2m+\sigma^2\right)\Delta_t}$ (the first and second 1-period-moments of a GBM)

$$d = \frac{e^{\left(2m+\sigma^2\right)\Delta t} + 1 - \sqrt{\left(e^{\left(2m+\sigma^2\right)\Delta t}+1\right)^2 - 4e^{2m\Delta t}}}{2e^{m\Delta t}}$$

Replacing $\Delta t = z^2$ and taking Mc Laurin expansion of $d(z)$, then replacing back Δt :

$$\approx 1 - \sigma\sqrt{\Delta t} + \frac{\sigma^2}{2}\Delta t + O\left(\Delta t^{3/2}\right)$$

- Since the first three terms are identical to the Taylor expansion of

$$d \approx e^{-\sigma\sqrt{\Delta t}}$$

with this approximation the error is of order $O(\Delta t^{3/2})$

- I only present this result because it is very popular in the literature, but in practice, I do not think it adds much value

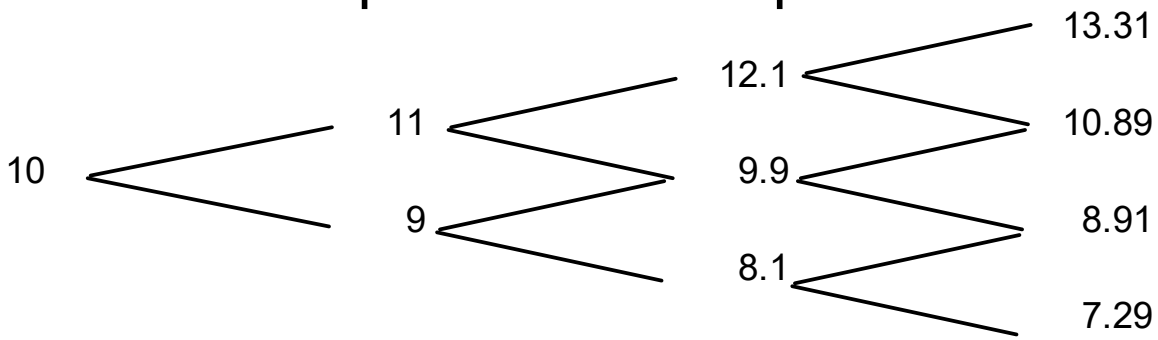
Expectation of a Payoff Function

- Let f(S_T) be some payoff function of the price at time T, then to compute its expectation first we compute the value f(S_T) in each state at time T, then we can compute the expectation in two ways:
  1. working back on the tree
  2. using the state probabilities

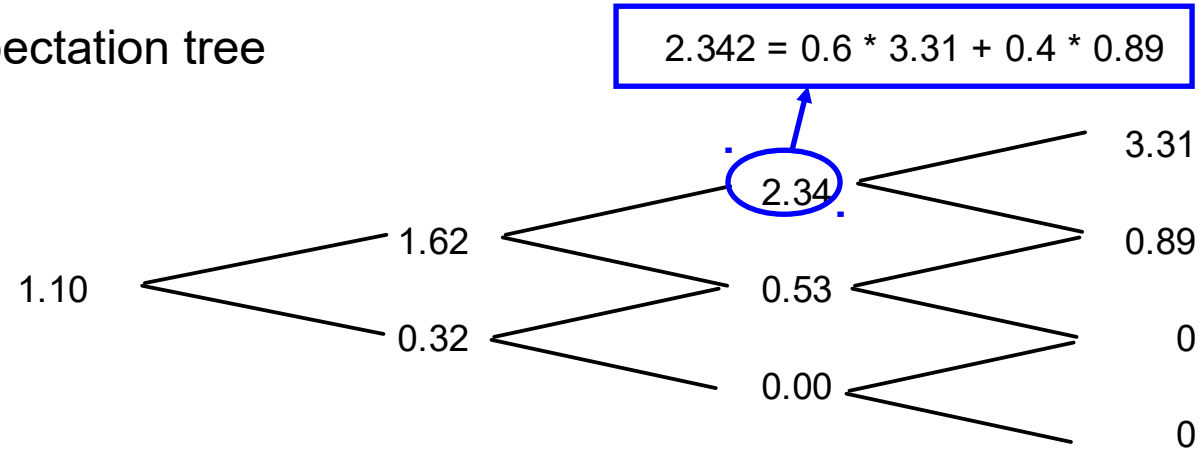
Expectation of a Function Example

- Let's consider the function max[S_3-10,0] applied to the Tree constructed in the previous example

Price tree



Function Expectation tree



Alternatively, using probabilities: 3.31*0.216+0.89*0.432 = 1.10

Summary

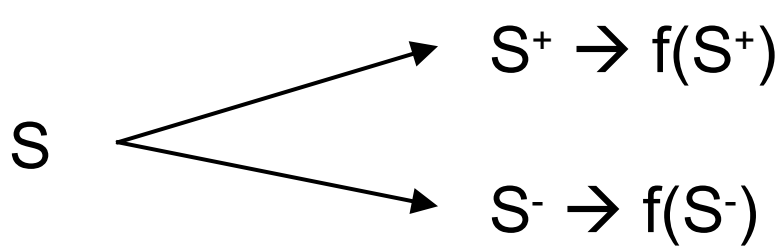
- So far we learned how to:
  - construct a binomial tree, either in price space or in log-return space
  - compute the expectation of a function on the tree
  - make the tree consistent with some given properties of the stochastic process of the underlying asset
- We understood the binomial probability function and the behavior when $\Delta T \to 0$

No Arbitrage Pricing

- Let $f(S_T)$ be the payoff of a derivative contract. We already know how to compute its expectation on a tree, but this is not sufficient for pricing. It still requires information about the risk appetite of buyer and seller, and that makes the price subjective
- If we can replicate the possible P&L generated by the payoff $f(S_T)$ with a portfolio of other securities of known price, then, by the **law of one price**, the price of the derivative contract must be identical to the price of its replicating portfolio

No Arbitrage Pricing, Single Period

- We want to price a derivative contract paying $f(S_T)$ at time T
- Let's consider a simple economy where a **non dividend paying** stock price at time T can assume only two values: S^+ and S^-



The value of the derivative contract in the end is just the cash flow implied by its payoff

No Arbitrage Pricing, Single Period

- Let's assume we can trade the stock and a invest or borrow money at fixed interest rate *r* continuously compounded
- We want to construct a portfolio investing β in the risk free rate and buying δ shares, which replicates the payoff of the derivative contract in all the possible final states of the world

$$\Pi = \delta S + \beta$$

- To do that we write a linear system

$$\begin{cases} V^+ = f\left(S^+\right) = \delta\, S^+ + \beta\, e^{r\,\Delta t} \\ V^- = f\left(S^-\right) = \delta\, S^- + \beta\, e^{r\,\Delta t} \end{cases} \quad \rightarrow \quad \begin{cases} \delta = \dfrac{V^+ - V^-}{S^+ - S^-} \\ \beta = \left(V^+ - \delta\, S^+\right) e^{-r\,\Delta t} \end{cases}$$

No Arbitrage Pricing, Single Period

- Note that δ can be interpreted as the sensitivity of the price of the derivative contract to changes in the price of the stock, i.e. the **hedging ratio**
- Note that we have not used the probability of an up or down move anywhere in the construction of the replication portfolio. We will come back to this.

Law of One Price

- Securities having same payoff must have the same price.
- If there is a price mismatch, in absence of trading restrictions and transaction costs, market participants will sell the most expensive asset and buy the cheapest one therefore influencing market prices and causing the arbitrage to close out.

No Arbitrage Pricing

- If the derivative contract cost **more** than its replicating portfolio, we can sell the derivative contract and buy the replicating portfolio. Hence we make a profit. At maturity the cash flows from the two positions will cancel out
- If the derivative contract cost **less** than its replicating portfolio, we can buy the derivative contract and sell the replicating portfolio. Hence we make a profit. At maturity the cash flows from the two positions will cancel out.
- This is what “arbitrage” means.

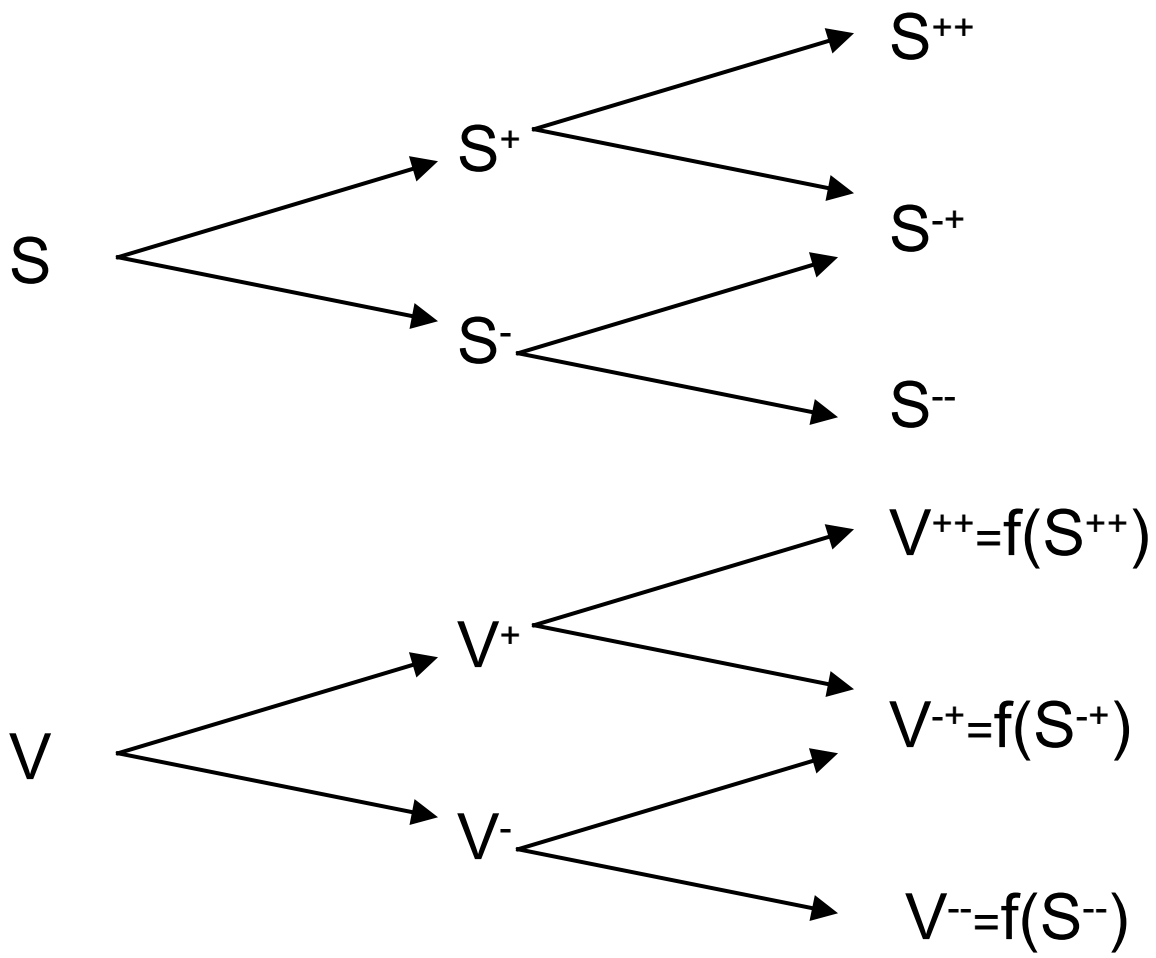
No Arbitrage Pricing

- In the results we obtained, the price of the derivative contract depends on S^+ and S^- and Z, but not on the probability p. How is that possible?
- Mathematically, state probabilities are irrelevant from a replication point of view
- Intuitively, if the probability of an up move increase, this should be reflected in the price of the stock, which should also increase, unless the risk of the stock increases too, causing the risk premium required by investors to hold it to be higher, which compensates for the increase in probability. In the end, if the price of the stock does not move, why should the derivative contract move? No reason!

No Arbitrage Pricing, Multi Period

- We know how to compute the price of the derivative contract for one single step. This could well be the last step of a multi-period tree. This means that if we repeat the argument for all possible last single steps of a multi-period tree, then we can iterate the reasoning backward in the tree, until we obtain the current price of the derivative contract

No Arbitrage Pricing, Multi Period



In the final state, the value of the derivative contract is given by the payoff itself

No Arbitrage Pricing, Multi Period

At step1, node “up”, we compute V^+

$$\delta^+ = \frac{V^{++} - V^{-+}}{S^{++} - S^{-+}}$$

$$\beta^+ = \left(V^{++} - \delta^+ S^{++}\right) e^{-r\Delta t}$$

$$V^+ = \delta^+ S^+ + \beta^+$$

$$\delta^- = \frac{V^{-+} - V^{--}}{S^{-+} - S^{--}}$$

$$\beta^- = \left(V^{--} - \delta^- S^{--}\right) e^{-r\Delta t}$$

$$V^- = \delta^- S^- + \beta^-$$

At step1, node “down”, we compute V^-

At step 0, main node, we compute V

$$\delta = \frac{V^+ - V^-}{S^+ - S^-}$$

$$\beta = \left(V^+ - \delta S^+\right) e^{-r\Delta t}$$

$$V = \delta S + \beta$$

No Arbitrage Pricing, Multi Period

- Working backward along the tree, we obtain the price of the derivative contract and the corresponding hedging strategy (replicating portfolio) in every possible state of the world at every time step.
- Note the hedging strategy is **dynamic**, i.e. at every time step we need to readjust the hedge position
- So not only we know the price, but we also know how to take advantage of the arbitrage if the price deviates from what it should be

No Arbitrage Pricing Example

Payoff = max[S-10,0]

r = 10%

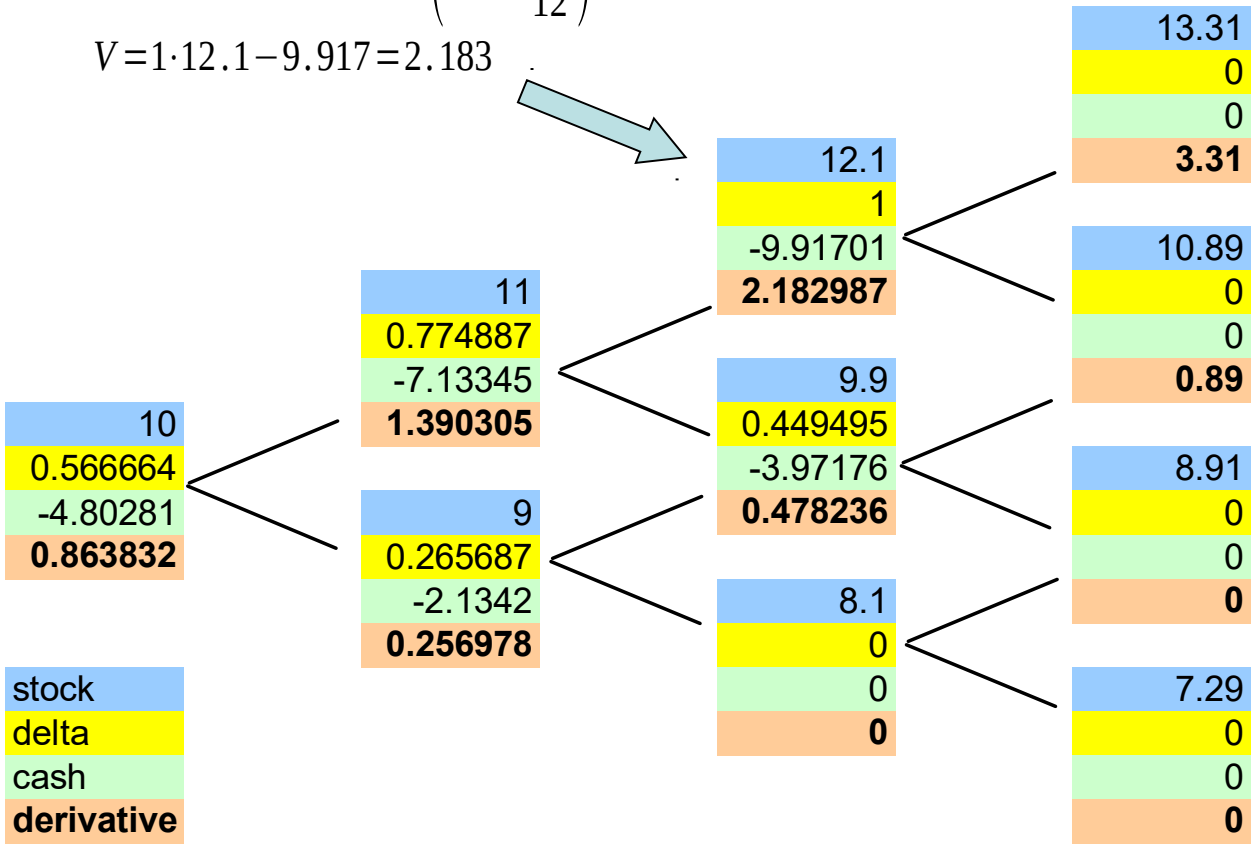
T=3/12 (3 months)

d=1.1, u=0.9

$$\delta = \frac{3.31 - 0.89}{13.31 - 10.89} = 1$$

$$\beta = (0.89 - 1 \cdot 10.89) \exp\left(-0.1 \frac{1}{12}\right) = -9.917$$

$$V = 1 \cdot 12.1 - 9.917 = 2.183$$



No Arbitrage Pricing

- Note that the replicating strategy is completely **self financing**, i.e. there are no cash flow injections nor withdrawal
- Suppose we sell the derivative and immediately hedge the position buying the replicating portfolio. Because we concluded that the price of the derivative is the same as the price of the replicating portfolio, the net value of our position after these transactions is zero.

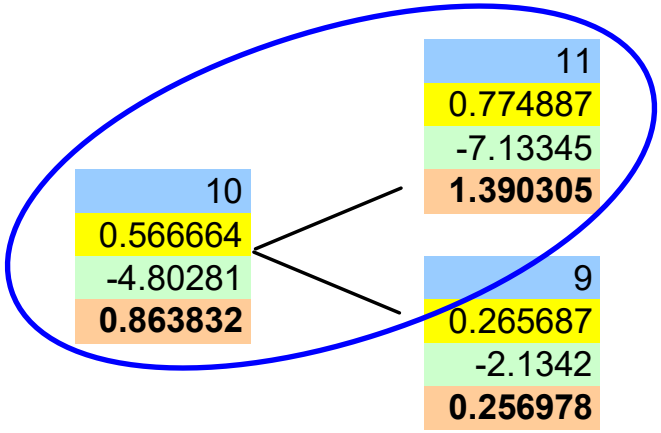
No Arbitrage Pricing Example

- At time 1, the value of the replicating portfolio still matches perfectly the value of the derivative contract by construction, therefore our total position is still worth zero.
- However, depending if the stock moved up or down, we need to rebalance appropriately the replicating portfolio for it to be an effective hedge over the next period.
- We want to show that this operation can be performed at zero cost.
- But this is verified by construction, because we are closing out the old hedge portfolio which is worth exactly the same as the derivative contract, and we are buying a new hedge portfolio which is also worth exactly the same as the derivative contract!

Hedge Rebalancing Example

- With reference to the previous example, we can illustrate the idea numerically. Let's focus for example on the "up" movement

Share price	10	11
Description	Time 0	Time 1
Sell derivative	0.863832	0
Buy 0.56666 shares	-5.66664	0
Borrow 4.8028	4.802808	0
Sell 0.56666 shares	0	6.23330385
Buy 0.7749 shares	0	-8.5237538
Pay 4.8028 loan & interests		-4.8429984
Borrow 7.1334		7.13344827
Total	0	0



- We have shown that the total net cash flow at time 1 in the "up" scenario case is zero

Notes on a Call Hedging Strategy

- It is self financing. If it wasn't we could have not concluded that the value of the derivative is the same as the replicating portfolio, as the strategy would have required random injection or withdrawal of capital in the portfolio
- We buy shares when the price increase and sell when the price goes down. That costs us money!
- We always borrow, and that costs us money too!

Where did we get so far?

- We learned how to construct a tree to fit the behavior of an asset price
- We learned about binomial distribution and how to compute expectation backward on a tree
- But then we learned how to price via no arbitrage and we did not use any of the above techniques
- So what was the point, was it just a waste of time?

Towards Risk Neutral Pricing

- It would be nice if we could combine the two frameworks, so that we could price a derivative contract computing its expectation backward (which is super easy), but obtain the same price we would obtain in the case of no arbitrage pricing

Toward Risk Neutral Pricing

$$\delta = \frac{V^+ - V^-}{S^+ - S^-}, \qquad \beta = \left(V^+ - \delta S^+\right) e^{-r\,\Delta t}$$

$$V = \delta S + \beta = \delta S + \left(V^+ - \delta S^+\right) e^{-r\,\Delta t}$$

$$= \left[\left(S e^{r\,\Delta t} - S^+\right)\delta + V^+\right] e^{-r\,\Delta t}$$

$$= \left[\left(\frac{S e^{r\,\Delta t} - S^+}{S^+ - S^-} + 1 - 1\right)\left(V^+ - V^-\right) + V^+\right] e^{-r\,\Delta t}$$

$$= \left[\left(\frac{S e^{r\,\Delta t} - S^-}{S^+ - S^-}\right) V^+ + \left(1 - \frac{S e^{r\,\Delta t} - S^-}{S^+ - S^-}\right) V^-\right] e^{-r\,\Delta t}$$

$$= \left[q V^+ + (1-q) V^-\right] e^{-r\,\Delta t} \quad \text{where} \quad q = \frac{S e^{r\,\Delta t} - S^-}{S^+ - S^-}$$

Hedge ratio and cash position computed for 1 step by no arbitrage

We manipulate the formula for the value of the derivative contract, until we can express it in a form which looks like "the discounted value of an expectation".
To achieve that we introduce a dummy probability *q*

Risk Neutral Pricing

- If we indicate up movements with $S^+ = uS$ and down movements as $S^- = dS$, we notice the probability q does not depends neither on the payoff, nor on the position in the tree

$$q = \frac{Se^{r\,\Delta t} - S^-}{S^+ - S^-} = \frac{e^{r\,\Delta t} - d}{u - d}$$

- q looks like a probability, however it is not a real probability, it is just a mathematical trick we pulled off to be able to price the derivative contract using a probability theory framework. For this reason it is called **risk neutral probability**.

Risk Neutral Probabilities

- To be able to use q as a probability, it must also satisfy all properties of a probability. I.e. it must be $0 \leq q \leq 1$.
- It is easy to demonstrate that this relation must hold by no arbitrage.

$$q = \frac{e^{r\,\Delta t} - d}{u - d}$$

$$q > 1 \quad \Rightarrow \quad e^{r\,\Delta t} > u$$

$$q < 0 \quad \Rightarrow \quad e^{r\,\Delta t} < d$$

Risk Neutral Probabilities

- If $q>1$, then the risk free rate has an expected return larger than the best possible return achievable in the stock. This is an arbitrage!
  - Any rationale investor would immediately short sell as many stocks as possible and invest in the risk free rate, realizing infinite profits.
- If $q<0$, then the risk free rate has an expected return lower than the worst possible return achievable in the stock. This is also an arbitrage!
  - Any rationale investor would immediately borrow and buy as many stocks as possible, realizing infinite profits.

Properties

- This is all confusing! Before we were matching expectation and volatility of the stock. Now we are pricing by no arbitrage, but which expectation and variance are we matching?
- What is the expectation of the stock price in one period?

$$E\left[S_T\right]=\left[quS+\left(1-q\right)dS\right]=S\left[q\left(u-d\right)+d\right]=S\left[\frac{e^{rT}-d}{u-d}\left(u-d\right)+d\right]=Se^{rT}$$

- We obtained the **forward price** (as it could be shown using a cash and carry argument) of the stock, which is also called the **risk neutral expectation**
- As per its variance, nothing can be said as it depends on the choice we make for *u* and *d*

Calibration to Process Properties

- We can still use the formulas derived before, except that now we want to match the *risk neutral* expectation of the stock price return, instead of the *real world* one (see implementation in terms of returns: parametersBinomGBMCRR.m)
- As long as the expectation of the tree matches the forward price, the model is risk neutral!
- In fact, the formula for the probability is the same as the one we derived before when matching the properties of the price return, except that the first moment is replaced by the risk neutral drift

$$q = \frac{\chi_1 - d}{u - d} = \frac{e^{r\,\Delta t} - d}{u - d}$$

The expectation we are matching is the one period risk neutral drift

$$\chi_1 = e^{r\,\Delta t}$$

Risk Neutral Pricing

- Thanks to the risk neutral probability q, we can now use the framework we devised initially to compute the value of the derivative contract.
- The only difference is the introduction of a discounting factor, i.e. the expectation computed backward at every step becomes the **risk neutral** expectation of the **present value** of the derivative

$$\begin{aligned}\left[q\,V^{+}+\left(1-q\right)V^{-}\right]e^{-r\,\Delta t} &= q\,e^{-r\,\Delta t}\,V^{+}+\left(1-q\right)e^{-r\,\Delta t}\,V^{-}\\ &= q\,PV\left(V^{+}\right)+\left(1-q\right)PV\left(V^{-}\right)=E^{Q}\left[PV\left(V\right)\right]\end{aligned}$$

- Because q is constant throughout the tree, we no longer need to solve for δ and β at every time step.

European Call Example

- A non-dividend paying stock follows a Geometric Brownian motion
- The interest rate is 7%, the volatility is 30%
- The spot price is 10.
- We want to price a 2 year call option strike 10 using a 3 step binomial tree, calibrated to the properties of the GBM.
- Note that we do not use the real world expected return of the stock, because, in risk neutral pricing, we only care about the risk neutral drift

European Call Example

- Using the formula obtained previously:

```
>> [d,u,p]=parametersBinomGBMCRR(7/100, 0/100, 0.3, 2, 3)
```

$$\chi_1 = \sqrt[N]{E\left[\frac{S_T}{S_0}\right]} = \sqrt[3]{e^{0.07\cdot 2}} = 1.0478$$

$$\chi_2 = \sqrt[N]{E\left[\left(\frac{S_T}{S_0}\right)^2\right]} = \sqrt[N]{Var\left[\frac{S_T}{S_0}\right] + E\left[\frac{S_T}{S_0}\right]^2} = \sqrt[3]{\left(e^{0.07\cdot 2}\right)^2 e^{0.3^2\cdot 2}} = 1.1657$$

$$d = \frac{\left(1+\chi_2\right) - \sqrt{\left(1+\chi_2\right)^2 - 4\chi_1^2}}{2\chi_1} = \frac{1+1.1657 - \sqrt{\left(1+1.1657\right)^2 - 4\cdot 1.0478^2}}{2\cdot 1.0478} = 0.7725$$

$$u = \frac{1}{d} = 1.2944$$

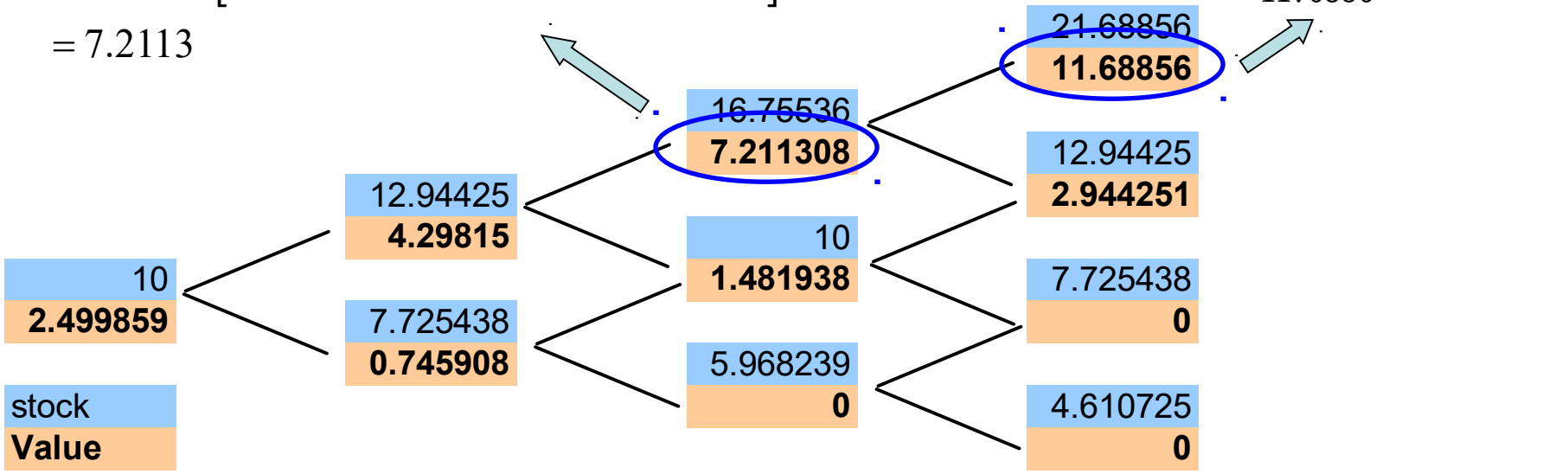
$$q = \frac{\chi_1 - d}{u - d} = \frac{1.0478 - 0.7725}{1.2944 - 0.7725} = 0.5274, \qquad 1 - q = 0.4726$$

$$PeriodDiscountFactor = e^{-r\frac{T}{N}} = e^{-0.07\cdot\frac{2}{3}} = 0.9544$$

European Call Example

$$e^{-r\frac{T}{N}}\left[qV^{+}+(1-q)V^{-}\right]$$
$$=0.9544\cdot\left[0.5274\cdot 11.6886+0.4726\cdot 2.9442\right]$$
$$=7.2113$$

$$\max\left[S-K,0\right]$$
$$=\max\left[21.6886-10,0\right]$$
$$=11.6886$$



- Note that only the stock price at the terminal time step are used in the computation

```
>> [d,u,p]=parametersBinomGBMCRR(7/100, 0/100, 3/100, 2, 3);
>> europeanBinomialPricer(d, u, p, 2, 3, 7/100, 10, @(S)max(0,S-10))
```

Computational Cost

- The computational cost is proportional to the number of nodes in the tree
- Because the number of nodes in the tree depends on N with the relation
  - NumNodes=1+2+3+…+(N+1)=(N+1)*(N+2)/2
- We can say that the computational cost in terms of number of time step is

  $\propto$((N+1)*(N+2)/2) → $O(N^2/2)$

European Payoffs

- For a European payoff, which is purely function of the value of the stock at expiration, we can skip all intermediate steps and use directly the state probabilities of the final step
- The expectation is just the product of the payoff at each state by the probability to reach that state.
- The discount factors are constants, therefore they can be factored out and multiplied separately

European Payoff Example

- Consider for example the payoff $f(S_T)$ in a 2 step tree
  - Let $Z=\exp(-r\,\Delta t)$
  - Value at time 2
    - Node up-up: f(uuS)
    - Node up-dn: f(udS)
    - Node dn-dn: f(ddS)
  - Value at time 1
    - Node up: Z [q f(uuS) + (1-q) f(udS)]
    - Node dn: Z [q f(udS) + (1-q) f(ddS)]
  - Value at time 0
    - Z { q Z [q f(uuS) + (1-q) f(udS)] + (1-q) Z [q f(udS) + (1-q) f(ddS)] }
      $= Z^2$ [q^2 f(uuS) + 2 q(1-q)f(udS) + $(1-q)^2$ f(ddS)]

European Payoffs

- Generalizing, for a European payoff(S_T)=f(S_T), considering a period *T* divided in *N* steps, assuming the stock can move either up or down with probabilities *u* and *d*, if the interest rate *r*, the value is just the sum of the products of the value of the payoff in all states of the final step by the correspondent risk neutral state probability, then multiplied by the discount factor for time *T*

$$V = \left(e^{-r\Delta t}\right)^N \sum_{j=0}^{N} P^{Q}\left(S_{N,j}\right) f\left(S_{N,j}\right) = \left(e^{-r\Delta t}\right)^N \sum_{j=0}^{N} \binom{N}{j} q^j (1-q)^{N-j} f\left(\underbrace{u^j d^{N-j} S_0}_{S_{N,j}}\right)$$

European Call Example

- With reference to the previous example, we show how to compute directly the value of the Call option, using only prices and probabilities at the last step

Spot		Payoff	Path Probability		n Paths	Product
Su^3d^0	21.688561	11.68856	$q^3(1-q)^0$	0.146679	1	1.714463
Su^2d^1	12.944251	2.944251	$q^2(1-q)^1$	0.131449	3	1.161059
Su^1d^2	7.7254376	0	$q^1(1-q)^2$	0.117801	3	0
Su^0d^3	4.6107255	0	$q^0(1-q)^3$	0.10557	1	0
					Sum	2.875522
					Call Price	**2.499859**

European Payoff Computational Cost

- This reduction allows a significant reduction in computational cost, which is now proportional to the number of nodes in the last time step, e.g.

  $\propto$(N+1) → O(N)

- Now the computational cost is linear in *N*, while before is was quadratic in *N*

Risk Neutral Distribution

- The binomial probability distribution obtained using the risk neutral probability *q* is called the **risk neutral probability density**
- The probabilities of reaching a node in the tree are called **state prices** and represent the undiscounted price of a claim paying 1 if S_T ends up in that state, 0 otherwise

$$q(j) = \left(\begin{array}{c} N \\ j \end{array} \right) = q^j (1-q)^{N-j}$$

Implementation Tricks

- The terms $Su^j d^{N-j}$ would seem to require the use of pow() operations. This is quite slow. With some ingenuity it is possible to avoid its use.
- After computing the first term, Sd^N the next one can be obtained simply by multiplying by (u/d).
- Similar technique can be used to avoid use of pow() in computing the path probabilities $p^j(1-p)^{N-j}$
- Binomial coefficient can also be computed in sequence, rather than one by one, which is faster. Given the term (N,j), the term (N,j+1) can be simply obtained as (N,j+1)=(N,j)*(N-j)/(j+1)

European Payoff Implementation

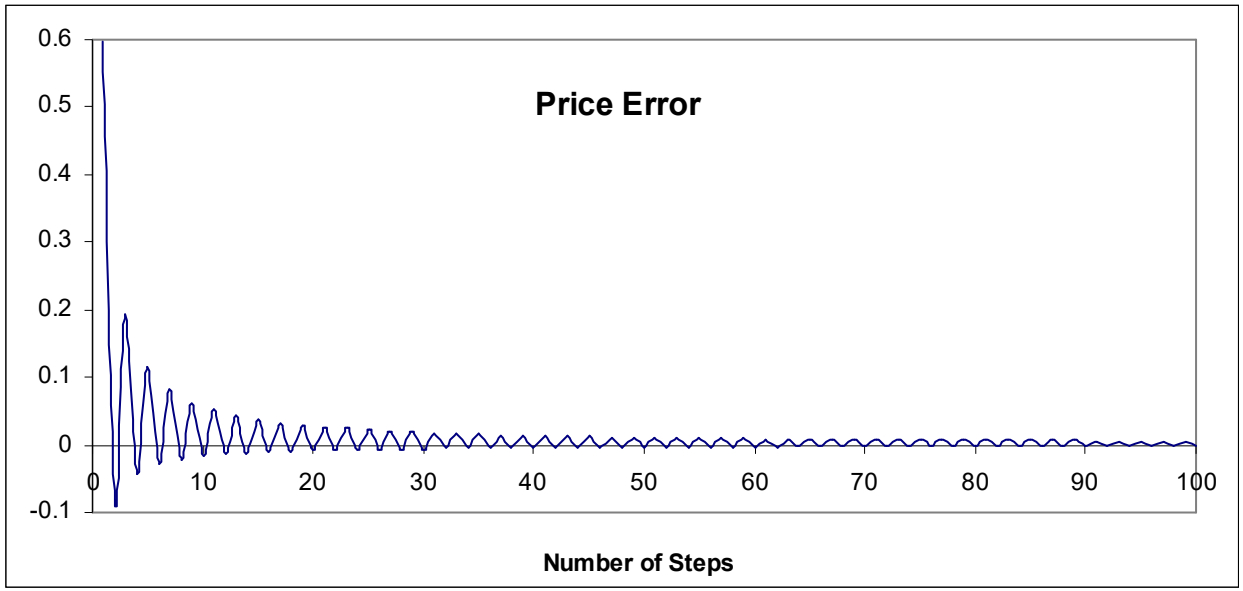
```
% europeanBinomialPricer.m
function res = europeanBinomialPricer( d, u, p, T, N, rf, spot, payoff )
    % Auxiliary variables
    ratioUd = u / d;
    ratioProb = p / ( 1.0 - p );

    % initialize loop variables
    price = spot * d ^ N; % Price corresponding to state i=0
    prob = ( 1 - p ) ^ N; % Probability of a path that reach state i=0
    binom = 1.0;
    expectation = prob * payoff( price );

    % loop over states in period N - 1 and compute the expectation
    for i = 1:N  % loop start from one, as we already added the case i=0
       price = price  * ratioUd;           % Price corresponding to state i
       prob = prob * ratioProb;            % Probability of a path that reach state i
       binom = binom * ( N - i + 1 ) / i;  % Number of paths which reach state i
       expectation = expectation + binom * prob * payoff( price );
    end

    res = exp( -rf * T ) * expectation;

% callPayoff.m
function res = callPayoff( spotPrice, strike )
    res = max( spotPrice - strike, 0.0 );

>> callWithStrike10 = @(x) callPayoff(x,10);
>> [d,u,p]=parametersBinomGBMCRR(7/100, 0/100, 3/100, 2, 100);
>> eurpeanBinomialPricer(d, u, p, 2, 100, 0.05, 2.95, callWithStrike10)
```

European Payoff Implementation

- **Sanity checks**: how do I know if my implementation is correct? By construction it must recover the forward and the variance, therefore if we use as payoff $f(S)=S$ and $f(S)=S^2$, the algorithm must recover exactly the first two moments "discounted", regardless of the number of nodes.

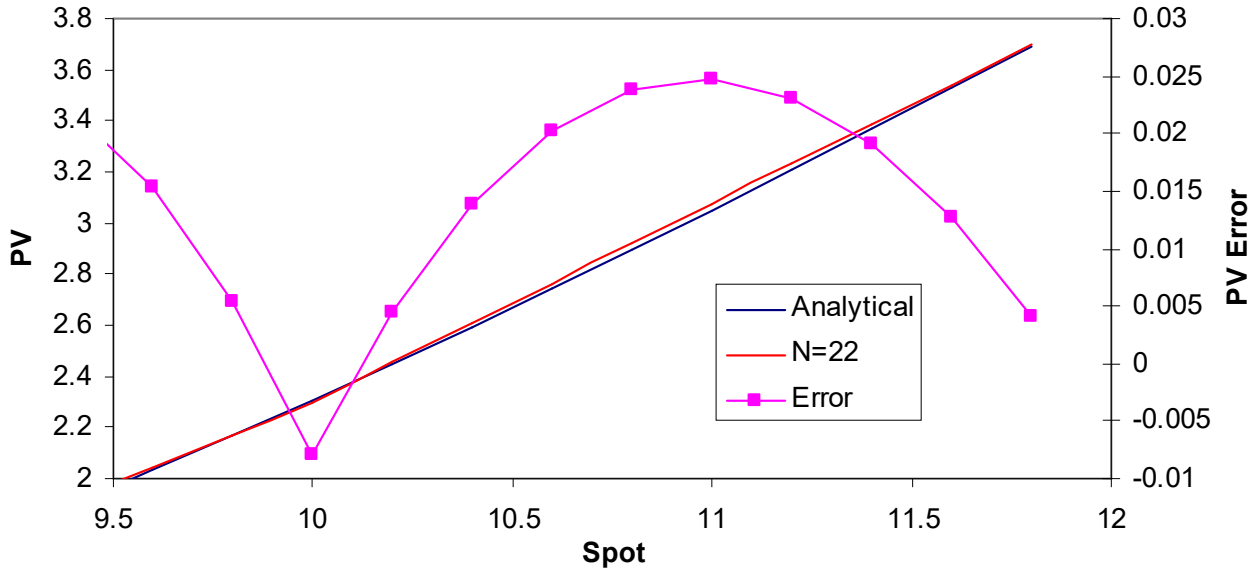
European Call Example

- The following chart shows the approximation error as a function of the number of periods in which we subdivide the 2 years
- Note the oscillatory nature of the convergence
- To put the error in perspective, the correct premium is 2.3067 (obtained via the B&S formula)




```
>> europeanBinomTest(10, 10, 7/100, 0/100, 30/100, 2, 100)
```

European Call Example



- We fix the strike, but we change the spot.
- We compare the price of the binomial tree with 22 steps vs the Black Scholes formula
- We print the two prices on the left axis and the error on the right axis
- We observe the error oscillations.
- Oscillations are problematic for Greeks, as they get amplified
- For European option stability is related to the position of discontinuities with respect to grid points (Wand 2002)

```
>> rf=7/100;vol=30/100;yield=0;T=2;spot=10;strike=10;steps=22;
>> tree=@(s)europeanBinomialPricer(d,u,p,T,steps,rf,s,@(x)max(0,x-strike));
>> blk=@(s)exp(-rf*T)*black(s*exp(rf*T),strike,vol,T,true);
>> s=[9.5:0.1:12]; for i=1:length(s), vtree(i)=tree(s(i)); vblk(i)=blk(s(i)); end
>> plot (s,vtree, s,vblk); hold on; yyaxis right; plot(s,vtree-vblk); hold off; yyaxis left;
```

Why Oscillations are Annoying?

- Let $V(S_0)$ be the value of the derivative computed using the tree conditional to S_0, we do Greeks via FD: $[V(S_0+h)-f(S_0-h)]/(2h)$, we change the position of the strike with respect to the tree-grid points. If in the two cases I happen to be at the top and the bottom of an oscillation, this may affect badly the accuracy of the derivative
- If we do Richardson extrapolation, using N_1 and N_2 intervals, if we are at the top and bottom of an oscillation, we will get very weird results

Summary

- We learned how to price on a tree using risk neutral expectation, therefore obtaining the price of a derivative we would have computed using a replication argument
- Note that all it took was to replace the original real world drift of the process driving the dynamic of the underlying asset with the risk neutral drift
- If we had known this in advance we could have saved some time!
- This is something that holds also for other models.

Further Readings

- 1979, Cox, Ross, Rubenstein, *Option Pricing, A Simplified Approach*
- Hull, *Option Future and Other Derivatives*, *Introduction to Binomial Trees*
- Shreve, Stochastic Calculus for Finance I: The Binomial Asset Pricing Model, Chapter 1
- Wilmott, On Quantitative Finance, *The Binomial Model*