**Rapport : Projet arbre trinomial Python/VBA**

*Apprentissage et application de la POO :*

L'apprentissage de la POO s'est avéré être un processus exigeant, nécessitant une compréhension approfondie de concepts tels que les classes, les attributs, et les méthodes, ainsi que leur organisation logique dans un programme. Le défi résidait dans le choix judicieux des attributs et des méthodes pour chaque classe, tout en veillant à leur pertinence et efficacité dans le cadre du projet.

*Création des classes clés :*

Pour la structure de l'arbre trinomial, j'ai dû concevoir quatre classes principales : marché, modèle, option et nœud. Chacune de ces classes avait un rôle spécifique et devait être définie pour refléter les aspects distincts de l'arbre trinomial. Ce processus impliquait de déterminer les attributs essentiels pour chaque classe et les méthodes nécessaires pour leur interaction. Au fur et à mesure de l'avancement du projet, j'ai pu constater les avantages substantiels de la POO. Non seulement elle a permis une meilleure organisation et structuration du code, mais elle a également offert une plus grande clarté dans la manipulation des différents objets du modèle.

*Approche initial et difficultés avec le dividende :*

Dans la première phase de mon projet, j'avais développé une version personnalisée de l'arbre trinomial, en m'écartant des directives fournies en cours. Bien que cette approche originale ait donné des résultats satisfaisants pour le pricing des options hors dividendes, elle s'est avérée inadaptée lors de l'introduction des dividendes dans le modèle.

*Réorientation vers une méthode éprouvée :*

Confronté au défi posé par l'intégration des dividendes, j'ai pris la décision de réorienter mon approche. Cela impliquait de reprendre le développement de l'arbre trinomial depuis le début, en suivant plus fidèlement les méthodes et les concepts enseignés en classe. Bien que cette réinitialisation ait demandé un effort supplémentaire, elle s'est avérée être très utile car cela m’a permis de comprendre pleinement chacune difficultés du projet.

*Implémentation efficace des dividendes :*

Une fois le modèle réorienté, l'intégration des dividendes est devenue nettement plus aisée. L'élément clé de cette réussite a été l'adoption de la méthode « nextMid », essentielle pour lier correctement les noeuds de l'arbre en tenant compte des dividendes. Cette méthode a permis d'attacher chaque noeud au bon prix forward ajusté par les dividendes, garantissant ainsi l'exactitude des calculs.

*Problèmes initiaux de performance :*

Au début de mon projet, l'un des défis majeurs que j'ai rencontrés concernait la performance du code, particulièrement en termes de vitesse et d'efficacité. Même avec la première version de mon arbre trinomial, que j'avais conçu de manière indépendante, je faisais face à une lenteur notable dans l'exécution, ce qui limitait ma capacité à étendre le modèle, surtout pour un nombre élevé de pas.

*Amélioration notable des performances :*

En adoptant la structure et les algorithmes recommandés en cours, j'ai réussi à optimiser considérablement la performance de mon code. Cette amélioration m'a permis de simuler des arbres trinomiaux jusqu'à 5000 pas, un grand progrès vis à vis de la version initiale de mon modèle qui planchait à 900 pas, ce qui m’a rassuré quant à mon choix de reprendre mon code du départ.

*Approche initiale des dates :*

Lors de la phase initiale de mon projet, j'avais l'intention d'implémenter un système de dates sophistiqué en utilisant la bibliothèque DateTime de Python. L'idée était de lier chaque étape de mon arbre trinomial à une date spécifique, créant ainsi une correspondance directe entre le temps écoulé et la progression de l'arbre. Tout semblait fonctionner correctement jusqu'à ce que je rencontre un obstacle majeur : lorsque le nombre de pas de l'arbre dépassait le nombre de jours entre la date de départ et la date de maturité.

*Confrontation à un problème inattendu :*

Ce problème s'est manifesté sous la forme d'une extension incorrecte de l'arbre au-delà de la date de maturité, chaque pas supplémentaire se traduisant par un jour de plus, même après avoir atteint la date de maturité. Cette anomalie a entraîné des résultats aberrants, et il m'a fallu un certain temps pour identifier la source de cette erreur. Mon souhait de voir chaque colonne de l'arbre représentée par une date précise s'est avéré être un point de friction inattendu dans le modèle.

*Retour à une méthode vu en cours :*

Après avoir évalué les problèmes liés à cette approche de gestion des dates, j'ai finalement décidé de revenir à la méthode enseignée en cours. Cela impliquait de renoncer à associer directement les dates à chaque colonne de l'arbre et d'adopter à la place une représentation basée sur un pourcentage de progression entre la date de départ et la date de fin. Cette modification a permis d'éliminer les incohérences rencontrées et d'aligner le modèle sur une logique plus robuste et fiable.

*Évolution de la méthode de calcul des probabilités :*

Le calcul initial des probabilités se basait sur une analyse rétrospective des nœuds, ce qui alourdissait le code car chaque nœuds devait connaître les nœuds qui « l’alimentait ». En adoptant une méthode de propagation des probabilités de transition, j'ai pu optimiser la performance du modèle, réduisant ainsi la complexité et améliorant l'efficacité du calcul des probabilités. Ainsi j’ai pu aborder plus sereinement le « pruning »

*Gestion des dividendes élevés et validation des dates de dividende :*

J'ai identifié et résolu un problème où un dividende trop élevé entraînait des prix forward négatifs. De plus, j'ai implémenté une vérification pour s'assurer que la date du dividende se situe entre la date de départ et la date de maturité, soulignant l'importance de la cohérence des données dans la modélisation financière.

*Initialisation de l'arbre trinomial et gestion des erreurs :*

La complexité de l'initialisation de l'arbre, où un noeud racine reçoit un arbre en attribut et vice versa, a été résolue en utilisant un paramètre optionnel. Cette méthode a facilité la construction de l'arbre tout en évitant les problèmes d'initialisation circulaire. La mise en place de mécanismes de gestion des erreurs a renforcé la robustesse et la fiabilité du code.

*Défis de visualisation de l'arbre avec Python :*

La visualisation de l'arbre trinomial a été explorée à travers les bibliothèques Python Matplotlib et NetworkX. Chaque bibliothèque présentait des avantages et des inconvénients en termes de performance et de qualité visuelle. Finalement, j'ai opté pour Matplotlib pour sa clarté visuelle, malgré ses limitations de performance, et j'ai désactivé la visualisation pour les arbres avec un grand nombre de pas.

*Calcul difficile de Gamma et Theta :*

Dans le cadre de mon projet sur l'arbre trinomial pour le pricing d'options, j'ai été confronté à une difficulté notable dans le calcul de certaines grecques, en particulier Gamma et Theta. Bien que le calcul de Delta se soit avéré précis, les tentatives de calcul de Gamma et Theta ont révélé des incohérences. Il s’est finalement avéré que mon gamma était faux car mon epsilon de prix était trop petit au départ 0.01€ et donc cela ne permettait pas d’obtenir une bonne dérivée seconde.

*Graphique du temps d’exécution :*

J’ai tracé un graphique permettant de donné l’écart de prix entre le prix obtenu par l’arbre trinomial et le prix de Black-Scholes multiplié par le nombre de pas en fonction du nombre de pas. On retrouve bien le « tunnel » dans lequel oscille notre courbe. Cela indique bien que notre pricing se précise avec le nombre de pas. On montre aussi sur un second graphique que le logarithme du temps d’éxécution est une fonction affine du logarithme du nombre de pas.

A graph showing a number of points

Description automatically generated with medium confidence

A graph with a red line

Description automatically generated

*Graphique écart de prix en fonction du strike :*

Tout d’abord, malgré tous les essais, je ne parviens pas à obtenir le même graphique que celui attendu. Le mien montre que lorsque notre option est « at the money » il est moins précis que lorsque le prix spot se trouve loin du strike. Les oscillations sont très fortes et très concentré.

A graph of a graph showing a number of points

Description automatically generated with medium confidence