Pricing d’Options par un Arbre Trinomial

# Description

Le pricer a été développé en python puis traduit en vba suivant la même méthode. Le code comporte 3 scripts principaux :

* pricer.py : les classes Underlying et Option avec des enums pour les types d’exercice et d’option
* node.py : les classes Node et Proba
* tree.py : la classe TrinomialTree

# Test du Pricer

On observe bien la convergence vers le prix de Black-Scholes dans le cas d’une option européenne sans dividende.



L’écart avec B-S en fonction du nombre de pas et du strike



On observe que le prix des options Européennes et souvent inférieur à celui des Américaines.

# Extensions

* On a ajouté le pruning pour accélérer (considérablement) le temps de calcul
* Codage d’une méthode pour représenter l’arbre en python avec un jeu de couleur sur la valeur de l’option à chaque nœud et un autre sur la taille des noeuds en fonction de la probabilité d’atteindre le nœud
* En python, on peut au lieu d’entrer le nombre de pas, assigner le niveau de précision souhaité (i.e. l’erreur acceptée) dans l’objet Option

# Remarques

On peut grâce à xlwings utiliser la fonction « pricer\_trino » contenu dans le fichier trinomial\_option\_pricer.py dans Excel.

Le graphique généré par python de l’arbre repose sur l’utilisation des librairies networkx et plotly. En effet, on convertit l’arbre en graphe via networkx pour pouvoir utiliser des fonctions que nous avions déjà pour représenter des graphes avec plotly. On a jugé bon de pouvoir convertir notre arbre en un graphe via une librairie tierce, la représentation étant une extension.

La méthode suivie pour la construction ne se base pas sur l’hypothèse d’une suite géométrique dans les prix des colonnes de l’arbre, si bien que nous avons un calcul de probabilités un peu différent (voir détail du calcul). Néanmoins, cela nous permet de ne pas faire de recherche dans le branchement du nœud le plus proche lors du détachement d’un dividende.

Le code traduit en vba supporte jusqu’à 500 pas, passé ce seuil le stack mémoire devient insuffisant, sans doute à cause d’un nombre d’appels de fonction trop grand entre les modules. Malheureusement, le code fonctionnant très bien en python, nous n’avons pas pris le temps de résoudre le problème en vba.

De plus, certaines fonctionnalités comme la représentation graphique ou la possibilité de donner la précision en argument ont été implémentées en python mais pas en vba.