Note de synthèse

Utilisation de méthodes de Monte Carlo pour la prévision météorologique

Baud Candice

Superviseur : Lindsten Fredrik - *Correspondant ENSAE* : Chopin Nicolas

Objectifs du projet

La prévision météorologique repose sur la quête de comprendre et d'anticiper les fluctuations de la nature à travers l'étude minutieuse des processus atmosphériques. En combinant des données scientifiques, des modèles mathématiques sophistiqués et une analyse attentive, la prévision météorologique vise à atténuer les risques liés aux intempéries, à optimiser les opérations liées au transport, à l'agriculture, à l'énergie et à diverses autres industries, tout en contribuant à la sécurité, au bien-être et à la résilience de notre société face aux aléas climatiques. Le but de ce projet en collaboration avec le SMHI (Institut de Météorologie Suédois) est d'implémenter des algorithmes de filtrage sur les données mesurées afin d'améliorer les prévisions météorologiques.

1 Données disponibles

Les données disponibles sont extraites de mesures effectuées au Nord de la Finlande. Le projet s'axe sur l'étude des sols [1] divisés en parcelles agricoles et forêts. Ces sols sont divisés en couches correspondant aux profondeurs. Pour prendre en compte la neige en hiver, on définit également des couches au-dessus du sol. Chacune de ces strates contient des variables telles que la quantité d'eau, la quantité de glace, la température, la densité de la neige, et l'âge de la neige.

A partir de ces données, on veut implémenter des algorithmes de filtrage à particules [2] ayant plusieurs objectifs. D'une part, nous souhaitons prendre en compte les imprécisions de mesure et de connaissance des processus météorologiques (le 'bruit') afin de filtrer nos données et récupérer une information de bonne qualité. D'autre part, nous souhaitons pouvoir déduire des informations sur certaines variables de certaines couches lorsque celles-ci ne sont pas mesurées mais que l'on a accès à des mesures sur d'autres couches ou d'autres variables. Le problème auquel nous faisons face est spatio-temporel car nos variables évoluent selon le temps mais s'influencent également spatialement entre couches. Cette grande dimensionnalité implique d'implémenter des algorithmes plus poussés que les filtres classiques.

2 Techniques de filtrage

On considère pour modéliser notre phénomène que l'on a un processus x_t (où x_t est multi-dimensionnel car il contient plusieurs variables) que l'on ne peut pas observer directement mais qu'à la place on observe y_t, un processus ayant un lien avec x_t. Par exemple, on peut imaginer que y_t contient le nombre de jours de précipitations et la température extérieure, et que x_t est une quantité de neige que l'on ne peut pas mesurer. On peut également imaginer que y_t soit une observation très imprécise de x_t.

On a donc défini deux processus x_t et y_t dont l'un est latent et l'autre est observé. On cherche à obtenir de l'information sur le processus latent, plus particulièrement sur sa loi de probabilité. Le filtrage à particules revient à générer des valeurs de x_t qui représentent des hypothèses sur la valeur réelle. Connaissant la valeur de y_t, on peut évaluer la pertinence de chaque valeur hypothétique de x_t et lui attribuer un poids. De cette manière, on souhaite approximer pour tout t la quantité $p(x_t|y_{1:t})$ (c'est à dire la loi de x_t conditionnellement aux observations de $y_{1:t}$) par une somme pondérée de mesures de diracs comme le montre la figure [1].

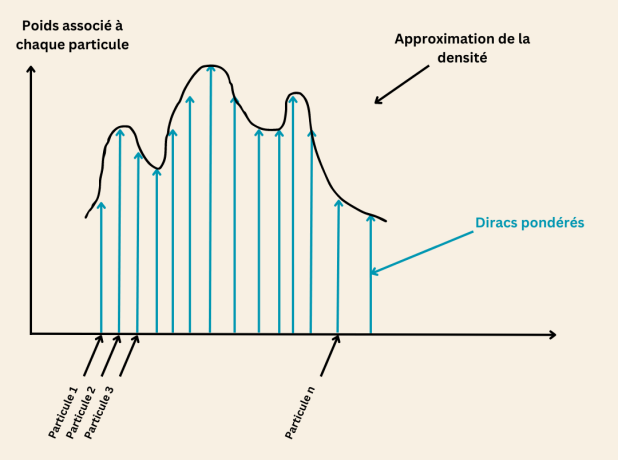


FIGURE 1 – Approximation de la distribution

Plus précisément, on définit :

$$x_t = f(x_{t-1}) + \epsilon_t \qquad y_t = g(x_t) + \delta_t \qquad \text{où } \epsilon_t \text{ et } \delta_t \text{ sont des bruits, généralement gaussiens.}$$

A l'initialisation, on génère N particules dont on va évaluer le poids w_t. Ces particules constituent une approximation de la densité $p(x_0|y_0)$. On va ensuite suivre les mêmes étapes détaillées ci-après pour approximer les densités $p(x_t|y_{1:t})$ $\forall t$. Partant de nos N particules pondérées $(x_t^i, w_t^i)_{i=1}^N$, on les rééchantillonne pour obtenir un système $(x_t^{a_t^i}, 1)_{i=1}^N$ où a_t^i représente l'indice de l'ancêtre lors du rééchantillonnage. Les particules ont toutes un poids égal car c'est leur multiplicité qui a été modifiée pour prendre en compte leur pertinence. Finalement, ayant des particules pertinentes au temps t, on les propage au temps $t+1$ avec l'équation régissant l'évolution de $x_{t+1}|x_t$ et on obtient $(x_{t+1}^i)_{i=1}^N$. On donne un poids à ces nouvelles particules $(x_{t+1}^i)_{i=1}^N$ suivant la formule $w_{t+1}^i = p(y_{t+1}|x_{t+1}^i)$. Cette approche est intuitive car le poids sera d'autant plus élevé que la probabilité d'observer y_{t+1} sachant x_{t+1}^i est élevée. En d'autres termes, ce poids évalue la cohérence de la mesure de y_{t+1} par rapport à la particule hypothétique x_{t+1}^i. Ces N particules pondérées approximent la densité $p(x_{t+1}|y_{1:t+1})$. On répète les étapes jusqu'au temps maximal considéré.

Partant des principes expliqués plus haut, notre but est de générer des hypothèses sur nos particules les plus pertinentes possibles, et pour cela, il convient d'utiliser au maximum l'information dont nous disposons. L'algorithme de base se nomme le *Bootstrap particle filter* [2] mais des variantes comme le *Fully adapted particle filter* [3] permettent d'inclure l'information de y à des temps supérieurs dans l'algorithme lors de l'étape de rééchantillonnage et donc de générer des particules plus pertinentes. Ces méthodes sont efficaces en petite dimension, mais leur précision diminue très nettement lors de problèmes plus complexes. Il convient alors d'utiliser d'autres algorithmes comme les *Nested sequential Monte Carlo* [4] ou le *Divide and Conquer sequential Monte Carlo* [5].

3 Implémentation et résultats

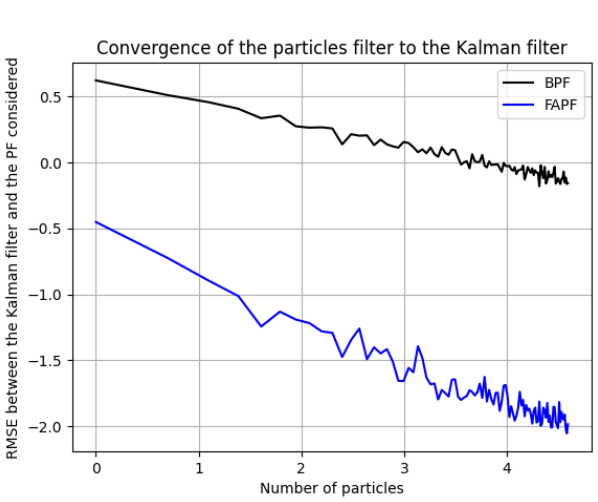
Modèles linéaires gaussiens

Dans un premier temps, j'implémente tous les algorithmes précédents sur un modèle linéaire gaussien, c'est à dire un modèle où les fonctions f et g sont linéaires.

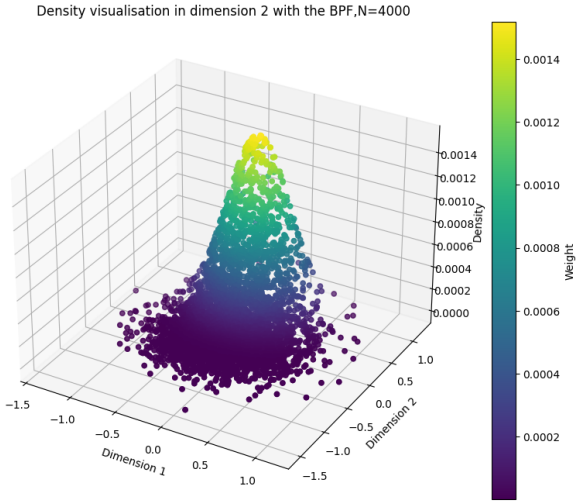
$$x_{t+1} = Ax_t + \epsilon_t \qquad y_t = Bx_t + \delta_t \qquad \epsilon_t \sim N(0, \sigma^2 I), \delta_t \sim N(0, \eta^2 I)$$

Ces modèles sont intéressants pour évaluer l'efficacité de nos algorithmes car il existe une solution analytique optimale donnée par le filtre de Kalman. Ce filtre nous donne la moyenne et la covariance,

ce qui caractérise complètement la loi dans le cas gaussien. On peut alors comparer nos filtres à particules à la solution optimale donnée par Kalman [6]. La figure [2a] présente l'écart entre notre filtre à particules et le filtre de Kalman à mesure que l'on augmente le nombre de particules. La figure [2b] présente la densité obtenue au temps $T = 20$ avec le *Bootstrap PF* pour un modèle linéaire gaussien de dimension 2 avec les paramètres $A = \begin{bmatrix} 0.1 & 0.5 \\ 0.4 & 0.3 \end{bmatrix}$, $B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, $\sigma^2 = 0.1, \delta^2 = 0.1$.



(a) Convergence vers le filtre de Kalman



(b) Densité de $p(x_T|y_{1:T})$

Le graphique de gauche [2a] présente l'écart (mesuré par RMSE) entre les solutions données par les filtres et la solution optimale donnée par le filtre de Kalman. Le tracé est en échelle logarithmique et nous permet de visualiser une convergence à la vitesse $\sqrt{N}$ pour le *Fully adapted PF*, ce qui est bien plus efficient que le *Bootstrap PF*. Il convient de noter que ces convergences dépendent de paramètres :

- **La dimension** : Lorsque la dimension augmente, le *Bootstrap PF* devient de plus en plus inefficient dû à un phénomène de dégénérescence des poids empêchant de conserver une diversité de particules suffisante. On n'explore alors pas assez l'espace lors de la génération de nouvelles particules, ce qui diminue la précision de l'approximation.

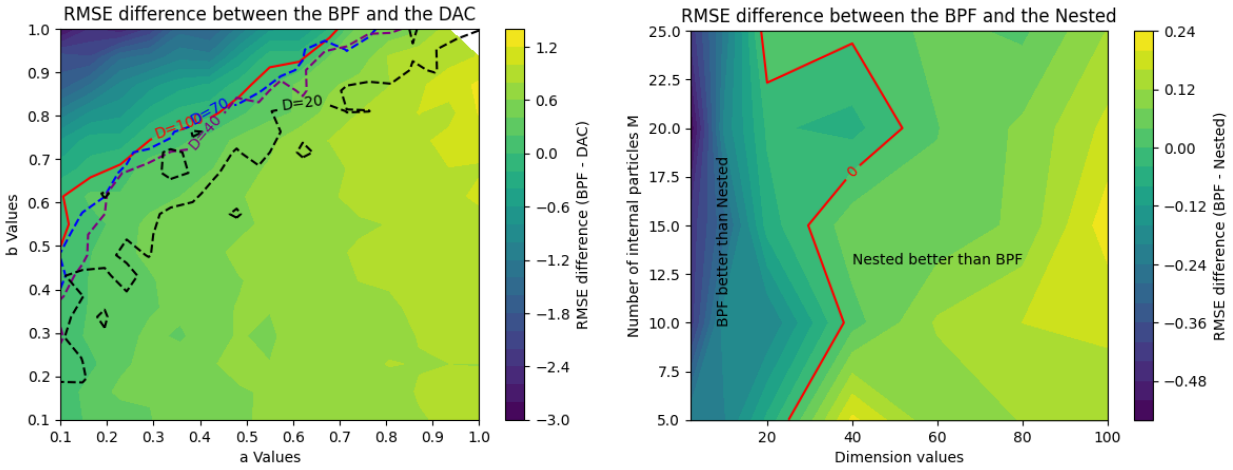
- **Le niveau de bruit** : Le *Fully adapted PF* prend en compte l'information contenue dans y_t pour tout t. Mais, si nos observations de y_t sont très imprécises (niveau de bruit élevé), l'information que l'on en extrait est de mauvaise qualité et impacte donc nos approximations. Ainsi, lorsque l'erreur de mesure de Y est faible, le *Fully adapted PF* gagne en efficacité comparativement au *Bootstrap PF*, qui lui ne prend pas en compte la valeur présente de y_t. A l'inverse, si la mesure de Y est peu précise, ce dernier n'a pas une grande valeur ajoutée par rapport au *Bootstrap PF* car le bruit ne permet pas d'extraire de l'information pertinente.

Dans les exemples précédents, nous nous sommes cantonés à une dimension de 5, ce qui est assez faible. En pratique en météorologie, les dimensions sont très largement supérieures à 10, et le *Bootstrap PF* est alors peu efficace. J'ai implémenté deux autres algorithmes, qui, en divisant notre problème multidimensionnel en plein de problèmes de plus petite dimension permettent de meilleures approximations.

La première méthode considérée est l'algorithme *Divide-and-Conquer*. Il divise une séquence donnée en un arbre comportant à ses feuilles des nombres unidimensionnels. On génère alors des sets de

particules unidimensionnelles pour chaque feuille et on remonte ensuite l'arbre en concaténant ces particules à chaque fois que des branches se rencontrent. On crée ainsi des particules de dimension de plus en plus grande jusqu'à arriver à la racine. De plus, à chaque concaténation, on ajuste les poids pour avoir une plus grande exactitude dans nos approximations.

La seconde méthode considérée est l'algorithme *Nested Monte Carlo*. Il divise le problème spatio-temporel en un problème d'abord spatial puis temporel. Pour chaque itération de t, et chaque particule N, on effectue une approximation interne avec M particules des distributions dans la direction spatiale. Une fois cette approximation effectuée, pour chaque particule externe, on choisit la meilleure particule interne pour chaque dimension, ce qui permet de créer une particule externe qui soit la plus pertinente possible. On pondère pour obtenir l'estimation à t. On n'a alors plus qu'à passer au temps suivant.



(a) DAC vs BPF pour différentes dimensions

(b) Nested SMC vs BPF, a = 0.2, b = 1.1

La figure [3a] présente la différence de RMSE entre le DAC et le BPF pour différentes matrices de transition et des variances unitaires. On définit $A = a \cdot I_D$ et $B = b \cdot I_D$. Ce graphe permet de voir que les résultats quant au BPF et au DAC sont très dépendants des ces paramètres. Pour certaines combinaisons de valeurs, même en grande dimension, le BPF se révèle plus avantageux. Ces cas restent minoritaires, ce qui confirme que le DAC apporte une réelle amélioration. Cette amélioration est d'autant plus importante que la dimension du problème croit. De la même manière, certains niveaux de bruit donnent un avantage au BPF, avantage qui se réduit à mesure que la dimension augmente.

La figure [3b] présente les performances du Nested SMC comparativement au BPF lorsque la dimension augmente et le nombre de particules internes augmente. Les algorithmes ont le même budget computationel : $M \cdot N$ particules pour le BPF ; et M particules internes et N externes pour le Nested SMC. Le Nested SMC apporte une réelle amélioration dès lors que la dimension devient importante quels que soient les paramètres du modèle (a, b, variances).

En d'autres termes, le DAC et le Nested SMC sont deux algorithmes efficaces pour des modèles linéaires gaussiens lorsque la dimension augmente. Le DAC est également meilleur en terme de performance et de temps de calcul comparativement au Nested SMC. Dans la suite, c'est donc le DAC que l'on va préférer.

Modèle de Lorenz

Une fois les implémentations dans le modèle linéaire gaussien effectuées, nous passons à l'application aux données météorologiques. On considère le modèle météorologique de Lorenz 1996, déterministe, et nous lui ajoutons du bruit gaussien lors de la simulation afin de le rendre stochastique. Le modèle s'écrit, en indicant le temps par t et la dimension par i,

$$x_{t+1}(i) = (x_t(i+1) - x_t(i-2))x_t(i-1) - x_t(i) + F + v_t \ , \qquad v_t \sim N(0, \sigma^2)$$
$$x_t(-1) = x_t(D-1), \qquad x_t(0) = x_t(D), \qquad x_t(D+1) = x_t(1)$$

On définit également y_t :

$$y_t = Hx_t + u_t \ , \qquad u_t \sim N(0, \delta^2 I_d)$$

Le modèle n'étant pas linéaire, il n'est pas possible d'utiliser le filtre de Kalman. Néanmoins, on peut implémenter le filtre 'étendu' de Kalman [7], qui, en linéarisant les équations permet d'obtenir une approximation. Ce filtre n'est pas une solution analytique ! Il est une autre approximation, dont nous allons évaluer la pertinence. Nous nous intéressons à l'écart en terme de RMSE entre les filtres considérés et les vraies valeurs de X. Ici, l'intérêt est d'évaluer la pertinence de nos prédictions météorologiques, en regardant à quel point nos approximations se rapprochent des vraies valeurs.

La figure [4] présente les RMSE des différents filtres pour un nombre de particules allant de 20 à 1000, avec un pas d'échantillonnage de 10. Le DAC se révèle à nouveau être le filtre le plus performant en terme de précision vis à vis des valeurs de X, suivi de peu par le filtre étendu de Kalman. Le BPF est quant à lui le moins précis, mais son RMSE diminue bien plus que celui du FAPF lorsque le nombre de particules augmente. Il convient de noter que si le DAC reste le plus performant, il est également le plus coûteux computationnellement, avec un temps de calcul de 310 min, contre une vingtaine pour le FAPF et une dizaine pour le BPF.

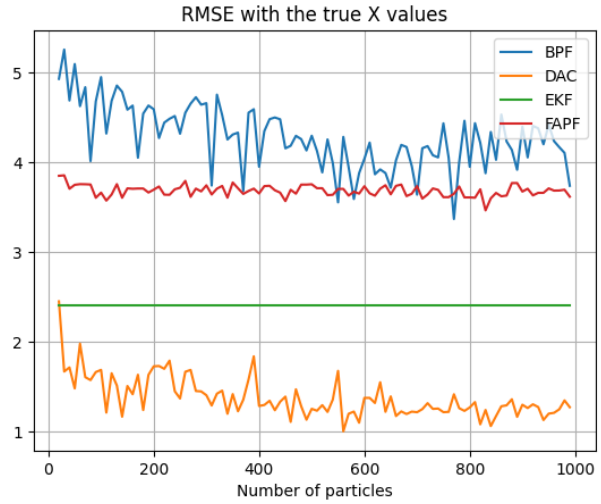


FIGURE 4 – RMSE des différents filtres

Conclusion

En conclusion, les filtres à particules sont particulièrement utiles pour obtenir des informations sur la loi d'un processus à partir de modèles spatio-temporels. En météorologie, la grande dimensionnalité des données implique d'utiliser des algorithmes plus poussés que ceux de base pour avoir de bons résultats.

Lorsque la dimension augmente, un principe en mathématiques est de diviser le problème en plein de problèmes de plus petite dimension. C'est exactement ce que font le *Nested SMC* et le *Divide and Conquer SMC*. Dans le premier cas, on filtre nos observations dans la dimension spatiale avec un premier jeu de particules, puis on filtre dans la dimension temporelle avec un deuxième jeu de particules issu du premier. Dans le second cas, on divise nos distributions en arbres presque binaires et l'on approxime des valeurs unidimensionnelles qu'on regroupe ensuite en repondérant.

Les performances de ces algorithmes dépendent certes du problème considéré, des dépendances spatio-temporelles, et de la connaissance des lois marginales, mais se révèlent systématiquement bien meilleurs dès que la dimension augmente. A la fois dans le modèle linéaire gaussien et dans le modèle de Lorenz, le DAC est le plus performant des algorithmes implémentés.

L'extension directe de ce projet est l'utilisation de ces algorithmes sur de vraies données météorologiques grâce à des centres de supercomputation comme le NSC (*National Supercomputer Center*) présent à l'université de Linköping et mis à disposition du SMHI.

Références

[1] P. le Moigne, "Surfex scientific documentation," vol. 8.1, pp. 107–224, 2018. [Online]. Available : https://www.umr-cnrm.fr/surfex/spip.php?rubrique11

[2] T. Schön and F. Lindsten, "Learning of dynamical systems," August 2017.

[3] A. Doucet and A. Johansen, *A Tutorial on Particle Filtering and Smoothing : Fifteen years later*, 2008.

[4] C. Naesseth, F. Lindsten, and T. Schön, "High-dimensional filtering using nested sequential monte carlo," *arXiv*, no. 1612.09162v1, 2016.

[5] F. R. Crucinio and A. M. Johansen, "A divide and conquer sequential monte carlo approach to high dimensional filtering," 2022.

[6] H. Masnadi-Shirazi, A. Masnadi-Shirazi, and M.-A. Dastgheib, "A step by step mathematical derivation and tutorial on kalman filters," 2019.

[7] T. Schön, "Solving nonlinear state estimation problems using particle filters," 2010.

[8] Pour accéder au code : `https://github.com/candicebaud/liu_smc`