Projet 2 MACS205 - Weighted Schemes in AIS

Julien Béguinot, Mathis Fitoussi

8 juillet 2024

Table des matières

1 Introduction : méthodes de Monte Carlo et Adaptive Importance Sampling

1.1 Les méthodes de Monte-Carlo

On rappels les résultats et le contexte générales des méthodes de **Monte-Carlo**. Ces méthodes sont utiles dans un large éventails de contextes scientifiques.En particulier c'est utile pour évaluer des intégrales de la forme $\int g f$ où f est une densité de probabilité.

Data: Nombre de particules n

1. On génère $X_1, ..., X_n$ de manière i.i.d selon f
2. On calcule

$$\hat{I}^{mc} = n^{-1} \sum_{i=1}^{n} g(X_i)$$

Result:

$$\hat{I}^{mc}$$

Algorithm 1: Algorithme de Monte Carlo

Ce genre d'estimateur est motivé par la loi des grands nombres et une variété de résultat en montre l'intérêt. En particulier pour des estimations dintégrales dans des espaces de grandes dimensions.

1.2 Importance Sampling

Les méthodes de Monte-Carlo on des limites. L'une d'elle et que l'on ne peut pas forcément générer des échantillons selon la loi f. On introduit alors une méthode apellé Importance Sampling. Si on considère une densité q pour laquel on peut générer des échantillons dont le support contient celui de f alors on exploite la formule $\int f g = \int \frac{fg}{q} q$.

L'algorithme d'importance sampling est alors,

Data: Nombre de particules n. Un échantilloneur q.

1. On génère $X_1, ..., X_n$ de manière i.i.d selon q
2. On calcule

$$\hat{I}^{IS} = n^{-1} \sum_{i=1}^{n} \frac{g(X_i) f(X_i)}{q(X_i)}$$

Result:

$$\hat{I}^{IS}$$

Algorithm 2: Algorithme d'Importance Sampling

1.3 L'adaptive Important Sampling

Lorsqu'on utilise une méthode d'Import Sampling une question naturelle apparait. Comment bien choisir la densité q? En effet, il apparait que plus q est proche de f plus l'estimateur sera efficace. On peut par exemple chercher à prendre q minimisant la divergence de Kullbackl-Leibler avec f. Mais cette minimisation induit également un calcul d'intégrale. On va donc améliorer notre choix de q au fur et à mesure de l'algorithme. C'est l'idée de l'**Adaptive Important Sampling**.

Cette remarque est notament justifié par les lemmes techniques montré dans [3].

Lemme 1. *In the AIS context the optimal sampler* $q \propto |g|$

Lemme 2. *In the bayseian AIS context the optimal sampler* $q \propto |g - \int g f| f$

On part donc d'une denisté q_0 puis au bout de n_0 itérations on choisit q_1 puis après n_1 itérations on détermine q_2 et ainsi de suite. la suite $(q_n)_{n\in\mathbb{N}}$. Cet algorithme est donc caractérisé par la suite $(n_t)_{t\in\mathbb{N}}$, la taille des **batch** et par la suite de densité $(q_n)_{n\in\mathbb{N}}$, la **politique**.

Pour un nombre de particules $N = \sum_{t=1}^{T} n_t$ fixé on doit donc considérer un compromis entre un grand nombre T de **mini-batch** ou un plus petit nombre de grand batch.

Une autre idée est la suivante. Pourquoi accorder la même importance à chaque batch alors que le choix de q est censé s'améliorer au cours des batchs? Ainsi, il faudrait mieux favoriser les derniers batchs et bien moins les premiers. C'est dans cette esprit qu'on introduit une méthode ponderés de l'Adaptive Importance Sampling, le **weigthed Adaptive Importance Sampling**.

Une nouvelle question se pose alors. Comment choisir au mieux les poids de cette méthode pondérée? C'est la question qui va nous intéresser dans la suite de ce rapport. Notamment à travers l'analyse des résultats de [3] (Portier et al.) et de [1] (Owen et al.). En particulier, on étends un résultats de [3] au cas pondérés.

2 Notations

On cherche à estimer via une méthode de Monte Carlo : $I = \int g f d\lambda$ où $g : \mathbb{R} \longrightarrow \mathbb{R}$ et f est une densité.

A chaque étape t, $t \in \{1, ..., T\}$ de l'algorithme d'échantillonage préférentiel adaptatif, on utilise la politique q_{t-1} pour générer n_t variables.

On note les variables $X_{n_{t-1}+1}, ..., X_{n_t}$, où $N = \sum_{i=1}^{T} n_t$.

L'estimation de l'étape t s'écrit donc :

$$\hat{I}_t = \frac{1}{t} \sum_{i=1}^{t} \frac{fg}{q_{t-1}}(X_i)$$

Dans le cadre bayésien qui nous intéresse, nous n'avons pas accés à la densité f. En effet nous obtenons f grace à la règle de Bayes donc à normalisation près. Ainsi, en général, nous avons accès à une version non normalisée de f, f_u tel que :

$$f = \frac{f_u}{\int f_u}$$

L'estimation de l'étape t s'écrit alors :

$$\hat{I}_t^{(norm)} = \sum_{i=1}^{t} \frac{f_u g}{q_{t-1}}(X_i) \Big/ \sum_{i=1}^{t} \frac{f_u}{q_{t-1}}(X_i)$$

En particulier,

$$\hat{I}_t^{(norm)} = \frac{\hat{I}_t(gf_u)}{\hat{I}_t(f_u)}$$

On cherche dans notre cas une pondération de ces estimations, qui s'écrit :

$$\hat{I}_\alpha^T = \sum_{t=1}^{T} \alpha_{T,t} \hat{I}_t$$

Si c'est un barycentre i.e. $\sum_{t=1}^{T} \alpha_{T,t} = 1$ alors on remarque notamment que c'est encore un estimateur non biaisé de I.

Ou encore dans le cas bayésien,

$$\hat{I}_\alpha^{T,(norm)} = \sum_{t=1}^{T} \alpha_{T,t} \hat{I}_t^{(norm)}$$

Ajoutons les notations suivantes qui décrivent l'algorithme d'échantillonage préférentiel adaptatif :

$$\Delta_t = \frac{fg}{q_{t-1}}(X_t) - I$$

$$M_n = \sum_{t=1}^{n} \Delta_j$$

Autes notations :

$$\mathrm{Var}(\hat{I}_t) = \sigma_t^2$$

Data: Le nombre T d'étape de l'AIS, la politique $(n_t)_{t=1,\dots,T}$ et une distribution initiale q_0
$S_0 = 0$ $N_0 = 0$
for *t=1,...,T* **do**
`/* Phase d'exploration */`
On tire $(X_{t,1}, \dots, X_{t,n_t})$ selon q_{t-1}
`/* Phase d'exploitation */`
`/* Mise à jour de l'estimateur */`

$$S_t = S_{t-1} + \sum_{i=1}^{n_t} \frac{f(X_{t,i})g(X_{t,i})}{q_{t-1}(X_{t,i})}$$

$$N_t = N_{t-1} + n_t$$

$$I_t = N_t^{-1} S_t$$

`/* Mise à jour de l'échantilloneur` q_t `*/`
On calcul q_t comme l'élément minimisant la divergence de Kullback-Leibler avec f. Cette divergence de KL étant estimé grâce à q_{t-1}
end
Result:

Algorithm 3: AIS à l'échelle de la politique

Data: Le nombre T d'étape de l'AIS, la politique $(n_t)_{t=1,\dots,T}$ et une distribution initiale q_0
for *j=1,...,N* **do**

`/* Phase d'exploration */`

On tire x_j selon q_{j-1}

`/* Phase d'exploitation */`

`/* Mise à jour de l'estimateur */`

$$S_j = S_{j-1} + \frac{f(X_j)g(X_j)}{q_{j-1}(X_j)}$$

$$I_j = j^{-1} S_j$$

`/* Mise à jour de l'échantilloneur` q_j `*/`

Si $j \in \{N_t = \sum_{s=1}^{t} n_s | t \geq 1\}$ on met à jour l'échantilloneur.

end
Result:

Algorithm 5: AIS à l'échelle d'un tirage

3 Poids optimaux

Le problème peut être vu comme celui de la minimisation de la variance de $\hat{I}$ sous contrainte.

$$\text{Var}(\hat{I}) = \sum_{t=1}^{T} \alpha_t^2 \sigma_t^2$$

On peut donc décrire le Lagrangien :

$$L((\alpha_i)_i, \beta) = \text{Var}(\hat{I}) - \beta\left(\sum_{t=1}^{T} \alpha_t - 1\right) = \sum_{t=1}^{T} \alpha_t^2 \sigma_t^2 - \beta\left(\sum_{t=1}^{T} \alpha_t - 1\right)$$

Soit $t \in \{1, ..., T\}$, dérivons cette expression par rapport à α_t et annulons la dérivée :

$$\frac{\partial L}{\partial \alpha_t}((\alpha_i)_i, \beta) = 0 = 2\alpha_t \sigma_t^2 - \beta$$

$$\implies \forall t \in \{1, ..., T\}, \alpha_t \propto \sigma_t^{-2}$$

4 Propriétés de Martingales

Dans cette section on retrouve les résultats donnés dans [3].
Montrons que (M_n) est une martingale pour la filtration $F_n = \sigma(X_1, ..., X_n)$:

$$\mathbb{E}(M_n | F_{n-1}) = \sum_{j=1}^{n} \mathbb{E}\left(\Delta_j | F_{n-1}\right)$$

$$\mathbb{E}(M_n | F_{n-1}) = \sum_{j=1}^{n-1} \mathbb{E}\left(\Delta_j | F_{n-1}\right) + \mathbb{E}(\Delta_n | F_{n-1})$$

$$\mathbb{E}(M_n | F_{n-1}) = M_{n-1} + \mathbb{E}\left(\frac{fg}{q_{n-1}}(X_n) - I | X_1, ..., X_{n-1}\right)$$

On sait que conditionnellement à $X_1, ..., X_{n-1}$, X_n suit la loi q_{n-1}, et donc :

$$\mathbb{E}(M_n | F_{n-1}) = M_{n-1} + \underbrace{\int \frac{fg}{q_{n-1}}(x) \times q_{n-1}(x) dx}_{=I} - I = M_{n-1}$$

Calculons l'erreur quadratique moyenne commise à chaque étape. Remarquons d'abord l'identité suivante :

$$\hat{I}_t = \frac{1}{n_t} \sum_{i=1}^{n_t} \frac{fg}{q_{t-1}}(X_i) = \frac{1}{n_t} \sum_{i=1}^{n_t} (\Delta_i + I) \implies \hat{I}_t - I = \frac{M_n}{n}$$

Ainsi,

$$EQM(t) = \mathbb{E}\left[\left(\hat{I}_t - I\right)^2\right] = \frac{1}{n^2}\mathbb{E}\left(M_n^2\right)$$

Or on sait que $\mathbb{E}\left(M_n^2\right)$ se calcule avec la partie prédictible de la décomposition de Doob de M_n^2. Procédons à son calcul :

$$\langle M \rangle_n = \sum_{j=1}^{n} \mathbb{E}\left(\Delta M_j^2 | F_{j-1}\right)$$

$$\langle M\rangle_n = \sum_{j=1}^{n} \mathbb{E}\left(\Delta_j^2 | F_{j-1}\right)$$

$$\langle M\rangle_n = \sum_{j=1}^{n} \mathbb{E}\left(\left(\frac{f^2g^2}{q_{j-1}^2}(X_j) + I^2 - 2I\frac{fg}{q_{j-1}}(X_j)\right)|F_{j-1}\right)$$

On sait que conditionnellement à $X_1,...,X_{n-1}$, X_n suit la loi q_{n-1}, et donc $\mathbb{E}\left(\frac{fg}{q_{j-1}}(X_j)|F_{j-1}\right) = I$. Ainsi,

$$\langle M\rangle_n = -nI^2 + \sum_{j=1}^{n} \mathbb{E}\left(\left(\frac{f^2g^2}{q_{j-1}^2}(X_j)\right)|F_{j-1}\right)$$

De même,

$$\langle M\rangle_n = -nI^2 + n\int \frac{f^2g^2}{q_{j-1}}$$

Enfin :

$$EQM(t) = \frac{1}{n^2}\mathbb{E}\left(M_n^2\right) = \frac{1}{n^2}\mathbb{E}\left(\langle M\rangle_n\right) = \frac{1}{n}\left(\mathbb{E}\left(\int \frac{f^2g^2}{q_{j-1}}\right) - I^2\right) = \frac{1}{n}\left(\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}\left(\frac{f^2g^2}{q_{j-1}^2}(X_j)\right) - I^2\right)$$

En posant :

$$V(q,\phi) = \int \frac{\left(\phi - q\int\phi\right)^2}{q}$$

On remarque que l'on peut également écrire :

$$EQM(t) = \frac{1}{n^2}\sum_{j=1}^{n}\mathbb{E}\left(V\left(q_{j-1}, fg\right)\right)$$

5 Estimation des poids optimaux

Reprenons les notations de l'algorithme 1.
On cherche à estimer les poids $\alpha_t \propto \sigma_t^{-2}$ et il faut donc parvenir à estimer les σ_t. On peut pour cela remarquer que $\sigma_t^2 = \mathrm{Var}\left(\hat{I}_t\right)$.

$$\mathrm{Var}\left(\hat{I}_t\right) \propto \sum_{i=1}^{n_t} \mathrm{Var}\left(\frac{fg}{q_{i-1}}\left(X_{t,i}\right)\right)$$

Or :

$$\mathrm{Var}\left(\frac{fg}{q_{i-1}}\left(X_{t,i}\right)\right) = \mathbb{E}\left[\left(\frac{fg}{q_{i-1}}\left(X_{t,i}\right) - \mathbb{E}\left(\frac{fg}{q_{i-1}}\left(X_{t,i}\right)\right)\right)^2\right]$$

$$\mathrm{Var}\left(\frac{fg}{q_{i-1}}\left(X_{t,i}\right)\right) = \mathbb{E}\left[\left(\frac{fg}{q_{i-1}}\left(X_{t,i}\right) - \mathbb{E}\left(\underbrace{\mathbb{E}\left(\frac{fg}{q_{i-1}}\left(X_{t,i}\right)|F_{i-1}\right)}_{=I}\right)\right)^2\right]$$

$$\mathrm{Var}\left(\frac{fg}{q_{i-1}}\left(X_{t,i}\right)\right) = \mathbb{E}\left[\left(\frac{fg}{q_{i-1}}\left(X_{t,i}\right) - I\right)^2\right]$$

On obtient donc :

$$\mathrm{Var}\left(\hat{I}_t\right) \propto \sum_{i=1}^{n_t} \mathbb{E}\left[\left(\frac{fg}{q_{i-1}}\left(X_{t,i}\right) - I\right)^2\right]$$

Et on aimerait proposer comme estimateur :

$$\hat{\sigma}_t^2 \propto \sum_{i=1}^{n_t} \left(\frac{fg}{q_{i-1}}\left(X_{t,i}\right) - I\right)^2$$

Remarquons tout d'abord que $\mathbb{E}\left(\hat{\sigma}_t^2\right) = \sigma_t^2$ donc l'estimateur est non-biaisé.

Cet estimateur a deux problèmes essentiels :
- On ne connaît évidemment pas la valeur exacte de I puisqu'on cherche à l'estimer.
- La méthode est dependante de la fonction g

Néanmoins, on peut remarquer que si on veut être équitablement "efficace" sur tout l'espace d'intégration, une bonne méthode consiste à prendre g la fonction constante. Ainsi, $I = \int fg = g(0)\int f = g(0)$ et l'estimateur précédent se simplifie en :

$$\hat{\sigma}_t^2 \propto \sum_{i=1}^{n_t} \left(\frac{f}{q_{i-1}}\left(X_{t,i}\right) - 1\right)^2$$

En ajoutant la condition sur les poids $\sum_{i=1}^{n_t} \frac{1}{\hat{\sigma}_i^2} = 1$, on a un estimateur à chaque étape.

6 Square root rule

Supposons maintenant que l'algorithme de l'AIS progresse au rythme suivant : $\mathrm{Var}\left(\hat{I}_k\right) = \sigma^2 k^{-y}$ pour un certain $y \in (0,1)$.

A quoi correspond cette hypothèse ?

— On suppose ici que l'algorithme permet un gain constant, c'est à dire qu'il n'est jamais excessivement plus performant à une étape qu'à une autre.

— Le cas $y = 0$ est le cas le plus pessimiste possible : il correspond à une progression nulle de l'algorithme en termes de qualité du résultat.

— Le cas $y = 1$ (ou même $y > 1$) est trop optimiste car il indiquerait une convergence bien plus rapide que ce qui est connu actuellement.

Notre estimation $\hat{I}$ dépend donc maintenant de la valeur que l'on choisit pour y :

$$\hat{I} = \hat{I}(y) = \frac{1}{\sum_{t=1}^{T} t^y} \sum_{t=1}^{T} t^y \hat{I}_t$$

Comme à l'accoutumée, on va donc essayer d'en minimiser la variance, cette fois en fonction de y, dans le but de montrer que prendre $y = \frac{1}{2}$ est un bon compromis.

On note :

$$\rho_T\left(y|x\right) = \frac{\mathrm{Var}\left(\hat{I}(x)\right)}{\mathrm{Var}\left(\hat{I}(y)\right)}$$

Cette valeur reflète l'inéfficacité de prendre x par rapport à un certain y.

Les résultats principaux de [1] sont alors :

Proposition 1.

$$\sup_{1 \le T < \infty} \sup_{0 \le x \le 1} \rho_T\left(\frac{1}{2}|x\right) \le \frac{9}{8}$$

$$\forall K \ge 2, y \ne \frac{1}{2}, \sup_{0 \le x \le 1} \rho_T\left(y|x\right) > \sup_{0 \le x \le 1} \rho_T\left(\frac{1}{2}|x\right)$$

— La première inégalité signifie qu'en prenant $y = \frac{1}{2}$, la variance ne peut être augmentée que d'un facteur $\frac{9}{8}$.

— La seconde signifie que $y = \frac{1}{2}$ est l'unique meilleur pari si on ne connait pas x.

7 Asymptotic Results for AIS and wAIS

On commence par rappeler les résultats de [3].

7.1 Théorème Centrale Limite pour AIS

Lemme 3. $\langle M \rangle_n = \sum_{j=1}^{n} V(q_{j-1}, fg)$ *où* $V(q,h) = \int \frac{\|h(x) - q(x)\int h\|_2^2}{q(x)} dx$

Démonstration. We have $\langle M \rangle_n = \sum_{j=1}^{n} \mathbb{E}[\|\Delta_j\|_2^2 | \mathscr{F}_{j-1}] = \sum_{j=1}^{n} V(q_{j-1}, fg)$ par indépendance conditionel. □

Theorème 1. *(Théorème centrale limite pour l'AIS)*
Supposons que :
—

$$V(q_n, fg) \longrightarrow V_* \geq 0, p.s.$$

— $\exists \eta > 0$ *tel que*

$$sup_{j \in \mathbb{N}} \int \frac{\|fg\|^{2+\eta}}{q_j^{1+\eta}} < \infty p.s.$$

Alors,

$$n^{\frac{1}{2}}(I_n - \int fg) \Longrightarrow \mathscr{N}(0, V_*)$$

On reprend la preuve fournie dans le matériel complémentaire de [3].

Démonstration. On commence par se ramener au cas scalaire. En effet d'après le lemme de Cramer-Wald il suffit de montrer que :

$$\forall \gamma \in \mathbb{R}^p, \langle \gamma, n(I_n - fg) \rangle \Longrightarrow \mathscr{N}(0, \gamma^{\perp} V_* \gamma).$$

On peut donc supposer dans la suite que fg est une fonction à valeur réel.

Or, on a que

$$\sqrt{n}(I_n - \int fg) = n^{-\frac{1}{2}} M_n$$

Par théorème (Corollaire 3.1 p. 58, [2]), en posant $X_{nj} = \frac{1}{\sqrt{n}} \Delta_j$ il faut montrer que :

1. $\sum_{j=1}^{n} \mathbb{E}[X_{nj}^2 | \mathscr{F}_{j-1}] \longrightarrow V_*$ en probabilité
2. $\forall \epsilon > 0, \sum_{j=1}^{n} \mathbb{E}[X_{nj}^2 \mathbb{1}_{|X_{nj}| > \epsilon} | \mathscr{F}_{j-1}] \longrightarrow 0$ en probabilité (condition de Lindeberg)

On commence par démontrer le premier point.

On a,

$$\sum_{j=1}^{n} \mathbb{E}[X_{nj}^2 | \mathscr{F}_{j-1}] = n^{-1} \langle M \rangle_n$$

$$= V_* + \frac{1}{n} \sum_{j=1}^{n} [V(q_{j-1}, fg) - V_*]$$

Or on a supposé que $V(q_n, fg) \longrightarrow V_* \geq 0, p.s.$ Donc d'après le lemme de Cesaro,

$$\sum_{j=1}^{n} \mathbb{E}[X_{nj}^2 | \mathscr{F}_{j-1}] \longrightarrow 0$$

presque surement. Et donc en particulier en probabilité.

Il reste a montrer le second point. Or, par définition de Δ_j

$$\sum_{j=1}^{n} \mathbb{E}[X_{nj}^2 \mathbb{1}_{|X_{nj}|>\epsilon} | \mathscr{F}_{j-1}] = \frac{1}{n} \sum_{j=1}^{n} \mathbb{E}[\Delta_j^2 \mathbb{1}_{|\Delta_j|>\sqrt{n}\epsilon} | F_{j-1}]$$

Notons, $w_{j-1} = \frac{fg}{q_{j-1}}$

$$\sum_{j=1}^{n} \mathbb{E}[X_{nj}^2 \mathbb{1}_{|X_{nj}|>\epsilon} | \mathscr{F}_{j-1}] = \int (w_{j-1}(x) - I)^2 q_{j-1}(x) \mathbb{1}_{|w_{j-1}(x)-I|>\sqrt{n}\epsilon} dx$$

Par inégalité triangulaire, puis positivité,

$$\leq \int (w_{j-1}(x) - I)^2 q_{j-1}(x) \mathbb{1}_{|w_{j-1}(x)|>\sqrt{n}\epsilon-|I|} dx$$

Par inégalité arithmético-géométrique (ou convexité de $x \mapsto x^2$)

$$\leq \int 2(w_{j-1}(x)^2 + I^2) q_{j-1}(x) \mathbb{1}_{|w_{j-1}(x)|>\sqrt{n}\epsilon-|I|} dx$$

$$\leq 2 \int w_{j-1}(x)^2 q_{j-1}(x) \mathbb{1}_{|w_{j-1}(x)|>\sqrt{n}\epsilon-|I|} dx + 2I^2 \int q_{j-1}(x) \mathbb{1}_{|w_{j-1}(x)|>\sqrt{n}\epsilon-|I|} dx$$

Soit $\eta > 0$ alors d'après l'inégalité de Markov,

$$\leq \frac{2}{(\epsilon\sqrt{n} - |I|)^\eta} \int \frac{|fg|^{2+\eta}}{q_{j-1}(x)^{1+\eta}} dx + \frac{2I^2}{\epsilon\sqrt{n} - |I|} \int |\frac{fg}{q_{j-1}}| q_{j-1} dx$$

Finalement,

$$\leq \frac{2}{(\epsilon\sqrt{n} - |I|)^\eta} \sup_j \int \frac{|fg|^{2+\eta}}{q_{j-1}(x)^{1+\eta}} dx + \frac{2I^2}{\epsilon\sqrt{n} - |I|} \int |\frac{fg}{q_{j-1}}| q_{j-1} dx$$

Ainsi, on a par hypothèse la converge presque sûr vers 0 et donc en particulier la convergence en probabilité.

□

Corollaire 4. *(TCL pour l'AIS normalisé) Supposons que les hypothèses du théorème préécédent soit vérifié pour $(f_u g, f_u)^\perp$ au lieu de fg alors :*

$$\sqrt{n}(I_n^{(norm)} - \int fg) \implies \mathscr{N}(0, UV_*U^\perp)$$

avec $U = (\mathscr{I}, -\int f_u g)$

Démonstration. On applique le théorème précédent au vecteur. Le corollaire est alors vraie d'après le lemme de Slutsky. □

7.2 Théorème Centrale Limite pour le wAIS

On va adapter l'argument de la martingale pour prpouver notre théorème.

Posons,

$$M_n^\alpha = \sum_{t=1}^{n} \alpha_t \Delta_t$$

Avec

$$\Delta_j = \frac{f(X_j)g(X_j)}{q_{j-1}(X_j)} - I$$

Dans ce contexte, on alors un estimateur pondérée par la suite α,

$$I_n^\alpha = \frac{1}{\sum_{t=1}^{n} \alpha_t} \sum_{t=1}^{n} \alpha_t \frac{f(X_t)g(X_t)}{q_{t-1}(X_t)}$$

Lemme 5. *M_n^α est une $\mathscr{F}_n$-martingale.*

Démonstration. Soit $n \in \mathbb{N}^*$ alors,

Comme M_{n-1}^α est $\mathscr{F}_{n-1}$ mesurable,

$$\mathbb{E}[M_n^\alpha | \mathscr{F}_{n-1}] = M_{n-1}^\alpha + \mathbb{E}[\alpha_n \Delta_n | \mathscr{F}_{n-1}]$$

Comme Δ_n est indépendant de $\mathscr{F}_{n-1}$,

$$= M_{n-1}^\alpha + \alpha_n \mathbb{E}[\Delta_n]$$

$$= M_{n-1}^\alpha$$

□

Lemme 6. $\langle M^\alpha \rangle_n = \sum_{j=1}^{n} \alpha_j^2 V(q_{j-1}, fg)$ *où* $V(q,h) = \int \frac{\|h(x) - q(x)\int h\|_2^2}{q(x)} dx$

Démonstration. $\langle M^\alpha \rangle_n = \sum_{j=1}^{n} \mathbb{E}[(\alpha_j \Delta_j)(\alpha_j \Delta_j)^\perp | \mathscr{F}_{j-1}] = \sum_{j=1}^{n} \alpha_j^2 \mathbb{E}[\|\Delta_j\|_2^2 | \mathscr{F}_{j-1}] = \sum_{j=1}^{n} \alpha_j^2 V(q_{j-1}, fg)$ □

Remarque 1. *Notons que pour les α_j constant de valeur 1 on retrouve bien le lemme de la version non pondérés.*

Theorème 2. *(TCL pour le wAIS) Supposons que :*

—

$$\sup_{n\in\mathbb{N}^*,1\leq j\leq n} n\alpha_j \Big/ \sum_{t=1}^{n}\alpha_t < \infty$$

—

$$V(q_n, fg) \longrightarrow V_* \geq 0, p.s.$$

— $\exists \eta > 0$ *tel que*

$$sup_{j\in\mathbb{N}} \int \frac{\|fg\|^{2+\eta}}{q_j^{1+\eta}} < \infty \, p.s.$$

Alors,

$$n^{\frac{1}{2}}(I_n^\alpha - \int fg) \Longrightarrow \mathcal{N}(0, V_*)$$

Remarque 2. *A noter que dans l'hypothèse, le $alpha_j$ ne peut pas être retiré si on ne fait pas d'hypothèse suplémentaire sur la suite des poids. Cependant, si on suppose par exemple que les poids sont tous positifs, ce qui parait raisonable dans notre contexte alors elle se récrit $sup_{n\in\mathbb{N}^*} n\alpha_n \Big/ \sum_{t=1}^{n}\alpha_t < \infty$*

Démonstration. On commence par remarquer que,

$$\sqrt{n}(I_n^\alpha - I) = \frac{\sqrt{n}}{\sum_{t=1}^{n}\alpha_t} M_n^\alpha$$

Posons $X_{nj} = \frac{\sqrt{n}}{\sum_{t=1}^{n}\alpha_t}\alpha_j\Delta_j$

On cherche a applique le corollaire 3 page 58 de [2]. Pour conclure il suffit alors de montrer que :

1. $\sum_{j=1}^{n}\mathbb{E}[X_{nj}^2|\mathcal{F}_{j-1}] \longrightarrow V_*$ en probabilité
2. $\forall\epsilon > 0, \sum_{j=1}^{n}\mathbb{E}[X_{nj}^2 \mathbb{1}_{|X_{nj}|>\epsilon}|\mathcal{F}_{j-1}] \longrightarrow 0$ en probabilité (condition de Lindeberg)

On commence par démontrer le premier point.

On a,

$$\sum_{j=1}^{n}\mathbb{E}[X_{nj}^2|\mathcal{F}_{j-1}] = \frac{n}{(\sum_{t=1}^{n}\alpha_t)^2}\langle M^\alpha\rangle_n$$

D'après le lemme ci-dessus.

$$= V_* + \frac{n}{(\sum_{t=1}^{n}\alpha_t)^2}\sum_{j=1}^{n}\alpha_t^2[V(q_{j-1}, fg) - V_*]$$

$$= V_* + \frac{1}{n}\sum_{j=1}^{n}(\frac{n\alpha_j}{(\sum_{t=1}^{n}\alpha_t)})^2[V(q_{j-1}, fg) - V_*]$$

Or on a supposé que $V(q_n, fg) \longrightarrow V_* \geq 0, p.s.$ et que $\frac{n\alpha_j}{(\sum_{t=1}^{n} \alpha_t)}$ était bornée disons majoré par une constante B. Donc d'après le lemme de Cesaro,

$$\sum_{j=1}^{n} \mathbb{E}[X_{nj}^2 | \mathscr{F}_{j-1}] \longrightarrow 0$$

presque surement. Et donc en particulier en probabilité.

Il reste a montrer le second point. Or, par définition de Δ_j on a,

$$\sum_{j=1}^{n} \mathbb{E}[X_{nj}^2 \mathbb{1}_{|X_{nj}|>\epsilon} | \mathscr{F}_{j-1}] = \sum_{j=1}^{n} \mathbb{E}[\frac{n\alpha_j^2}{(\sum \alpha_t)^2} \Delta_j^2 \mathbb{1}_{|\Delta_j|>(\frac{\sum \alpha_t}{n\alpha_j})\sqrt{n}\epsilon} | \mathscr{F}_{j-1}]$$

$$\leq B^2 \sum_{j=1}^{n} \mathbb{E}[\Delta_j^2 \mathbb{1}_{|\Delta_j|>\sqrt{n}B\epsilon} | \mathscr{F}_{j-1}]$$

Or dans la démonstration en l'absence de ponderation on a montré sous les même hypothèses que :

$$\forall \epsilon > 0, \sum_{j=1}^{n} \mathbb{E}[\Delta_j^2 \mathbb{1}_{|\Delta_j|>\sqrt{n}\epsilon} | \mathscr{F}_{j-1}] \longrightarrow 0 \, p.s.$$

On l'applique dans notre cas avec ϵ remplacé par $B\epsilon$ et on obtient également le résultat voulu.

□

Dans le contexte baysésien, de la même facon que pour l'AIS classique on considère un estimateur normalisé.

$$\hat{I}_n^{\alpha,(norm)} = \frac{\sum_{t=1}^{n} \alpha_t \frac{g(X_t) f_u(X_t)}{q_{t-1}(X_t)}}{\sum_{t=1}^{n} \alpha_t \frac{f_u(X_t)}{q_{t-1}(X_t)}} = \frac{I_n^{\alpha,(norm)}(g f_u)}{I_n^{\alpha,(norm)}(f_u)}$$

Corollaire 7. *(TCL pour le wAIS normalisé) Supposons que les hypothèses du théorème précédent soit vérifiées pour* $(f_u g, f_u)^{\perp}$ *au lieu de* $f\,g$ *alors :*

$$\sqrt{n}\left(I_n^{(norm),\alpha} - \int f g\right) \Longrightarrow \mathcal{N}(0, UV_* U^{\perp})$$

avec $U = (\mathcal{I}, -\int f_u g)$

Démonstration. On applique le théorème précédent au vecteur. Le corollaire est alors vraie d'après le lemme de Slutsky. □

7.3 Interpretation du théorème

Commencons par regarder les hypothèses de notre théorème. Les deux dernières hypothèses sont les mêmes que pour le théorème de conergence de l'AIS de [3]. La première est une hypothèse sur la suite qui est raisonable. En effet, cette hypothèse est par exemple vérifié pour des suites bornés $0 < \alpha_- < \alpha_t < \alpha_+$, des suites en progressions arithmétique et géométriques. En particulier elle est vérifiée pour la "square root rule" de [1]. Par contre, l'hypothèse n'est pas vérifié pour des suites du type $\alpha_t = \frac{1}{t}$, c'est compréhensible car alors dans ce cas on donne du poids au premiers batchs qui sont les moins bons et on ne bénéficie pas des derniers.

En ce qui concerne le résultat du théorème, il nous donne la normalité assymptotique de notre estimateur avec le même V_* que dans le théorème de l'AIS. Ce résultat est donc plutôt pessimiste, en effet, il montre qu'assymptotiquement les méthodes pondérés du type wAIS ne sont pas meilleurs que la méthode AIS. Heuristiquement on comprend bien ce résultat, en effet, une fois que l'échantilloneur à convergé, il ne bouge plus et donc tous les batchs deviennent équivalents. En particulier pour le cas de [3] les variances n'ont pas de raisons de beaucoup varier d'une étape à une autre. En conclusion, les méthodes pondérés ont un intérêt pour un nombre fini de particules mais pas dans le régime assymptotique.

8 Optimalité assymptotique pour l'AIS et le wAIS

8.1 Optimalité assymptotique de l'AIS

On se place dans le contexte où la politique q_n provient d'un modèle paramètrique $(q_\theta)_{\theta\in\Theta}$ avec Θ un compact de $\mathbb{R}^n$. On se place dans le cadre de la M-estimation.

Alors on rappele les théorèmes de [3]

Theorème 3. *(Consistance de la politique) Soit $M(x) = sup_{\theta\in\Theta} m_\theta(x)$. On suppose que Θ est un compact de $\mathbb{R}^n$ et que :*

1. $$\int M < \infty$$
2. $$sup_{\theta\in\Theta} \int \frac{M^2}{q_\theta} < \infty$$
3. $$\forall \theta \neq \theta^*, \int m_\theta > \int m_{\theta_*}$$
4. *L'application $\theta \mapsto m_\theta(x)$ est continue.*

$$\textit{Alors, } \theta_n \longrightarrow \theta^* \textit{ p.s.}$$

Démonstration. Dans le matériel supplémentaire de [3] □

Theorème 4. *(Optimalité assymptotique de l'AIS)*

Supposons que :
— *Les hypothèse du théorème de consistance de la politique sont vérifiées.*
— *$\exists \eta > 0$ tel que*

$$sup_{\theta\in\Theta} \int \frac{\|fg\|^{2+\eta}}{q_j^{1+\eta}} < \infty \; p.s.$$

Alors,

$$n^{\frac{1}{2}}(I_n - \int fg) \implies \mathcal{N}(0, V(q_{\theta_*}, fg))$$

Corollaire 8. *(Optimalité assymptotique de l'AIS normalisé)*
Supposons que les hypothèses du théorème précédent soit vérifiées pour $(f_u g, f_u)^\perp$ au lieu de fg alors :

$$\sqrt{n}(I_n^{(norm),\alpha} - \int fg) \implies \mathcal{N}(0, UV(q_{\theta_*}, fg))U^\perp)$$

avec $U = (\mathcal{I}, -\int f_u g)$

8.2 Optimalité assymptotique pour le wAIS

Theorème 5. *(Optimalité assymptotique de le wAIS)*

Supposons que :

—

$$\sup_{n\in\mathbb{N}^*, 1\leq j\leq n} n\alpha_j / \sum_{t=1}^{n} \alpha_t < \infty$$

— *Les hypothèse du théorème de consitance de la politique sont vérifié.*

— $\exists \eta > 0$ *tel que*

$$sup_{\theta\in\Theta} \int \frac{\|fg\|^{2+\eta}}{q_j^{1+\eta}} < \infty \, p.s.$$

Alors,

$$n^{\frac{1}{2}}(I_n - \int fg) \Longrightarrow \mathcal{N}(0, V(q_{\theta_*}, fg))$$

9 Résultats expérimentaux et conclusion

Nos expérience montre l'intérêt des méthodes pondérés dans l'accélération de la convergence des méthodes d'importance sampling.

Pour les figures suivantes, on se fixe un budget en terme de nombre total de variables tirées pour estimer une intégrale. On fixe $\sum_{t=1}^{T} n_t = 2^{10}$.
On considère de plus une politique de la forme $n_t = n, \forall t$, et donc $nT = 2^{10}$.
Dans les figures suivantes, on a en abscisse n.

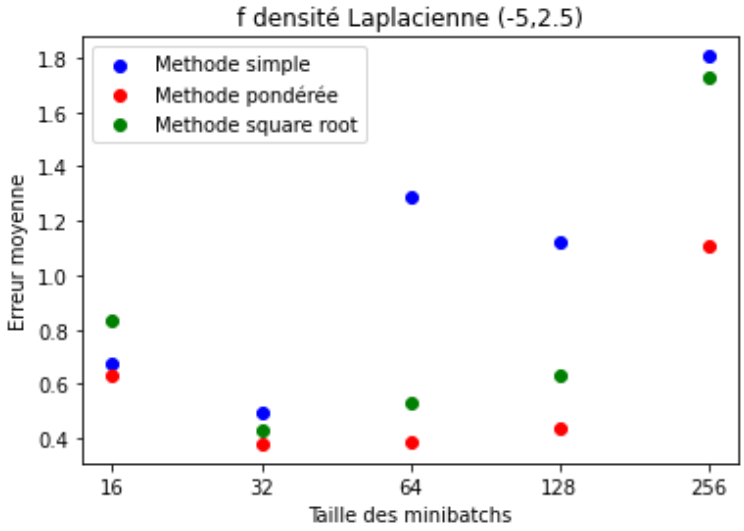

FIGURE 1 – Erreur pour $g : x \mapsto x$ et f une densité Laplacienne de paramètres (-5,2.5)

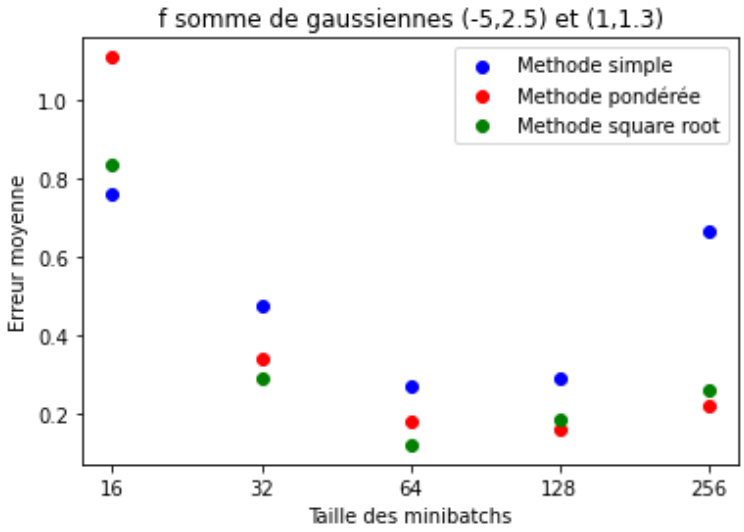

FIGURE 2 – Erreur pour $g : x \mapsto x$ et f une somme de gaussiennes de paramètres (-5,2.5) et (1,1.3)

Dans le cas des **mini-batchs** la méthode de "square root rule " [1] est la meilleur, notamment car elle est plus robuste dans son estimation de la variance. Pour des batchs suffisament grands, l'estimateur de la variance de [3] devient sufisament robuste. Dans ce cas, c'est la méthodes de [3] qui est donc la plus performante.

Références

[1] Yi Zhou ART B. OWEN. "The square root rule for adaptive importance sampling". In : (2019). URL : https://arxiv.org/abs/1901.02976.

[2] P. HALL et C. C. HEYDE. "Martingale limit theory and its application. Academic Press". In : (1980).

[3] François PORTIER et Bernard DELYON. "Asymptotic optimality of adaptive importance sampling". In : *Advances in Neural Information Processing Systems*. Sous la dir. de S. BENGIO et al. T. 31. Curran Associates, Inc., 2018. URL : https://proceedings.neurips.cc/paper/2018/file/1bc0249a6412ef49b07fe6f62e6dc8de-Paper.pdf.

10 Annexe : Code

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats
from math import exp
import scipy.special
import warnings
warnings.filterwarnings("ignore")
#Quelques fonctions    int grer :

def f(x):
    mu = -5
    b = 2.5
    return(np.exp(-abs(x-mu)/b)/(2*b))

def gamma(x):
    if x<=0:
        return 0

    theta = 0.5
    k = 9

    return( (x**(k-1)) * np.exp(-x/theta)/(scipy.special.gamma(k)*(theta**k)) )

def gaussian1(x):
    mu = -5
    sigma = 3
    return scipy.stats.norm.pdf(x,mu,sigma)

def gaussian2(x):
    mu = 1
    sigma = 1.3
    return scipy.stats.norm.pdf(x,mu,sigma)

def gaussian_mix(x):
    return 0.5*(gaussian1(x) + gaussian2(x))

def g(x):
    return x

#La fonction principale faisant appel aux autres

def run(f,g,n,m0,sigma0,methods):
    '''output[i] est le r sultat de la methode i dans la liste methods'''
    output = []
    for method in methods:
        if method == 'simple':
            output.append(gaussian_AIS(f,g,n,m0,sigma0))
        if method == 'weighted':
            output.append(gaussian_wAIS(f,g,n,m0,sigma0))
        if method == 'square_root':
            output.append(gaussian_srAIS(f,g,n,m0,sigma0))
```

```
52          if method == 'bayesian':
53              output.append(bayesian_gaussian_wAIS(f,g,n,m0,sigma0))
54      return output
55
56  #Les fonctions permettant de d finir et de minimiser la KL pour une q gaussienne
57
58  def KL(f,mu,sigma,mui,sigmai,Xt):
59      return np.mean(np.log10(f(Xt)/scipy.stats.norm.pdf(Xt,mu,sigma)) * (f(Xt)) / (scipy.stats.norm.
60
61  def KL_grad(f,mu,sigma,mui,sigmai,Xt):
62      alpha = (Xt - mu)**2
63
64      gradmu = np.mean((f(Xt)/(scipy.stats.norm.pdf(Xt,mui,sigmai) * np.log(10)))*(mu - Xt)/(sigma**2
65      gradsig = np.mean((f(Xt)/(scipy.stats.norm.pdf(Xt,mui,sigmai) * np.log(10)))*(sigma**2 - alpha)
66
67      return gradmu,gradsig
68
69  #print(KL_grad(f,0,1,-2,2,np.array([-1,2,-4,1,0])))
70  #print((KL(f,10**-3,1,-2,2,np.array([-1,2,-4,1,0]))-KL(f,0,1,-2,2,np.array([-1,2,-4,1,0])))*(10**3))
71  #print((KL(f,0,1+10**-3,-2,2,np.array([-1,2,-4,1,0]))-KL(f,0,1,-2,2,np.array([-1,2,-4,1,0])))*(10**
72
73  def minimisation_KL(f,mui,sigmai,Xt):
74
75      def kl(theta):
76          return KL(f,theta[0],theta[1],mui,sigmai,Xt)
77
78      def kl_grad(theta):
79          return KL_grad(f,theta[0],theta[1],mui,sigmai,Xt)
80
81      try:
82          m,s=scipy.optimize.minimize(kl,x0=(mui,sigmai),jac=kl_grad).x
83      except:
84          #print("Erreur survenue durant l'optimisation, on garde les deux param tres pr c dents d
85          return mui,sigmai
86      if s<0:
87          #print("L'optimisation donne une variance n gative, on garde les param tres pr c dents"
88          return mui,sigmai
89      return m,s
90
91  #Les fonctions de calcul d'int grale
92
93  def gaussian_AIS(f,g,n,m0,sigma0):
94      '''
95      Approche par une densit  gaussienne classique
96      f est une densit
97      g est une fonction    int grer
98      n est la politique d'  chantillonage
99      m0 et sigma0 sont la moyenne et l' cart  type de la distribution initiale
100     '''
101
102     s = 0
103     N = 0
104     number_of_steps = len(n)
105     m = [m0]
```

```
    sigma = [sigma0]
    I = []

    for t in range(number_of_steps):
        #Tirage des X_(t,i), phase d'exploration
        Xt = np.random.normal(loc = m[t],scale = sigma[t], size = n[t])

        #Ajout des nouvelles valeurs    la somme lobale (m j de l'estimateur)
        partial_sum = np.sum(f(Xt)*g(Xt)/scipy.stats.norm.pdf(Xt,m[t],sigma[t]))
        s+=partial_sum
        N += n[t]
        I.append(s/N)

        #Mise    jour de l'  chantilloneur
        muOpti,sigmaOpti = minimisation_KL(f,m[t],sigma[t],Xt)
        m.append(muOpti)
        sigma.append(sigmaOpti)

    return I[-1]

def gaussian_wAIS(f,g,n,m0,sigma0):
    '''
    Approche par une densit  gaussienne, schema pond r
    f est une densit
    g est une fonction    int grer
    n est la politique d'  chantillonage
    m0 et sigma0 sont la moyenne et l' cart  type de la distribution initiale
    '''

    number_of_steps = len(n)
    m = [m0]
    sigma = [sigma0]
    I = []
    estimated_variance=[]

    for t in range(number_of_steps):
        #Tirage des X_(t,i), phase d'exploration
        Xt = np.random.normal(loc = m[t],scale = sigma[t], size = n[t])

        #Ajout des nouvelles valeurs    la somme globale (m j de l'estimateur)
        partial_sum = np.sum(f(Xt)*g(Xt)/scipy.stats.norm.pdf(Xt,m[t],sigma[t]))
        I.append(partial_sum/n[t])

        #Estimation de la variance pour cette  tape
        estimated_variance.append(np.sum(((f(Xt)/scipy.stats.norm.pdf(Xt,m[t],sigma[t])) - 1)**2))

        #Mise    jour de l'  chantilloneur
        muOpti,sigmaOpti = minimisation_KL(f,m[t],sigma[t],Xt)
        m.append(muOpti)
        sigma.append(sigmaOpti)

    #Calcul des poids normalis s:
    alpha = 1/np.array(estimated_variance)
    alpha_normalized = alpha/np.sum(alpha)
```

```
160
161     try:
162         assert(abs(np.sum(alpha_normalized)-1)<10**-3)
163     except:
164         print("Erreur d'assertion W. L' cart     0 est : " + str(abs(np.sum(alpha_normalized)-1)))
165         print("Les poids sont " + str(alpha_normalized))
166     return np.sum(alpha_normalized*np.array(I))
167
168
169 def gaussian_srAIS(f,g,n,m0,sigma0):
170     '''
171     Approche par une densit  gaussienne, schema pond r  par des racines carr es
172     f est une densit
173     g est une fonction    int grer
174     n est la politique d'  chantillonage
175     m0 et sigma0 sont la moyenne et l' cart  type de la distribution initiale
176     '''
177
178     number_of_steps = len(n)
179     m = [m0]
180     sigma = [sigma0]
181     I = []
182     estimated_variance=[]
183
184     for t in range(number_of_steps):
185         #Tirage des X_(t,i), phase d'exploration
186         Xt = np.random.normal(loc = m[t],scale = sigma[t], size = n[t])
187
188         #Ajout des nouvelles valeurs    la somme lobale (m  j  de l'estimateur)
189         partial_sum = np.sum(f(Xt)*g(Xt)/scipy.stats.norm.pdf(Xt,m[t],sigma[t]))
190         I.append(partial_sum/n[t])
191
192         #Mise    jour de l'  chantilloneur
193         muOpti,sigmaOpti = minimisation_KL(f,m[t],sigma[t],Xt)
194         m.append(muOpti)
195         sigma.append(sigmaOpti)
196
197     #Calcul des poids normalis s:
198     alpha = [(k+1)**(1/2) for k in range(number_of_steps)]
199     alpha_normalized = alpha/np.sum(alpha)
200     try:
201         assert(abs(np.sum(alpha_normalized)-1)<10**-3)
202     except:
203         print("Erreur d'assertion SR. L' cart     0 est : " + str(abs(np.sum(alpha_normalized)-1)))
204         print("Les poids sont " + str(alpha_normalized))
205
206     return np.sum(alpha_normalized*np.array(I))
207
208 def gaussian_wAIS_livePlot(f,g,n,m0,sigma0):
209     '''
210     Approche par une densit  gaussienne, schema pond r
211     f est une densit
212     g est une fonction    int grer
213     n est la politique d'  chantillonage
```

```
    m0 et sigma0 sont la moyenne et l' cart  type de la distribution initiale
    '''
    # plt.close()
    # plt.figure()
    plt.ion()
    plt.show()
    fig,ax=plt.subplots()
    number_of_steps = len(n)
    m = [m0]
    sigma = [sigma0]
    I = []
    estimated_variance=[]

    for t in range(number_of_steps):
        #Tirage des X_(t,i), phase d'exploration
        Xt = np.random.normal(loc = m[t],scale = sigma[t], size = n[t])

        #Ajout des nouvelles valeurs    la somme globale (m  j  de l'estimateur)
        partial_sum = np.sum(f(Xt)*g(Xt)/scipy.stats.norm.pdf(Xt,m[t],sigma[t]))
        I.append(partial_sum/n[t])

        #Estimation de la variance pour cette  tape
        estimated_variance.append(np.sum(((f(Xt)/scipy.stats.norm.pdf(Xt,m[t],sigma[t])) - 1)**2))

        #Mise    jour de l'  chantilloneur
        muOpti,sigmaOpti = minimisation_KL(f,m[t],sigma[t],Xt)
        m.append(muOpti)
        sigma.append(sigmaOpti)

        #affichage live
        Z = []
        Lq = []
        Lf = []
        a = -13
        b = 3
        steps = 150
        for z in range(steps):
            abs = a + z*(b-a)/steps
            Z.append(abs)
            Lq.append(scipy.stats.norm.pdf(abs,muOpti,sigmaOpti))
            Lf.append(f(abs))

        ax.clear()
        plt.plot(Z,Lq,color="blue",label = "q actuel")
        plt.plot(Z,Lf,color="red",label = "f cible")
        plt.legend()
        ax.set_ylim(0,0.38)
        ax.set_xlim(a,b)
        plt.gcf().canvas.draw()
        plt.pause(0.01)

        m.append(muOpti)
        sigma.append(sigmaOpti)

```

```
    #Calcul des poids normalis s:
    alpha = [(k+1)**(1/2) for k in range(number_of_steps)]
    alpha_normalized = alpha/np.sum(alpha)
    try:
        assert(abs(np.sum(alpha_normalized)-1)<10**-3)
    except:
        print("Erreur d'assertion SR. L' cart     0 est : " + str(abs(np.sum(alpha_normalized)-1)))
        print("Les poids sont " + str(alpha_normalized))

    return np.sum(alpha_normalized*np.array(I))


def bayesian_gaussian_wAIS(f,g,n,m0,sigma0):
    '''
    Approche par une densit  gaussienne, schema pond r
    f est une fonction L1
    g est une fonction    int grer
    n est la politique d'  chantillonage
    m0 et sigma0 sont la moyenne et l' cart  type de la distribution initiale
    '''

    number_of_steps = len(n)
    m = [m0]
    sigma = [sigma0]
    I = []
    estimated_variance=[]

    for t in range(number_of_steps):
        #Tirage des X_(t,i), phase d'exploration
        Xt = np.random.normal(loc = m[t],scale = sigma[t], size = n[t])

        #Ajout des nouvelles valeurs    la somme globale (m  j  de l'estimateur)
        partial_sum = np.sum(f(Xt)*g(Xt)/scipy.stats.norm.pdf(Xt,m[t],sigma[t]))
        partial_sum2 = np.sum(f(Xt)/scipy.stats.norm.pdf(Xt,m[t],sigma[t]))
        I.append(partial_sum/partial_sum2)

        #Estimation de la variance pour cette  tape
        estimated_variance.append(np.sum(((f(Xt)/scipy.stats.norm.pdf(Xt,m[t],sigma[t])) - 1)**2))

        #Mise    jour de l'  chantilloneur
        muOpti,sigmaOpti = minimisation_KL(f,m[t],sigma[t],Xt)
        m.append(muOpti)
        sigma.append(sigmaOpti)

    #Calcul des poids normalis s:
    alpha = 1/np.array(estimated_variance)
    alpha_normalized = alpha/np.sum(alpha)

    try:
        assert(abs(np.sum(alpha_normalized)-1)<10**-3)
    except:
        print("Erreur d'assertion W. L' cart     0 est : " + str(abs(np.sum(alpha_normalized)-1)))
        print("Les poids sont " + str(alpha_normalized))
    return np.sum(alpha_normalized*np.array(I))
```

```
n=[20 for i in range(10)]
methods = ['simple','weighted','square_root']
results = run(gaussian1,g,n,0,1,methods)

for i in range(len(methods)):
    print('Le r sultat pour la m thode '+ methods[i]+' est : ' + str(results[i]) + ', soit une er

#Cellule pour le calcul, longue

results =[]

for i in range(50):
    results.append(run(f,g,n,0,1,methods))

#Cellule pour l'affichage des r sultats, ex cution rapide

error_simple=[]
error_w=[]
error_sr=[]

for result in results:
    error_simple.append(abs((result[0])+5))
    error_w.append(abs((result[1])+5))
    error_sr.append(abs((result[2])+5))

print('En moyenne sur 50 tests : \n')
print("L'erreur pour la m thode " + methods[0]+' est : ' + str(np.mean(error_simple)))
print("L'erreur pour la m thode " + methods[1]+' est : ' + str(np.mean(error_w)))
print("L'erreur pour la m thode " + methods[2]+' est : ' + str(np.mean(error_sr)))

errors_s =[]
errors_w=[]
errors_sr=[]
from time import time
import datetime
t = time()
methods = ['simple','weighted','square_root']

results=[]

starting_value=4
end_value = 7
max_samples = 10

number_of_samples = 40
print("Le nombre de variables    tirer pour cette batterie de tests est : " + str(2**max_samples *
for i in range(starting_value,end_value + 1):
    print("On en est    i = "+ str(i))
    results=[]
    n=[2**i for h in range(2**(max_samples-i))]
    for i in range(number_of_samples):
        results.append(run(f,g,n,0,1,methods))
    error_simple=[]
```

```
    error_w=[]
    error_sr=[]
    for result in results:
        error_simple.append(abs((result[0])+5))
        error_w.append(abs((result[1])+5))
        error_sr.append(abs((result[2])+5))
    errors_s.append(np.mean(error_simple))
    errors_w.append(np.mean(error_w))
    errors_sr.append(np.mean(error_sr))

X = [str(i+starting_value) for i in range(len(errors_s))]
plt.scatter(X,errors_s,color="blue",label="Methode simple")
plt.scatter(X,errors_w,color="red",label="Methode pond r e")
plt.scatter(X,errors_sr,color="green",label="Methode square root")
plt.xlabel("i")
plt.title("f densit  Laplacienne (-5,2.5)")
plt.legend()
plt.show()

print("Temps    coul    : "+ str(time()-t) + "s.")

errors_s =[]
errors_w=[]
errors_sr=[]
methods = ['simple','weighted','square_root']

t = time()

results=[]

starting_value=4
end_value = 8
max_samples = 10

number_of_samples = 50
print("Le nombre de variables    tirer pour cette batterie de tests est : " + str(2**max_samples * 
for i in range(starting_value,end_value + 1):
    print("On en est    i = "+ str(i))
    results=[]
    n=[2**i for h in range(2**(max_samples-i))]
    for i in range(number_of_samples):
        results.append(run(gaussian_mix,g,n,-4,1,methods))
    error_simple=[]
    error_w=[]
    error_sr=[]
    for result in results:
        error_simple.append(abs((result[0])+2))
        error_w.append(abs((result[1])+2))
        error_sr.append(abs((result[2])+2))
    errors_s.append(np.mean(error_simple))
    errors_w.append(np.mean(error_w))
    errors_sr.append(np.mean(error_sr))

X = [str(i+starting_value) for i in range(len(errors_s))]
```

```
plt.scatter(X,errors_s,color="blue",label="Methode simple")
plt.scatter(X,errors_w,color="red",label="Methode pond r e")
plt.scatter(X,errors_sr,color="green",label="Methode square root")
plt.xlabel("i")
plt.title("f somme de gaussiennes (-5,2.5) et (1,1.3)")
plt.legend()
plt.show()

print("Temps    coul    : "+ str(time()-t) + "s.")

#Cellule pour le calcul, longue
methods = ['bayesian']
results =[]
n=[2**12 for i in range(2**4)]

def not_f(x):
    return f(x)

for i in range(20):
    results.append(run(not_f,g,n,0,1,methods))

#Cellule pour l'affichage des r sultats, ex cution rapide

error_w=[]


for result in results:
    error_w.append(abs((result[0])+5))

print('En moyenne sur 50 tests : \n')
print("L'erreur pour la m thode " + methods[0]+' est : ' + str(np.mean(error_w)))

print(len(results))
```