M2 Course content

Introduction
- Language reminders:
  - VBA
  - Python
- Classes in VBA
- Classes in Python
- Project: Option pricing with a lattice

10/4/2025 21:50:02

Pricing model landscape

- Here are the three types of pricing models:

	Closed-form formulas	Trees	Monte-Carlo
Examples	Black-Scholes, Heston, SABR	Hull & White	Longstaff-Schwartz
European ex.	Yes	Yes	Yes
American ex.	No	Yes	Hard
Path dependent	No	Hard	Yes
Speed	Instantaneous	Fast	Slow
Precision	Perfect	Error ↘ 1/Nb Steps	Error ↘ 1/√Nb Draws
Memory	Light	Depends	Depends

Lattice node VBA class

- Goal #1, practice PDE and trees:
  - Simulate a stock price in a recombining lattice
  - Price European and American calls and puts
  - Assess convergence with Black-Scholes
- Goal #2, code and use classes:
  - Define classes for the lattice node and tree
  - Build a tree with reconnecting nodes
  - Price options on this lattice

Stock price behavior

- In the « *Derivatives Pricing and Stochastic Calculus* » course by Elie & Kharroubi we can see for Black & Scholes p. 61:

• **The risky asset.** We suppose that the risky asset S is given by the SDE

$$\begin{cases} dS_t &= S_t\,(\mu\,dt + \sigma\,dW_t)\,, \quad t \in [0,T] \\ S_0 &= s_0 \end{cases} \tag{6.2.1}$$

where μ and σ are two constants such that $\sigma > 0$.

- We can use those assumptions for the interest rate and the volatility
- But the dividend is discrete and it is simplistic to define μ as: μ = rate - dividend

Discrete dividends

- Dividends in a near future are known
- Dividends in several years can be approximated as a % of the stock price
- Example with a stock price S_0 = 100 €
  - First annual dividend = 3 €
  - Second annual dividend = 2 € + 1% of X_{1Y}
  - Third annual dividend = 1.4 € + 1.6% of X_{2Y}
  - Fourth annual dividend = 1 € + 2% of X_{3Y}
  - …
  - Tenth annual dividend = 3% of X_{10Y}

10/4/2025 21:50:02

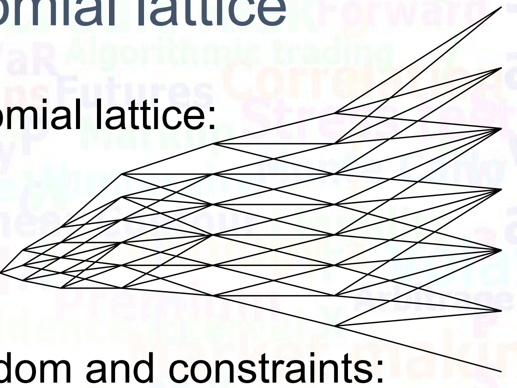
Dividend model

- The dividend is discrete and moves from constant to proportional to the stock price
- Example of possible functional form:
  - $D_t = \rho\,(S_0\, e^{-\lambda(t-t_0)} + S_t\,(1- e^{-\lambda(t-t_0)}))$ on ex-div. dates
  - $D_t = 0$ on all other dates
- $dS_t = S_t\, r\, dt + S_t\, \sigma\, dW_t – D_t$

10/4/2025 21:50:02

Trinomial lattice

- Example of trinomial lattice:

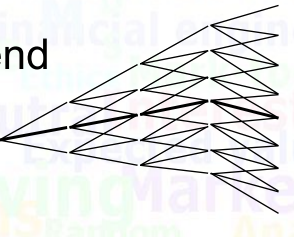


- Degrees of freedom and constraints:

Degrees of freedom	Constraints
Time steps	Match expected value
Nodes values	Match variance
Choice of next middle node	Positive probabilities
Transition probabilities	Sum of probabilities = 1

Trinomial tree assumptions

- We use the following assumptions:
  - Time steps are all equal: Δt
  - The middle node is equal to the forward price
  - Nodes values are geometric series: $\alpha = S_{up}/S_{mid}$
  - The next middle node is the closest to the forward price
- Example with a dividend on the fifth date:



Lattice building

- Start from the root on date d_0: N_{start} with $S_{start} = s_0$
- Branch to the next central node: N_{mid}

$$S_{mid} = S_{start} \exp(r\, \Delta t) - D_1$$

- Build other nodes on date d_1: $S_{up} = S_{mid}\, \alpha$
- α is defined by a multiple of the standard deviation over one time step ≈ $S_t\, \sigma\, \mathrm{sqr}(\Delta t)$:

$$S_{up} - S_{mid} \approx \sqrt{3}\, StdDev = \sqrt{3}\, S_{mid} \sqrt{e^{\sigma^2 \Delta t} - 1}$$

- Divide by S_{mid}: $\alpha \approx 1 + \sqrt{3}\, StdDev / S_{mid}$
- The actual formula is: $\alpha = e^{\sigma\sqrt{3\Delta t}}$
  - Please note: $\sqrt{3}$ is indicative. $\sqrt{2}$ as well as 2 work too

10/4/2025 21:50:02

Probabilities (1)

- For each node N we calculate the probabilities:
  - The sum of the three probabilities is equal to 1:

$$p_{up} + p_{mid} + p_{down} = 1$$

  - Matching the expected value means:

$$E_{i+1} = p_{up}\, S_{i+1,up} + p_{mid}\, S_{i+1,mid} + p_{down}\, S_{i+1,down} = E\left[S_{t_{i+1}} \middle| S_{t_i}\right]$$

  - With $S_{t_{i+1}} = S_{t_i} e^{\left(r - \frac{1}{2}\sigma^2\right)\Delta t + \sigma W_{\Delta t}} - D_{t_{i+1}}$

$$E\left[S_{t_{i+1}} \middle| S_{t_i}\right] = S_{t_i} e^{r\Delta t} - D_{t_{i+1}}$$

10/4/2025 21:50:02

Probabilities (2)

- Matching the variance on the step after N means:

$$V_{i+1} + E_{i+1}^2 = p_{up} S_{i+1,up}^2 + p_{mid} S_{i+1,mid}^2 + p_{down} S_{i+1,down}^2$$
$$V_{i+1} = E\left[\left(S_{t_{i+1}} - E\left[S_{t_{i+1}} \middle| S_{t_i}\right]\right)^2 \middle| S_{t_i}\right]$$

  - $V_{i+1} = E\left[S_{t_i}^2\left(e^{\left(r-\frac{1}{2}\sigma^2\right)\Delta t+\sigma W_{\Delta t}} - e^{r\Delta t}\right)^2 \middle| S_{t_i}\right]$
  - $V_{i+1} = S_{t_i}^2\, e^{2r\Delta t}\, E\left[e^{-\sigma^2\Delta t+2\sigma W_{\Delta t}} - 2e^{-\frac{1}{2}\sigma^2\Delta t+\sigma W_{\Delta t}} + 1\right]$
  - $V_{i+1} = S_{t_i}^2\, e^{2r\Delta t}\left(e^{\sigma^2\Delta t} - 1\right)$

10/4/2025 21:50:02

Probabilities (3)

- The three constraints make a linear system:
  - $1 = p_{up} + p_{mid} + p_{down}$
  - $E_{i+1} = p_{up}\, S_{i+1,up} + p_{mid}\, S_{i+1,mid} + p_{down}\, S_{i+1,down}$
  - $V_{i+1} + E_{i+1}^2 = p_{up}\, S_{i+1,up}^2 + p_{mid}\, S_{i+1,mid}^2 + p_{down}\, S_{i+1,down}^2$
- That can be written as a matrix product:

$$\begin{pmatrix} 1 & 1 & 1 \\ S_{i+1,up} & S_{i+1,mid} & S_{i+1,down} \\ S_{i+1,up}^2 & S_{i+1,mid}^2 & S_{i+1,down}^2 \end{pmatrix}\begin{pmatrix} p_{up} \\ p_{mid} \\ p_{down} \end{pmatrix} = \begin{pmatrix} 1 \\ E_{i+1} \\ V_{i+1} + E_{i+1}^2 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & S_{i+1,mid} & 0 \\ 0 & 0 & S_{i+1,mid}^2 \end{pmatrix}\begin{pmatrix} 1 & 1 & 1 \\ \alpha & 1 & \alpha^{-1} \\ \alpha^2 & 1 & \alpha^{-2} \end{pmatrix}\begin{pmatrix} p_{up} \\ p_{mid} \\ p_{down} \end{pmatrix} = \begin{pmatrix} 1 \\ E_{i+1} \\ V_{i+1} + E_{i+1}^2 \end{pmatrix}$$

10/4/2025 21:50:02

Probabilities (4)

- The system has a unique solution with α>1 and S_{i+1}>0:

$$\begin{pmatrix} 1 & 1 & 1 \\ \alpha & 1 & \alpha^{-1} \\ \alpha^2 & 1 & \alpha^{-2} \end{pmatrix} \begin{pmatrix} p_{up} \\ p_{mid} \\ p_{down} \end{pmatrix} = \begin{pmatrix} 1 \\ S_{i+1,mid}^{-1} E_{i+1} \\ S_{i+1,mid}^{-2} \left(V_{i+1} + E_{i+1}^2 \right) \end{pmatrix}$$

- Solve by substitutions:

$$\begin{pmatrix} 1 & 1 & 1 \\ \alpha - 1 & 0 & \alpha^{-1} - 1 \\ \alpha^2 - 1 & 0 & \alpha^{-2} - 1 \end{pmatrix} \begin{pmatrix} p_{up} \\ p_{mid} \\ p_{down} \end{pmatrix} = \begin{pmatrix} 1 \\ S_{i+1,mid}^{-1} E_{i+1} - 1 \\ S_{i+1,mid}^{-2} \left(V_{i+1} + E_{i+1}^2 \right) - 1 \end{pmatrix}$$

- Subtract (α+1) times the second line to the third to get:

$$(1 - \alpha)(\alpha^{-2} - 1)\, p_{down} \\ = S_{i+1,mid}^{-2} \left(V_{i+1} + E_{i+1}^2 \right) - 1 - (\alpha + 1)\left(S_{i+1,mid}^{-1} E_{i+1} - 1 \right)$$

- p_{down} has no dimension (good!), but is not guaranteed to be in [0, 1]

10/4/2025 21:50:02

Probabilities (5)

$$p_{down} = \frac{S_{i+1,mid}^{-2} \left(V_{i+1} + E_{i+1}^2 \right) - 1 - (\alpha + 1)\left(S_{i+1,mid}^{-1} E_{i+1} - 1 \right)}{(1 - \alpha)(\alpha^{-2} - 1)}$$

- When there is no dividend, $S_{i+1,mid} = E_{i+1}$ and we have:

$$p_{down} = \frac{S_{i+1,mid}^{-2} V_{i+1}}{(1 - \alpha)(\alpha^{-2} - 1)} = \frac{e^{\sigma^2 \Delta t} - 1}{(1 - \alpha)(\alpha^{-2} - 1)}$$

- Use the second row to solve p_{up}:

$$(\alpha - 1) p_{up} + (\alpha^{-1} - 1) p_{down} = S_{i+1,mid}^{-1} E_{i+1} - 1$$

- Without div: $p_{up} = \frac{p_{down}}{\alpha}$
- Use the sum to solve p_{mid}

European option pricing

- Build a lattice with N steps and $t_N = T$, the option maturity
- Then value the payoff on each node on T
  - On ITM nodes the payoff is for a call: $S_N - K$
  - On OTM nodes the payoff is equal to 0
- Propagate the net future value on t_{N-1} nodes:
  - NFV_{N-1} = sum of probability x next node value x DF
  - DF is the discount factor = $\exp(-r\,\Delta t)$
- NFV_0 = NPV is the net present value of the option

10/4/2025 21:50:02

American option pricing

- Same as European options except for the early exercise
- At any time before the maturity the option holder can exercise the option
- On each interim node the value is the maximum of two quantities:
  - Hold: value = discounted value like for European case
  - Exercise: value = $\max(S_i - K, 0)$ for a call

10/4/2025 21:50:02

Assignment (1)

- Build a trinomial lattice and an option pricer
  - Start with few steps, no dividend, European exercise
  - Try to have as much code as possible in the classes
  - Check that the price converges to Black formula
  - Describe the gap as a function of the number of steps
  - Describe the gap as a function of the strike
- Add the American vs. European exercise
  - Study the difference of price American – European
  - Observe the effect of the interest rate (>0 or <0)
  - Describe the cases where the two prices are equal

10/4/2025 21:50:02

Assignment (2)

- Add the dividend
  - With interest rates = 0, see the impact on the call and the put
- Other possible extensions:
  - Graph tree
    - Option value
    - Exercise frontier
  - Search the limit of number of steps (time, memory)
    - Try to remove nodes with very low probability
      - Describe the effect on run time for 1000 steps
    - Extend the number of steps
  - Set the number of steps from a user-defined precision

10/4/2025 21:50:02

Assignment (3)

- Recursive pricing:
  - Either locate the code in the Model/Tree class:

```
' Call recursive pricing
Let OptionPricerRecVBA = model.Price(opt, t.Root)
```

  - Or in the Node class:

```
optionPrice: float = model.tree.root.priceRecursive(opt)
```

- Choose, in VBA and in Python, the design you prefer

10/4/2025 21:50:02

Assignment (4)

- Derivative:
  - Simple design in VBA:

```
    Let shift = startPrice * 0.01
    Let priceUp = OptionPricerRecVBA(nSteps, startDate, matDate, _
      startPrice + shift, rate, volat, div, exDate, optType, exer, k, False)
    Let priceDown = OptionPricerRecVBA(nSteps, startDate, matDate, _
      startPrice - shift, rate, volat, div, exDate, optType, exer, k, False)
    Let OptionPricerRecVBADelta = (priceUp - priceDown) / (2 * shift)
End Function
```

  - In Python we can pass a function as a parameter:

```
class OneDimDerivative:

    def __init__(self, function: Callable[[object, float], float],
                 other_parameters: object, shift: float = 1):
        self.f: Callable[[object, float], float] = function
        self.param: object = other_parameters
        self.shift: float = shift

    2 usages
    def first(self, x: float) -> float:
        return (self.f(self.param, x + self.shift)
                - self.f(self.param, x - self.shift)) / (2 * self.shift)
```

```
def OptionDeltaTreeBackward(market_range: numpy.array, pricer_range: numpy.array, option_range: numpy.array) -> float:
    return OneDimDerivative(cast((object, float), _PriceTreeBackward_OneDimPrice),
                            OptionPricingParam(market_range, pricer_range, option_range),
                            market_range[0] * OptionPricingParam.UND_SHIFT).first(market_range[0])
```

Assignment (5)

- Other trick to calculate delta and gamma:
  - Add up and down nodes around the root
  - Calculate the option prices in the whole tree
  - Deduce the delta from the option and underlying prices on the up and down nodes
  - Ditto for the gamma from the three nodes
- There is no similar variation for vega and vomma

Assignment (6)

- Report: short and informative
  - When you write 10 pages, I read 250
- Express:
  - What you did
  - Your noticeable findings
  - Your difficulties…
  - …and how you overcame them (when you did)
- I love nice graphs


Edi VERBOVCI

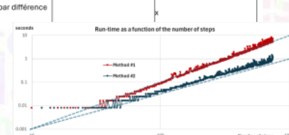
Timothée DANGLETERRE

Rapport : étude de cas – Pricing par la méthode de l'arbre trinomial

1. **Facultés du code :**

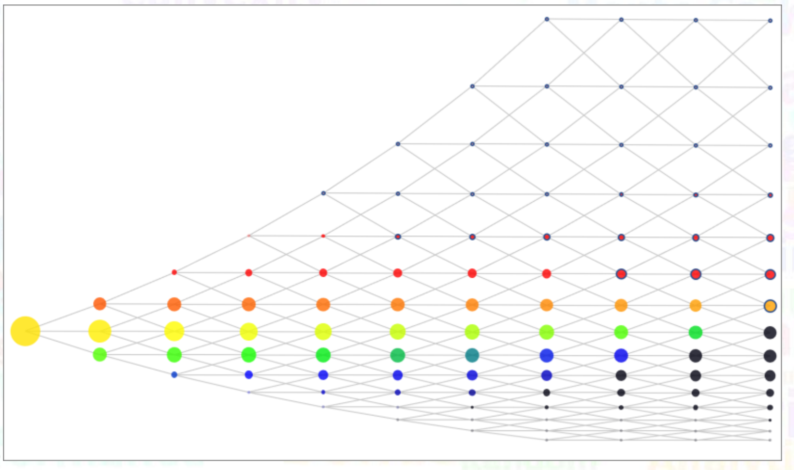
Dans le cadre de ce projet, nous avons réalisé en Python et VBA un code orienté objet permettant de construire un arbre trinomial pour valoriser une option Américaine / Européenne en tenant compte d'un éventuel versement de dividende. Les deux codes sont proches mais n'ont pas tout à fait les mêmes fonctionnalités. Les applications possibles sont résumées dans le tableau ci-dessous :

	VBA	Python
Construction de l'arbre trinomial (+1000 pas)	x	x
Méthode de pricing	Récursif & Backward	Backward
Représentation graphique de l'arbre	x	
Pruning au niveau de l'arbre	x	x
Calcul des Grecques par la méthode de Black & Scholes	x	
Calcul des Grecques au premier ordre par différence finie		x



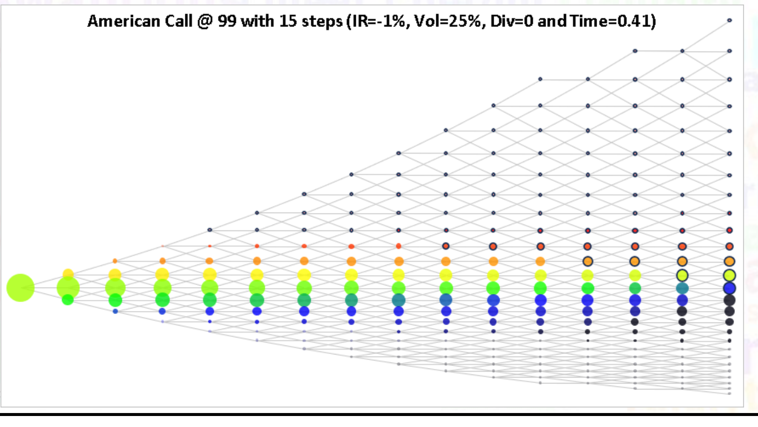
Example of tree with 10 steps

- 1Y ATM American call (vol=30%, int. rate=-1%)



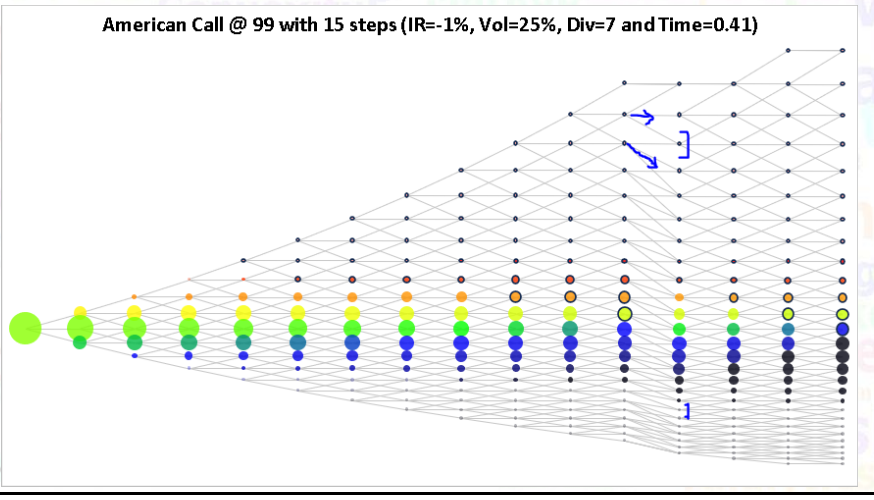
Effect of discrete dividends (1)

- In general, the middle nodes are aligned:
  - The j^{th} node above the trunk on date i connects with the j^{th} node above the trunk on date i+1 as next mid



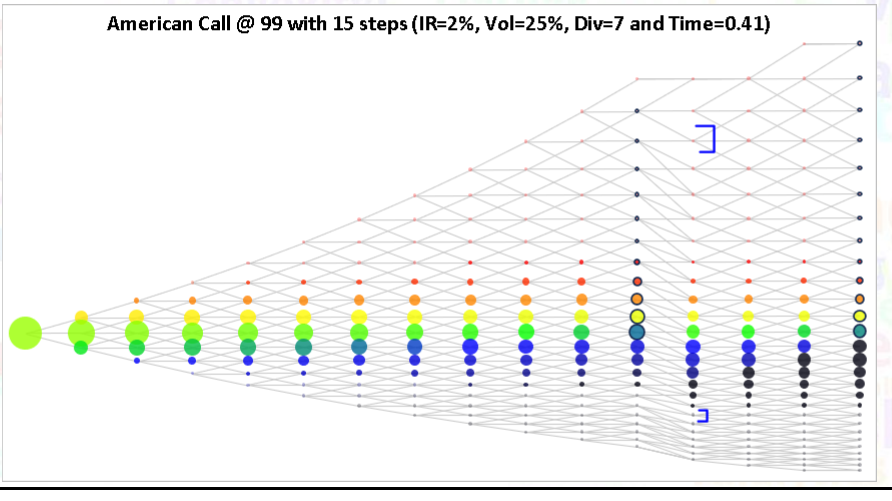
Effect of discrete dividends (2)

- But that doesn't hold with dividends:



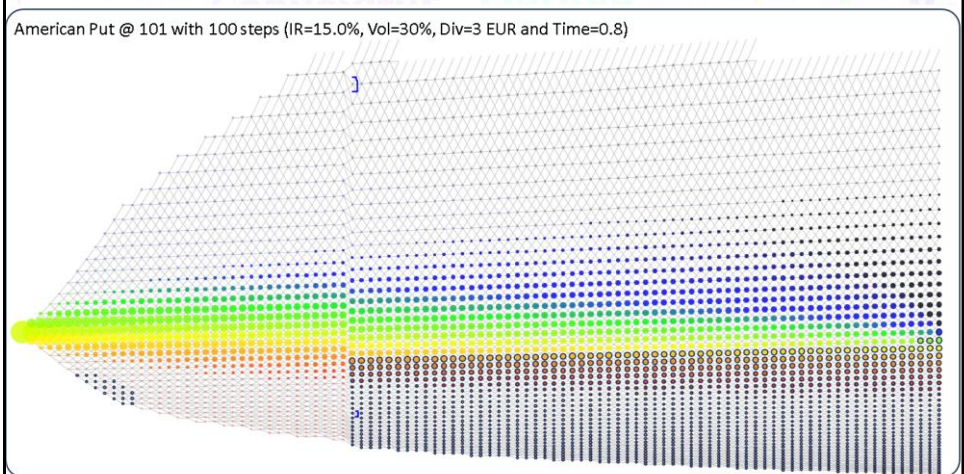
Effect of discrete dividends (3)

- Same with positive interest rate:



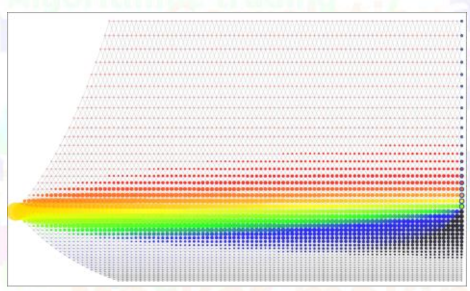
Effect of discrete dividends (4)

- Other example with a put:

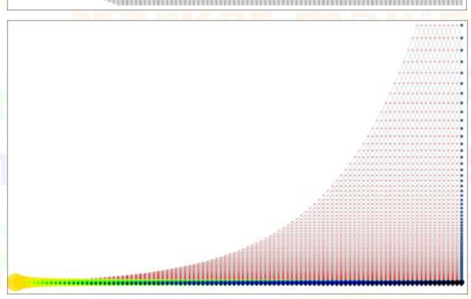


Example of tree with 100 steps

- With prices between 30 and 300:



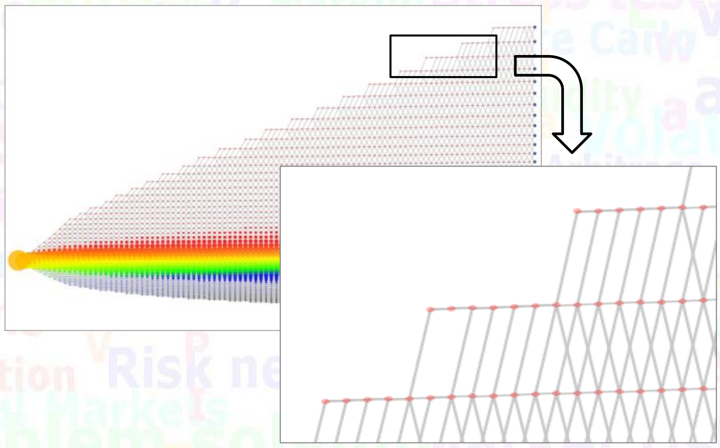
- …and between 1 and 10,000:



10/4/2025 21:50:02

Tree with side connections cut

- Tree with mid connection below 1E-7:



Tree "pruning" (1)

- It consists in not allocating nodes which probability is too small
- A typical design consists in branching to the middle node only, with a probability = 1
- The "monomial" branching can be triggered by one of two designs:
  1. by the probability to reach a node, calculated from the root
  2. or by a number of standard deviations of the node from the middle one

Tree "pruning" (2)

- We can calculate the probability to reach a node by:
  - starting with the root node with probability = 100%
  - any other node has a probability equal to the sum of previous nodes connected to it x transition probability
    - Example: the top node has a probability equal to the top node on the previous date x transition probability
    - If the move up probability is, say, 0.18, the probability to be on the top node decreases like 0.18^i, with i the number of the column date. After 9 steps the top node probability is close to 2.10^{-7}.

10/4/2025 21:50:02

Tree "pruning" (3)

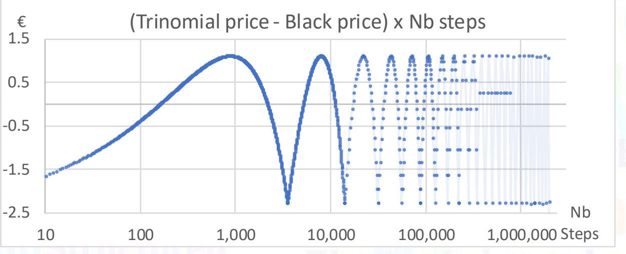
- We can also deduce a node probability by its number of standard deviations from the middle:
  - The process on date i is: $\frac{S_{t_i}}{S_{forward_i}} = e^{\sigma W_{i\,\Delta t} - \frac{1}{2}\sigma^2 \Delta t}$
  - $ln\left(\frac{S_{t_i}}{S_{forward_i}}\right)$ standard deviation = $\sigma\sqrt{i\,\Delta t}$
  - The relative space between nodes is $\sigma\sqrt{3\Delta t}$
  - If we allocate nodes over 4 standard deviations we need k nodes on either sides:

$k\sigma\sqrt{3\Delta t} \geq 4\sigma\sqrt{i\,\Delta t}$, which simplifies into: $k \geq 4\sqrt{\frac{i}{3}}$

10/4/2025 21:50:02

Calculate number of steps (1)

- (Tree – BS) x NbSteps gap lies in a tunnel

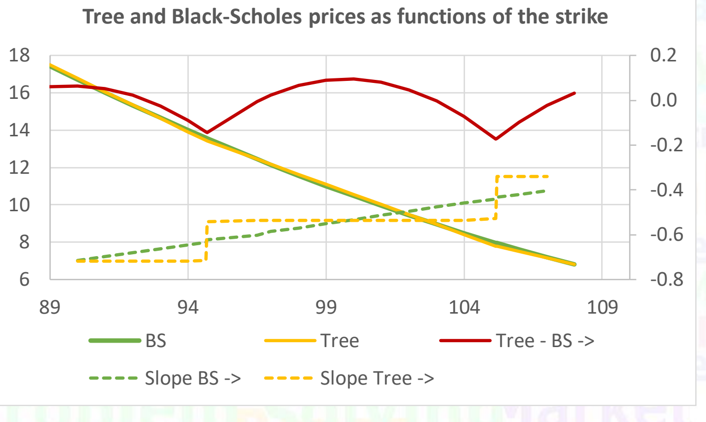


- => Gap = f(S_0, σ, K, T, r) / NbSteps
- f(S_0, σ, K, T, r) is a function of other parameters: vol, strike, maturity, etc.

10/4/2025 21:50:02

Calculate number of steps (2)

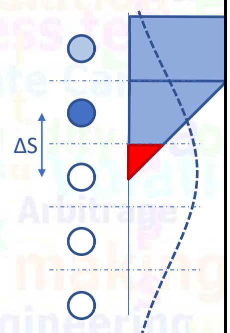
- Tree vs. Black Price function of K (10 steps):



10/4/2025 21:50:02

Calculate number of steps (3)

- Idea: consider the ATM forward option
- Gap ≈ lost value of node where S = K
- Value of that node with BS:



$$\text{Red triangle x density} = \frac{1}{2}\left(\frac{\Delta S}{2}\right)^2 \frac{1}{\sqrt{2\pi}\,\Sigma}$$

$$\Delta S = Fwd(T)\ (\alpha - 1)$$

$$\Delta S = (S_0\ e^{rT} - D)\left(e^{r\Delta t + \sigma\sqrt{3\Delta t}} - 1\right)$$

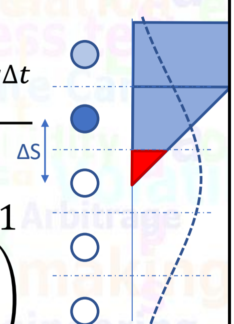
- Σ is the standard deviation of the stock distribution on the maturity date T

$$\Sigma = S_0\ e^{rT}\sqrt{e^{\sigma^2 T} - 1}$$

10/4/2025 21:50:02

Calculate number of steps (4)

- Gap function of Δt (i.e. of NbSteps) is:



$$Gap(NbSteps) \approx \frac{3\ S_0}{8\sqrt{2\pi}} \frac{\left(e^{\sigma^2\Delta t} - 1\right)e^{2r\Delta t}}{\sqrt{e^{\sigma^2 T} - 1}}$$

- The main dependency on Δt is in $e^{\sigma^2\Delta t} - 1$

$$\Delta t \approx \frac{1}{\sigma^2} ln\left(1 + \frac{8\sqrt{2\pi}\ Gap\ \sqrt{e^{\sigma^2 T} - 1}}{3\ S_0}\right)$$

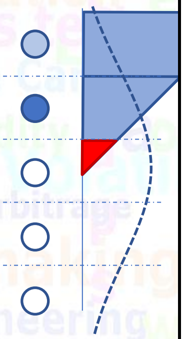
$$NbSteps = \frac{T}{\Delta t} \approx \frac{\sigma^2\ T}{ln\left(1 + \frac{8\sqrt{2\pi}\ Gap\ \sqrt{e^{\sigma^2 T} - 1}}{3\ S_0}\right)}$$

10/4/2025 21:50:02

Calculate number of steps (5)

Remarks:

- $\frac{Gap}{S_0}$ is a relative precision on the stock price
- For small gaps, NbSteps = T / Δt is inversely proportional to the gap (and the gap to NbSteps)
- Simplified formulas:



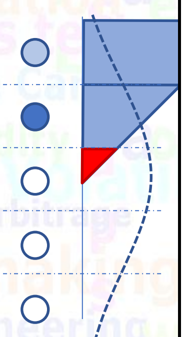
$$Gap \approx \frac{3\,S_0}{8\sqrt{2\pi}} \frac{\sigma^2 \Delta t}{\sqrt{e^{\sigma^2 T}-1}} \qquad \Delta t \approx \frac{8\sqrt{2\pi}}{3} \frac{Gap}{S_0} \frac{\sqrt{e^{\sigma^2 T}-1}}{\sigma^2}$$

$$NbSteps = \frac{T}{\Delta t} \approx \frac{3}{8\sqrt{2\pi}} \frac{S_0}{Gap} \frac{\sigma^2 T}{\sqrt{e^{\sigma^2 T}-1}}$$

Calculate number of steps (6)

Remarks:

- With T = 1 year, σ = 30%, we have:
  - $\frac{Gap}{S_0} \approx \frac{1}{N} \frac{3}{8\sqrt{2\pi}} \frac{0.3^2}{\sqrt{e^{0.3^2}-1}}$
  - N=1: $\frac{Gap}{S_0}$ = 4%
  - N=10: $\frac{Gap}{S_0}$ = 0.4%
  - N=100: $\frac{Gap}{S_0}$ = 0.04%
  - N=1000: $\frac{Gap}{S_0}$ = 0.004%



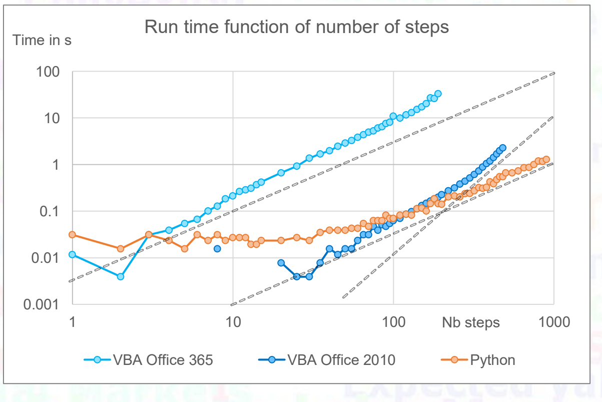
How to check prices?

- You can check several results:
  - European options without dividends compared to Black-Scholes prices
  - European calls with strike = 0, compared with the forward price on maturity, including with dividends
  - American options are harder to test, but the risk of error is lower (very little code is involved) and you can check the exercise frontier

27-Nov-19 21:50:02

Performance: VBA vs. Python

- VBA performance has been poor in 365 version:

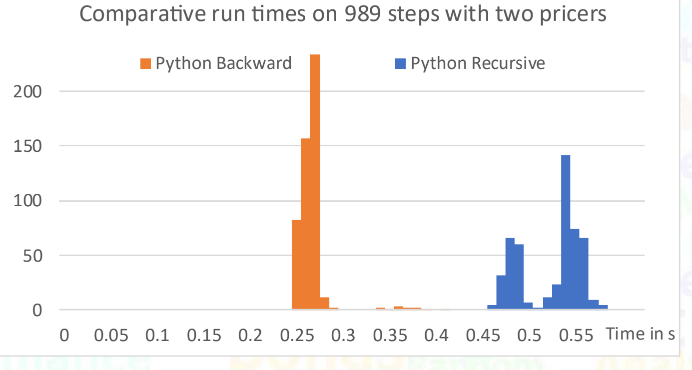


Recursive vs. backward model

- Backward model:
  - Go to last date
    - Price up and down nodes
  - Go to preceding date on the trunk
    - Price up and down nodes
  - Repeat until reaching the root
- This requires a backward link on trunk nodes
  - In Python: define a TrunkNode class that inherits from the Node class

Recursive vs. backward model

- Comparative run times between the recursive and backward pricers in Python
  - Histogram of run times on a sample of 500 runs



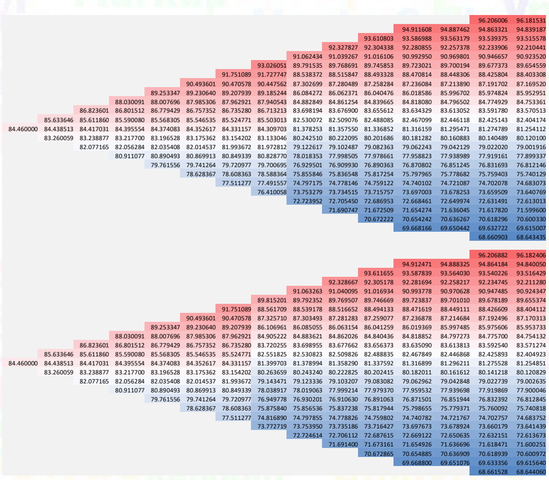
Gap analysis: VBA vs. Python

- Test on random parameters:
  - Draw a set of random parameters
  - Price with VBA and Python
  - Store set of parameters if:
    - The gap is larger than 1.10^{-13}
    - Or there is a negative probability

Start date	Start price	ıterest rate	Volatility	Dividend	Div ex-date	Maturity	Type	Exercise	Strike	Nb steps	VBA	Py Rec	Py Backward	Gap 1	Gap 2	Run test
23-Aug-22	88.71	1.46%	30.95%	1.36	08-Dec-22	07-Apr-23	Put	European	70.40	46	1.882848	1.882848	1.882848	-3.11E-15	0	On
06-Aug-22	117.04	3.80%	44.15%	10.04	29-Apr-23	08-May-23	Put	American	122.97	81	25.482674	AssertionError: Err	NbStep=81	#VALUE!	#VALUE!	
08-Sep-22	86.50	11.64%	49.14%	5.25	29-May-23	13-Dec-23	Put	American	69.13	70	7.877375	AssertionError: Err	NbStep=70	#VALUE!	#VALUE!	
30-Nov-22	101.49	9.30%	3.91%	0.61	15-Oct-23	29-Oct-23	Call	American	73.13	94	34.065895	34.065895	34.065895	3.41E-13	0	
12-Aug-22	114.21	-4.60%	45.57%	6.17	11-Aug-23	01-Feb-24	Call	European	121.40	81	17.189573	AssertionError: Err	NbStep=81	#VALUE!	#VALUE!	
12-Jan-22	100.60	-2.07%	43.71%	2.21	01-Feb-22	04-Sep-22	Put	American	134.34	90	41.606851	41.606851	41.606851	4.19E-13	0	
19-Mar-22	128.08	-3.09%	40.63%	3.39	24-Aug-22	03-Apr-23	Put	European	199.60	69	85.186486	85.186486	85.186486	4.97E-13	0	
22-Feb-22	84.46	-1.97%	7.14%	2.91	27-Mar-22	29-Apr-22	Put	American	70.14	14	0.000000	0.000000	0.000000	-2.02E-08	0	
14-May-22	81.70	-3.97%	38.45%	1.86	01-May-23	13-Jun-23	Put	European	127.98	91	55.845025	55.845025	55.845025	4.05E-13	0	
28-Jan-22	156.32	10.98%	42.66%	3.16	22-Jul-22	28-Oct-22	Call	American	109.78	91	55.288951	55.288951	55.288951	-5.47E-13	0	
18-Feb-22	163.73	9.71%	10.77%	1.66	02-Apr-22	07-Dec-22	Call	European	108.88	75	61.348146	61.348146	61.348146	-5.47E-13	0	

Example of gap analysis

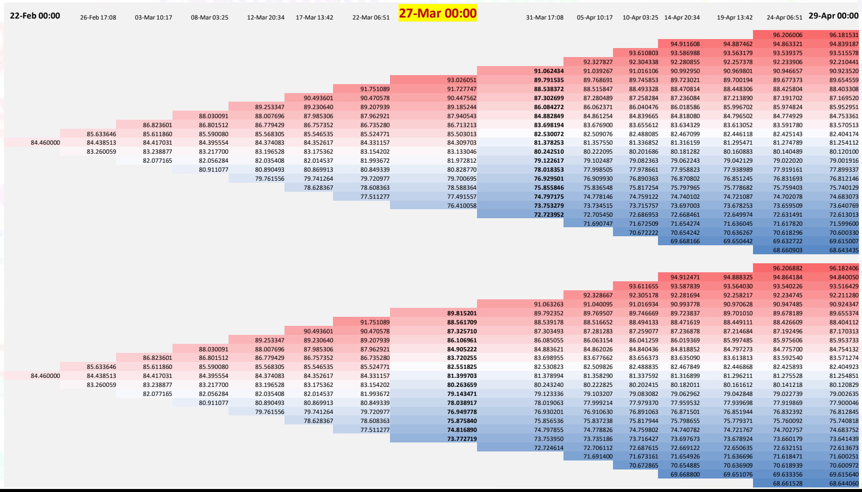
- Price differ by 2.10^{-8}, for a put very OTM
- Print underlying:
- What happens?



Market	Start date	22-Feb-22
	Start price	84.46 €
	Interest rate	-1.97%
	Volatility	7.14%
	Dividend	2.91 €
	Div ex-date	27-Mar-22
Trade	Maturity	29-Apr-22
	Type	Put
	Exercise	American
	Strike	70.14 €
Model	Nb steps	14

The dividend periods differ

- The dividend is not paid on the same period!



Equality between dates (1)

- Initial date + n x time step can seem different from the ex-dividend date
- Solution: define the equality with a tolerance

```
' returns True if the div falls between the current date (excluded) and the next one (included)
Public Function IsDivAfterThisDate() As Boolean
    Dim divDate As Date
    Let divDate = Me.TheTree.Market.DivExDate
    Let IsDivAfterThisDate = Not (Me.AreEqualDates(Me.ColumnDate, divDate)) _
        And Me.ColumnDate < divDate _
        And (divDate < Me.NextDate() Or Me.AreEqualDates(divDate, Me.NextDate()))
End Function

' returns True if the two dates are closer than a small fraction of time
Function AreEqualDates(ByVal d1 As Date, ByVal d2 As Date) As Boolean
    Let AreEqualDates = Abs(d1 - d2) < 1 / Me.TheTree.nbSteps / 1000
End Function
```

Equality between dates (2)

- In Python, as a member in TruncNode:

```
class TruncNode(Node):
  # constructor with reference to the Node constructor
  def __init__(self, precNode, colDate, price, tree):
    Node.__init__(self, price, self, tree)
    self.precMid: TruncNode = precNode
    self.columnDate = colDate
    self.nextMid: TruncNode = None
    exDivDate = self.tree.mkt.divExDate
    # means self.columnDate < exDivDate <= self.nextDate()
    self.isDivInTheFollowingPeriod = not(self.areSameDates(self.columnDate, exDivDate)) \
      and self.columnDate < exDivDate \
      and (exDivDate < self.nextDate() or self.areSameDates(exDivDate, self.nextDate()))

  # this equality between dates avoids small numerical errors that make dates look different
  def areSameDates(self, d1, d2):
    return abs(d1 - d2) < datetime.timedelta(days=1) / self.tree.nbSteps / 1000
```

Price precision: lower in Python

- The largest gaps between VBA and Python are around 1.10^{-11} € on stock prices around 100 €
- Observing the transition probabilities (without dividend), we can see that Python values are 1000 times more instable with low volatilities:

Start date	08-Jun-22	
Start price	112.53	€
Interest rate	2.00%	
Volatility	**0.44%**	
Dividend	0.00	€
Div ex-date	19-Jun-22	
Maturity	08-Jul-22	
Type	Call	
Exercise	European	
Strike	107.44	€
Nb steps	6	

VBA, down probabilities

		0.166741008854**2450**
	0.166741008854**4540**	0.166741008854**2450**
0.1667410088543840	0.1667410088543840	0.1667410088543840
	0.16674100885438**60**	0.166741008854**3130**
		0.166741008854**2420**

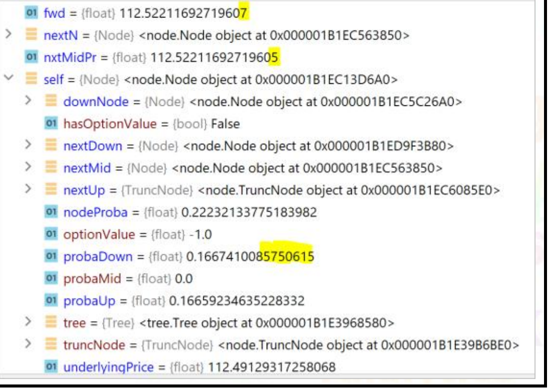
Python

		0.16674100**87146660**
	0.166741008854**4570**	0.16674100**87146660**
0.1667410088543950	0.1667410088543950	0.1667410088543950
	0.166741008854**4570**	0.16674100**85750610**
		0.166741008854**2700**

- Why?!

Analysis

- Underlying prices differ only by the last 2 digits
- The same for forward rates
- Variances are all exactly equal
- In the debugger, the only visible gap is between the **forward** and the value on the **next node**, just on the last digit
- Is it enough?



Detailed calculation

- The down probability has the following code:

```
self.probaDown = ((variance + fwd * fwd) / (nxtMidPr * nxtMidPr) – 1
- (a + 1) * (fwd / nxtMidPr - 1)) / ((1 - a) * (1 / (a * a) - 1))
```

$$p_{down} = \frac{S_{i+1,j'}^{-2}\left(V_{i+1,j} + E_{i+1,j}^{2}\right) - 1 - (\alpha + 1)\left(S_{i+1,j'}^{-1}E_{i+1,j} - 1\right)}{(1-\alpha)(\alpha^{-2} - 1)}$$

- The numerator is 2.7 10^{-7} and the denominator 1.6 10^{-6}. Therefore a small error in the numerator is magnified in p_{down}.
- The solution is to use the simplified formula when there is no dividend (store 3 values in tree)

Other precision points

- It also helps to rephrase the math formula:

$$\frac{V_{i+1,j} + E_{i+1,j}^2}{S_{i+1,j'}^2} - 1$$

- It is less precise when the next node and forward values are close than this variation:

$$\frac{V_{i+1,j} + (E_{i+1,j} + S_{i+1,j'})(E_{i+1,j} - S_{i+1,j'})}{S_{i+1,j'}^2}$$

- Ditto for the other term:

$$\frac{E_{i+1,j}}{S_{i+1,j'}} - 1 = \frac{E_{i+1,j} - S_{i+1,j'}}{S_{i+1,j'}}$$

Organization

- For questions, please use Moodle **Forum**.
- Please post your spreadsheet on Moodle before Sunday 2nd of November
- Please include your name in the files, like this:
  - Folder_Anna-M_Julien-P…
    - Anna-M_Julien-P_...xlsm
    - Anna-M_Julien-P_...: Word or pdf file
    - main.py,
    - tree.py, etc.

10/4/2025 21:50:02