École Nationale Des Sciences Appliquées
Agadir

Application de la Méthode Monte Carlo à la Finance : Options et Risques de Portefeuille

---

Documentation

Élève :
Salma Elaissari

Enseignant :
Prof.BrahimEl Asri

Filière:
FID

3 février 2025

Résumé

Ce rapport est une documentation technique et analytique approfondie qui détaille la conception, l'implémentation et les résultats d'une application dédiée à l'application de la méthode de simulation Monte Carlo dans le domaine financier. L'objectif principal de ce projet est de fournir un outil performant et interactif pour répondre aux besoins complexes de la finance moderne, notamment la tarification des options, l'analyse de portefeuille et la gestion des risques. La simulation Monte Carlo, en tant que méthode numérique puissante, permet de modéliser l'incertitude et la variabilité des marchés financiers en générant un grand nombre de scénarios possibles pour les prix des actifs, offrant ainsi une base solide pour la prise de décision dans des environnements incertains.
Il vise à fournir une compréhension approfondie des concepts mathématiques, des choix technologiques et des fonctionnalités pratiques de l'application, tout en mettant en lumière son utilité pour la tarification des options, l'analyse de portefeuille et la gestion des risques. Structuré en plusieurs sections, le rapport aborde successivement l'architecture générale de l'application, les fondements théoriques de la simulation Monte Carlo, les modèles mathématiques utilisés (tels que le mouvement brownien géométrique), les techniques d'optimisation des calculs, et les méthodes de visualisation des résultats.
Enfin, le rapport présente des études de cas concrets pour illustrer l'efficacité de l'application dans des scénarios financiers réels. Ces exemples démontrent comment la simulation Monte Carlo peut être utilisée pour évaluer le risque d'un portefeuille, optimiser la répartition des actifs et prédire les performances futures. Le rapport conclut par une discussion sur les avantages et les limites de la méthode, ainsi que des perspectives d'amélioration, telles que l'intégration de modèles plus complexes ou l'utilisation de l'apprentissage automatique pour affiner les prédictions.

Ce document s'adresse à un public varié, incluant les professionnels de la finance cherchant à mieux comprendre les outils numériques disponibles, les développeurs techniques intéressés par les aspects d'implémentation, et les chercheurs souhaitant explorer les applications pratiques des méthodes numériques en finance. En combinant rigueur scientifique, explications accessibles et exemples concrets, ce rapport vise à être une ressource précieuse pour tous ceux qui s'intéressent à l'intersection entre la finance et les technologies numériques.

Table des matières

1 Introduction :

La méthode de Monte-Carlo est une technique mathématique et statistique qui utilise des échantillons aléatoires pour estimer des résultats numériques dans des situations d'incertitude. Son nom provient du célèbre casino de Monte-Carlo, en raison de l'élément de hasard qui est central à cette approche.

1.1 Définition et Principe

La méthode de Monte-Carlo consiste à simuler un grand nombre de scénarios aléatoires pour modéliser des systèmes complexes ou évaluer des probabilités. En générant des échantillons aléatoires, elle permet d'obtenir une distribution de résultats possibles et d'estimer la probabilité associée à chacun d'eux. Par exemple, dans le cadre d'une simulation, on peut estimer la probabilité d'un événement en répétant l'expérience un grand nombre de fois et en observant la fréquence des résultats. Cette méthode est un outil puissant pour l'analyse quantitative dans divers domaines, permettant d'évaluer des scénarios complexes sous incertitude. Sa capacité à fournir une gamme de résultats basés sur des simulations répétées en fait une technique précieuse pour la prise de décision. Les applications de la méthode de Monte-Carlo sont variées et incluent :

- **Calculs financiers :** Estimation du risque et évaluation d'options financières.
- **Physique :** Modélisation de phénomènes physiques complexes, comme les interactions moléculaires.
- **Ingénierie :** Analyse de la fiabilité des systèmes et optimisation des processus.
- **Statistiques :** Estimation d'intégrales multiples et résolution de problèmes qui seraient autrement intractables par des méthodes analytiques

1.2 Objectifs du rapport

L'objectif principal de cette étude est d'explorer les fondements théoriques de la méthode de Monte Carlo en mettant en lumière ses bases mathématiques et probabilistes, ses principes de fonctionnement, et ses avantages dans les contextes complexes. Par la suite, cette recherche sera illustrée par un exemple pratique dans le domaine financier, portant sur l'évaluation des risques. Enfin, une interface interactive sera développée pour démontrer la simulation Monte Carlo, rendant la méthode accessible et visuellement compréhensible.

Ce double objectif permet de combiner une analyse théorique rigoureuse avec une application concrète et utile dans la finance.

- **Objectifs Théoriques :**
  - Explorer les fondements mathématiques de la méthode Monte Carlo.
  - Identifier les forces et les limites théoriques.
  - Établir un cadre général d'application.
- **Objectifs Pratiques :**
  - Appliquer la méthode Monte Carlo à un exemple concret en finance
  - Construire une interface interactive pour la simulation

- **Objectifs Méthodologiques :**
  - Combiner théorie et pratique
  - Concevoir un processus reproductible

1.3 Description du projet

Ce projet développe une application interactive qui intègre une simulation Monte Carlo pour la tarification des options (y compris les options exotiques) et l'analyse de portefeuille en utilisant des données financières en temps réel de Yahoo Finance. L'application est structurée avec une interface utilisateur soignée via Dash Bootstrap Components, permettant aux utilisateurs d'entrer des paramètres personnalisés pour les simulations et d'obtenir des visualisations interactives des trajectoires de prix et des métriques de risque (VaR, CVaR, ratio de Sharpe).

- **Fonctionnalités du programme** :
  - **Simulation Monte Carlo pour la tarification des options :**
    - Tarification d'options vanilles (Call, Put) et exotiques (Asiatiques, Barrières, Lookback, etc.).
    - Génération de multiples trajectoires de prix d'actifs sous-jacents selon un modèle de diffusion géométrique brownien (GBM).
    - Calcul de l'espérance du prix d'option avec intervalle de confiance.
    - Affichage des trajectoires simulées sous forme de graphique interactif.
  - **Analyse de portefeuille :**
    - Importation de données financières en temps réel depuis Yahoo Finance.
    - Estimation du Value at Risk (VaR) et Conditional Value at Risk (CVaR) via simulation Monte Carlo.
    - Calcul du ratio de Sharpe pour mesurer la rentabilité ajustée au risque.
    - Visualisation des performances et du risque du portefeuille avec des graphiques interactifs.
  - **Interface utilisateur interactive (Dash & Bootstrap :**
    - Sélection dynamique des actifs financiers via un champ de recherche (Yahoo Finance).
    - Entrée des paramètres pour la simulation (nombre de simulations, maturité, volatilité, etc.).
    - Génération de rapports graphiques et tableaux de résultats.

2 Explication de la méthode :

2.1 Explication détaillée :

2.1.1 Principe de fonctionnement :

La méthode de Monte-Carlo est une technique statistique qui utilise l'échantillonnage aléatoire pour estimer des résultats dans des situations d'incertitude. Son principe de fonctionnement consistera dans :

— **Échantillonnage Aléatoire :** La méthode repose sur la génération de nombres aléatoires pour simuler différents scénarios possibles. Ces scénarios sont utilisés pour représenter les incertitudes inhérentes aux variables d'entrée d'un modèle mathématique.

— **Modélisation Mathématique :** Un modèle mathématique est établi pour relier les variables d'entrée (facteurs influents) aux variables de sortie (résultats). Ce modèle peut être une simple équation ou un système complexe d'équations.

— **Simulation Répétée :**La méthode consiste à exécuter un grand nombre de simulations (parfois des milliers ou des millions) en utilisant des valeurs d'entrée générées aléatoirement. Chaque simulation produit un résultat qui contribue à une distribution globale des résultats possibles.

— **Analyse des Résultats :**À l'issue des simulations, les résultats sont analysés pour estimer la probabilité de différents résultats et pour déterminer les intervalles de confiance. Cela permet d'obtenir une vue d'ensemble des risques et incertitudes associés à la situation modélisée.

En résumé, la méthode de Monte-Carlo transforme l'incertitude en informations exploitables grâce à une approche systématique d'échantillonnage et de simulation, permettant ainsi d'estimer des résultats dans des contextes variés.

2.1.2 Avantages de la méthode :

La méthode de Monte-Carlo présente plusieurs avantages significatifs qui en font un outil précieux dans divers domaines d'application. Voici les principaux avantages :

— **Flexibilité :** La méthode de Monte-Carlo peut modéliser une large gamme de systèmes complexes, y compris ceux avec des relations non linéaires et des variables aléatoires. Elle s'adapte facilement à différents types de modèles et d'hypothèses, ce qui permet une personnalisation selon les besoins spécifiques de l'analyse.

— **Gestion de l'incertitude :**Contrairement aux méthodes déterministes qui supposent des valeurs fixes pour les variables, la méthode de Monte-Carlo intègre des distributions de probabilité pour les variables d'entrée. Cela permet de capturer l'incertitude et la variabilité inhérentes aux systèmes analysés, offrant ainsi une représentation plus réaliste des scénarios possibles.

— **Génération de scénarios multiples :**En exécutant un grand nombre de simulations (des milliers ou millions), la méthode fournit une vue d'ensemble des résultats possibles. Cela aide à évaluer la probabilité de divers résultats et à comprendre les risques associés.

— **Analyse des risques :**La méthode est particulièrement efficace pour l'analyse des risques financiers et opérationnels. Elle permet d'identifier les impacts potentiels des différentes incertitudes sur les résultats, facilitant ainsi la prise de décisions éclairées.

— **Précision accrue :**À mesure que le nombre d'échantillons augmente, la précision des estimations s'améliore grâce à la loi des grands nombres. Cela signifie que les résultats obtenus deviennent plus fiables avec un échantillonnage suffisant.

— **Applications variées :**La méthode est utilisée dans divers domaines tels que la finance, l'ingénierie, la gestion de projet, et même en sciences physiques. Sa polyvalence en fait un outil adapté à de nombreux types d'analyses et simulations.

En résumé, la méthode de Monte-Carlo est un outil puissant pour modéliser l'incertitude et analyser des systèmes complexes, offrant flexibilité, précision et une meilleure compréhension des risques associés aux décisions prises dans divers contextes.

2.1.3 Inconvénients de la méthode :

La méthode de Monte-Carlo, bien qu'elle soit une technique puissante pour l'analyse statistique et la modélisation d'incertitudes, présente également plusieurs inconvénients. Voici les principaux défis associés à son utilisation :

— **Dépendance aux Données d'Entrée :** La précision des résultats de la méthode de Monte-Carlo dépend fortement des données d'entrée et des distributions de probabilité choisies. Si ces données sont incorrectes ou mal spécifiées, les résultats peuvent être trompeurs, ce qui compromet la fiabilité des conclusions tirées.

— **Exigences en Calcul :** La méthode nécessite souvent des ressources informatiques importantes, surtout lorsqu'il s'agit de simuler un grand nombre de scénarios. Les simulations peuvent prendre beaucoup de temps, ce qui peut être un obstacle dans des contextes où des résultats rapides sont nécessaires.

— **Complexité du Modèle :** Pour des modèles complexes, la mise en œuvre de la méthode peut nécessiter une expertise approfondie en mathématiques et en statistiques. Cela peut limiter son utilisation par des analystes qui ne possèdent pas les compétences requises.

— **Erreurs d'Échantillonnage :**Étant donné que la méthode repose sur l'échantillonnage aléatoire, il existe toujours un risque d'obtenir des résultats aberrants ou non représentatifs, surtout si le nombre de simulations est insuffisant. Cela peut mener à une estimation incorrecte des probabilités ou des résultats.

— **Problèmes avec les Événements Rares :** La méthode de Monte-Carlo peut être inefficace pour estimer la probabilité d'événements rares, car elle nécessite un nombre très élevé de simulations pour obtenir une estimation précise. Cela peut rendre le processus long et coûteux.

— **Biais Potentiel des Générateurs Aléatoires :**Si le générateur de nombres aléatoires utilisé est biaisé ou peu fiable, cela peut affecter la qualité des simulations. Un générateur qui produit des nombres non uniformément distribués peut conduire à des résultats erronés.

En résumé, bien que la méthode de Monte-Carlo soit un outil précieux pour l'analyse statistique et la modélisation sous incertitude, ses inconvénients doivent être soigneusement considérés pour garantir l'exactitude et la fiabilité des résultats obtenus.

2.1.4 Simulation de la méthode de Monte Carlo en Finance :

2.1.4.1 Exemples d'utilisations dans la finance : La simulation de Monte-Carlo est une technique largement utilisée en finance pour modéliser l'incertitude et évaluer les risques associés à divers investissements. Elle est largement utilisée dans le domaine financier à diverses fins, notamment :

— **Analyse des Risques :**La simulation de Monte Carlo est utilisée pour évaluer et quantifier les risques financiers. En simulant différents scénarios de marché et en générant plusieurs résultats, elle permet d'estimer l'éventail des rendements ou des pertes d'un portefeuille. Elle aide les analystes à évaluer la probabilité de différents niveaux de risque, tels que la Value-at-Risk (VaR) ou l'Expected Shortfall (ES), et à comprendre la distribution des résultats possibles.

— **Optimisation de Portefeuille :** La simulation de Monte Carlo peut être utilisée pour optimiser des portefeuilles d'investissement en tenant compte de différentes stratégies d'allocation d'actifs. En simulant la performance de différentes compositions de portefeuille et méthodes de rééquilibrage, elle aide les investisseurs à déterminer l'allocation optimale qui maximise les rendements tout en gérant les risques dans les contraintes spécifiées. Elle permet d'évaluer la probabilité que le portefeuille atteigne un certain niveau de valeur à la retraite.

— **Tarification des Options :** La simulation de Monte Carlo est utilisée dans les modèles de tarification des options, tels que le modèle de Black-Scholes, pour estimer la valeur des options et d'autres titres dérivés. En simulant les trajectoires des prix des actifs sous-jacents, elle aide à calculer la distribution de probabilité des paiements des options et à déterminer les prix justes des options.

— **Évaluation du Risque de Crédit :** La simulation de Monte Carlo est utilisée pour évaluer le risque de crédit en simulant les résultats potentiels des prêts ou des portefeuilles de crédit. Elle aide à estimer la probabilité de défaut, la perte en cas de défaut, et d'autres indicateurs liés au crédit. En incorporant des facteurs tels que les corrélations de défaut, les taux de récupération et les variables macroéconomiques, elle permet de modéliser les risques des portefeuilles de crédit et de calculer les ajustements de valeur du crédit (CVA).

— **Prévisions Financières et Analyse de Scénarios :** La simulation de Monte Carlo est utilisée dans la modélisation financière pour générer des prévisions et évaluer l'impact de différents scénarios sur la performance financière. Elle permet aux analystes d'intégrer l'incertitude et la variabilité dans des paramètres clés tels que les taux de croissance des ventes, les taux d'intérêt ou les taux de change. En simulant plusieurs scénarios, elle offre des perspectives sur l'éventail des résultats possibles et aide à éclairer les décisions stratégiques.

— **Tests de Résilience (Stress Testing) :**La simulation de Monte Carlo est utilisée dans les exercices de stress testing pour évaluer la résilience des institutions financières ou des portefeuilles sous des conditions de marché extrêmes. Elle aide à évaluer l'impact de chocs sévères sur la suffisance des fonds propres, la liquidité et d'autres indicateurs de risque. En simulant un large éventail de scénarios défavorables, elle aide à identifier les vulnérabilités et à concevoir des stratégies de gestion des risques.

2.1.4.2 Matrice de Covariance : La matrice de covariance est une matrice carrée qui représente la covariance entre plusieurs variables. En finance, elle est souvent utilisée pour analyser les relations entre les rendements de différents actifs au sein d'un portefeuille. Cela permet de comprendre comment les actifs interagissent entre eux, en termes de risque et de corrélation.
La covariance entre deux variables X et Y se calcule à l'aide de la formule suivante :

$$\text{Cov}(X, Y) = E[(X - \mu_X)(Y - \mu_Y)]$$

Où :

- $\text{Cov}(X, Y)$: Covariance entre les variables X et Y,
- E : Espérance ou valeur moyenne,
- X et Y : Variables aléatoires ou séries de données,
- μ_X et μ_Y : Moyennes (ou valeurs attendues) de X et Y.

2.1.4.3 Construction de la matrice de covariance : La matrice de covariance est constituée en calculant les covariances entre toutes les paires de variables d'un ensemble de données.

Pour un ensemble de données contenant n variables, la matrice de covariance sera une matrice $n \times n$, où chaque élément $\text{Cov}(X_i, X_j)$ représente la covariance entre la $i^{\text{ème}}$ et la $j^{\text{ème}}$ variable.

La matrice de covariance peut être représentée comme suit :

$$\text{Cov}(X) = \begin{bmatrix} \text{Cov}(X_1, X_1) & \text{Cov}(X_1, X_2) & \cdots & \text{Cov}(X_1, X_n) \\ \text{Cov}(X_2, X_1) & \text{Cov}(X_2, X_2) & \cdots & \text{Cov}(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}(X_n, X_1) & \text{Cov}(X_n, X_2) & \cdots & \text{Cov}(X_n, X_n) \end{bmatrix}$$

2.1.4.4 Rendement Espéré d'un Portefeuille : Le rendement espéré d'un portefeuille est une mesure du rendement moyen anticipé qu'un investisseur peut attendre d'un portefeuille d'investissements. Il prend en compte les poids attribués à chaque actif dans le portefeuille ainsi que leurs rendements espérés respectifs.

- **Calcul du rendement espéré** Il existe deux manières principales de calculer le rendement espéré d'un portefeuille.
  - **Première méthode**

$$\text{Rendement Espéré du Portefeuille} = \sum_{i=1}^{n} \text{Poids}_i \cdot \text{Rendement Espéré}_i$$

  Où :

  - Rendement Espéré du Portefeuille : Moyenne pondérée des rendements espérés des actifs individuels dans le portefeuille.
  - $\text{Rendement Espéré}_i$: Rendement espéré de l'actif i.
  - Poids_i : Poids attribué à l'actif i dans le portefeuille.

Cette formule calcule le rendement espéré d'un portefeuille en sommant les produits du rendement espéré de chaque actif (Rendement Espéré$_i$) et de son poids correspondant (Poids$_i$) dans le portefeuille. Elle suppose que les rendements espérés fournis pour chaque actif sont représentatifs des performances futures.

— **Deuxième méthode**

$$\text{Rendement Espéré du Portefeuille} = \text{Temps} \cdot \sum_{i=1}^{n} \text{Poids}_i \cdot \text{Rendement Moyen}_i$$

Où :

— Rendement Espéré du Portefeuille : Moyenne pondérée des rendements moyens des actifs individuels, ajustée pour une période donnée.
— Rendement Moyen$_i$: Rendement moyen de l'actif i.
— Poids$_i$: Poids attribué à l'actif i dans le portefeuille.
— Temps : Période sur laquelle le rendement espéré est calculé.

Cette formule prend en compte les rendements moyens pondérés de chaque actif dans le portefeuille ainsi que la période temporelle (**Temps**) pour estimer le rendement espéré sur un horizon donné.

— **Importance du rendement espéré** Le rendement espéré d'un portefeuille fournit une estimation du rendement moyen qu'un investisseur peut attendre, en se basant sur les performances anticipées des actifs individuels et leurs pondérations respectives. C'est une métrique essentielle utilisée dans l'analyse et la prise de décision concernant les portefeuilles.

2.1.4.5 Écart-type et Volatilité d'un Portefeuille : L'écart-type et la volatilité sont des mesures fondamentales du risque associé à un portefeuille. Elles permettent d'évaluer la variabilité des rendements autour de leur moyenne.

— **Formule de l'écart-type d'un portefeuille :** L'écart-type d'un portefeuille est calculé à l'aide de la formule suivante :

$$\text{Écart-type du Portefeuille} = \sqrt{\mathbf{w}^\top \Sigma \mathbf{w}}$$

Où :

— Écart-type du Portefeuille (*std*) : Mesure de la dispersion ou du risque global du portefeuille.
— $\mathbf{w}$: Vecteur des poids attribués à chaque actif dans le portefeuille.
— Σ : Matrice de covariance des rendements des actifs.
— T : Période de temps sur laquelle les rendements du portefeuille sont calculés.

— **Lien avec la volatilité :** La volatilité est souvent définie comme l'écart-type annualisé des rendements d'un portefeuille. Pour annualiser l'écart-type, la formule est donnée par :

$$\text{Volatilité} = \text{Écart-type} \times \sqrt{T}$$

Dans ce contexte, $T = 252$, correspond au nombre de jours de trading dans une année. Ainsi, pour calculer la volatilité, on ajuste l'écart-type en fonction de la racine carrée du nombre de périodes dans l'année de trading.

Ces mesures sont essentielles pour comprendre le risque associé à un portefeuille d'investissement. Une volatilité plus élevée indique un risque accru et une incertitude plus grande concernant les rendements futurs. L'écart-type et la volatilité sont également utilisés dans l'optimisation de portefeuille pour maximiser les rendements ajustés au risque.

2.1.4.6 La méthode VaR Monte-Carlo : La méthode de Monte-Carlo pour calculer la Value-at-Risk (VaR) est une technique qui utilise la génération de scénarios aléatoires pour évaluer la probabilité de pertes importantes d'un portefeuille. Elle permet d'estimer la perte maximale attendue à un niveau de confiance donné sur une période donnée.

Étapes de la méthode

1. **Modélisation des rendements des actifs** :
   - Les rendements des actifs sont modélisés en utilisant une distribution statistique (souvent normale) ou des données historiques.
   - La matrice de covariance des rendements est utilisée pour capturer les relations entre les actifs.
2. **Simulation de scénarios** :
   - Des milliers de scénarios de rendements possibles sont générés en utilisant des tirages aléatoires selon la distribution choisie.
3. **Calcul des rendements du portefeuille** :
   - Pour chaque scénario, le rendement du portefeuille est calculé à l'aide de la formule :

   $$R_{\text{portefeuille}} = \sum_{i=1}^{n} \text{Poids}_i \cdot R_i$$

   Où Poids_i est le poids de l'actif i et R_i est le rendement simulé de l'actif i.
4. **Construction de la distribution des pertes** :
   - Les rendements du portefeuille sont triés pour construire une distribution des pertes potentielles.
5. **Calcul de la VaR** :
   - La VaR correspond à la perte au quantile $(1-\alpha)$ de la distribution des pertes :

   $$\text{VaR} = \text{Quantile}_{1-\alpha}(\text{Distribution des pertes})$$

   Où α est le niveau de confiance choisi (par exemple, 95% ou 99%).

2.1.4.7 La méthode CVaR Monte-Carlo : La Conditional Value at Risk (CVaR), ou espérance de perte, est une mesure avancée du risque qui complète la Value at Risk (VaR). Contrairement à la VaR, qui se limite à estimer une perte maximale à un certain niveau de confiance, la CVaR quantifie la perte moyenne des scénarios se trouvant au-delà du seuil de la VaR. Cela en fait une mesure plus robuste, particulièrement utile pour évaluer les risques extrêmes dans les queues de distribution.

Formule mathématique

La CVaR est calculée à l'aide de l'intégrale suivante :

$$\text{CVaR} = \frac{1}{1-c} \int_{-\infty}^{\text{VaR}} x \cdot p(x)\, dx$$

Où :

- $p(x)$: La densité de probabilité des rendements.
- c : Le niveau de confiance (par exemple, $c = 0.95$ pour un niveau de confiance de 95%).
- VaR : La Value at Risk au niveau de confiance c.

Applications

- **Gestion des risques** : Identifier et quantifier les pertes extrêmes pour mieux protéger un portefeuille.
- **Optimisation de portefeuille** : Intégrer la CVaR dans les fonctions d'objectif pour minimiser le risque global.
- **Analyse des risques extrêmes** : Compléter la VaR, qui peut être insuffisante pour des distributions non normales.

La CVaR est une mesure puissante qui surmonte les limitations de la VaR en se concentrant sur les scénarios extrêmes. Elle est largement utilisée en finance pour améliorer la gestion et l'optimisation des risques.

2.1.4.8 Ratio de Sharpe et Simulation Monte Carlo : Le ratio de Sharpe est un indicateur permettant d'évaluer la rentabilité ajustée au risque d'un portefeuille. Il est défini par :

$$S = \frac{E[R_p] - R_f}{\sigma_p} \tag{1}$$

où :

- $E[R_p]$ est le rendement espéré du portefeuille,
- R_f est le taux sans risque,
- σ_p est l'écart-type (volatilité) du portefeuille.

Estimation du Ratio de Sharpe via Monte Carlo

La dynamique des prix des actifs suit un Mouvement Brownien Géométrique (GBM), modélisé par l'équation de Black-Scholes :

$$dS_t = \mu S_t dt + \sigma S_t dW_t \tag{2}$$

où W_t est un mouvement brownien. Sa discrétisation selon la méthode d'Euler donne :

$$S_{t+\Delta t} = S_t e^{(\mu - \frac{1}{2}\sigma^2)\Delta t + \sigma\sqrt{\Delta t}Z} \tag{3}$$

avec $Z \sim \mathcal{N}(0, 1)$.

Calcul des Rendements du Portefeuille :

Le rendement du portefeuille est donné par :

$$R_p = \sum_{i=1}^{n} w_i R_i \tag{4}$$

où w_i est le poids de l'actif i et R_i son rendement simulé.

Estimation du Ratio de Sharpe : Après avoir généré plusieurs trajectoires et calculé les rendements simulés

— $E[R_p]$ est obtenu en moyennant les rendements simulés.

— σ_p est estimé comme l'écart-type empirique des rendements simulés.

— Le ratio de Sharpe est alors calculé pour chaque scénario.

Interprétation et Analyse :

— **Sensibilité au nombre de simulations** : Plus le nombre de simulations est élevé, plus l'estimation est précise.

— **Comparaison avec une approche historique** : Analyse des différences entre la simulation et les données réelles.

— **Impact du choix du taux sans risque** R_f : Influence de ce paramètre sur le ratio de Sharpe.

— **Optimisation du portefeuille** : Recherche de la composition optimale en maximisant le ratio de Sharpe.

2.1.5 Explication Mathématique de la Simulation de Monte Carlo :

La simulation de Monte Carlo repose sur la génération de nombres aléatoires à partir de distributions de probabilité spécifiées. Ces nombres aléatoires sont utilisés pour échantillonner les variables d'entrée et simuler différents scénarios. Le processus de simulation consiste à échantillonner de manière répététives les variables d'entrée et à évaluer le modèle afin d'obtenir les résultats.

2.1.5.1 Exemple : Estimation de la Valeur d'un Portefeuille d'Investissement

Considérons un exemple simple où nous souhaitons estimer la valeur d'un portefeuille d'investissement après une certaine période. Nous avons les variables d'entrée suivantes :

— **Montant initial de l'investissement** I_0 suivant une distribution normale avec une moyenne μ_I et un écart-type σ_I.

— **Taux de rendement annuel** R suivant une distribution log-normale avec des paramètres μ' et σ'.

Étapes de la Simulation de Monte Carlo :

1. Définir le Problème : L'objectif est d'estimer la valeur du portefeuille d'investissement après une période spécifiée.

2. Modéliser le Système : Supposons que la valeur du portefeuille à la fin de la période soit donnée par la formule suivante :

$$I_T = I_0 \cdot (1 + R)^T$$

Où :

— I_T est la valeur du portefeuille à la fin de la période.

— I_0 est l'investissement initial.

— R est le taux de rendement annuel.

3. Définir les Distributions d'Entrée : Spécifiez les distributions de probabilité pour les variables d'entrée :

— $I_0 \sim \mathcal{N}(\mu_I, \sigma_I^2)$, une distribution normale.

— $R \sim \mathcal{LN}(\mu', \sigma'^2)$, une distribution log-normale.

4. Générer des Échantillons Aléatoires : Générez des valeurs aléatoires pour I_0 et R en fonction de leurs distributions spécifiées.

5. Effectuer les Simulations : Pour chaque simulation, insérez les valeurs générées pour I_0 et R dans le modèle pour calculer la valeur du portefeuille I_T. Répétez ce processus plusieurs fois (par exemple, N fois) pour obtenir N valeurs simulées du portefeuille.

6. Analyser les Résultats : Analysez les N valeurs simulées du portefeuille et calculez des statistiques récapitulatives, telles que :

— La **moyenne** des valeurs simulées.

— L'**écart-type**, qui mesure la variabilité des valeurs simulées.

— Les **percentiles**, par exemple le 5e et le 95e percentile, pour obtenir une idée de la plage des valeurs possibles du portefeuille.

7. Valider et Affiner : Validez les résultats de la simulation en les comparant avec les valeurs réelles du portefeuille ou des données historiques. Si nécessaire, affinez le modèle ou les distributions d'entrée pour améliorer la précision des résultats.

En exécutant un grand nombre de simulations et en agrégeant les résultats, la simulation de Monte Carlo fournit un cadre statistique pour estimer le comportement et l'incertitude du portefeuille d'investissement. Elle permet aux analystes des risques de prendre des décisions éclairées, d'évaluer les risques potentiels et d'examiner différentes stratégies de manière probabiliste.

2.1.5.2 Exemple : Tarification des Options Considérons un exemple où nous souhaitons modéliser l'évolution du prix d'un actif financier. **1.Définir le Problème :** Estimer le prix d'une option européenne avec payoff $\Phi(S_T)$, où S_T est le prix du sous-jacent à maturité T. **2.Modéliser le Système :** Modèle de prix sous-jacent (GBM sous mesure risque-neutre) :

$$dS_t = rS_t dt + \sigma S_t dW_t$$
$$S_T = S_0 \exp\left(\left(r - \frac{\sigma^2}{2}\right)T + \sigma\sqrt{T}Z\right), \quad Z \sim \mathcal{N}(0,1)$$

3.Définir les Distributions d'Entrée :

— Paramètres : S_0, K, r, σ, T

— Distribution gaussienne : $Z \sim \mathcal{N}(0,1)$

4.Générer des Échantillons Aléatoires : Pour $N = 10^6$ simulations :

$$\{Z_i\}_{i=1}^N \xrightarrow{\text{Box-Muller}} \text{Échantillons gaussiens}$$

5.Effectuer les Simulations : Pour chaque Z_i :

$$S_T^{(i)} = S_0 \exp\left(\left(r - \frac{\sigma^2}{2}\right)T + \sigma\sqrt{T}Z_i\right)$$
$$\text{Payoff}_i = \max(S_T^{(i)} - K, 0) \quad \text{(Call européen)}$$

6.Analyser les Résultats :

$$\text{Prix}_{\text{MC}} = e^{-rT} \cdot \frac{1}{N} \sum_{i=1}^{N} \text{Payoff}_i$$

Erreur standard :

$$\epsilon = \frac{\sigma_{\text{payoff}}}{\sqrt{N}} \quad \text{où } \sigma_{\text{payoff}} = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (\text{Payoff}_i - \overline{\text{Payoff}})^2}$$

7.Valider et Affiner : Comparaison avec Black-Scholes :

$$C_{\text{BS}} = S_0 \Phi(d_1) - K e^{-rT} \Phi(d_2)$$
$$d_1 = \frac{\ln(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}, \quad d_2 = d_1 - \sigma\sqrt{T}$$

3 Architecture et Implémentation de l'Application :

3.1 Introduction :

Cette application vise à fournir un outil interactif pour la tarification d'options et l'analyse de portefeuille en utilisant la simulation Monte Carlo. Elle intègre Dash pour l'interface utilisateur, Plotly pour la visualisation, et Yahoo Finance pour l'importation des données de marché.

3.2 Architecture Générale de l'Application :

L'application est structurée selon une architecture modulaire comprenant :

3.2.1 Frontend : Interface Utilisateur (UI) :

L'interface utilisateur est construite avec Dash et Plotly, offrant une expérience interactive et fluide. **Technologies utilisées :**Dash : framework basé sur Flask pour créer des interfaces web interactives. Plotly : bibliothèque pour la visualisation de données et graphiques dynamiques. Dash Bootstrap Components (DBC) : pour styliser l'interface avec des éléments réactifs.

Fonctionnalités :

- Sélection des paramètres d'option et des simulations Monte Carlo.
- Visualisation interactive des trajectoires simulées.
- Affichage des résultats et indicateurs de performance du portefeuille.

3.2.2 Backend : Modèles et Calculs

Le backend est chargé d'exécuter les simulations et d'effectuer les calculs financiers.

Technologies utilisées :

- Python (NumPy, Pandas, SciPy) pour les calculs et l'optimisation.

Principaux modules :

- Simulation Monte Carlo des prix des actifs selon un Mouvement Brownien Géométrique (GBM).
- Tarification des options et calculs des métriques de risque (VaR, CVaR).
- Analyse de portefeuille basée sur les statistiques des actifs sélectionnés.

3.2.3 Données : Récupération et Prétraitement

L'application utilise des données financières en temps réel via Yahoo Finance.

Technologies utilisées :

- yfinance : pour récupérer les prix des actifs.
- Pandas : pour nettoyer, interpoler et structurer les données.

Étapes du traitement :

- Nettoyage des données (suppression des valeurs aberrantes, interpolation des données manquantes).
- Conversion des devises si nécessaire.
- Estimation de la volatilité et des rendements historiques.

3.2.4 API : Communication et Intégration

Une API REST est mise en place pour exposer les résultats des simulations.
Technologies possibles :

- Flask RESTful ou FastAPI pour gérer les requêtes API.
- JSON comme format d'échange des données.

Fonctionnalités :

- Récupération des résultats des simulations Monte Carlo.
- Exposition des métriques de risque des portefeuilles analysés.
- Intégration avec d'autres systèmes financiers pour stocker ou exploiter les données.

3.3 Implémentation des Algorithmes de Simulation Monte Carlo :

3.3.1 Modélisation des Trajectoires de Prix :

La dynamique du prix d'un actif S_t est modélisée par l'équation différentielle stochastique suivante :

$$dS_t = \mu S_t dt + \sigma S_t dW_t \tag{5}$$

où :

- S_t est le prix de l'actif à l'instant t.
- μ est le rendement moyen.
- σ est la volatilité de l'actif.
- W_t est un mouvement brownien standard.

En discrétisant cette équation en utilisant la méthode d'Euler-Maruyama, on obtient l'expression suivante :

$$S_{t+\Delta t} = S_t \exp\left(\left(\mu - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t} \cdot Z\right) \tag{6}$$

où :

- Δt est le pas de temps de la simulation.
- $Z \sim \mathcal{N}(0, 1)$ est une variable aléatoire suivant une loi normale centrée réduite.

3.4 Traitement et Importation des Données :

Pour réaliser des simulations Monte Carlo précises et analyser un portefeuille, il est essentiel d'obtenir des données financières fiables et bien traitées. Cette section décrit les étapes d'importation, de nettoyage et d'optimisation des données de marché.

3.4.1 Acquisition des Données

L'application utilise l'API `Yahoo Finance` via la bibliothèque `yfinance` pour récupérer les données financières des actifs sous-jacents. Les données importées incluent :

- Les prix historiques (ouverts, fermés, plus hauts, plus bas, ajustés).
- La volatilité implicite estimée à partir des variations de prix.
- Les dividendes et fractionnements d'actions pour un ajustement correct des séries de prix.

L'importation est effectuée dynamiquement en fonction des actifs sélectionnés par l'utilisateur.

3.4.2 Prétraitement et Nettoyage des Données

Les données brutes peuvent contenir des valeurs manquantes ou être sujettes à des ajustements en raison de fractionnements d'actions ou de distributions de dividendes. Pour garantir la qualité des données, plusieurs traitements sont appliqués :

- **Interpolation des valeurs manquantes** : Les données manquantes sont comblées à l'aide de méthodes comme l'interpolation linéaire ou l'extrapolation basée sur des moyennes mobiles.
- **Ajustement des prix** : Les prix sont corrigés pour refléter correctement les fractionnements et dividendes, garantissant ainsi une continuité temporelle des données.
- **Filtrage des anomalies** : Suppression des valeurs aberrantes détectées par des méthodes statistiques (écart-type, test de Grubbs).

3.4.3 Optimisation des Performances

L'importation des données peut être coûteuse en termes de temps et de ressources. Pour améliorer la réactivité de l'application, plusieurs techniques d'optimisation sont mises en place :

- **Mise en cache des données** : Utilisation de `joblib` ou `Redis` pour stocker temporairement les données fréquemment utilisées et éviter des requêtes répétées à Yahoo Finance.
- **Requêtes asynchrones** : L'importation des données est réalisée de manière asynchrone pour ne pas bloquer l'interface utilisateur.
- **Vectorisation avec NumPy et Pandas** : Les opérations sur les séries temporelles sont effectuées avec des structures de données optimisées pour le calcul matriciel.

Ces optimisations garantissent un accès rapide et fluide aux données tout en réduisant la charge serveur.

3.5 Interface Utilisateur avec Dash :

L'interface utilisateur de l'application est construite avec `Dash`, un framework basé sur `Flask`, `Plotly` et `React`, permettant de créer des tableaux de bord interactifs pour la visualisation et l'analyse financière.

3.5.1 Composants de Dash

Dash fournit plusieurs bibliothèques facilitant la construction d'interfaces web dynamiques :

— **dash_core_components (dcc)** : Contient des éléments interactifs comme des graphiques, des curseurs, des menus déroulants et des champs de saisie.

— **dash_html_components (html)** : Permet de structurer la page en ajoutant des éléments HTML classiques (div, boutons, titres, tableaux).

— **dash_bootstrap_components (dbc)** : Fournit des composants stylisés facilitant la mise en page et l'ergonomie (boutons, cartes, alertes, grilles).

L'agencement de l'interface est conçu de manière modulaire pour s'adapter à différentes tailles d'écran et offrir une navigation fluide.

3.5.2 Graphiques Interactifs avec Plotly

Les simulations Monte Carlo et l'analyse de portefeuille génèrent des volumes importants de données qui nécessitent une représentation claire et intuitive. Pour cela, l'application utilise `Plotly`, une bibliothèque puissante de visualisation interactive :

— **Affichage des trajectoires simulées** : Visualisation des scénarios de prix générés par Monte Carlo sous forme de courbes superposées.

— **Histogrammes de distribution des prix** : Représentation des probabilités d'atteinte de certains niveaux de prix à l'échéance.

— **Graphes de performance du portefeuille** : Suivi de l'évolution des rendements et des mesures de risque (VaR, CVaR, Ratio de Sharpe).

L'interaction avec ces graphiques est assurée par `Dash` : l'utilisateur peut zoomer, sélectionner des plages de données et modifier les paramètres des simulations en temps réel.

3.5.3 Tableaux de Bord Dynamiques

L'un des atouts majeurs de Dash est la possibilité de mettre à jour dynamiquement les composants de l'interface sans recharger la page. L'application implémente plusieurs mécanismes pour une expérience utilisateur fluide :

— **Callbacks de Dash** : Mise à jour des graphiques et résultats dès qu'un paramètre est modifié.

— **Stockage en mémoire** : Utilisation des composants `dcc.Store` et `Redis` pour conserver les résultats des simulations et éviter des recalculs inutiles.

— **Mises à jour en temps réel** : Rafraîchissement automatique des données de marché et recalcul des indicateurs en arrière-plan.

Grâce à cette architecture dynamique, l'utilisateur peut interagir directement avec les simulations, ajuster les hypothèses et observer instantanément les impacts sur les résultats.

3.5.4 Interface Utilisateur :

L'interface utilisateur de l'application a été conçue pour offrir une expérience intuitive et interactive, permettant aux utilisateurs de configurer les simulations, visualiser les résultats et explorer les données financières de manière dynamique. Elle se compose de plusieurs sections clés, notamment un panneau de configuration pour ajuster les paramètres de simulation (tels que la volatilité, le rendement attendu et le nombre de scénarios), une zone d'affichage des résultats avec des graphiques interactifs, et un tableau de bord résumant les indicateurs de performance et de risque. La figure ci-dessous montre un aperçu de l'interface, mettant en évidence ces éléments et leur organisation.

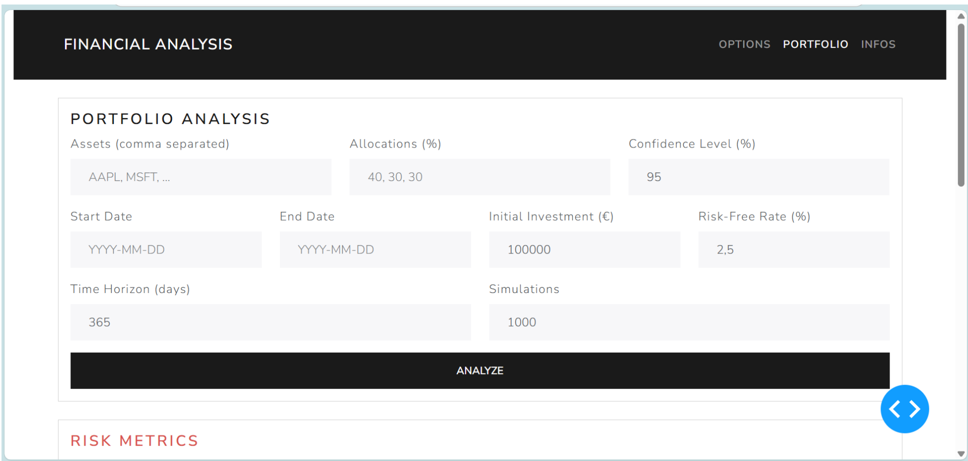

FIGURE 1 – Aperçu de l'interface utilisateur de l'application.

3.6 Visualisation des Résultats :

La visualisation des résultats est une étape clé pour interpréter les simulations Monte Carlo. L'application propose plusieurs types de graphiques interactifs et indicateurs pour analyser les données :

3.6.1 Graphiques interactifs

— **Trajectoires Monte Carlo** : Visualisation des chemins simulés pour le prix de l'actif sous-jacent. Chaque trajectoire représente une évolution possible du prix dans le temps.

— **Convergence des simulations** : Graphique montrant l'évolution de la moyenne des prix simulés en fonction du nombre de scénarios. Cela permet de vérifier la stabilité des résultats.

— **Distribution des rendements** : Histogramme interactif montrant la distribution des rendements du portefeuille ou de l'actif à l'horizon de temps choisi.

3.6.2 Distribution des prix à l'expiration

— **Histogramme des prix finaux** : Représentation graphique de la distribution des prix simulés à la date d'expiration.

— **Intervalles de confiance** : Calcul et affichage des intervalles de confiance à 95% pour les prix finaux. Par exemple :

$$\text{Intervalle de confiance} = \left[\overline{S_T} - 1.96 \cdot \frac{\sigma}{\sqrt{N}}, \overline{S_T} + 1.96 \cdot \frac{\sigma}{\sqrt{N}}\right]$$

où $\overline{S_T}$ est la moyenne des prix simulés, σ est l'écart-type, et N est le nombre de simulations.

3.6.3 Indicateurs clés

Les indicateurs suivants sont calculés et affichés pour chaque simulation :

— **Prix moyen de l'option** : Moyenne des prix simulés à l'expiration.

— **Intervalle de confiance** : Fournit une plage de valeurs dans laquelle le prix final a une probabilité donnée de se situer.

— **Convergence des simulations** : Mesure de la stabilité des résultats en fonction du nombre de scénarios.

— **VaR (Value at Risk)** : Estimation de la perte maximale potentielle à un niveau de confiance donné (par exemple, 95%).

— **CVaR (Conditional Value at Risk)** : Espérance des pertes au-delà de la VaR, fournissant une mesure du risque extrême.

3.6.4 Exemple de visualisation

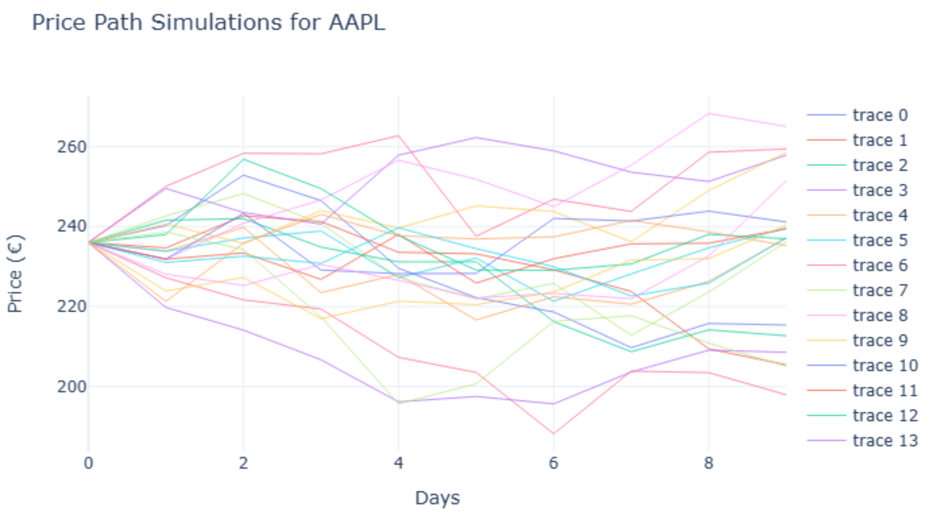

FIGURE 2 – Exemple de trajectoires Monte Carlo pour le prix d'un actif sous-jacent.

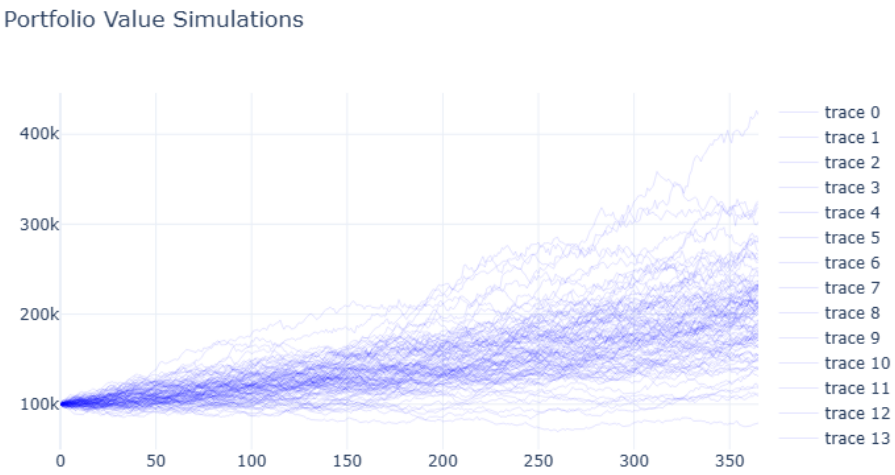

FIGURE 3 – Distribution des prix à l'expiration avec intervalles de confiance.

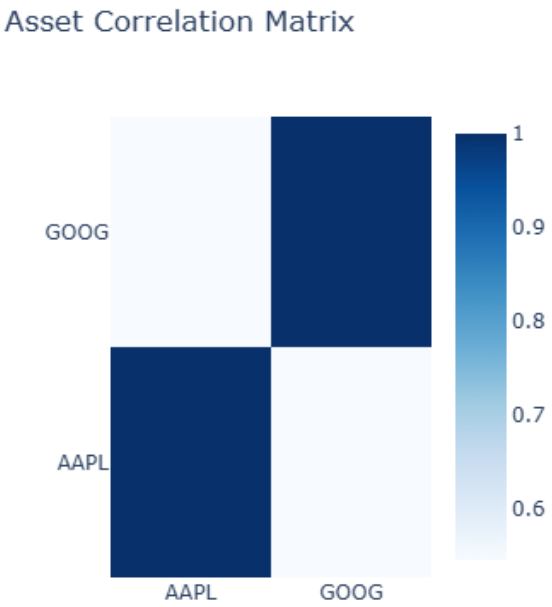

FIGURE 4 – Matrice de Corrélation des Actifs.

FIGURE 5 – rendements cumulés historiques.

3.7 Analyse de Portefeuille et Mesures de Risque :

L'analyse de portefeuille et la gestion des risques sont des composantes essentielles de l'application. Les fonctionnalités suivantes sont proposées pour évaluer et visualiser les risques associés à un portefeuille d'actifs :

3.7.1 Calcul des Mesures de Risque

— **Value at Risk (VaR)** : La VaR est une mesure statistique qui estime la perte maximale potentielle d'un portefeuille sur un horizon de temps donné, avec un niveau de confiance spécifié (par exemple, 95%). Elle est calculée comme suit :

$$\text{VaR}_\alpha = F^{-1}(1 - \alpha)$$

où F^{-1} est la fonction quantile de la distribution des rendements, et α est le niveau de confiance.

— **Conditional Value at Risk (CVaR)** : Le CVaR, également appelé Expected Shortfall, mesure l'espérance des pertes au-delà de la VaR. Il fournit une estimation plus complète du risque extrême :

$$\text{CVaR}_\alpha = E[X \mid X \geq \text{VaR}_\alpha]$$

où X représente les pertes du portefeuille.

3.7.2 Visualisation des Risques

— **Distribution des rendements** : Un histogramme interactif montre la distribution des rendements du portefeuille, mettant en évidence les queues de distribution et les risques extrêmes.

— **Graphique de la VaR et du CVaR** : Visualisation des niveaux de VaR et CVaR sur la distribution des rendements, permettant une interprétation intuitive des risques.

— **Analyse de sensibilité** : Graphiques montrant l'impact de la variation des poids des actifs sur les mesures de risque.

3.7.3 Simulation de Portefeuilles

— **Ajustement dynamique des poids** : L'utilisateur peut ajuster les poids des actifs dans le portefeuille et observer en temps réel l'impact sur les mesures de risque et les rendements attendus.

— **Optimisation de portefeuille** : Simulation de différents portefeuilles en fonction de critères tels que la maximisation du ratio de Sharpe ou la minimisation du risque.

— **Comparaison de portefeuilles** : Visualisation côte à côte des performances et des risques de plusieurs portefeuilles simulés.

3.7.4 Exemple de Visualisation

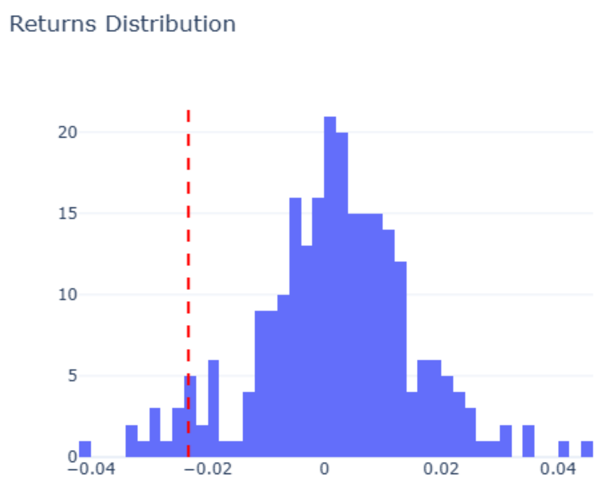

FIGURE 6 – Distribution des rendements du portefeuille avec indication de la VaR et du CVaR.

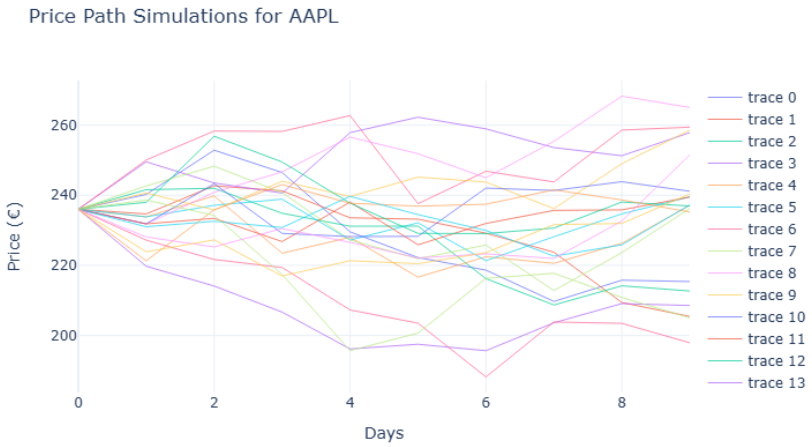

FIGURE 7 – Simulation de différents portefeuilles avec ajustement dynamique des poids.

4 Conclusion :

Ce projet a permis de concevoir et de développer une application dédiée à l'application de la méthode de simulation Monte Carlo dans le domaine financier, avec un focus particulier sur la tarification des options, l'analyse de portefeuille et la gestion des risques. En combinant des concepts mathématiques rigoureux, des technologies modernes et une interface utilisateur interactive, l'application offre un outil puissant pour modéliser l'incertitude des marchés financiers et prendre des décisions éclairées.

Les simulations Monte Carlo, basées sur le modèle de mouvement brownien géométrique (GBM), ont démontré leur efficacité pour prédire les trajectoires de prix des actifs et évaluer les risques associés à différents scénarios. Grâce à des techniques d'optimisation telles que la réduction de variance et la parallélisation, les calculs sont réalisés de manière rapide et précise, même pour un grand nombre de simulations. L'interface utilisateur, développée avec Dash et Plotly, permet une visualisation intuitive des résultats, incluant des graphiques interactifs, des distributions de prix et des indicateurs clés comme la Value at Risk (VaR) et la Conditional Value at Risk (CVaR).

Les données financières, récupérées en temps réel via l'API Yahoo Finance, ont été intégrées de manière fluide dans l'application, offrant aux utilisateurs la possibilité d'analyser des portefeuilles complexes et d'ajuster dynamiquement les poids des actifs. Les études de cas présentées dans ce rapport ont illustré l'utilité pratique de l'application dans des scénarios réels, mettant en lumière sa capacité à fournir des insights précieux pour la gestion des risques et l'optimisation des portefeuilles.

Cependant, ce projet ouvre également la voie à des améliorations futures. Parmi les perspectives envisageables, on peut citer l'intégration de modèles plus sophistiqués (tels que les modèles de sauts ou les processus de volatilité stochastique), l'utilisation de techniques d'apprentissage automatique pour affiner les prédictions, et l'extension de l'application à d'autres classes d'actifs ou marchés financiers. De plus, une collaboration avec des professionnels de la finance pourrait permettre d'adapter l'outil à des besoins spécifiques et d'enrichir ses fonctionnalités.

En conclusion, ce projet a non seulement atteint ses objectifs initiaux en fournissant une plateforme robuste et interactive pour la simulation Monte Carlo en finance, mais il a également posé les bases pour des développements futurs. Il représente une contribution significative à l'intersection entre la finance quantitative et les technologies numériques, offrant des opportunités tant pour les chercheurs que pour les praticiens du domaine financier.

5 Annexe

Le code source présenté dans cette annexe est disponible sous licence open-source et peut être réutilisé ou modifié selon les besoins. Pour plus de détails sur l'implémentation ou pour contribuer au projet, veuillez consulter le dépôt GitHub associé à ce projet.

```
import numpy as np
import pandas as pd
import yfinance as yf
import dash
import dash_bootstrap_components as dbc
from dash import dcc, html, Input, Output, State, callback
from dash.exceptions import PreventUpdate
import plotly.graph_objects as go

class MonteCarloPricing:
    def __init__(self, ticker, K=None, T_days=None, r=None, sigma=None, simulations=1000, option_type='european', avg_type='arithmetic'):
        # Valider et garantir des param tres positifs.
        self.ticker = ticker
        self.simulations = max(100, int(simulations))  # Minimum 100 simulations

        # R cup rer les donn es boursi res avec gestion des erreurs
        try:
            self.data = self._fetch_stock_data()
        except Exception as e:
            print(f"Data fetch error: {e}")
            self.data = self._default_stock_data()

        # D finir les param tres avec des valeurs de repli
        self.S0 = float(self.data.get('price', 100))
        self.K = float(K if K is not None else self.data['options']['strike'])

        # Garantir un nombre de jours jusqu'  l' chance  positif
        self.steps = max(10, int(T_days if T_days is not None else self.data['options']['days_to_expiry']))
        self.T = self.steps / 365  # Convertir en ann es

        # Valider les autres param tres num riques
        self.r = float(r if r is not None else self.data.get('risk_free_rate', 0.05))
        self.sigma = float(sigma if sigma is not None else self.data.get('volatility', 0.2))

        self.option_type = option_type
        self.avg_type = avg_type
        self.dt = self.T / self.steps

    def _default_stock_data(self):
        """Fournir des donn es par d faut en cas d' chec  de la r cup ration"""
        return {
            'price': 100,
            'volatility': 0.2,
            'risk_free_rate': 0.05,
```

```
            'options': {
                'strike': 100,
                'days_to_expiry': 30
            }
        }

    def _fetch_stock_data(self):
        """R cup rer les donn es des actions et options depuis Yahoo Finance"""
        try:
            # Obtenir les informations sur le ticker de l'action
            stock = yf.Ticker(self.ticker)

            # R cup rer le prix actuel de l'action
            price = float(stock.history(period="1d")['Close'].iloc[0])

            # Obtenir les donn es d'options (prochaine    chance   )
            options = stock.options
            if not options:
                raise ValueError(f"No options data available for {self.ticker}")

            # Extraire la cha ne d'options de la prochaine    chance
            next_expiry = options[0]
            option_chain = stock.option_chain(next_expiry)

            # Calculer le prix d'exercice moyen et la volatilit  implicite des options d'achat
            calls = option_chain.calls
            avg_strike = float(calls['strike'].mean())
            implied_vol = float(calls['impliedVolatility'].mean())

            # D terminer le nombre de jours jusqu'   l'   chance
            expiry_date = pd.to_datetime(next_expiry)
            days_to_expiry = max(10, (expiry_date - pd.Timestamp.now()).days)

            # Estimer le taux sans risque (approxim     partir du rendement des bons du Tr sor    10 ans)
            risk_free_rate = float(yf.Ticker('^TNX').history(period='1d')['Close'].iloc[0] / 100)

            return {
                'price': price,
                'volatility': implied_vol,
                'risk_free_rate': risk_free_rate,
                'options': {
                    'expiry_date': next_expiry,
                    'days_to_expiry': days_to_expiry,
                    'strike': avg_strike
                }
            }
        except Exception as e:
            print(f"Error fetching data for {self.ticker}: {e}")
            return self._default_stock_data()

    def simulate(self):
```

```python
        """Simuler les trajectoires de prix avec une gestion robuste des erreurs"""
        try:
            np.random.seed(42)

            # V rifier que les dimensions des matrices sont positives
            steps = max(10, int(self.steps))
            sims = max(100, int(self.simulations))

            S = np.zeros((steps, sims))
            S[0] = self.S0

            drift = (self.r - 0.5*self.sigma**2)*self.dt
            vol = self.sigma*np.sqrt(self.dt)

            for t in range(1, steps):
                Z = np.random.normal(0, 1, sims)
                S[t] = S[t-1] * np.exp(drift + vol*Z)

            return S
        except Exception as e:
            print(f"Simulation error: {e}")
            # Fournir une simulation de secours avec des param tres minimaux en cas  d erreur
            return np.full((10, 100), self.S0)

    def calculate_prices(self, S):
        discount = np.exp(-self.r*self.T)

        if self.option_type == 'asian':
            if self.avg_type == 'arithmetic':
                avg_price = np.mean(S, axis=0)
            else:  # geometric
                avg_price = np.exp(np.mean(np.log(S), axis=0))

            call = discount * np.mean(np.maximum(avg_price - self.K, 0))
            put = discount * np.mean(np.maximum(self.K - avg_price, 0))
        else:  # europ en
            call = discount * np.mean(np.maximum(S[-1] - self.K, 0))
            put = discount * np.mean(np.maximum(self.K - S[-1], 0))

        return call, put


app = dash.Dash(__name__,
                external_stylesheets=[dbc.themes.LUX],
                suppress_callback_exceptions=True)

app.title = "Financial Simulations Suite"

app.layout = dbc.Container([
    dbc.Navbar(
        dbc.Container([
            dbc.NavbarBrand("Analyse Financi re ", className="ms-2"),
            dbc.Nav([
                dbc.NavLink("Options", href="/options", active="exact"),
```

```
                dbc.NavLink("Portfeuille", href="/portfolio", active="exact"),
                dbc.NavLink("Infos", href="/infos", active="exact"),
            ], className="ms-auto", navbar=True)
        ]),
        color="primary",
        dark=True,
        className="mb-4"
    ),

    dcc.Location(id='url'),
    html.Div(id='page-content')
], fluid=True)


options_layout = dbc.Container([
    dbc.Card([
        dbc.CardBody([
            html.H4("Calculateur de Tarification d'Options ", className="card-title text-primary"),

            dbc.Row([
                dbc.Col([
                    dbc.Label("Stock Ticker"),
                    dbc.Input(id="opt-ticker", value="AAPL", type="text")
                ], md=3),

                dbc.Col([
                    dbc.Label("prix d'exercice ( )"),
                    dbc.Input(id="opt-strike", type="number", placeholder="Auto-detect")
                ], md=3),

                dbc.Col([
                    dbc.Label("Jours jusqu'  l'  chance  "),
                    dbc.Input(id="opt-days", type="number", placeholder="Auto-detect")
                ], md=3),

                dbc.Col([
                    dbc.Label("Type d'option"),
                    dcc.Dropdown(
                        id='opt-type',
                        options=[
                            {'label': 'Europ enne', 'value': 'european'},
                            {'label': 'Asiatique', 'value': 'asian'}
                        ],
                        value='european'
                    )
                ], md=3),
            ], className="mb-3"),

            dbc.Row([
                dbc.Col([
                    dbc.Label("Taux Sans Risque (%)"),
```

```
                    dbc.Input(id="opt-rate", type="number", placeholder="Auto-detect")
                ], md=3),

                dbc.Col([
                    dbc.Label("Volatilit  (%)"),
                    dbc.Input(id="opt-vol", type="number", placeholder="Auto-detect")
                ], md=3),

                dbc.Col([
                    dbc.Label("Simulations"),
                    dbc.Input(id="opt-sims", value=1000, type="number")
                ], md=3),

                dbc.Col([
                    dbc.Label("Type de moyenne"),
                    dcc.Dropdown(
                        id='opt-avg-type',
                        options=[
                            {'label': 'G om trique', 'value': 'geometric'}
                        ],
                        value='geometric',  # Valeur par d faut forc e
                        disabled=True,      # Emp che la modification
                        style={'backgroundColor': '#f8f9fa'}  # Style visuel pour indiquer le statut d sactiv
                    )
                ], md=3),
            ]),

            dbc.Button("Calculate",
                      id="opt-calculate",
                      color="primary",
                      className="mt-3 w-100")
        ])
    ], className="mb-4"),

    dbc.Row([
        dbc.Col(dcc.Graph(id="opt-paths"), md=8),
        dbc.Col([
            dbc.Card([
                dbc.CardBody([
                    html.H4("Results", className="text-success"),
                    html.Div(id="opt-results", className="mt-3 fs-5")
                ])
            ])
        ], md=4)
    ])
])


portfolio_layout = dbc.Container([
    dbc.Card([
        dbc.CardBody([
```

```
            html.H4("Analyse de Portefeuille", className="card-title text-primary"),

            dbc.Row([
                dbc.Col([
                    dbc.Label("Actifs (s par s par des virgules)"),
                    dbc.Input(id="pf-tickers", placeholder="AAPL, MSFT, ...")
                ], md=4),

                dbc.Col([
                    dbc.Label("Allocations (%)"),
                    dbc.Input(id="pf-allocations", placeholder="40, 30, 30")
                ], md=4),

                dbc.Col([
                    dbc.Label("Niveau de confiance (%)"),
                    dbc.Input(id="pf-confidence", value=95, type="number")
                ], md=4),
            ], className="mb-3"),

            dbc.Row([
                dbc.Col([
                    dbc.Label("Date de d but"),
                    dbc.Input(id="pf-start", placeholder="YYYY-MM-DD")
                ], md=3),

                dbc.Col([
                    dbc.Label("Date de fin"),
                    dbc.Input(id="pf-end", placeholder="YYYY-MM-DD")
                ], md=3),

                dbc.Col([
                    dbc.Label("Investissement initial (   )"),
                    dbc.Input(id="pf-initial", value=100000, type="number")
                ], md=3),

                dbc.Col([
                    dbc.Label("Taux sans risque (%)"),
                    dbc.Input(id="pf-risk-free", value=2.5, type="number")
                ], md=3),
            ], className="mb-3"),

            dbc.Row([
                dbc.Col([
                    dbc.Label("Horizon temporel (jours)"),
                    dbc.Input(id="pf-horizon", value=365, type="number")
                ], md=6),

                dbc.Col([
                    dbc.Label("Simulations"),
                    dbc.Input(id="pf-sims", value=1000, type="number")
                ], md=6),
```

```
            ]),

            dbc.Button("Analyser",
                       id="pf-analyze",
                       color="primary",
                       className="mt-3 w-100")
        ])
    ], className="mb-4"),

    dbc.Card([
        dbc.CardBody([
            html.H4("M triques de risque", className="text-danger"),
            html.Div(id="pf-risk-metrics", className="mt-3 fs-5")
        ])
    ], className="mb-4"),

    dbc.Row([
        dbc.Col(dcc.Graph(id="pf-simulations"), md=8),
        dbc.Col(dcc.Graph(id="pf-correlation"), md=4)
    ]),

    dbc.Row([
        dbc.Col(dcc.Graph(id="pf-distribution"), md=6),
        dbc.Col(dcc.Graph(id="pf-returns"), md=6)
    ])
])


info_layout = dbc.Container([
    dbc.Card([
        dbc.CardBody([
            # Titre principal
            html.H2("M thode Monte Carlo", className="text-primary mb-4"),

            # Section : Principe G n ral
            html.H4("Principe G n ral", className="mt-4"),
            dcc.Markdown(
                '''
                La m thode Monte Carlo est une technique num rique qui utilise l' chantillonnage al atoire
                pour r soudre des probl mes math matiques ou physiques complexes. Son nom vient du quartier
                de Monte Carlo  Monaco, r put  pour ses casinos et les jeux de hasard.
                ''',
                className="mb-3"
            ),

            # Section : Application en Finance
            html.H4("Application en Finance", className="mt-4"),
            dcc.Markdown(
                '''
                **Utilisations typiques :**
                - Valorisation d'options exotiques
                - Analyse de risque de portefeuille
                - Pr visions de march s complexes
```

```
                - Calculs de Value at Risk (VaR)

                **Avantages :**
                - Flexibilit  dans la mod lisation
                - Gestion de dimensions multiples
                - Pr cision am lior e avec plus de simulations
                ''',
                className="mb-3"
            ),

            # Section : Dans cette Application
            html.H4("Dans cette Application", className="mt-4"),
            dcc.Markdown(
                '''
                - **Options :** Simulation de trajectoires browniennes g om triques
                - **Portefeuille :** Mod lisation des corr lations entre actifs
                - **Param tres cl s :**
                    - Nombre de simulations (pr cision vs performance)
                    - Volatilit  (mesure du risque)
                    - Horizon temporel (p riode de projection)
                ''',
                className="mb-3"
            ),

            # Section :  quations  Cl s (corrig e)
            html.H4(" quations  Cl s", className="mt-4"),
            dcc.Markdown(
                r"""
                **Mouvement Brownien G om trique :**
                $$
                dS_t = \mu S_t \, dt + \sigma S_t \, dW_t
                $$

                **Discounted Payoff (Options) :**
                $$
                \text{Prix} = \mathbb{E}\left[ e^{-rT} \cdot \text{Payoff}(S_T) \right]
                $$

                **Value at Risk (Portefeuille) :**
                $$
                \text{VaR}_\alpha = \inf \left\{ l \in \mathbb{R} : \mathbb{P}(L > l) \leq 1 - \alpha \right\}
                $$
                """,
                className="mb-3",
                mathjax=True  # Activation de MathJax si n cessaire
            )
        ])
    ], className="mt-4")
], fluid=True)

@callback(
    Output("page-content", "children"),
    Input("url", "pathname")
```

```
)
def render_page(pathname):
    if pathname == "/portfolio":
        return portfolio_layout
    elif pathname == "/infos":
        return info_layout
    return options_layout


@callback(
    [Output("opt-paths", "figure"),
     Output("opt-results", "children")],
    Input("opt-calculate", "n_clicks"),
    [State("opt-ticker", "value"),
     State("opt-strike", "value"),
     State("opt-days", "value"),
     State("opt-rate", "value"),
     State("opt-vol", "value"),
     State("opt-sims", "value"),
     State("opt-type", "value"),
     State("opt-avg-type", "value")]
)
def update_options(_, ticker, K, T, r, vol, sims, option_type, avg_type):
    if not ticker:
        raise PreventUpdate

    # Cr er un mod le avec des param tres d tect s automatiquement
    model = MonteCarloPricing(
        ticker,
        K=K,  # Sera utilis  si aucun param tre n'est sp cifi
        T_days=T,  # Sera utilis  si aucun param tre n'est sp cifi
        r=r/100 if r else None,  # Convertir en d cimal si fourni
        sigma=vol/100 if vol else None,  # Convertir en d cimal si fourni
        simulations=sims,
        option_type=option_type,
        avg_type=avg_type
    )

    paths = model.simulate()
    call, put = model.calculate_prices(paths)

    fig = go.Figure()
    for i in range(min(20, sims)):
        fig.add_trace(go.Scatter(
            y=paths[:, i],
            mode='lines',
            line=dict(width=1),
            opacity=0.7
        ))
    fig.update_layout(
        title=f"Price Path Simulations for {ticker}",
        xaxis_title="Days",
        yaxis_title="Price (   )",
        template="plotly_white"
    )
```

```python

    results = f"""
    Current Price:    {model.S0:.2f}
    Call Price:    {call:.2f}
    Put Price:    {put:.2f}

    Details:
    Strike:    {model.K:.2f}
    Volatility: {model.sigma*100:.2f}%
    Risk-Free Rate: {model.r*100:.2f}%
    Days to Expiry: {int(model.steps)}
    Simulations: {sims:,}
    Type: {option_type.capitalize()}{' ('+avg_type+')' if option_type == 'asian' else ''}
    """

    return fig, results

def calculate_risk_metrics(returns, confidence, risk_free_rate):
    var = np.percentile(returns, 100 - confidence)
    cvar = returns[returns <= var].mean()
    sharpe = (returns.mean() * 252 - risk_free_rate/100) / (returns.std() * np.sqrt(252))
    return {'var': var, 'cvar': cvar, 'sharpe': sharpe}

def download_asset_data(tickers, start, end):
    data = pd.DataFrame()
    for t in tickers:
        try:
            df = yf.download(t, start=start, end=end)['Adj Close']
            data[t] = df
        except:
            continue
    return data.dropna()

@callback(
    [Output("pf-simulations", "figure"),
     Output("pf-correlation", "figure"),
     Output("pf-distribution", "figure"),
     Output("pf-returns", "figure"),
     Output("pf-risk-metrics", "children")],
    Input("pf-analyze", "n_clicks"),
    [State("pf-tickers", "value"),
     State("pf-allocations", "value"),
     State("pf-start", "value"),
     State("pf-end", "value"),
     State("pf-initial", "value"),
     State("pf-horizon", "value"),
     State("pf-sims", "value"),
     State("pf-confidence", "value"),
     State("pf-risk-free", "value")]
)
def update_portfolio(_, tickers, allocs, start, end, initial, horizon, sims, confidence, risk_free):
    if not tickers or not allocs:
        raise PreventUpdate

```

```python
    try:
        tickers = [t.strip() for t in tickers.split(",")]
        allocs = [float(a.strip())/100 for a in allocs.split(",")]

        if len(tickers) != len(allocs) or abs(sum(allocs)-1) > 0.01:
            raise ValueError("Invalid allocations")

        data = download_asset_data(tickers, start, end)
        if data.empty:
            raise ValueError("No data downloaded")

        returns = data.pct_change().dropna()
        portfolio_returns = returns.dot(allocs)

        # Calcul des m triques de risque
        risk_metrics = calculate_risk_metrics(portfolio_returns, confidence, risk_free)

        # Simulation Monte Carlo
        mu = portfolio_returns.mean() * 252
        sigma = portfolio_returns.std() * np.sqrt(252)
        simulations = []
        for _ in range(sims):
            prices = [initial]
            for _ in range(horizon):
                ret = np.random.normal(mu/252, sigma/np.sqrt(252))
                prices.append(prices[-1] * np.exp(ret))
            simulations.append(prices)

        # G n ration des graphiques
        sim_fig = go.Figure()
        for s in simulations[:100]:
            sim_fig.add_trace(go.Scatter(y=s, line=dict(width=1, color='blue'), opacity=0.1))
        sim_fig.update_layout(title="Portfolio Value Simulations", template="plotly_white")

        corr_fig = go.Figure(go.Heatmap(
            z=returns.corr().values,
            x=returns.columns,
            y=returns.columns,
            colorscale='Blues'
        )).update_layout(title="Asset Correlation Matrix")

        dist_fig = go.Figure(go.Histogram(x=portfolio_returns, nbinsx=50))
        dist_fig.add_vline(x=risk_metrics['var'], line_dash="dash", line_color="red")
        dist_fig.update_layout(title="Returns Distribution", template="plotly_white")

        cum_returns = (1 + returns).cumprod()
        ret_fig = go.Figure()
        for col in cum_returns:
            ret_fig.add_trace(go.Scatter(x=cum_returns.index, y=cum_returns[col], name=col))
```

```
567         ret_fig.update_layout(title="Historical Cumulative Returns",
    template="plotly_white")
568
569         metrics_text = f"""
570         VaR ({confidence}%): {risk_metrics['var']:.2%}
571         CVaR ({confidence}%): {risk_metrics['cvar']:.2%}
572         Sharpe Ratio: {risk_metrics['sharpe']:.2f}
573         """
574
575         return sim_fig, corr_fig, dist_fig, ret_fig, metrics_text
576
577     except Exception as e:
578         print(f"Error: {str(e)}")
579         return [go.Figure()]*4 + [f"Error: {str(e)}"]
580
581
582 if __name__ == "__main__":
583     app.run_server(debug=True, port=8050)
```