Projet C++
Portefeuille d'actifs et calcul de rendement

Bousselmame Adam
Moniz Antoine

27 février 2026

Table des matières

1 Présentation du sujet

Ce projet a pour objectif de concevoir, en langage C++, un système permettant de modéliser un portefeuille d'actifs financiers. Il repose sur une approche orientée objet et s'articule autour de deux classes principales : la classe `Asset`, représentant un actif financier individuel, et la classe `Portfolio`, représentant un ensemble d'actifs détenus en différentes quantités.

Le programme doit permettre la gestion dynamique des positions, le calcul du rendement attendu du portefeuille ainsi que l'estimation de sa variance à partir d'une matrice de corrélation. Il inclut également la surcharge de certains opérateurs et la mise en place d'un mécanisme robuste de gestion des exceptions. Ce travail vise à mobiliser les notions vues en cours, notamment l'utilisation des conteneurs standards tels que `std::map` et `std::vector`, ainsi que les principes fondamentaux de la programmation orientée objet.

2 Modélisation mathématique

Le rendement attendu d'un portefeuille est défini comme la moyenne pondérée des rendements attendus des actifs qui le composent. Formellement, il s'écrit :

$$E(R_p) = \sum_{i=1}^{n} w_i E(R_i) \tag{1}$$

où w_i représente le poids de l'actif i dans le portefeuille et $E(R_i)$ son rendement attendu. Le poids est déterminé à partir de la valeur de marché de chaque position :

$$w_i = \frac{P_i \times q_i}{\sum_{j=1}^{n} P_j \times q_j} \tag{2}$$

avec P_i le prix de l'actif et q_i la quantité détenue.

La variance du portefeuille repose sur l'interaction entre les volatilités individuelles et les corrélations entre actifs. Elle est donnée par :

$$Var(R_p) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \sigma_i \sigma_j \rho_{ij} \tag{3}$$

où σ_i désigne la volatilité de l'actif i et ρ_{ij} la corrélation entre les actifs i et j. Cette expression correspond à la forme développée de l'écriture matricielle classique :

$$Var(R_p) = w^T \Sigma w \tag{4}$$

où Σ est la matrice de covariance.

3 Structure orientée objet

La classe `Asset` encapsule les caractéristiques financières d'un actif. Elle contient son nom, son prix, son rendement attendu ainsi que sa volatilité. Deux constructeurs ont été définis : un constructeur paramétré permettant d'initialiser les attributs, et un constructeur par défaut nécessaire à l'utilisation du conteneur `std::map`. Des méthodes d'accès constantes permettent de consulter les informations sans modifier l'état de l'objet.

La classe `Portfolio` assure la gestion des positions. Elle repose sur une structure de type :

```
std::map<std::string, std::pair<Asset, double>>
```

Chaque entrée associe le nom d'un actif à une paire contenant l'objet `Asset` correspondant et la quantité détenue. Ce choix garantit un accès rapide par nom, évite les doublons et facilite la gestion des positions.

4 Fonctionnalités implémentées

L'ajout d'une position est assuré par une méthode qui vérifie si l'actif est déjà présent dans le portefeuille. Dans ce cas, la quantité est incrémentée ; sinon, une nouvelle entrée est créée. La suppression d'une position est également prévue. Si l'actif demandé n'existe pas, une exception de type `std::out_of_range` est levée afin d'assurer la robustesse du programme.

Le calcul du rendement attendu suit une démarche en deux étapes : calcul de la valeur totale du portefeuille, puis détermination des poids et application de la formule de moyenne pondérée.

Le calcul de la variance nécessite une vérification préalable de la cohérence dimensionnelle entre le nombre d'actifs et la matrice de corrélation fournie. En cas d'incohérence, une exception `std::invalid_argument` est déclenchée. Le calcul est ensuite réalisé au moyen d'une double boucle parcourant l'ensemble des actifs.

Deux opérateurs ont été surchargés. L'opérateur `+` permet de fusionner deux portefeuilles en additionnant les quantités des actifs communs. L'opérateur `[]` autorise l'accès direct à une position via son nom ; une exception est levée si l'actif n'est pas présent.

5 Gestion des exceptions

La gestion des erreurs constitue un aspect du projet. Deux types d'exceptions ont été mis en place : `std::out_of_range` pour signaler un accès invalide à un actif et `std::invalid_argument` pour indiquer une matrice de corrélation mal dimensionnée. Ce mécanisme garantit un comportement prévisible du programme en cas d'erreur.

6 Bonus

6.1 Affichage du portefeuille

Une méthode spécifique d'affichage a été implémentée afin de visualiser de manière structurée le contenu du portefeuille. Cette méthode, déclarée constante, permet de parcourir l'ensemble des positions et d'afficher pour chaque actif :

— son nom,
— la quantité détenue,
— son prix unitaire,
— la valeur de la position,
— son poids dans le portefeuille (exprimé en pourcentage).

La valeur d'une position est calculée comme le produit du prix unitaire par la quantité détenue. Le poids est ensuite déterminé en rapportant cette valeur à la valeur totale du portefeuille :

$$w_i = \frac{P_i \times q_i}{\sum_{j=1}^{n} P_j \times q_j} \tag{5}$$

L'affichage inclut également la valeur totale du portefeuille ainsi que son rendement attendu.

Cette fonctionnalité améliore la lisibilité des résultats et permet de vérifier visuellement la cohérence des calculs effectués. Elle établit un lien direct entre les formules mathématiques présentées et leur implémentation concrète dans le code.

Le choix d'implémenter cette méthode en tant que fonction constante garantit qu'elle ne modifie pas l'état interne du portefeuille, respectant ainsi les principes de const-correctness.

6.2 Calcul du ratio de Sharpe

Afin de compléter l'analyse du portefeuille, une méthode de calcul du ratio de Sharpe a été implémentée. Le ratio de Sharpe constitue une mesure de performance ajustée du risque, largement utilisée en gestion de portefeuille.

Il est défini par :

$$Sharpe = \frac{E(R_p) - r_f}{\sigma_p} \tag{6}$$

où :

— $E(R_p)$ représente le rendement attendu du portefeuille,

— r_f est le taux sans risque,

— σ_p est la volatilité du portefeuille.

Cette mesure permet d'évaluer le rendement excédentaire obtenu par unité de risque pris. Un ratio de Sharpe élevé indique que le portefeuille offre une rémunération attractive au regard de sa volatilité.

Dans l'implémentation, la volatilité est obtenue à partir de la racine carrée de la variance précédemment calculée. Une vérification supplémentaire est effectuée afin d'éviter une division par zéro en cas de volatilité nulle.

L'ajout de cette fonctionnalité permet de relier directement les calculs techniques (rendement et variance) à un indicateur synthétique de performance financière, renforçant ainsi la cohérence économique du projet.

7 Difficultés rencontrées

La première difficulté a concerné l'utilisation de `std::map`, qui impose la présence d'un constructeur par défaut dans la classe `Asset`. L'absence initiale de ce constructeur a entraîné une erreur de compilation, corrigée par l'ajout d'un constructeur adéquat.

Un second problème est apparu lors de la génération des fichiers temporaires du projet, en raison d'un conflit avec un dossier synchronisé. Le déplacement du projet hors du dossier concerné a permis de résoudre cette difficulté.

Enfin, la mise en cohérence entre l'ordre des actifs et la matrice de corrélation a nécessité une attention particulière afin d'assurer la validité du calcul de variance.

8 Perspectives d'amélioration

Ce projet pourrait être prolongé de plusieurs manières. Il serait notamment possible d'intégrer une matrice de covariance complète et d'implémenter une optimisation moyenne-variance selon l'approche de Markowitz. D'autres extensions incluraient l'ajout de contraintes d'investissement, l'importation de données à partir de fichiers externes ou encore le calcul d'autres indicateurs de performance.

9 Conclusion

Ce projet respecte l'ensemble des exigences techniques fixées : programmation orientée objet, utilisation des conteneurs standards, surcharge d'opérateurs et gestion des exceptions. Il met en œuvre des concepts fondamentaux de la gestion de portefeuille et constitue une base solide pour des développements plus avancés en finance quantitative.