Comparaison des différentes méthodes de calcul des contributions à la VaR

Maxime Jactat supervisé par Paul Demey

16/07/2024

Résumé

Comprendre les contributions de chaque actif au risque total est crucial pour optimiser la gestion du portefeuille et minimiser les pertes potentielles. Cette approche permet d'identifier les sources de risque et de prendre des décisions éclairées pour une allocation plus efficace des ressources. Pour faire face au problème d'additivité de la VaR, il est nécessaire de trouver les contributions à la VaR de chaque actif du portefeuille. Nous verrons sur ce papier deux formules de la contribution à la VaR issues de la littérature [1] et une méthode de calcul proposée par l'éditeur QRM dans son outil éponyme. L'objectif est de déterminer quelle méthode est la plus efficace, la moins biaisée et la plus précise. Pour cela nous verrons rapidement la définition de la VaR et le problème d'additivité inhérent à la VaR. Ensuite, nous aborderons le mode opératoire que j'ai appliqué pour comparer les différentes méthodes de calcul des contributions à la VaR. Et enfin, on pourra comparer les différentes méthodes.

Table des matières

1 Introduction à la Value at Risk (VaR)

La Value at Risk (VaR) est une mesure de risque utilisée dans la finance pour estimer la perte potentielle maximale d'un portefeuille d'actifs sur une période donnée, avec un certain niveau de confiance. Essentiellement, la VaR quantifie le pire scénario auquel on peut s'attendre, à l'exception de cas très rares. Elle est définie comme un certain quantile α de la distribution des pertes potentielles. En termes simples, la VaR à un niveau de confiance α indique la perte maximale qui ne devrait pas être dépassée avec une probabilité de $1-\alpha$.

$$\text{VaR}(\alpha) = \inf\{x \in \mathbb{R} \mid P(X \leq x) \geq 1-\alpha\}$$

On notera que le x appartiendra à un vecteur de PnL dans le cadre d'une modélisation numérique. Et ainsi la probabilité que ma variable aléatoire soit inférieure à la VaR est égale à $1-\alpha$:

$$P(X < \text{VaR}_\alpha) = 1-\alpha$$

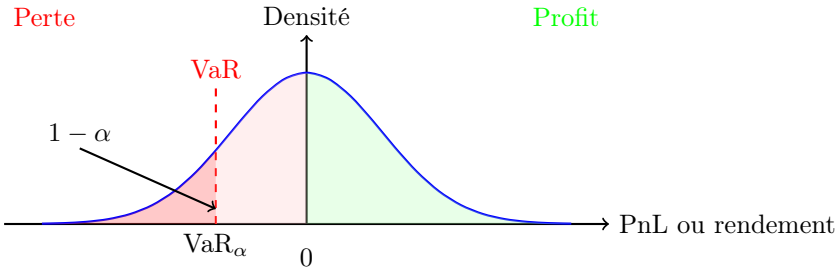


FIGURE 1 – Illustration de la VaR_α

La VaR est un outil d'ALM essentiel et connaître la contribution des actifs du portefeuille à la VaR est un enjeu pour affiner le pilotage de l'ALM suite au calcul de la VaR.

2 Problème d'additivité de la VaR

L'une des principales limitations de la VaR est son manque d'additivité. Autrement dit, la VaR d'un portefeuille n'est pas nécessairement égale à la somme des VaR des composants individuels du portefeuille. Ceci est dû au fait que la VaR ne tient pas compte de la diversification et des corrélations entre les différents actifs du portefeuille P. Formellement, si $\text{VaR}(P)$ représente la VaR du portefeuille et VaR_i représente la VaR du ième actif, alors généralement :

$$\text{VaR}(P) \neq \sum_i \text{VaR}_i$$

En pratique, on aimerai bien connaître les contributions à la VaR de chacun des actifs de notre portefeuille. Et la VaR individuelle ne répond pas à la question (car elle n'est pas additive) donc il faudrait définir autrement cette contribution à la VaR.

3 Mode opératoire pour comparer les méthodes de calcul des CVaR_i

Pour résoudre ce problème d'additivité, nous introduisons les Contributions à la Value at Risk (CVaR_i), de manière à ce que la VaR du portefeuille soit égale à la somme des CVaRi des composants individuels. En d'autres termes, nous cherchons des valeurs CVaR_i telles que :

$$\text{VaR}(P) = \sum_i \text{CVaR}_i$$

Pour tester les différentes méthodes de calcul des contributions à la VaR, on utilisera le code en annexe. L'objectif de ce papier est de constater le biais des estimateurs des CVaR_i et leur variance.

Pour cela, on considère n contrats dont les rendements suivent une loi multivariée normale centrée. Le fait que la loi soit normale et plus particulièrement centrée ne changera pas la conclusion de ce papier mais facilitera les calculs par la suite.

Soit $\{\eta_i\}_{i\in[\![1,n]\!]}$ les rendements des n contrats dans notre portefeuille à un horizon T (représentant les PnL_i de chacun des contrats si on multiplie par le nominal_i).

Par définition, si P_t est la valeur du portefeuille à la date t, le rendement global η_G est donné par :

$$\eta_G = \frac{P_T}{P_0} - 1$$

avec T un certain horizon et $P_t = \sum_{i=1}^{n} N_i(t)$, où N_i est le nominal du contrat i à t. Ainsi :

$$P_T = \sum_{i=1}^{n} N_i(1 + \eta_i)$$

En injectant dans le rendement global η_G et en posant $w_i = \frac{N_i}{P_0}$ le poids du contrat i sur le portefeuille, on a :

$$\eta_G = \sum_{i=1}^{n} w_i \eta_i$$

On notera qu'une somme de gaussiennes centrées est une gaussienne centrée de variance $w'\Sigma w$, avec Σ la matrice de covariance des n contrats et w le vecteur des poids des contrats.

On a alors que le rendement global suit une loi normale centrée :

$$\eta_G \sim \mathcal{N}(0, w'\Sigma w) \sim \sigma_{\eta_G} \mathcal{N}(0, 1)$$

avec $\sigma_{\eta_G} = \sqrt{w'\Sigma w}$ et on pose Φ^{-1} la loi normale inverse centrée réduite

On a alors les formules théoriques de la VaR et de la CVaR_i par définition de la VaR puis par formule d'Euler (car la VaR est 1-homogène, voir la prochaine partie) :

$$\mathrm{VaR}_\alpha = \sigma_{\eta_G} \Phi^{-1}(1 - \alpha)$$

$$\mathrm{CVaR}_i = w_i \frac{\partial \mathrm{VaR}_\alpha}{\partial w_i}$$

Donc le vecteur $(\mathrm{CVaR})_i$ est donné par dérivation matricielle :

$$(CVaR) = w \cdot \frac{\partial \mathrm{VaR}_\alpha}{\partial w} = \frac{\Sigma \cdot w}{\sigma_{\eta_G}} \Phi^{-1}(1 - \alpha) = \frac{\Sigma \cdot w}{\sigma^2_{\eta_G}} \mathrm{VaR}_\alpha$$

Ces deux formules théoriques vont nous permettre de comparer les prochaines méthodes numériques que nous utiliserons pour approximer les contributions à la VaR.

Voici le procédé d'expérimentation empirique pour analyser les trois méthodes que nous allons voir :

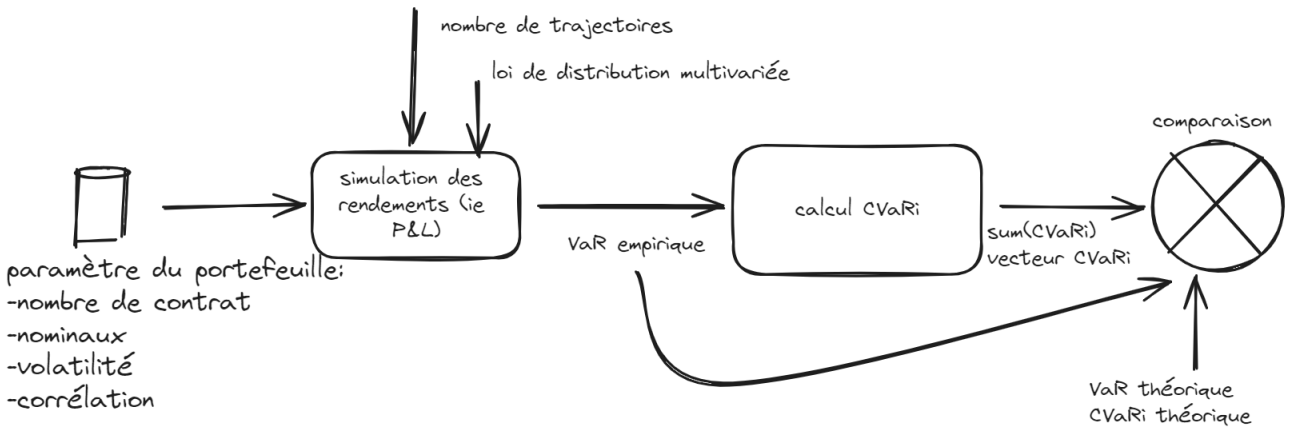


FIGURE 2 – Mode opératoire

En repétant plusieurs fois ce mode opératoire, nous pouvons tracer un histogramme de la VaR empirique $\hat{\mathrm{VaR}}_\alpha$ avec $\sum \mathrm{CVaR}_i$ calculé par différentes méthodes.

4 Première méthode : Différence finie

Cette méthode consiste à utiliser les dérivées pour calculer les CVaR_i. En effet, sous certaines conditions (nombre de points suffisant, homogénéité de la VaR), la CVaR de chaque actif peut être interprétée comme la dérivée partielle de la VaR du portefeuille par rapport à la position dans cet actif. Formellement, si w_i est la pondération du ième actif, alors d'après la formule de Euler :

$$\text{CVaR}_i = w_i \frac{\partial \text{VaR}(P)}{\partial w_i}$$

Pour démontrer la formule d'Euler à partir d'une fonction α-homogène, il suffit de dériver par le terme "d'homothétie" puis d'évaluer en 1 la dérivé et en α=1.

Pour utiliser cette formule sous forme dérivée de la CVaR_i, on peut utiliser la méthode des différences finies à un point et à deux points :

*** Méthode des différences finies à un point** : Cette méthode consiste à approximer la dérivée en utilisant une petite perturbation δ autour du point d'intérêt. La dérivée de la CVaR_i est alors approximée par :

$$\text{CVaR}_i = w_i \frac{\partial \text{VaR}_i}{\partial w_i} \approx \frac{\text{VaR}_i(w_i + \delta) - \text{VaR}_i(w_i)}{\delta}$$

*** Méthode des différences finies à deux points**[1] : Cette méthode utilise deux petites perturbations δ de part et d'autre du point d'intérêt pour obtenir une approximation plus précise de la dérivée. On parle aussi de formule de différence finie centrée (dans mes graphiques, je l'appelle ud pour up down), ce qui se comprend graphiquement. La formule s'écrit alors :

$$\text{CVaR}_i = w_i \frac{\partial \text{VaR}_i}{\partial w_i} \approx \frac{\text{VaR}_i(w_i + \delta) - \text{VaR}_i(w_i - \delta)}{2\delta}$$

Ces méthodes permettent d'obtenir une approximation numérique de la dérivée de la VaR en fonction de w_i.

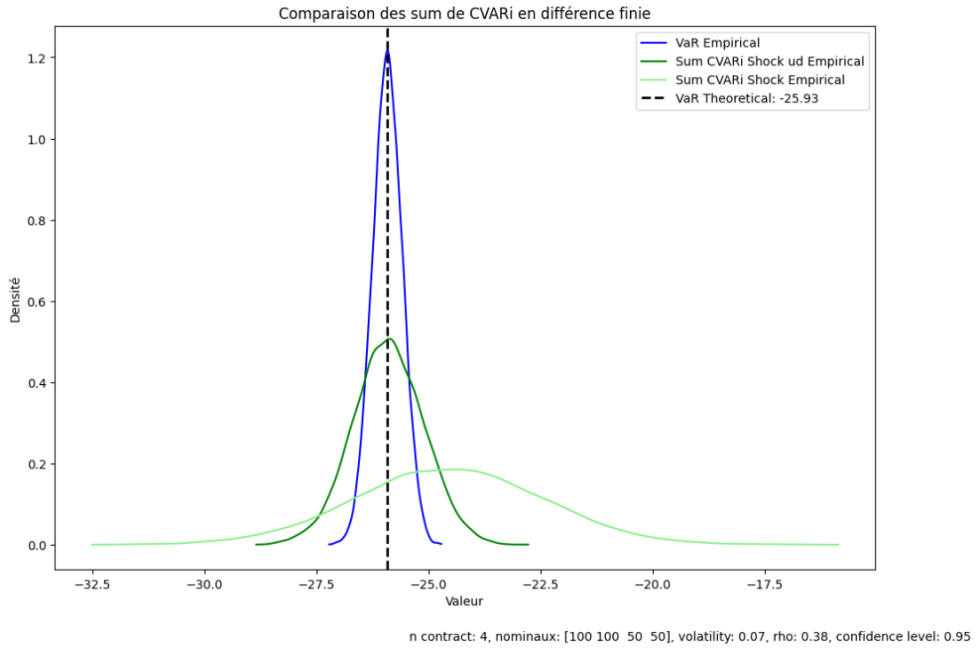


FIGURE 3 – VaR empirique VS $\sum \text{CVaR}_i$ par différence finie

Plus la distribution est proche de la VaR théorique en pointillé, plus l'estimateur est précis. De plus, plus une courbe est étalée, comme celle en vert clair, plus sa variance est élevée, ce qui indique que c'est une mesure moins précise. En fait ce graphe permet de conclure qu'une différence finie à deux points est plus performante que la formule de numérisation classique de la dérivé à un point.

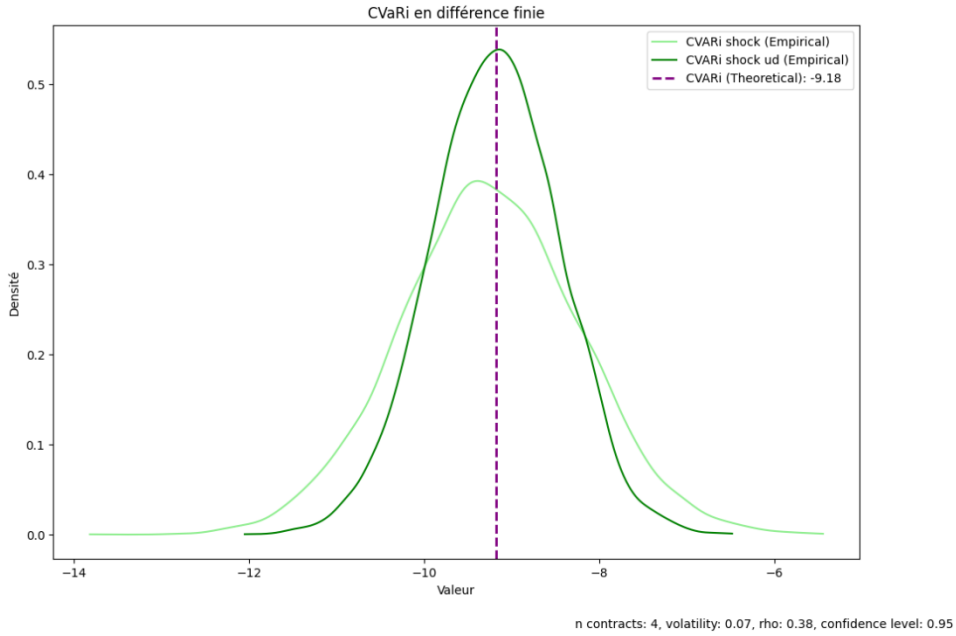


FIGURE 4 – CVaR_i théorique VS CVaR_i par différence finie

Même si la VaR par différence finie simple est biaisée, sa $CVaR_i$ semble moins biaisée. Néanmoins, et sans surprise, la méthode par différence finie centrée est meilleure.

5 Deuxième méthode : Extraction de scénario et méthode du noyau

Cette méthode repose sur une approche probabiliste, où l'on extrait des scénarios spécifiques et ajuste la variance avec une fenêtre de kernel.

1. Extraction de scénarios : On simule un grand nombre de scénarios de pertes pour le portefeuille et ses composants.

$$\text{CVaR}_i = E[PnL_i \mid PnL = \text{VaR}]$$

On retrouve bien la VaR car $\sum_i PnL_i = PnL$ et l'espérance est linéaire. Néanmoins d'un point de vue numérique, cette méthode n'est pas stable en vue de la variance des contributions à la VaR comme le montre le graphique ci-dessous.

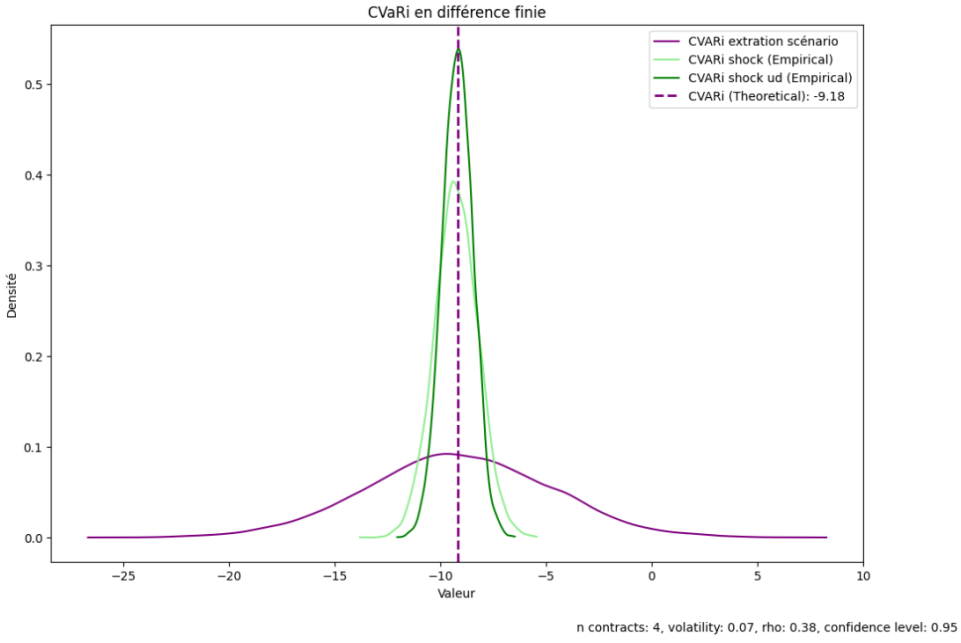


FIGURE 5 – CVaR_i différence finie VS CVaR_i par extraction de scénario

2. Fenêtre kernel[1] : Pour atténuer la variance élevée des résultats, on applique une méthode de lissage par kernel. Le but est d'extraire plusieurs scénarios autour du scénario cible, ie $\{PnL = \text{VaR}\}$, puis de les pondérer. Cette technique permet de lisser les distributions de pertes estimées et de réduire l'impact des variations extrêmes.

$$\text{CVaR}_i \approx \frac{\sum_s K(PnL(s) - \text{VaR}) \cdot PnL_i(s)}{\sum_s K(PnL(s) - \text{VaR})}$$

Les noyaux sont définis comme suit $K : \mathbb{R} \to \mathbb{R}$:

1. **Non-négativité** : $\forall x \in \mathbb{R}, K(x) \geq 0$
2. **Normalisation** : $\int_{-\infty}^{\infty} K(x)\,dx = 1$
3. **Symétrie** : $\forall x \in \mathbb{R}, K(x) = K(-x)$
4. **Largeur de bande h (optionnel)** : $\forall x \in \mathbb{R}, |x| > h \implies K(x) = 0$
5. **Continuité** : $K \in \mathcal{CM}^0(\mathbb{R})$

Nous utilisons trois types de noyaux : le rectangle, le triangle et le gaussien.

1. **Noyau rectangulaire** :

$$K_{\square}(x) = \begin{cases} \frac{1}{2h} & \text{si } \left|\frac{x}{h}\right| \leq 1 \\ 0 & \text{sinon} \end{cases}$$

2. **Noyau triangulaire** :

$$K_{\triangle}(x) = \frac{1}{h} \max\left(1 - \frac{|x|}{h}, 0\right)$$

3. **Noyau gaussien** :

$$K_{\mathcal{N}}(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

où $\sigma = \frac{h}{\sqrt{6}}$. En effet, on veut que la variance du noyau triangulaire est égale à la variance du noyau gaussien. Pour le noyau triangulaire, on calcule l'intégrale sur $\mathbb{R}$:

$$\text{Var}(K_{\triangle}) = \int_{-\infty}^{\infty} x^2 K_{\triangle}(x)\,dx = \frac{h^2}{6}.$$

Ainsi, en égalant cette variance à celle du noyau gaussien, nous retrouvons la formule : $\sigma^2 = \frac{h^2}{6} \implies \sigma = \frac{h}{\sqrt{6}}$. Les formules ci-dessus montrent comment chaque type de noyau est défini et utilisé en fonction de la variable x et de la largeur de bande h. Le papier en bibliographie [1], nous propose d'optimiser la largeur de bande comme suit :

$$h = 2.6 \frac{\sigma_p}{n^{1/5}}$$

où σ_p est la volatilité du portefeuille et n est le nombre de trajectoires.

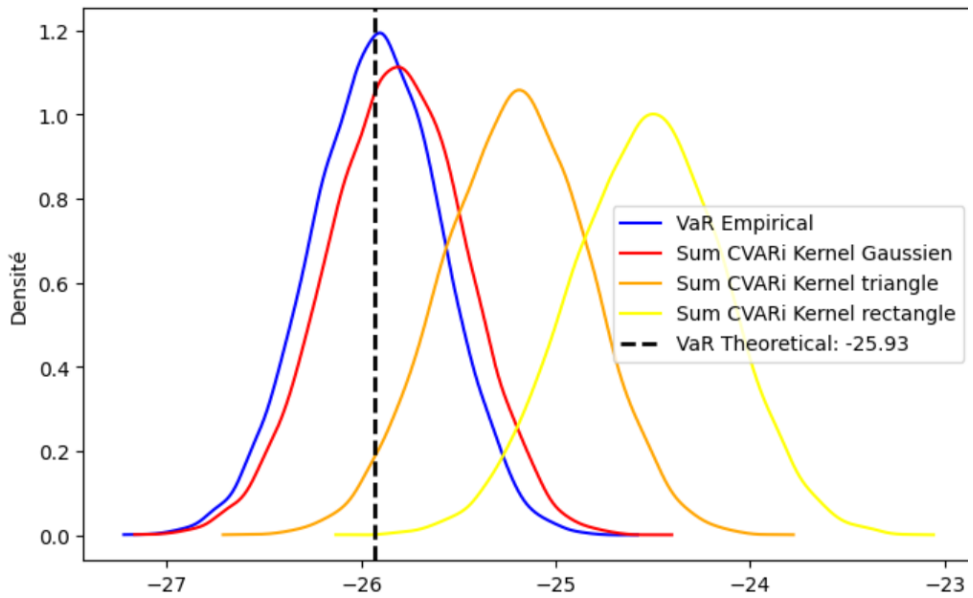


FIGURE 6 – Comparaison des $\sum \text{CVaR}_i^{[K]}$ par rapport à la $\hat{\text{VaR}}_\alpha$

On remarque sur cette figure 5 que la variance des différents noyaux suivent la variance de la VaR empirique $\hat{\text{VaR}}_\alpha$. Néanmoins, ils sont tous plus ou moins biaisés.

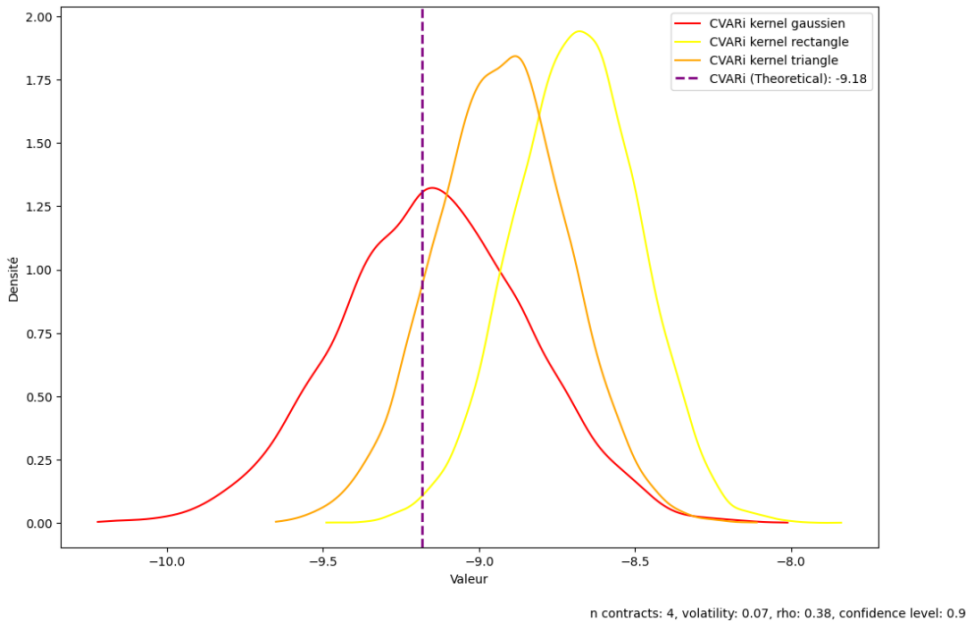


FIGURE 7 – Comparaison des $\text{CVaR}_i^{[K]}$

Le noyau gaussien est moins biaisé par rapport à la VaR mais il a une variance plus importante sur les contributions par rapport aux autres noyaux. Par ailleurs, on notera qu'en observant l'échelle du graphe, la méthode par noyau est bien meilleure que la méthode par différence finie. Dans ce dernier graphe, il serait intéressant de rebaser les contributions à la VaR par noyau triangulaire afin de retirer le biais.

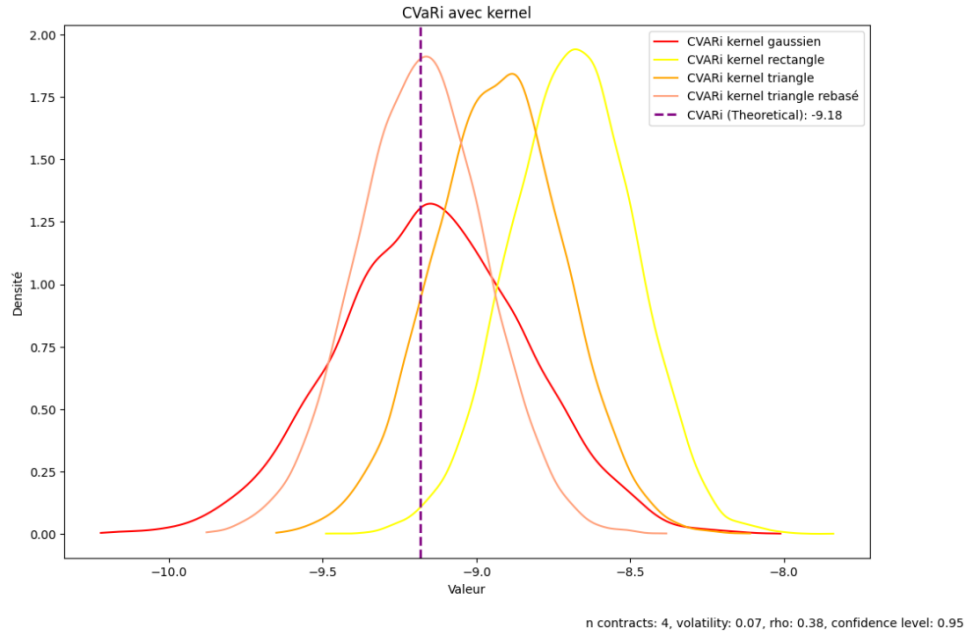


FIGURE 8 – $\text{CVaR}^{[\triangle]}$ rebasé

En rebasant, nous retirons le biais de la $\sum \text{CVaR}_i^{[\triangle]}$ et en plus on obtient une CVaR_i avec une meilleur variance que la méthode du noyau gaussien.

La formule pour la contribution à la VaR par méthode du noyau K rebasé sur la VaR $\text{CVaR}_i[K, R]$ est donnée par :

$$\text{CVaR}_i^{[K,R]} = \text{VaR}\frac{\text{CVaR}_i^{[K]}}{\sum \text{CVaR}_j^{[K]}} = \frac{\sum_{j=1}^{N} K(PL(j) - \text{VaR}; h)PL_i(j)}{\sum_{j=1}^{N} K(PL(j) - \text{VaR}; h)PL(j)}$$

6 Troisième méthode : Correspondance VaR-ES

Cette méthode repose sur la relation entre la VaR et l'Expected Shortfall (ES). Soit X une variable aléatoire qui suit la même loi que nos PnL de telle sorte qu'on ait $X = \sum w_i X_i$ avec le PnL_i le PnL du contrat_i.

$$\text{ES}_\beta = \mathbb{E}[X \mid X \leq \text{VaR}_\beta]$$

On cherche un paramètre β tel que la VaR au niveau de confiance α soit égale à l'Expected Shortfall (ES) à un certain niveau β. On notera qu'intuitivement $\alpha \leq \beta$. L'ES, étant une espérance conditionnelle, est linéaire, ce qui facilite la décomposition en termes de contributions individuelles.

1. Détermination de β : On trouve β tel que $\text{VaR}(\alpha) = \text{ES}(\beta)$. Dans le cadre d'une modélisation numérique, on utilisera tout simplement une dichotomie.
2. Calcul des ES_i : En utilisant la linéarité de l'ES, on peut facilement calculer les contributions individuelles. $\text{ES}_i(\beta) = \mathbb{E}[Xi \mid X \leq VaR(\beta)]$. En effet, pour calculer $\text{ES}(\beta)$ il suffit de prendre $(1-\beta)10^n$ nombre de scénarios, avec 10^n nombre de trajectoires total, correspondant aux PnL $\leq \text{VaR}_\beta$. Puis pour avoir les contributions ES_i il suffit de prendre les PnL_i associé aux PnL $= \text{VaR}_\beta$.

La correspondance entre la $\text{CVaR}_i(\alpha)$ et l'$\text{ES}_i(\beta)$ dans le cas gaussien est vérifié mathématiquement en annexe. Cette méthode de correspondance s'avère souvent être une bonne approximation numérique des CVaR_i. Et d'autre part, il garantit que $\sum ES_i(\beta) = VaR(\alpha)$ dans la limite de la précision de notre dichotomie numérique.

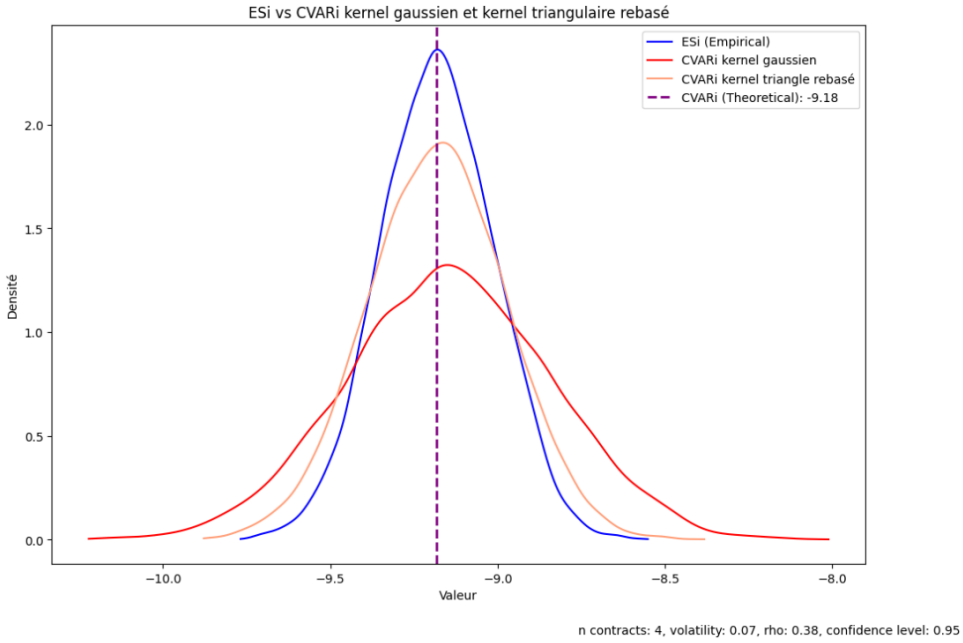


FIGURE 9 – $\text{CVaR}_i^{[K]}$ VS ES_i

La méthode de correspondance entre la VaR et l'ES semble être la plus appropriée pour approximer les contributions à la VaR dans le cadre de loi multivariée gaussienne.

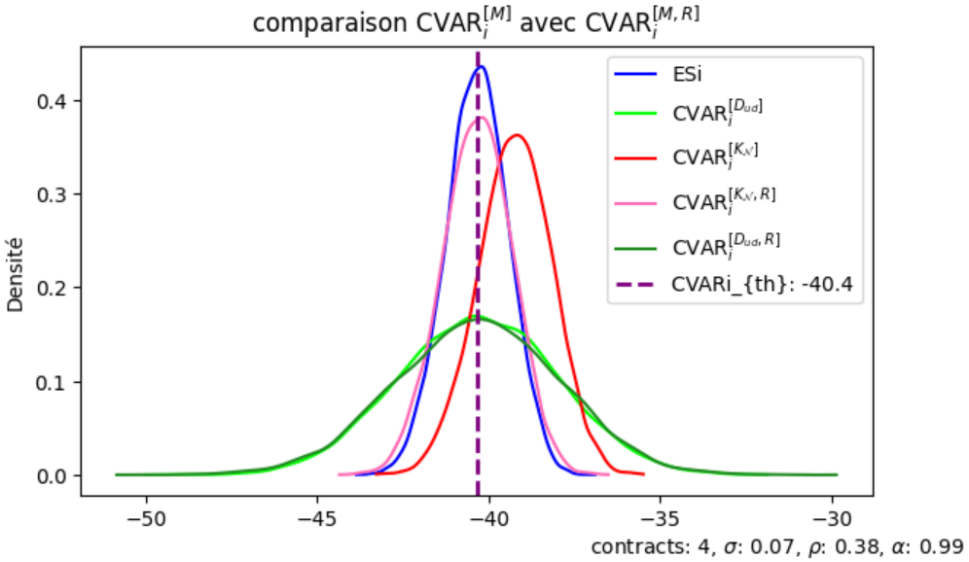
7 Comparaison des différentes méthodes

Pour réduire la variabilité des simulations, cf annexe 9.5, et mettre toutes les méthodes sur un pied d'égalité, nous allons rebaser chaque méthode $M \in \{K : \text{kernel}, D : \text{différence finie}\}$. La version rebasé R de la CVaR pour la méthode M est donnée par :

$$\text{CVaR}_i^{[M,R]} = \frac{\text{CVaR}_i^{[M]}}{\sum_j \text{CVaR}_j^{[M]}} \text{VaR}$$

où $\text{CVaR}_i^{[M]}$ est la CVaR_i pour la méthode M.

On notera qu'il est inutile de regarder les $\sum \text{CVaR}_i^{[M,R]}$. En effet, elles se superposeront à la courbe de la $\hat{\text{VaR}}_\alpha$. Néanmoins, il est intéressant de regarder l'impact sur les CVaR_i de chaque méthode.



Méthode	μ	σ	Min	Max	κ	γ	N	τ (s)
ES_i	-40.3	0.9	-43.9	-36.9	-0.079	-0.068	1200	0.017
$CVaR_i^{[K_{\mathcal{N}},R]}$	-40.3	1.0	-44.4	-36.5	-0.061	-0.080	3033	0.001
$CVaR_i^{[\delta,ud,R]}$	-40.3	2.4	-50.9	-29.9	0.061	-0.054	10000	0.002

TABLE 1 – Tableau des statistiques pour chaque méthode rebasée sur 10000 simulations

En vert, on a les statistiques les plus intéressantes.

— μ (Moyenne) : La moyenne arithmétique des valeurs dans la série de données. Elle donne une indication du centre de la distribution par rapport à la $\text{VaR}_t h = -40,35$.

— σ (Écart type) : La mesure de la dispersion des valeurs par rapport à la moyenne. Un écart type plus petit indique que les valeurs sont plus concentrées autour de la moyenne.

— **Min** : Point le plus à gauche de la distribution.

— **Max** : Point le plus à droite de la distribution.

— κ (Kurtosis normalisée) : Une mesure du "pointu" de la distribution. La kurtosis évalue la hauteur des queues de la distribution. Une valeur de κ proche de 0 indique une distribution normale.

— γ (Skewness) : Une mesure de l'asymétrie de la distribution. Une skewness proche de 0 indique une distribution symétrique.

— **N** : Nombre de trajectoires utilisées pour l'approximation (10 000 trajectoires pour 1 000 itérations du mode opératoire). Cela représente le nombre total d'échantillons ou de simulations sur lesquelles les $\text{CVaR}_i^{[M]}$ ont été calculées. Dans le cadre d'un noyau sur une fenêtre h, 890 trajectoires sont utilisés pour le calcul $\text{CVaR}_i^{[K,\triangle,\square]}$

— τ (s) : Temps de calcul moyen pour les fonctions python, des différentes méthodes de calcul de la CVaR, utilisées en annexe.

8 Conclusion

En conclusion, bien que la VaR présente des limitations en termes d'additivité, les méthodes étudiées permettent de décomposer la VaR du portefeuille en contributions individuelles de manière cohérente, facilitant ainsi la gestion et l'évaluation des risques. Nous pouvons déterminer l'actif qui contribue le plus à la VaR. Parmi ces méthodes, les approches basées sur l'Expected Shortfall (ES) et les noyaux (Kernel) semblent donner les meilleurs résultats, comparées à la méthode par différences finies, qui présente une variance d'estimation plus élevée. De plus, la méthode par ES semble légèrement meilleure que celle par Kernel, d'un point de vue du compromis entre variance et biais de l'estimateur.

Perspectives

Pour aller plus loin dans ce projet, plusieurs pistes peuvent être envisagées :

— Explorer la modification des paramètres d'entrée du mode opératoire : les lois de distribution multivariées (skew, loi inverse gaussienne, log-normale), les paramètres du portefeuille, l'impact de α sur le coefficient relatif de variation $\frac{\sigma_{\text{CvaR}_i}}{\mu_{\text{CvaR}_i}}$...etc. J'ai déjà effectué plusieurs modifications des paramètres d'entrée, ce qui n'avait pas changé la conclusion de ce papier. Néanmoins, une étude approfondie et rigoureuse peut nous permettre une meilleure compréhension du problème.

— Optimiser les hyperparamètres : la largeur de bande h pour la méthode du noyau, le δ pour la différence finie. Pour ces paramètres, j'ai suivi mes tests numériques ou les recommandations d'Epperlein.

— Impact de l'utilisation d'un quasi Monte Carlo sur la vitesse de convergence (Sobol)

Références

[1] Eduardo Epperlein and Alan Smillie, Cracking VaR with Kernels, August, 2006

9 Annexe

9.1 Différence finie à gauche puis à 2n points

Différence finie à gauche :

$$\mathrm{CVaR}_i^{[left]} = w_i \frac{\partial \mathrm{VaR}^{[left]}}{\partial w_i} \approx \frac{\mathrm{VaR}(w_i) - \mathrm{VaR}(w_i - \delta)}{\delta}$$

Cette différence finie semble aussi biaisée mais à gauche comme montrera la comparaison graphique des $\sum \mathrm{CVaR}_i^{[D]}$, ce qui est symétrique avec la différence finie à droite.

Différence finie à 2n points :

La formule pour estimer la dérivée d'une fonction $f(x)$ en utilisant la méthode des différences finies à $2n$ points est donnée par :

$$\frac{1}{\delta} \sum_{i=-n}^{n} c_i f(x + i\delta) \xrightarrow[\delta \to 0]{} f'(x) \text{ où } c_i = \frac{(-1)^i}{i\delta} \sum_{\substack{j=-n \\ j \neq i}}^{n} \frac{(-1)^j}{i - j}$$

- Exemple à 4 points :

$$\frac{\mp \mathrm{VaR}(w_i \pm 2\delta) \pm 8\mathrm{VaR}(w_i \pm \delta)}{12\delta} \xrightarrow[\hat{\delta} \to 12\%^-]{} \mathrm{CVaR}_i$$

- Exemple à 6 points :

$$\frac{\mp \mathrm{VaR}(w_i \pm 3\delta) \pm 9\mathrm{VaR}(w_i \pm 2\delta) \mp 45\mathrm{VaR}(w_i \pm \delta)}{60\delta} \xrightarrow[\hat{\delta} \to 60\%^-]{} \mathrm{CVaR}_i$$

Mon choix de $\hat{\delta}$ est empirique et doit être ajusté en fonction du nombre de points utilisés dans la différence finie. Pour la différence finie centrée, j'ai suivi la recommandation de Epperlein [1], qui conseille un δ de 10%.

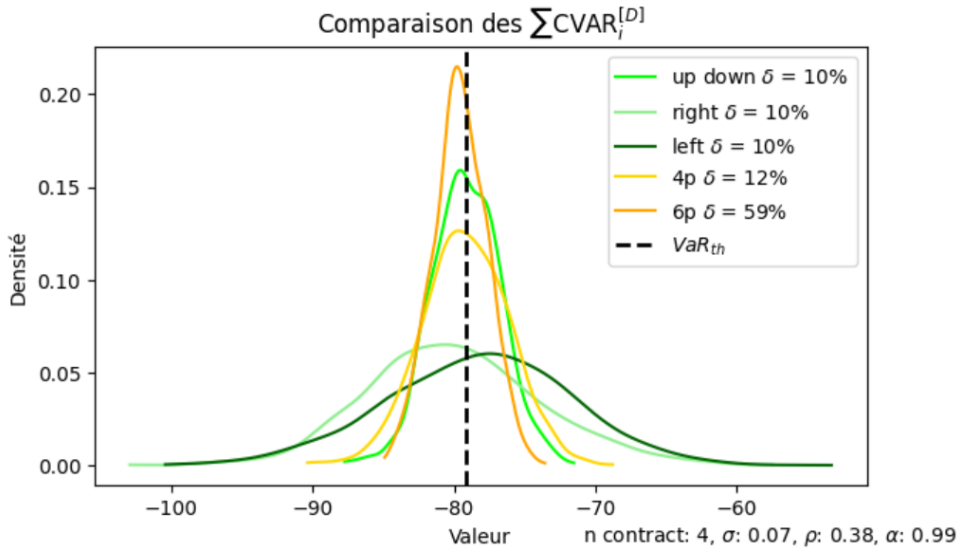


FIGURE 10 – $\mathrm{CVaR}_i^{[D,4p]}$ VS $\mathrm{CVaR}_i^{[D,6p]}$

La différence finie à 6 points semble être la meilleure malgré que $\delta = \frac{2}{3}$. Théoriquement l'approximation est d'ordre 6. Donc plus on augmente le nombre de points de la différence finie, plus l'erreur d'approximation est négligeable. Néanmoins d'un point de vue numérique, il y a des problèmes d'annulation de terme, de précision numérique des fractions...etc. Ici je n'arrive pas à avoir une différence finie à 4 points meilleure que celle à 2 points.

9.2 Noyaux Épanechnikov et Quartic

1. **Noyau Épanechnikov** :

$$K_{\mathrm{E}}(x) = \begin{cases} \frac{3}{4h}\left(1 - \left(\frac{x}{h}\right)^2\right) & \text{si } \left|\frac{x}{h}\right| \leq 1 \\ 0 & \text{sinon} \end{cases}$$

2. **Noyau Quartic** :

$$K_{\mathrm{Q}}(x) = \begin{cases} \frac{15}{16h}\left(1 - \left(\frac{x}{h}\right)^2\right)^2 & \text{si } \left|\frac{x}{h}\right| \leq 1 \\ 0 & \text{sinon} \end{cases}$$

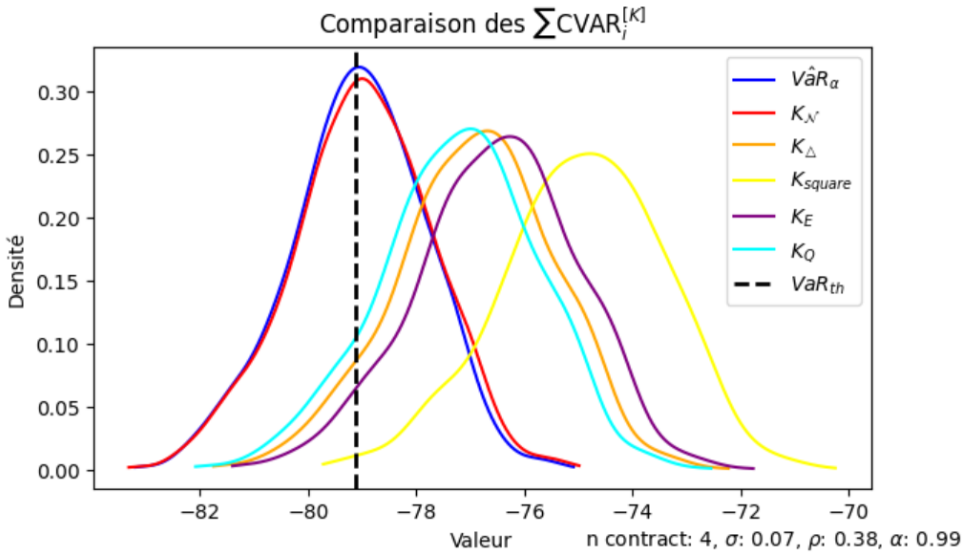


FIGURE 11 – K_E et K_Q

Résumé des Noyaux Comparés :
— $K_{\mathcal{N}}$: Smooth, mais peut être sensible aux valeurs extrêmes.
— K_E : Efficace et optimal pour la variance ; moins sensible aux valeurs extrêmes.
— K_Q : Plus lisse que l'Epanechnikov ; excellent pour la douceur.
— $K_\triangle$: Linéairement décroissant, simple et intuitif.
— $K_\square$: Très simple, mais peut introduire des discontinuités.

9.3 Correspondance VaR-ES cas gaussien

Soit $X = \sum w_i X_i$, où les $X_i \sim \mathcal{N}(\mu_i, \sigma_i)$. Alors $X \sim \mathcal{N}(\mu_X, \sigma_{\eta_G})$ où $\mu_X = \sum w_i \mu_i$ et $\sigma_{\eta_G} = \sqrt{w'\Sigma w}$. Par ailleurs, on notera pour la suite :

$$\forall z \in \mathbb{R}, \quad \Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} \exp\left(-\frac{t^2}{2}\right) dt, \quad \phi(z) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right)$$

Calcul VaR_α et CVaR_α

$$\mathrm{VaR}_\alpha = \mu_X + \sigma_{\eta_G} \Phi^{-1}(1-\alpha)$$

En reprenant la formule des dérivés partielles grâce au théorème d'Euler, on a $\mathrm{CVaR}_\alpha = w \frac{\partial \mathrm{VaR}_\alpha}{\partial w}$. Si on note $\mu = (\mu_i)_i$, on a :

$$\mathrm{CVaR}_\alpha = \mu + \phi^{-1}(1-\alpha)\frac{w^T \Sigma w}{\sigma_{\eta_G}} = \mu + \frac{w^T \Sigma w (\mathrm{VaR}_\alpha - \mu_X)}{\sigma^2_{\eta_G}}$$

Calcul $\mathbf{ES}_\beta$

Nous avons,

$$\mathrm{ES}_\beta = \mathbb{E}[X \mid X \leq \mathrm{VaR}_\beta] = \frac{\int_{-\infty}^{\mathrm{VaR}_\beta} x f_X(x)\,dx}{\int_{-\infty}^{\mathrm{VaR}_\beta} f_X(x)\,dx}$$

avec $X \sim \sigma_{\eta_G}\mathcal{N}(0,1)$. Or,

$$\int_{-\infty}^{\mathrm{VaR}_\beta} f_X(x)\,dx = \int_{-\infty}^{\mathrm{VaR}_\beta} \frac{\exp(-\frac{x^2}{2\sigma_{\eta_G}^2})}{\sqrt{2\pi\sigma_{\eta_G}^2}} = \mathbb{P}(X \leq \mathrm{VaR}_\beta) = 1-\beta.$$

En utilisant le changement de variable $u = \frac{x-\mu_X}{\sigma_{\eta_G}}$, sachant que par convention la VaR est négative, on a bien $u \in \mathrm{Bij}(\,]-\infty, \mathrm{VaR}_\beta],\mathbb{R}\,) \cap C^1(\,]-\infty,\mathrm{VaR}_\beta],\mathbb{R}\,)$. Ainsi, l'intégrale devient :

$$\int_{-\infty}^{\mathrm{VaR}_\beta} x f_X(x)\,dx = \mu_X \Phi\left(\frac{\mathrm{VaR}_\beta - \mu_X}{\sigma_{\eta_G}}\right) - \sigma_{\eta_G}\phi\left(\frac{\mathrm{VaR}_\beta - \mu_X}{\sigma_{\eta_G}}\right)$$

En remplaçant $\frac{\mathrm{VaR}_\beta - \mu_X}{\sigma_{\eta_G}} = \Phi^{-1}(1-\beta)$, on obtient :

$$\mathrm{ES}_\beta = \mu_X - \frac{\sigma_{\eta_G}\phi\left(\Phi^{-1}(1-\beta)\right)}{1-\beta}.$$

Calcul $\mathbf{ES}_i(\beta)$

Nous avons :

$$\mathrm{ES}_i(\beta) = \mathbb{E}[X_i \mid X \leq \mathrm{VaR}_\beta] = \int_{-\infty}^{+\infty} x_i f_{X_i|X\leq \mathrm{VaR}_\beta}(x_i)\,dx_i = \int_{-\infty}^{+\infty} x_i \frac{\int_{-\infty}^{\mathrm{VaR}_\beta} f_{X_i,X}(x_i,x)\,dx}{\int_{-\infty}^{\mathrm{VaR}_\beta} f_X(x)\,dx}\,dx_i$$

Or $f_{X_i|X} = \frac{f_{X_i,X}}{f_X}$, en inversant les intégrales et sachant que $\mathbb{E}[X_i \mid X = x] = \int_{-\infty}^{+\infty} x_i f_{X_i|X=x}(x_i)\,dx_i$:

$$\mathrm{ES}_i(\beta) = \frac{1}{1-\beta}\int_{-\infty}^{\mathrm{VaR}_\beta}\left(\int_{-\infty}^{+\infty} x_i f_{X_i|X=x}(x_i)\,dx_i\right) f_X(x)\,dx = \frac{1}{1-\beta}\int_{-\infty}^{\mathrm{VaR}_\beta} \mathbb{E}[X_i \mid X=x] f_X(x)\,dx$$

Il suffit alors de calculer une extraction de scénario en x :

$$\mathbb{E}[X_i \mid X = x] = \int_{-\infty}^{+\infty} x_i f_{X_i|X}(x_i)\,dx_i = \int_{-\infty}^{+\infty} x_i \frac{f_{X_i,X}(x_i,x)}{f_X(x)}\,dx_i$$

La densité conjointe $f_{X_i,X}$ est :

$$f_{X_i,X}(x_i,x) = \frac{1}{2\pi\sigma_i\sigma_{\eta_G}\sqrt{1-\rho_i^2}}\exp\left(-\frac{1}{2(1-\rho_i^2)}\left[\left(\frac{x_i-\mu_i}{\sigma_i}\right)^2 + \left(\frac{x-\mu_X}{\sigma_{\eta_G}}\right)^2 - 2\rho_i\frac{(x_i-\mu_i)(x-\mu_X)}{\sigma_i\sigma_{\eta_G}}\right]\right),$$

où $\rho_i = \frac{\mathrm{Cov}(X_i,X)}{\sigma_i\sigma_{\eta_G}}$.

La densité marginale f_X est : $f_X(x) = \exp\left(-\frac{(x-\mu_X)^2}{2\sigma_{\eta_G}^2}\right)/\sqrt{2\pi\sigma_{\eta_G}^2}$.

Substituons les expressions des densités conjointe et marginale, regroupons les exponentielles puis appliquons une idendité remarquable pour simplifier l'expression :

$$f_{X_i|X}(x_i \mid x) = \frac{1}{\sqrt{2\pi\sigma_i^2(1-\rho_i^2)}}\exp\left(-\frac{1}{2}\left(\frac{x_i - \mu_i - \rho_i\frac{\sigma_i}{\sigma_{\eta_G}}(x-\mu_X)}{\sigma_i\sqrt{1-\rho_i^2}}\right)^2\right).$$

On reconnaît alors une forme canonique, ainsi $X_i \mid X \sim \mathcal{N}\left(\mu_i + \rho_i\frac{\sigma_i}{\sigma_{\eta_G}}(x-\mu_X), \sigma_i^2(1-\rho_i^2)\right)$, d'où :

$$\mathbb{E}[X_i \mid X = x] = \mu_i + \rho_i\frac{\sigma_i}{\sigma_{\eta_G}}(x-\mu_X).$$

Introduisons cette relation dans l'expression de $\mathrm{ES}_i(\beta)$:

$$\mathrm{ES}_i(\beta) = \frac{1}{1-\beta}\left[\int_{-\infty}^{\mathrm{VaR}_\beta}\mu_i f_X(x)\,dx + \rho_i\frac{\sigma_i}{\sigma_{\eta_G}}\int_{-\infty}^{\mathrm{VaR}_\beta}(x-\mu_X)f_X(x)\,dx\right].$$

Sachant que :

$$\int_{-\infty}^{\mathrm{VaR}_\beta} f_X(x)\,dx = 1-\beta \text{ et } \int_{-\infty}^{\mathrm{VaR}_\beta} x f_X(x)\,dx = \mu_X \cdot (1-\beta) - \sigma_{\eta_G}\phi(\Phi^{-1}(1-\beta)),$$

Nous avons alors :

$$\mathrm{ES}_i(\beta) = \frac{1}{1-\beta}\left[\mu_i(1-\beta) + \rho_i \frac{\sigma_i}{\sigma_{\eta_G}}\left[\mu_X \cdot (1-\beta) - \sigma_{\eta_G}\phi(\Phi^{-1}(1-\beta)) - \mu_X \cdot (1-\beta)\right]\right].$$

Simplifions davantage en rappelant $\rho_i = \frac{\mathrm{Cov}(X_i,X)}{\sigma_i \sigma_{\eta_G}} = \frac{(\Sigma w)_i}{\sigma_i \sigma_{\eta_G}}$:

$$\mathrm{ES}_i(\beta) = \mu_i - \frac{(\Sigma w)_i \phi(\Phi^{-1}(1-\beta))}{\sigma_{\eta_G}(1-\beta)}.$$

Ainsi, en isolant $\mathrm{ES}_\beta - \mu_X$, puis en rappelant qu'on a choisi β tel que $\mathrm{ES}_\beta = \mathrm{VaR}_\alpha$, nous obtenons :

$$\mathrm{ES}_i(\beta) = \mu_i + \frac{(\Sigma w)_i(\mathrm{VaR}_\alpha - \mu_X)}{\sigma^2_{\eta_G}} = \mathrm{CVaR}_i(\alpha)$$

9.4 VaR individuelle rebasée

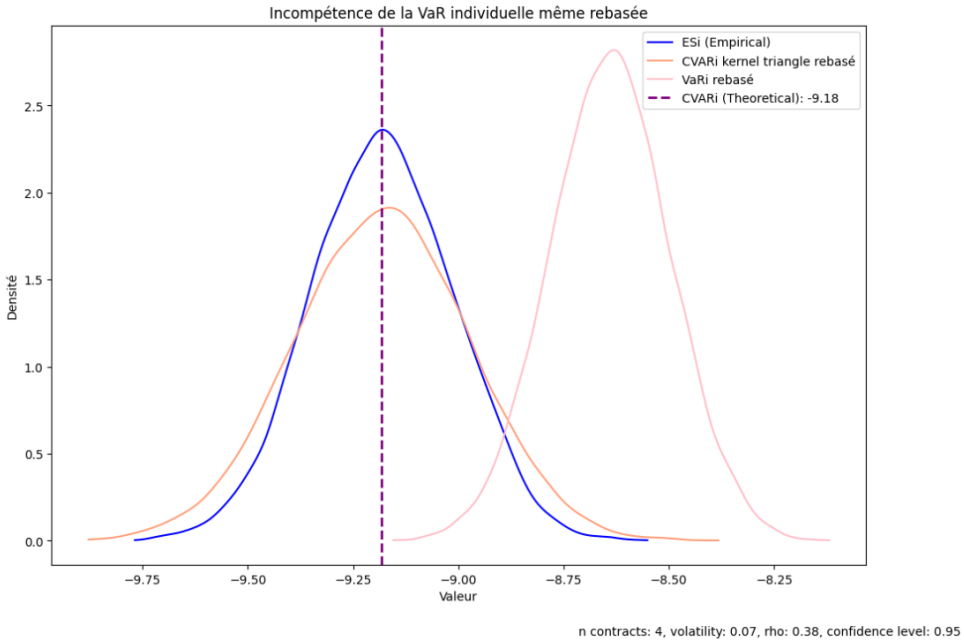


FIGURE 12 – Comparaison de la VaR_i^R avec les autres $CVaR_i$

On peut voir que même en rebasant la VaR individuelle, de telle sorte que sa somme soit égale à la VaR, chaque terme de la somme présente un biais par rapport à sa CVaR_i.

$$\mathrm{VaR} = \sum_i \frac{\mathrm{VaR}_i.\mathrm{VaR}}{\sum_i \mathrm{VaR}_i}$$

Le terme individuel rebasé $\mathrm{VaR}_i^{[}R]$ serait alors :

$$\mathrm{VaR}_i^{[}R] = \frac{\mathrm{VaR}_i.\mathrm{VaR}(P)}{\sum_i \mathrm{VaR}_i}$$

9.5 Vitesse de convergence en $1/\sqrt{n}$

On notera que la VaR empirique $\hat{\mathrm{VaR}}_\alpha$ a une certaine erreur par rapport à la VaR théorique VaR_{th}. Cette erreur est principalement due à la variabilité inhérente aux simulations. Le théorème de la limite centrale (TLC) pour les quantiles nous dit que la $\hat{\mathrm{VaR}}_\alpha$ converge en distribution vers une normale. Plus précisément, nous avons :

$$\sqrt{n}\left(\hat{\mathrm{VaR}}_\alpha - \mathrm{VaR}_\alpha\right) \xrightarrow{d} \mathcal{N}(0, \sigma_\alpha^2),$$

où σ_α^2 est la variance asymptotique de $\hat{\text{VaR}}_\alpha$. On retrouve bien cela sur le graphique suivant en échelle logarithmique.

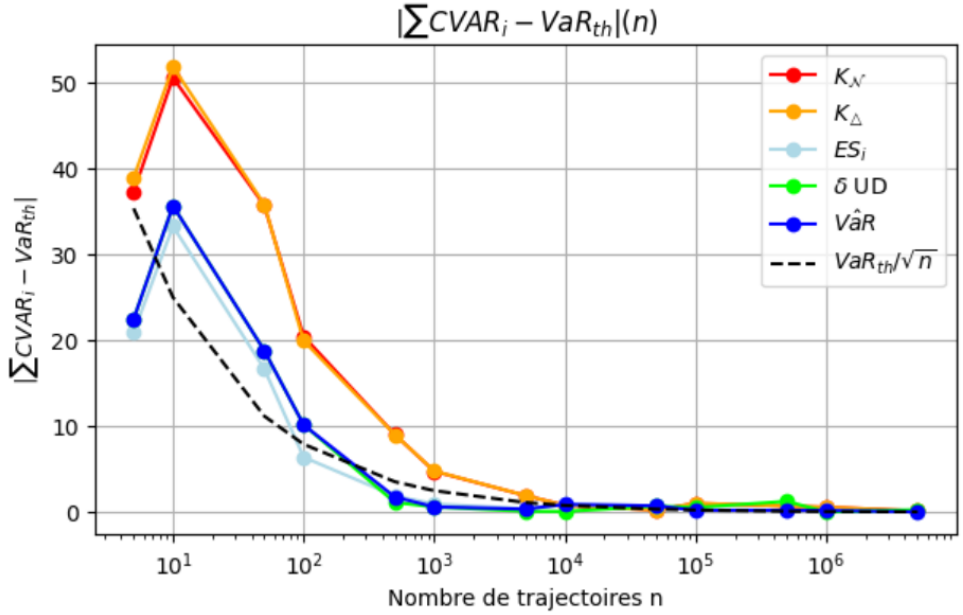


FIGURE 13 – Erreur absolue

On retrouve bien, selon les différents estimateurs $\hat{\text{VaR}}_\alpha$, une décroissance en $1/\sqrt{n}$. Rq : J'ai choisi un nombre de trajectoire de 10000 en accord avec [1], les tableaux excel et les temps de calcul sur python.

9.6 Code

```
import numpy as np
from scipy.stats import gaussian_kde, shapiro, kstest, norm, qmc
import matplotlib.pyplot as plt
import math
import time

# Exemple d'utilisation
n_components = 4 #Nombre de contrat
nb_traj = 10000  # Nombre de trajectoire
# Exemple de volatilite a 10 pourcent annuel a horizon 6 mois
volatility = 0.1 * math.sqrt(0.5)
rho = 0.38  # Coefficient de correlation unique
#On notera une sur-exposition aux marches europeens
confidence_level = 0.99 #a titre d'exemple
nominales = np.array([100, 100, 50, 50])  # Nominales des actifs a titre d exemple

# Matrice de correlation
corr_matrix = np.full((n_components, n_components), rho)
np.fill_diagonal(corr_matrix, 1)
# Vecteur de poids des actifs
weights = np.ones(n_components) / n_components
# Generation des rendements aleatoires
mean_returns = np.zeros(n_components)
cov_matrix = np.diag([volatility] * n_components)
            @ corr_matrix
            @ np.diag([volatility] * n_components)
#volatilite sur tout le portefeuille
portfolio_volatility = np.sqrt(np.dot(nominales,
                               np.dot(cov_matrix, nominales)))
returns = np.random.multivariate_normal(mean_returns,
```

```
            cov_matrix, nb_traj)
#returns = np.exp(returns)
#skew_params = np.array([5, -3, 2,2,2])#np.ones(5)*0
#returns = generate_skewed_multivariate(mean_returns,
            cov_matrix, nb_traj, skew_params)

# Calcul des P&L pour chaque contrat
PnL_matrix = nominales * returns


# Fonctions existantes
def kernel_function(x, kernel_type="gaussian"):
    bandwidth = 2.6 * portfolio_volatility / (nb_traj) ** (1/5)
    x_scaled = x / bandwidth
    if kernel_type == "square":
        return np.where(np.abs(x_scaled) <= 1, 1/(2*bandwidth), 0)
    elif kernel_type == "triangle":
        return np.where(np.abs(x_scaled) <= 1,
        (1-np.abs(x_scaled))/bandwidth, 0)
    elif kernel_type == "gaussian":
        stdev = bandwidth / math.sqrt(6)
        return stdev * norm.pdf(x)
    elif kernel_type == "epanechnikov":
        return np.where(np.abs(x_scaled) <= 1,
        3/(4*bandwidth)*(1-x_scaled**2),0)
    elif kernel_type == "quartic":
        return np.where(np.abs(x_scaled) <= 1,
        15/(16*bandwidth)*(1-x_scaled**2)**2,0)
    else: raise ValueError("Unsupported kernel type")

def calculate_VaR(PnL_total, confidence_level=0.95):
    #attention np.percentile prend des valeurs de percentile en pourcentage
    return np.percentile(PnL_total, (1 - confidence_level)* 100)

def calculate_es(pnl_list, beta, lets_go=False):
    var_beta = np.percentile(pnl_list, (1 - beta)* 100)
    pnl_tail = [x for x in pnl_list if x <= var_beta]
    return np.mean(pnl_tail)

def calculate_CVaRi(PnL_matrix, confidence_level=0.95, kernel_type):
    n_scenarios, n_components = PnL_matrix.shape
    PnL_total = np.sum(PnL_matrix, axis=1)
    VaR = calculate_VaR(PnL_total, confidence_level)
    distances = (PnL_total - VaR)
    weights = kernel_function(distances, kernel_type)
    weights /= np.sum(weights)
    CVaRi = np.zeros(n_components)
    for i in range(n_components):
        CVaRi[i] = np.sum(weights * PnL_matrix[:, i])
    return CVaRi, VaR

def extract_scenarios(PnL_matrix, confidence_level=0.95):
#mean(PnLi tel que PnL = VaR(alpha))
    n_scenarios, n_components = PnL_matrix.shape
    PnL_total = np.sum(PnL_matrix, axis=1)
    VaR = calculate_VaR(PnL_total, confidence_level)
    scenario_indices = np.where(PnL_total == VaR)[0]
    if len(scenario_indices) == 0:
        closest_index = np.argmin(np.abs(PnL_total - VaR))
        scenario_indices = [closest_index]
    PnL_scenarios = PnL_matrix[scenario_indices, :]
```

```
    return PnL_scenarios,VaR

def calculate_es_individual_from_beta(pnl_matrix, beta):
#ESi(beta) = mean( PnLi tel que PnL < VaR(beta))
    _, num_components = pnl_matrix.shape
    es_list = np.zeros(num_components)
    var_beta = np.percentile(np.sum(pnl_matrix, axis=1), (1 - beta)* 100)
    pnl_filtered_indices = np.sum(pnl_matrix, axis=1) <= var_beta
    for i in range(num_components):
        pnl_component = pnl_matrix[pnl_filtered_indices, i]
        pnl_component_sorted = np.sort(pnl_component)
        es_component = np.mean(pnl_component_sorted)
        es_list[i] = es_component
    return es_list

def find_beta_for_es_equals_var(pnl_list, alpha, tol=1e-5):
#on cherche beta tel que VaR(alpha)=ES(beta)
    target_var = calculate_VaR(pnl_list, alpha)
    alpha = 1 - alpha
    low_beta = 0
    high_beta = 4*alpha#depend du contexte dichotomique
    while high_beta - low_beta > tol:
        mid_beta = (low_beta + high_beta) / 2
        es_mid_beta = calculate_es(pnl_list, 1-mid_beta)
        if es_mid_beta > target_var:
            high_beta = mid_beta
        else:
            low_beta = mid_beta
    return (low_beta + high_beta) / 2

portfolio_volatility = np.sqrt(np.dot(nominales,
                                      np.dot(covariance_matrix, nominales)))

def calculate_theoretical_VaR(nominales, covariance_matrix, confidence_level):
    z = norm.ppf(confidence_level)#1.6448
    VaR_theoretical = -z * portfolio_volatility
    return VaR_theoretical
#nominales = np.array(List[int])
def calculate_theoretical_CVaRi(nominales, covariance_matrix, confidence_level):
    z = norm.ppf(confidence_level)#1.6448
    marginal_contributions=covariance_matrix @ nominales
    CVaRi_theoretical=-z * marginal_contributions/portfolio_volatility
    return CVaRi_theoretical*nominales

VaR_initial = calculate_VaR(PNL.sum(axis=1), confidence_level)

def calculate_CVaR_shock(PNL, returns, nominales,
                         confidence_level=0.95, shock=1):
    CVaR_i = np.zeros(len(nominales))
    for i in range(len(nominales)):#nbr de contrat
        nominales_perturbees = nominales.copy()
        nominales_perturbees[i] *=(1+(shock / 100))
        PNL_perturbe = nominales_perturbees * returns
        VaR_perturbee = calculate_VaR(np.sum(PNL_perturbe, axis=1),
                                      confidence_level)
        CVaR_i[i] = (VaR_perturbee - VaR_initial) / (shock / 100)
    return CVaR_i

def calculate_CVaR_shock_up_down(PNL, returns, nominales,
                                 confidence_level=0.95,shock=1):
    CVaR_i = np.zeros(len(nominales))
```

```
    for i in range(len(nominales)):  # nombre de contrats
        nominales_perturbees_up   = nominales.copy()
        nominales_perturbees_down = nominales.copy()
        nominales_perturbees_up[i]   *= (1 + (shock / 100))
        nominales_perturbees_down[i] *= (1 - (shock / 100))
        PNL_perturbe_up = nominales_perturbees_up     * returns
        PNL_perturbe_down = nominales_perturbees_down * returns
        VaR_perturbee_up = calculate_VaR(np.sum(PNL_perturbe_up, axis=1),
                                         confidence_level)
        VaR_perturbee_down = calculate_VaR(np.sum(PNL_perturbe_down, axis=1),
                                         confidence_level)
        CVaR_i[i] = (VaR_perturbee_up - VaR_perturbee_down)
                    / (2 * (shock / 100))
    return CVaR_i
def calculate_cvar_shock_4p(PNL, returns, nominales,
confidence_level=0.99, shock=1):
    VaR_initial = calculate_VaR(PNL.sum(axis=1), confidence_level)
    n = len(nominales)
    CVAR_i = np.zeros(n)
    delta = shock / 100.0
    for i in range(n):
        nominales_plus_2 = nominales.copy()
        nominales_plus_1 = nominales.copy()
        nominales_moins_1 = nominales.copy()
        nominales_moins_2 = nominales.copy()
        nominales_plus_2[i] *= (1 + 2 * delta)
        nominales_plus_1[i] *= (1 + delta)
        nominales_moins_1[i] *= (1 - delta)
        nominales_moins_2[i] *= (1 - 2 * delta)
        VaR_plus_2 = calculate_VaR((nominales_plus_2 *
        returns).sum(axis=1), confidence_level)
        VaR_plus_1 = calculate_VaR((nominales_plus_1 *
        returns).sum(axis=1), confidence_level)
        VaR_moins_1 = calculate_VaR((nominales_moins_1 *
        returns).sum(axis=1), confidence_level)
        VaR_moins_2 = calculate_VaR((nominales_moins_2 *
        returns).sum(axis=1), confidence_level)
        CVAR_i[i] = -(VaR_plus_2 - 8 * VaR_plus_1 + 8 *
        VaR_moins_1 - VaR_moins_2) / (12 * delta)
    return CVAR_i

def calculate_cvar_shock_6p(PNL, returns, nominales,
confidence_level=0.99, shock=1):
    VaR_initial = calculate_VaR(PNL.sum(axis=1), confidence_level)
    n = len(nominales)
    CVAR_i = np.zeros(n)
    delta = shock / 100.0
    for i in range(n):
        nominales_plus_3 = nominales.copy()
        nominales_plus_2 = nominales.copy()
        nominales_plus_1 = nominales.copy()
        nominales_moins_1 = nominales.copy()
        nominales_moins_2 = nominales.copy()
        nominales_moins_3 = nominales.copy()
        nominales_plus_3[i] *= (1 + 3 * delta)
        nominales_plus_2[i] *= (1 + 2 * delta)
        nominales_plus_1[i] *= (1 + delta)
        nominales_moins_1[i] *= (1 - delta)
        nominales_moins_2[i] *= (1 - 2 * delta)
        nominales_moins_3[i] *= (1 - 3 * delta)
        VaR_plus_3 = calculate_VaR((nominales_plus_3 *
```

```
        returns).sum(axis=1), confidence_level)
        VaR_plus_2 = calculate_VaR((nominales_plus_2 *
        returns).sum(axis=1), confidence_level)
        VaR_plus_1 = calculate_VaR((nominales_plus_1 *
        returns).sum(axis=1), confidence_level)
        VaR_moins_1 = calculate_VaR((nominales_moins_1 *
        returns).sum(axis=1), confidence_level)
        VaR_moins_2 = calculate_VaR((nominales_moins_2 *
        returns).sum(axis=1), confidence_level)
        VaR_moins_3 = calculate_VaR((nominales_moins_3 *
        returns).sum(axis=1), confidence_level)
        CVAR_i[i] = (VaR_plus_3 - 9 * VaR_plus_2 + 45 *
        VaR_plus_1 - 45 * VaR_moins_1 + 9 * VaR_moins_2 -
        VaR_moins_3) / (60 * delta)
    return CVAR_i
```

9.7 Temps de calcul

1. Temps moyen pour la fonction Python `calculate_CVaR_shock` : 0.002 secondes. Pour chaque contrat je vais choquer les $n_{trajectoires}$ de mon PnL_i : O($n_{contrat}$*$n_{trajectoires}$)
2. Temps moyen pour la fonction Python `calculate_CVaRi_kernel` : 0.001 secondes. J'applique une fenêtre qu'à un nombre limité de point pour chaque contrat : O($n_{contrat}$)
3. Temps moyen pour la fonction Python `calculate_es_individual_from_beta` : 0.017 .secondes On applique d'abord une dichotomie pour trouver β, tel que $ES_\beta = V\hat{a}R_\alpha$. Complexité $O(log_2(\frac{4(1-\alpha)}{\epsilon})$ Puis je fais la moyenne de $(1-\beta)n$ trajectoires pour chaque contrat. Finalement ma complexité est de : $O(log_2(frac4(1-\alpha)\epsilon)$ + O($n_{contrat}$*$(1-\beta)n_{trajectoires}$)

9.8 Contact

— chemin jupyter notebook : `U:\DFIN\DFINB\10-QD & SI METIER\PR_QUALITE_DONNEES\ORGANISATION\Contrats RH_QD\Maxime JACTAT\method CVAR analysis.ipynb`
— mail pro jusqu'au 31/08/2024 : maxime.jactat@caissedespots.fr
— mail perso : maxime.jactat@gmail.com
— LinkedIn: Maxime Jactat.
— Github: CVAR-KD-analysis