Élèves: Juan SUZANO et Tahya DEDDAH

Analyse du risque de crédit

1. Introduction:

La finance ou la science de gestion de l'argent est une branche très importante d'où la nécessité de la mise en place des différents outils pour l'étudier.

Grâce à sa puissance , l'informatique quantique est devenu récemment parmi les outils les plus utilisés en analyse financière afin de rendre plusieurs tâches plus facile en terme de complexité et de temps.

2. Application de l'informatique quantique sur la finance:

Parmi les cas d'utilisation de l'informatique quantique en finance on cite :

- **Optimisation quantique :**
  Les problèmes d'optimisation sont à la base de plusieurs problèmes financiers. La plupart de ces problèmes ont un ordre de difficulté de NP et donc c'est extrêmement difficile aux ordinateurs classiques de les résoudres efficacement. En revanche l'informatique quantique donne des meilleurs solutions pour ce genre de problèmes en les encodant à des problèmes physiques équivalents et en formulant ces problèmes physiques de telle sorte que leur état d'énergie le plus bas soit mathématiquement équivalent à la solution du problèmes que nous voulons résoudre. Tout en utilisant des événements de tunnel quantique pour conduire le système à son minimum.
  Certaines méthodes implémentant ce principe quantique ont été déjà implémenté:
  - Trajectoire de négociation optimale *:* ce sont des algorithmes mises en place pour minimiser le coût de la négociation.
  - Opportunités d'arbitrage optimales : il s'agit de réaliser de bénéfices des prix différents d'un même actifs sans risque.
  - Sélection optimale des caractéristiques dans la notation du crédit : il s'agit de déterminer, avant d'accorder un prêt , quelles caractéristiques des clients sont les plus pertinentes pour l'évaluation des risques.

- **Classification des données:**
  La notation de crédit se repose sur une méthode de machine learning appelé classification . En effet chaque client est exprimé par un vecteur ,appartenant à une classe donnée, et qui a comme coordonnées les différentes caractéristique de client.
  En recevant un nouveau vecteur, le programme doit déterminer la classe à laquelle il est susceptible d'appartenir. Ce domaine de machine learning est essentiellement utilisé en finance pour la détection des fraudes.

- **Analyse des composantes principales (ACP):**
  L'ACP consiste à projeter un ensemble des données sur un nombres des axes (souvent deux) qui contiennent la plupart d'information afin de faciliter les interprétations de ces données.
  L'ACP est utilisé en finance pour avoir une vision globale des trajectoires de taux d'intérêt. Cependant il était récemment montré qu'une version quantique d'ACP existe avec une vitesse d'exécution extrêmement plus grande

- **Analyse du risque du crédit (sujet de notre projet):**
  Le risque est quantifié mathématiquement par la valeur de la perte (loss), Value at Risk(VaR) et la valeur conditionnelle à risque (CVaR) ces concepts sont importants dans l'optimisation du portefeuille.
  Ces trois valeurs sont estimés classiquement en utilisant la méthode de Monte Carlo.
  En appliquant l'algorithme de Quantum amplitude estimation QAE en informatique quantique on peut déterminer ces trois valeurs a une excellente précision et une accélération quadratique par rapport à la méthode de Monte Carlo

3. Analyse du risque de crédit :

1-Definition de problème:

On veut analyser le risque du crédit (calcul de la perte ,VaR et CVaR) pour un portefeuille de K assets. La probabilité par défaut de de chaque asset k suit une loi d'indépendance conditionnelle gaussienne. Autrement dit soit Z une variable aléatoire de distribution normale modélisant l'évolution de notre portefeuille, la probabilité par défaut d'un asset k est donnée par la relation suivante:

$$p_k(z) = F\left(\frac{F^{-1}(p_k^0) - \sqrt{\rho_k}z}{\sqrt{1-\rho_k}}\right)$$

Où F est la fonction de distribution cumulé de Z ,p_k_0 la probabilité par défaut de l'asset k pour z=0 et rho_k_0 la sensibilité de la probabilité par défaut.

Passons maintenant à la valeur de la perte totale (Loss) donnée ci-dessous:

$$L = \sum_{k=1}^{K} \lambda_k X_k(Z)$$

Avec lambda_k la perte par défaut de l'asset k et X_k une variable aléatoire suivant une loi de Bernoulli représentant l'événement par défaut de l'actif k.

Intéressons au VaR qui mesure la répartition des pertes d'un portefeuille et qui est definit par :

$$\mathrm{VaR}_\alpha(L) = \inf\{x \mid \mathbb{P}[L <= x] \geq 1-\alpha\}$$

Avec alpha le niveau de sécurité : 0< alpha<1

En fin définissons la CVaR qui est la perte attendue qui se produit lorsque le point d'arrêt de la VaR est atteint:

$$\mathrm{CVaR}_\alpha(L) = \mathbb{E}[L \mid L \geq \mathrm{VaR}_\alpha(L)].$$

2- Modéle d'incertitude:

Nous construisons maintenant un circuit qui charge le modèle d'incertitude. Ceci peut être réalisé en créant un état quantique dans un registre de nz qubits qui représente Z le suivi d'une distribution normale standard.

Cet état est ensuite utilisé pour contrôler les Y- rotations d'un qubit sur un second registre de qubits de K qubits, où l'état |1> de qubit k représente l'événement par défaut de l'actif k. L'état quantique résultant peut être écrit comme:

$$|\Psi\rangle = \sum_{i=0}^{2^{n_z}-1} \sqrt{p_z^i}|z_i\rangle \bigotimes_{k=1}^{K} \left(\sqrt{1-p_k(z_i)}|0\rangle + \sqrt{p_k(z_i)}|1\rangle\right),$$

3.Résolution du problème:

On commence par résoudre le problème sans utilisation de QAE pour pouvoir ensuite comparer les résultats avec celles de QAE.

a. Sans QAE

L'approche utilisée pour résoudre le problème sans recourir à l'estimation consiste à assembler le circuit qui représente le problème, à simuler et à analyser le résultat de la simulation. Pour assembler le circuit, nous devons d'abord construire le circuit du modèle d'incertitude en utilisant la fonction GaussianConditionalIndependenceModel() et ensuite on utilise GaussianConditionalIndependenceModel.num_target_qubits pour déterminer le nombre de qubits nécessaires pour représenter le modèle d'incertitude. on a ainsi ce qu'il faut pour assembler le circuit à simuler.

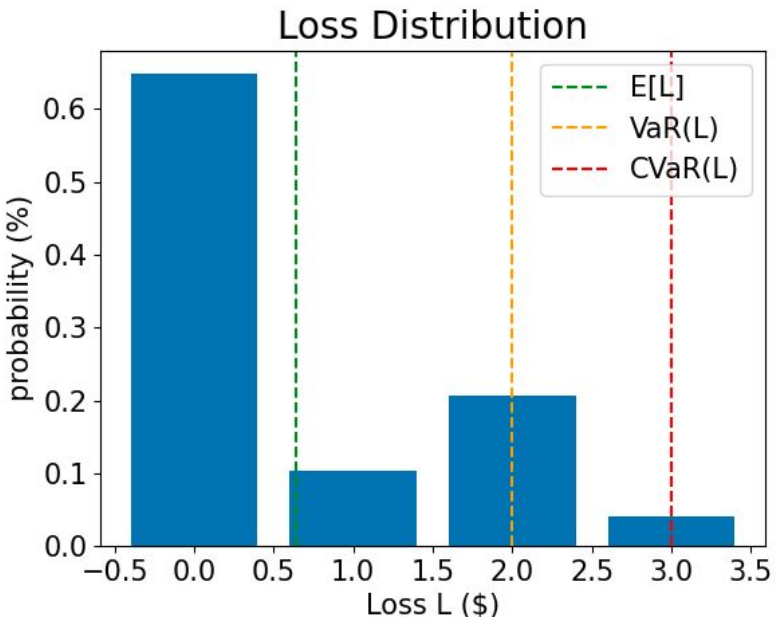
La simulation a été réalisée par la fonction execute() en utilisant le backend "statevector_simulator". Après avoir effectué la simulation, on peut facilement accéder au résultat grâce à la fonction result(). Avec cela, on peut calculer le vecteur de probabilité de perte de chaque actif. C'est-à-dire la probabilité qu'un actif présente une perte de 0, 1, 2 ou 3. Finalement, on trouve Expected Loss, Value at Risk (VaR), Conditional Value at Risk (CVaR) et

la probabilité d'avoir VaR comme perte (CDF) en utilisant simplement leurs définitions mathématiques.

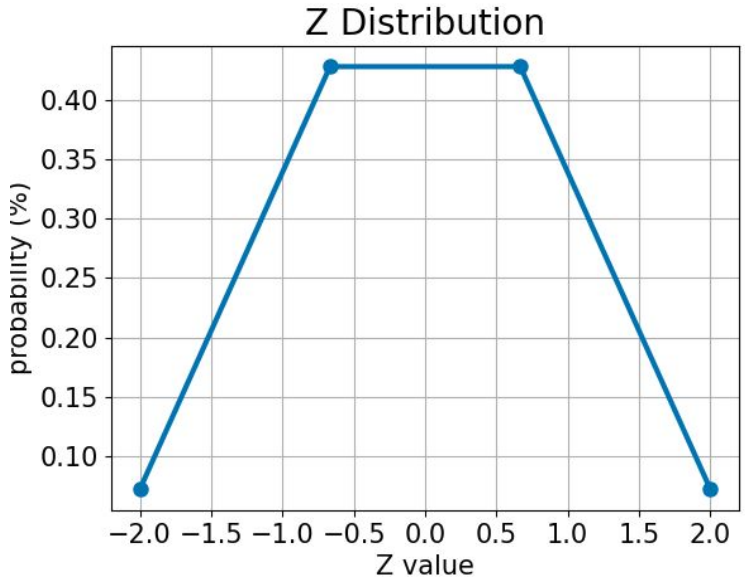
En exécutant notre programme on trouve les résultats suivants :

- Expected Loss E[L] : 0.6409
- Value at Risk VaR[L] : 2.0000
- P[L <= VaR[L]] (cdf) : 0.9591
- Conditional Value at Risk CVaR[L] : 3.0000

Ces résultats sont représentés graphiquement dans l'image ci-dessous :



Nous pouvons également afficher la distribution de Z:



b. Avec QAE

La résolution avec QAE passe par 4 étapes calcule de la perte (expected loss), la fonction de distribution cumulative (cdf), Value at Risk (VaR) puis Conditional Value at Risk (CVar).

Note : pour chaque partie de la solution avec QAE (i, ii, iii et iv), nous indiquerons brièvement certaines parties du code qui nous semblent les plus pertinentes. Certains détails de mise en œuvre peuvent être omis afin de ne pas rendre le rapport trop long.

i. Expected Loss

Pour estimer la perte attendue, nous appliquons d'abord un opérateur de somme pondérée pour résumer les pertes individuelles à la perte totale:

$$\mathcal{S} : |x_1, \ldots, x_K\rangle_K |0\rangle_{n_S} \mapsto |x_1, \ldots, x_K\rangle_K |\lambda_1 x_1 + \ldots + \lambda_K x_K\rangle_{n_S}.$$

Le nombre requis de qubits pour représenter le résultat est donné par:

$$n_s = \lfloor \log_2(\lambda_1 + \ldots + \lambda_K) \rfloor + 1.$$

Une fois que nous avons la distribution de la perte totale dans un registre quantique, nous pouvons utiliser les techniques décrites dans [Woerner2019] pour mapper une perte totale $L \in \{0, \ldots, 2^{n_s} - 1\}$ à l'amplitude d'un qubit objectif par un opérateur :

$$|L\rangle_{n_s}|0\rangle \mapsto |L\rangle_{n_s}\left(\sqrt{1-L/(2^{n_s}-1)}|0\rangle + \sqrt{L/(2^{n_s}-1)}|1\rangle\right),$$

ce qui permet d'exécuter une estimation d'amplitude pour évaluer la perte attendue.

Commentaires sur la mise en œuvre:

Pour générer le circuit de la somme pondérée, nous utilisons la fonction WeightedSumOperator() qui construit le circuit de la somme pondérée.

En raison de la nature du problème, la réalisation d'une estimation Expected Loss est un "multivariate problem". On utilise donc la fonction MultivariateProblem() pour définir le problème. Cette fonction prend comme paramètre le circuit du modèle gaussien, le circuit de la somme pondérée et une fonction de type UnivariatePiecewiseLinearObjective().

Après avoir défini le problème multivarié, on utilise les fonctions multivariable.num_target_qubits() et multivariable.required_ancillas() pour définir la quantité de qubits et la quantité de qubits auxiliaires pour résoudre le problème. (Ces deux fonctions ont été utilisées à d'autres moments tout au long du code, mais nous les omettrons à partir de maintenant.) On a ainsi tout ce qu'il faut pour assembler le circuit qui sera utilisé dans l'estimation.

Enfin, nous utilisons la classe AmplitudeEstimation avec cinq qubits d'évaluation pour créer l'objet qui sera simulé à l'aide de la fonction run(). (Comme cette étape a été utilisée pour estimer toutes les valeurs calculées (EL, CDF, VaR et CVaR), elle sera également omise à partir de maintenant.).

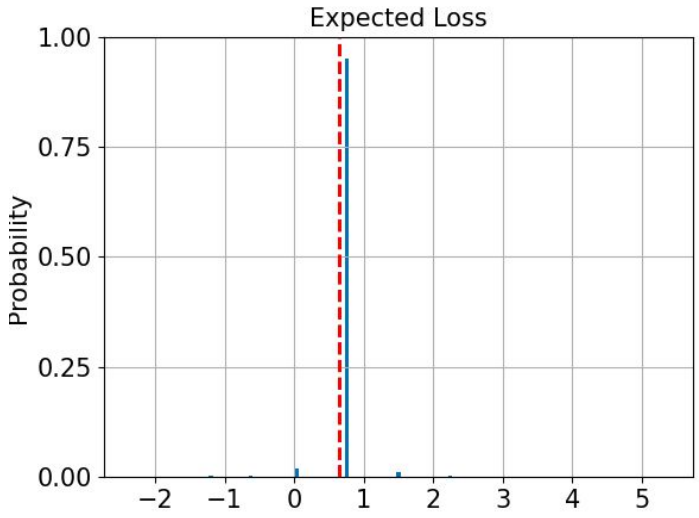
Résultats:

Les résultats qui nous intéressent sont "estimation" et "max_probability". La première est la valeur estimée de la Expected Loss, tandis que la seconde est la probabilité correspondante à cette valeur d'Expected Loss.

On trouve comme résultats les valeurs suivants:

Exact Value : 0.6409
Estimated Value: 0.7548
Probability: 0.9507

Nous pouvons afficher ce résultat graphiquement comme dans l'image ci-dessous où les barres bleues représentent les valeurs possibles de Expected Loss estimée et la ligne rouge pointillée représente la valeur exacte de Expected Loss.



ii. CDF

Après le calcule de la perte totale on s'intéresse au calcul de la Fonction de distribution cumulative (qui pourrait également être estimée efficacement en utilisant des techniques classiques).

Pour estimer la fonction de distribution cumulative (CDF) de la perte. Classiquement, cela implique soit d'évaluer toutes les combinaisons possibles d'actifs défaillants, soit de nombreux échantillons classiques dans une simulation Monte Carlo. Les algorithmes basés sur QAE ont le potentiel d'accélérer considérablement cette analyse à l'avenir.

Pour estimer le CDF, c'est-à-dire la probabilité $\mathbb{P}[L \leq x]$, nous appliquons à nouveau $\mathcal{S}$ pour calculer la perte totale, puis appliquons un comparateur qui, pour une valeur x donnée, agit comme:

$$\mathscr{C} : |L\rangle_n |0\rangle \mapsto \begin{cases} |L\rangle_n |1\rangle & \text{if} \quad L \leq x \\ |L\rangle_n |0\rangle & \text{if} \quad L > x. \end{cases}$$

L'état quantique résultant peut être écrit comme:

$$\sum_{L=0}^{x} \sqrt{p_L} |L\rangle_{n_s} |1\rangle + \sum_{L=x+1}^{2^{n_s}-1} \sqrt{p_L} |L\rangle_{n_s} |1\rangle,$$

Où nous supposons directement les valeurs de perte résumées et les probabilités correspondantes au lieu de présenter les détails du modèle d'incertitude.

Le CDF (x) est égal à la probabilité de mesurer |1> dans le qubit objectif et QAE peut être directement utilisé pour l'estimer.

Commentaires sur la mise en œuvre:

Pour l'estimation de la CDF, on doit utiliser la fonction FixedValueComparator() pour construire le circuit qui effectue les comparaisons nécessaires à la méthode. Ensuite, on construit le circuit et on effectue l'estimation comme on l'a fait au point précédent.

Résultats:

Après l'implémentation du programme on trouve les valeurs suivantes pour CDF:

Exact value: 0.9591
Estimated value: 0.9619
Probability: 0.9958

iii. Value at Risk (VaR)

L'estimation de la VaR consiste à effectuer l'estimation de la CDF, puis on effectue une "bisection search" pour trouver la valeur de la perte qui correspond à la CDF estimée. Cette valeur sera donc la valeur estimée de la VaR.

Commentaires sur la mise en œuvre:

Comme l'estimation du VaR implique l'estimation du CDF, on a de nouveau mis en place le circuit nécessaire pour estimer la CDF. Toutefois, deux nouveaux éléments étaient nécessaires. On a créé une fonction "objectif" où X est le deuxième paramètre de la classe FixedValueComparator. Cet argument représente "la valeur à comparer". De cette façon, lorsque nous passons l'objet objectif à la fonction bisection_search(), on a dans son retour la VaR estimée (la VaR correspond au résultat "level" renvoyé par la fonction bisection_search()).

Résultats:

Estimated Value at Risk: 2
Exact Value at Risk: 2
Estimated Probability: 0.962
Exact Probability: 0.959

iv. Conditional Value at Risk (CVar)

Enfin on calcule la valeur de la CVaR. La CVaR est la valeur attendue de la perte à condition qu'elle soit supérieure ou égale à la VaR. Pour tester ce condition nous évaluons une fonction objectif linéaire f(L), dépendante de la perte totale L, qui est donnée par:

$$f(L) = \begin{cases} 0 & \text{if} \quad L \leq VaR \\ L & \text{if} \quad L > VaR. \end{cases}$$

Pour normaliser, nous devons diviser la valeur attendue résultante par la probabilité VaR, c'est-à-dire $\mathbb{P}[L \leq VaR]$.

Commentaires sur la mise en œuvre:

Pour le calcul de la CVaR, nous devons accéder aux valeurs de Expected Loss, de la VaR et du CDF. Par conséquent, dans la mise en œuvre du calcul du CVaR, on utilise des éléments de tous les postes précédents.

Résultats:
On trouve les résultats suivants pour CVaR:

Exact CVaR:	3.0000
Estimated CVaR:	3.8670
Probability:	0.7146

En comparant les valeurs de L,VaR et CVaR par les deux méthodes on voit que les résultats sont similaires.

4. Repository et Utilisation du Code

Le code du projet a été publié sur le lien https://github.com/juansuzano/Projet_Final_Informatique_Quantique et comporte deux fichiers : Projet_Final.py et bisection_search.py. Le premier fichier est le code qui implémente toutes les fonctionnalités décrites dans ce rapport, à l'exception de la fonction de bisection_search implémentée dans le fichier bisection_search.py.

Nous n'avons pas étudié la mise en œuvre de blablabla.py car nous avons pensé qu'elle dépasserait un peu le cadre de notre projet.

Pour exécuter le code, placez les deux fichiers dans le même dossier et lancez la commande "$ python Projet_final.py" en ayant installé les bibliothèques qiskit, numpy et matplotlib.

L'exécution complète du code prend environ 7 minutes sur mon ordinateur. 30 secondes pour EL, 30 secondes pour CDF, 30 secondes pour VaR et 5 minutes pour CVaR.

Tous les graphiques contenus dans ce rapport ont été générés avec le code fourni. Cependant, les lignes de code responsables de l'affichage des images sont commentées pour faciliter l'exécution du code.

5. Conclusion:

En utilisant l'informatique quantique plus précisément le QAE , On a pu analyser le risque pour un portefeuille en calculant la perte ,VaR et CVaR avec une excellente précision et une accélération quadratique par rapport à la méthode classique.

Ce travail nous a permis d'approfondir les connaissances acquises en cours, ainsi que d'acquérir des connaissances sur des choses qui n'étaient pas traitées en cours. Avec la réalisation de ce projet, ainsi qu'avec tout le contenu appris en cours, nous savons, si nécessaire dans notre carrière professionnelle, où et comment trouver les informations nécessaires pour résoudre des différents problèmes dans le domaine de l'informatique quantitative. Nous avons également pu apprendre plusieurs concepts liés au domaine de l'économie qui ont certainement servi à élargir nos horizons en tant que futurs ingénieurs.