

Projet VBA-Python : Pricing d'options avec arbre trinomial

Ce projet a été réalisé dans le cadre du cours « Programmation comparée VBA-Python » du master 272 Ingénierie Economique et Financière de l'université Paris-Dauphine par :

- Thibault Charbonnier
- Baptiste Dufour

Il a consisté en la création d'un Pricer d'option via un arbre trinomial comprenant les capacités suivantes :

- Pricer des options Européennes, Américaines, Bermudiennes et Digitales
- Prendre un compte un versement de dividende du sous-jacent
- Fournir une interface graphique pour chaque projet (Python & VBA)
- Lier Python et Excel via Xlwings

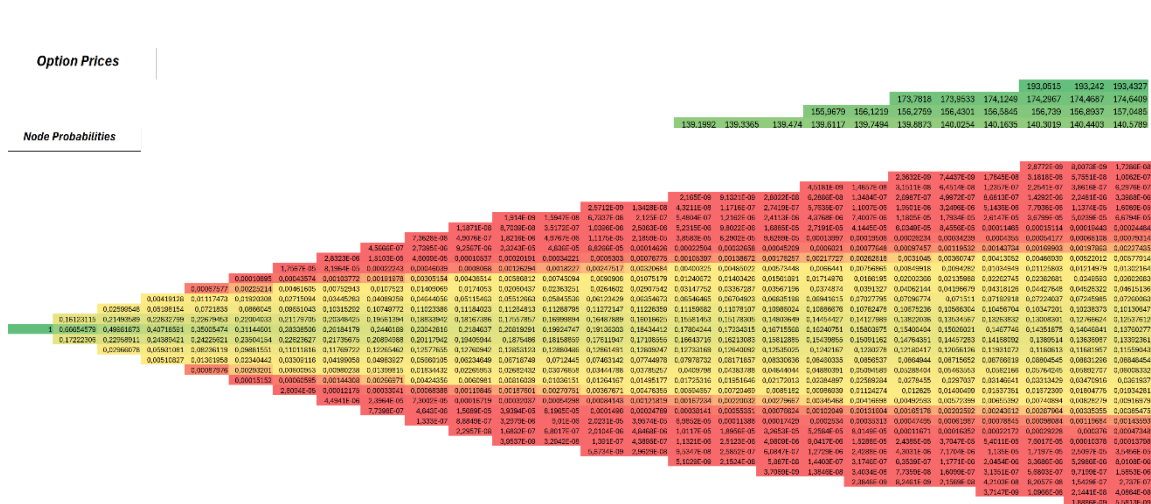
Dans ce document nous aborderons quelques points que nous considérons intéressants, principalement les graphiques qui permettent d'apprécier nos résultats visuellement et les axes d'amélioration supplémentaires que nous avons apportés.

Le projet VBA

Notre projet VBA se décompose en 7 feuilles :

- L'interface de pricing avec les paramètres et les résultats.
- 3 feuilles de plot : prix du sous-jacent, prix de l'option, probabilités des nœuds.
- 1 feuille contenant la distribution des Grecques selon le spot.
- 2 feuilles pour étudier deux phénomènes : la convergence et le gap.

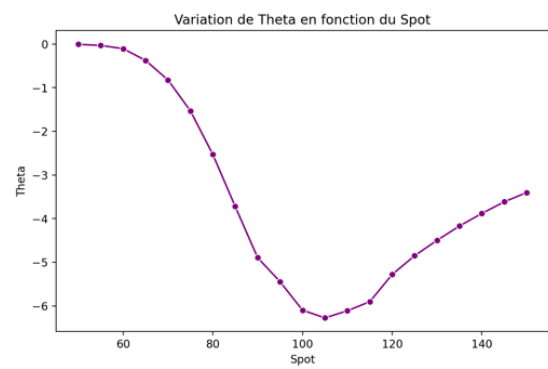
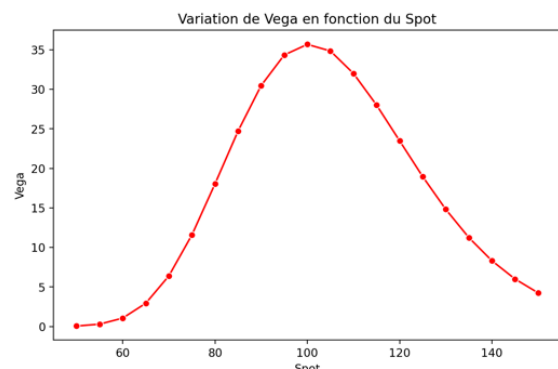
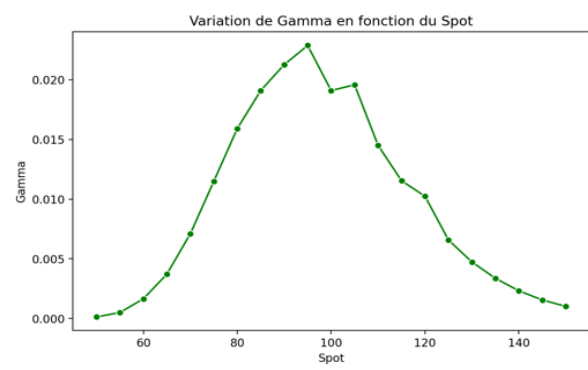
Plots intéressants :

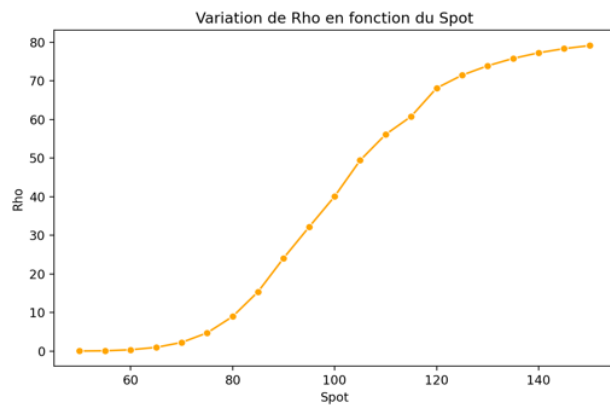


[illegible]

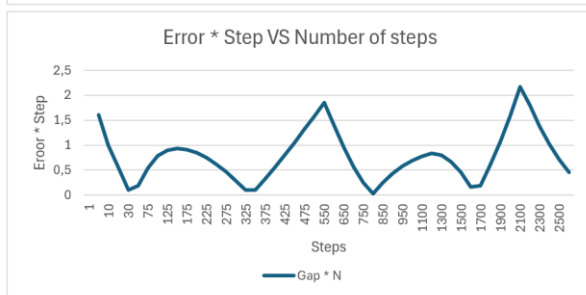
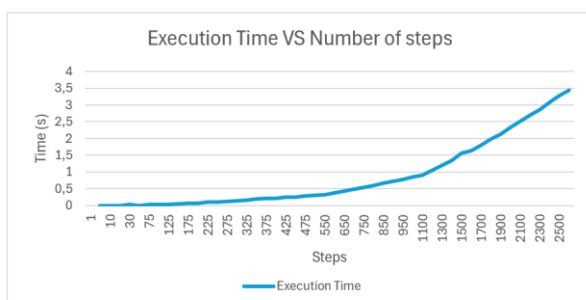
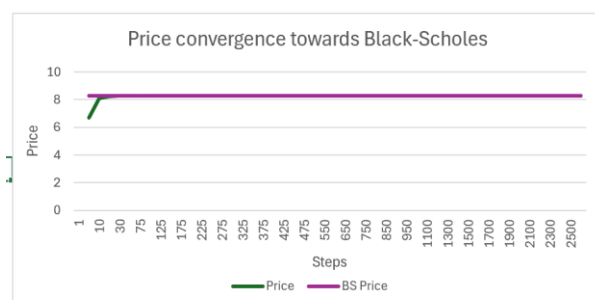
The graph illustrates the relationship between Delta and Spot. The x-axis, labeled 'Spot', ranges from 60 to 140. The y-axis, labeled 'Delta', ranges from 0.0 to 1.0. The curve shows Delta increasing as Spot increases, starting near 0.0 at Spot 50 and reaching 1.0 at Spot 150.

Spot	Delta
50	0.00
60	0.01
70	0.04
80	0.15
90	0.37
100	0.57
110	0.75
120	0.90
130	0.94
140	0.97
150	0.99

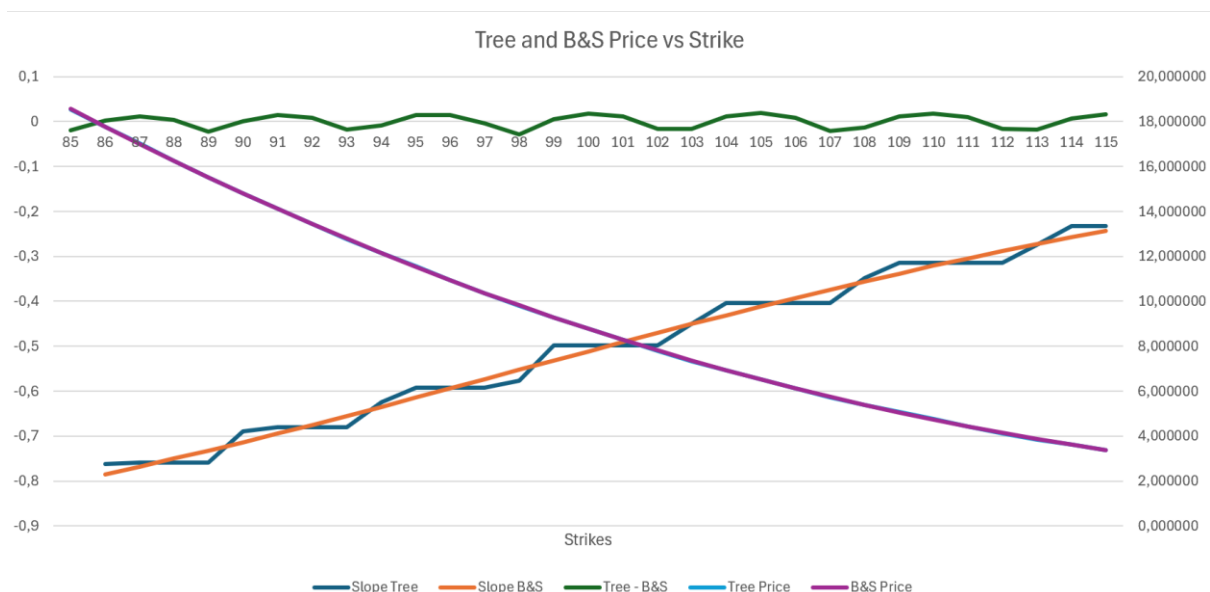




Convergence :



Analyse des gap et pentes :



Calcul des grecques :

On calcule les grecques par différences finies et on compare avec les grecques de Black & Scholes.

	Greeks Tree	Greeks B&S
Delta Δ	0,570	0,500
Gamma Γ	0,020	0,022
Vega v	35,666	36,168
Theta Θ	-5,813	-5,879
Rho P	40,365	40,496

Nombre de step en fonction de la précision :

On a également ajouté la possibilité de déterminer le nombre de step de l'arbre en fonction d'une précision voulue.

Sur notre feuille, on définit l'erreur relative comme :

$$\frac{Gap}{S_0} \approx \frac{1}{N} \frac{3}{8\sqrt{2\pi}} \frac{\sigma^2}{\sqrt{e^{\sigma^2} - 1}}$$

Et la précision comme étant l'inverse de l'erreur relative.

On a alors la relation suivante qui nous permet d'avoir le nombre de step :

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

Sur l'interface on laisse l'utilisateur choisir :

Mode (*)	Precision	Mode (*)	Step
NbSteps (**)	81	NbSteps (**)	1000
Precision	3000	Precision	0

Le projet Python

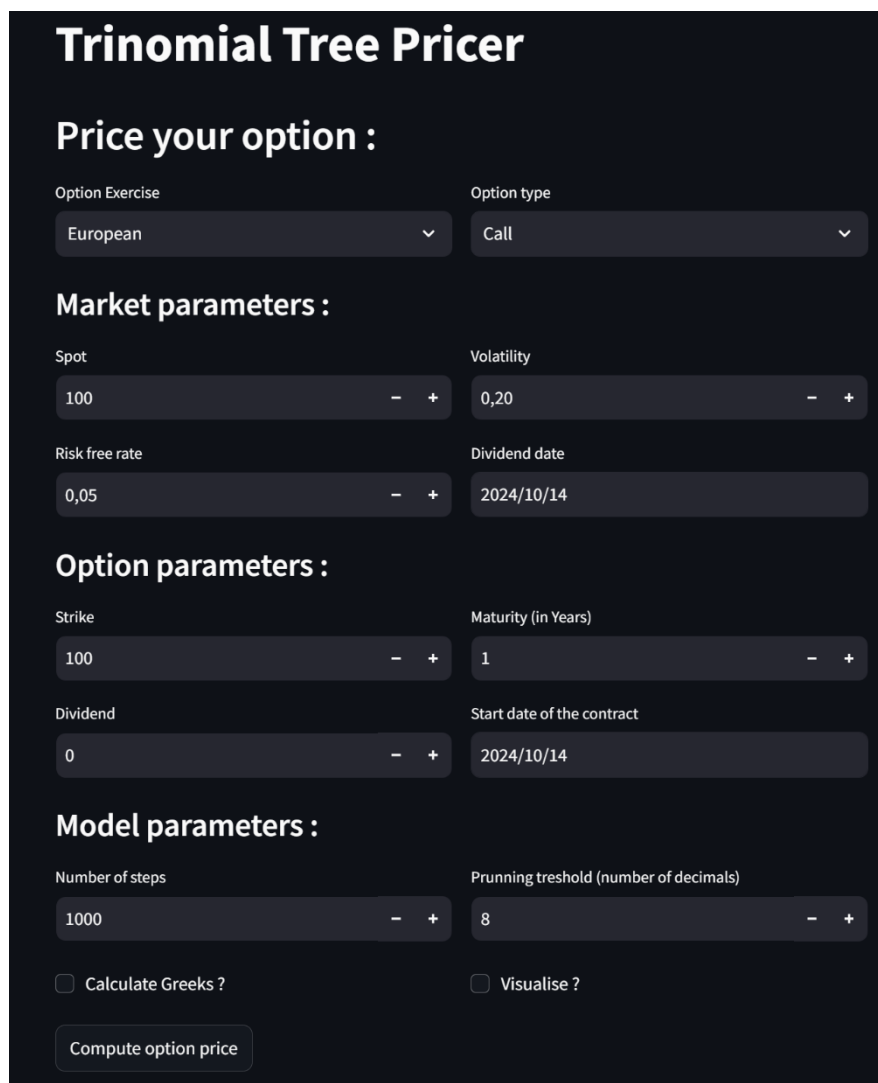
Pour réaliser ce projet en Python, voici les librairies utilisées :

Matplotlib (Visualisation), Scipy (Loi normale), Pandas (Dataframes), Streamlit (Interface Graphique).

Interface

Nous avons créé une interface graphique avec le package Streamlit en 100% Python afin que l'utilisateur puisse interagir avec le Pricer. **Pour lancer l'interface, entrez la commande « `streamlit run app.py` » dans le cmd à la racine du projet.**

- Paramétrisation sur l'interface :



Trinomial Tree Pricer

Price your option :

Option Exercise: European ▾ Option type: Call ▾

Market parameters :

Spot: 100 - + Volatility: 0,20 - +

Risk free rate: 0,05 - + Dividend date: 2024/10/14

Option parameters :

Strike: 100 - + Maturity (in Years): 1 - +

Dividend: 0 - + Start date of the contract: 2024/10/14

Model parameters :

Number of steps: 1000 - + Pruning treshold (number of decimals): 8 - +

☐ Calculate Greeks ? ☐ Visualise ?

Compute option price

- **Résultats sur l'interface :**

Results

Pricing :

Option price : 10.45139

VS Close formula price : 10.45058

Performance :

Tree generated in : 0.647 sec

Option priced in : 0.086 sec

Total time : 0.733 sec

Option Greeks	Value
Delta	0.6462
Gamma	0.0186
Vega	37.5589
Theta	6.4001
Rho	53.0875

XLwings

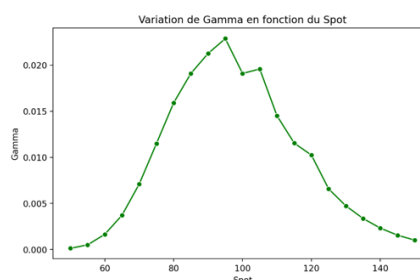
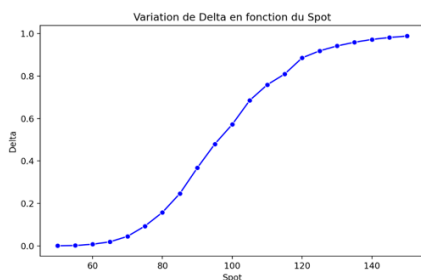
Nous avons également lié notre projet python au projet Excel via XLwings.

Nous avons donc créé 4 subroutines sous python et accessibles sous Excel permettant de :

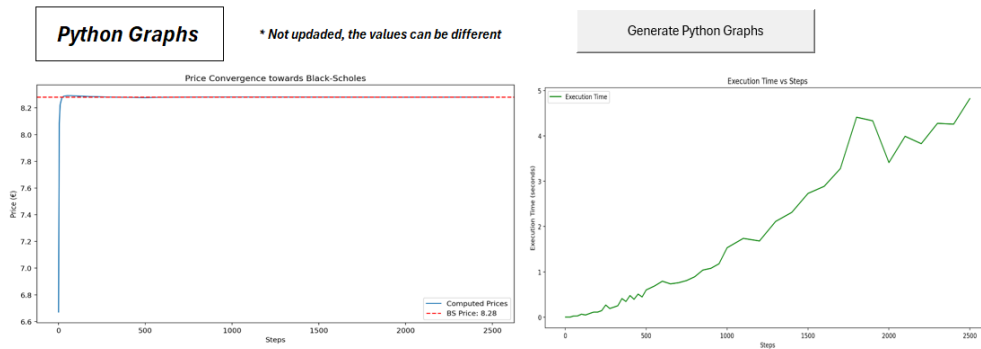
- Pricer une option européenne ou américaine avec dividende.
- Pricer une option avec la méthode de libération de l'allocation de mémoire.
- Générer la distribution des Grecs selon le spot.
- Générer les graphiques de convergence de la même manière que Excel mais via python.

Voici comment nous avons intégré sous Excel ses options.

Python	Tree Price	8.282	Price Python
	Tree Generation	0.77	
	Option Pricing	0.13	
	Total Time	0.90	Tree Memory Allocation Price (No div)
	Memory Alloc Price	8.282	
	Memory Alloc Time	0.77	



Génération des graphiques des grecs



Afin de récupérer les données utilisateurs, nous avons créé des fonctions spéciales récupérant les données de l'excel via python.

Dans ce projet nous avons intégré des fonctionnalités supplémentaires comme le calcul d'une option sans dividende avec la **méthode de la libération de la mémoire** et la création de graphiques permettant de visualiser la distribution des Grecs selon le spot.

Types d'option supplémentaires

Nous avons ajouté quelques fonctionnalités additionnelles à notre pricer.

Par exemple, nous avons ajouté deux types d'options supplémentaires :

- Options Bermudiennes :

De la même manière que les options Américaines, on doit pouvoir exécuter l'option avant la maturité.

Pour gérer la possibilité d'exécution, on définit une liste de time step où l'option peut être exécutée de la manière suivante :

```
@cached_property
def exercise_steps(self) -> list[int]:
    """
    Définit une liste qui contient tous les timeSteps où l'option peut être exécutée
    """
    if isinstance(self.option, BermudeanCallOption) or isinstance(self.option, BermudeanPutOption):
        exercise_dates = self.option.exercise_dates
        time_delta_in_days = self.time_delta * 365
        return [ceil((ex_date - self.option.start_date).days/time_delta_in_days) for ex_date in exercise_dates]

    elif isinstance(self.option, AmericanCallOption) or isinstance(self.option, AmericanPutOption):
        return [i for i in range(self.nb_steps)]

    else:
        return []
```

Sur l'interface, on rajoute le choix des dates d'exécution :

Add exercise date

Exercise date 1

2024/10/17

Exercise date 2

2025/01/09

- Options Digitales :

Pour l'option digitale, on change le payoff de l'option :

```
@dataclass
class DigitalCallOption(Option):
    coupon : float = 1

    def payoff(self, price : float) -> float:
        if price > self.strike:
            return self.coupon
        else:
            return 0
```

```
@dataclass
class DigitalPutOption(Option):
    coupon : float = 1

    def payoff(self, price : float) -> float:
        if price < self.strike:
            return self.coupon
        else:
            return 0
```

Sur l'interface, on ajoute le choix du « coupon » de l'option digitale :

Digital coupon

1 - +