Summary

Monte Carlo methods for numerical weather prediction

Baud Candice

Supervisor : Lindsten Fredrik - *ENSAE supervisor* : Chopin Nicolas

Motivation

Weather forecast is based on the goal to understand and anticipate the fluctuations of nature through the study of atmospheric processes. By combining scientific data, mathematical models, and careful analysis, weather forecasting aims to mitigate the risks associated with severe weather, optimize operations related to transportation, agriculture, energy, and various other industries, all while contributing to the safety, well-being, and resilience of our society in the face of climatic uncertainties. The goal of this project in collaboration with the SMHI (Swedish Meteorological Institute) is to implement particle filtering algorithms on measured data in order to enhance weather predictions.

1 Available data

The available data is extracted from measurements conducted in Northern Finland. The project focuses on the study of soils [1], divided into agricultural land and forests. These soils are divided into layers corresponding to depths. We also define layers above the ground for snow during winter. For each layer, variables such as water quantity, ice quantity, temperature, snow density, and snow age are of interest.

Based on this data, the goal is to implement particle filtering algorithms [2] with multiple objectives. On one hand, we aim to account for measurement uncertainties and knowledge gaps in meteorological processes (referred to as 'noise') in order to filter our data and obtain high-quality information. On the other hand, we aim to deduce information about certain variables in specific layers when those variables are not directly measured, but measurements are available for other layers or other variables. The challenge we are facing is spatio-temporal, as our variables evolve over time but also influence each other spatially between layers. This high dimensionality requires the implementation of more advanced algorithms than traditional filters.

2 Filtering principles

To model our phenomenon, we consider a process x_t (where x_t is multi-dimensional as it contains multiple variables), but we cannot directly observe it or measure it. Instead, we observe y_t, another process related to x_t. For instance, we can imagine that y_t contains the quantity of rain and the outside temperature, while x_t represents an unobservable quantity like the amount of snow. Alternatively, y_t could be a very noisy or imprecise observation of x_t.

Thus, we have defined two processes, x_t and y_t, where one is latent (unobserved) and the other one is observed. We aim to obtain information about the latent process, specifically its probability distribution. Particle filtering involves generating values of x_t that represent hypotheses about the actual values. With knowledge of the value of y_t, we can assess the relevance of each hypothetical value of x_t and assign it a weight.

In this way, for each time step t, we approximate the quantity $p(x_t|y_{1:t})$ using a weighted sum of Dirac measures, as illustrated in Figure [1]. The abscis of each arrow is the particle value, and its ordinate is its weight. By having an infinite number of particles, we recover the density at each point.

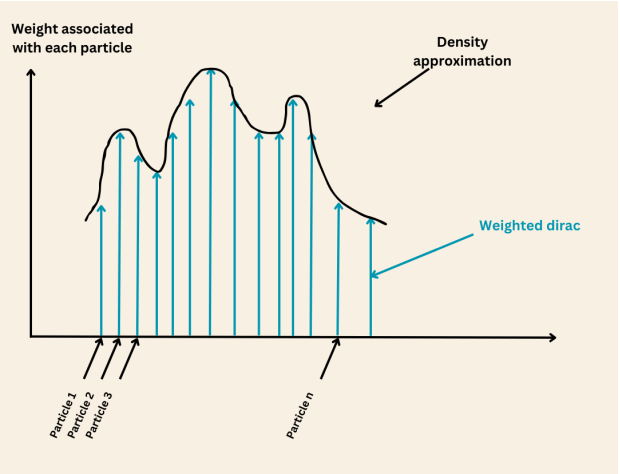


FIGURE 1 – Approximation by a weighted sum of Diracs

More precisely, we define :

$$x_t = f(x_{t-1}) + \epsilon_t \qquad y_t = g(x_t) + \delta_t \qquad \text{where } \epsilon_t \text{ and } \delta_t \text{ are noise terms, typically Gaussian.}$$

During initialization, we generate N particles and evaluate their weights w_t. These particles constitute an approximation of the density $p(x_0|y_0)$. We will then follow the same steps outlined below to approximate the densities $p(x_t|y_{1:t})$ $\forall t$. Starting with our weighted N particles $(x_t^i, w_t^i)i = 1^N$, we resample them to obtain a system $(x_t^{a_t^i}, 1)_{i=1}^N$, where a_t^i denotes the ancestor index of particle i at t. The particles all have equal weight since their multiplicity was adjusted during resampling. Finally, having relevant particles at time t, we propagate them to time $t+1$ using the equation governing the evolution of $x_{t+1}|x_t$ and obtain $(x_{t+1}^i)_{i=1}^N$. We assign a weight to these new particles $(x_{t+1}^i)_{i=1}^N$ according to the formula $w_{t+1}^i = p(y_{t+1}|x_{t+1}^i)$. This approach is intuitive since the weight will be higher if the probability of observing y_{t+1} given x_{t+1}^i is high. In other words, this weight evaluates the consistency of the measurement of y_{t+1} with respect to the hypothetical particle x_{t+1}^i. These N weighted particles approximate the density $p(x_{t+1}|y_{1:t+1})$. We repeat these steps until the maximum considered time.

Based on the principles explained above, our goal is to generate hypotheses about our particles that are as relevant as possible, and for that purpose, it's essential to use the most the available information possible. The most basic and fundamental algorithm is known as the *Bootstrap particle filter* [2], but variations like the *Fully adapted particle filter* [3] allow the inclusion of information from future times in the algorithm during the resampling step, thus generating more relevant particles. These methods are effective in low dimensions, but their accuracy significantly diminishes for more complex problems. In such cases, it is better to implement other algorithms like the *Nested sequential Monte Carlo* [4] or the *Divide and Conquer sequential Monte Carlo* [5].

3 Implementation and results

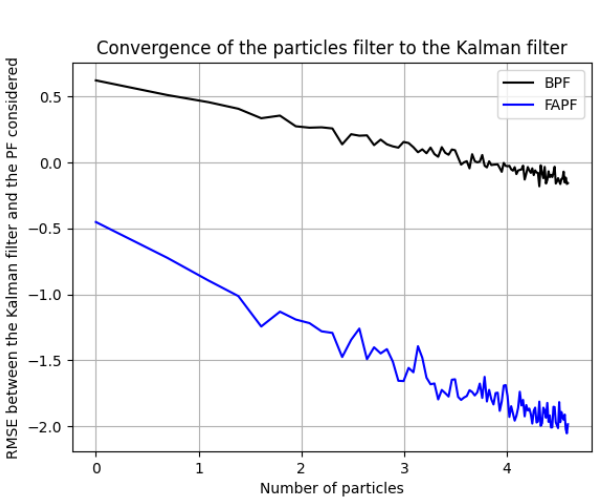
Linear Gaussian Model

I start by implementing all the aforementioned algorithms on a Linear Gaussian model, which means that the functions f and g are linear.

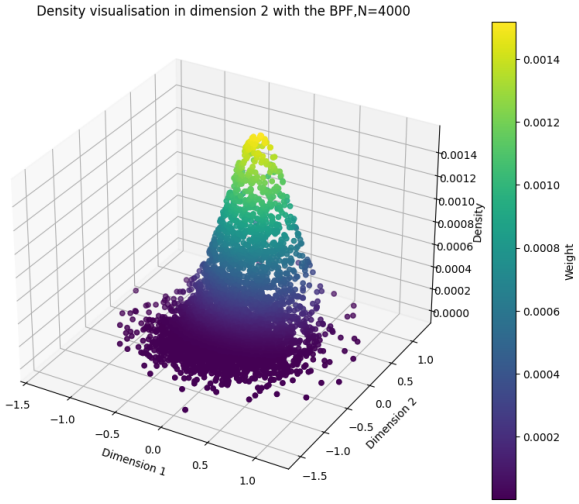
$$x_{t+1} = Ax_t + \epsilon_t \qquad y_t = Bx_t + \delta_t \qquad \epsilon_t \sim N(0, \sigma^2 I), \delta_t \sim N(0, \eta^2 I)$$

These models are interesting for evaluating the effectiveness of our algorithms because there exists an optimal analytical solution provided by the Kalman filter. This filter gives us the mean vector and

the covariance matrix, which completely characterizes the distribution in the Gaussian case. We can then compare our particle filters to the optimal solution provided by Kalman [6]. Figure [2a] shows the difference between our particle filter and the Kalman filter as we increase the number of particles. Figure [2b] shows the density obtained at time T = 20 with the Bootstrap PF for a 2-dimensional Gaussian linear model with parameters $A = \begin{bmatrix} 0.1 & 0.5 \\ 0.4 & 0.3 \end{bmatrix}$, $B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, $\sigma^2 = 0.1, \delta^2 = 0.1$.



(a) Convergence to the Kalman filter



(b) Density of $p(x_T|y_{1:T})$

The left graph [2a] presents the difference (measured by RMSE) between the solutions provided by the filters and the optimal solution given by the Kalman filter. The plot is in a logarithmic scale and allows us to visualize convergence at a rate of $\sqrt{N}$ for the Fully Adapted PF, which is much more efficient than the Bootstrap PF. It should be noted that these convergences depend on parameters :

- **The dimension** : When the dimension increases, the Bootstrap PF becomes increasingly inefficient due to a phenomenon of weight degeneracy that prevents maintaining a sufficient diversity of particles. This means that we do not explore the space enough when generating new particles, which decreases the accuracy of the approximation.

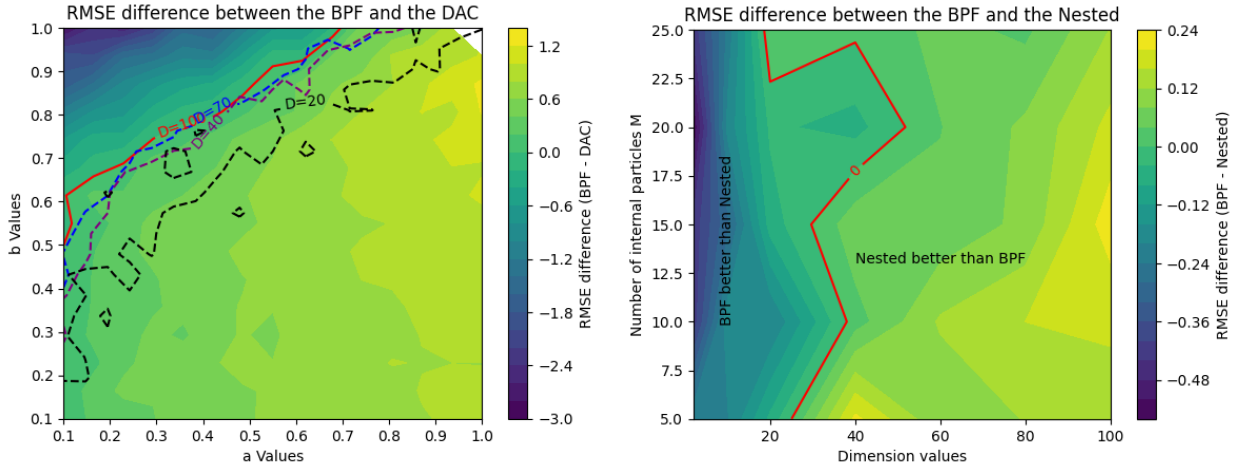
- **The noise level** : The Fully Adapted PF takes into account the information contained in y_t for all t. However, if our observations of y_t are highly inaccurate (high noise level), the information extracted from them is of poor quality and thus affects our approximations. Therefore, when the measurement error of Y is low, the Fully Adapted PF becomes more efficient compared to the Bootstrap PF, which does not take into account the current value of y_t. Conversely, if the measurement of Y is imprecise, the latter does not provide significant added value compared to the Bootstrap PF because the noise does not allow the extraction of relevant information.

In the previous examples, we restricted ourselves to a dimension of 5, which is quite low. In real meteorological data, dimensions are often much larger than 10, and the Bootstrap PF is then not very accurate. I have implemented two other algorithms that, by dividing our multidimensional problem into many smaller-dimensional problems, allow for better approximations.

The first method considered is the *Divide-and-Conquer* algorithm. It divides a given sequence into a tree with one-dimensional numbers at its leaves. One generates sets of one-dimensional particles for each leaf and then moves up the tree by concatenating these particles whenever branches meet. This

way, particles of increasingly higher dimensions are created until reaching the root. Moreover, at each concatenation, weights are adjusted to achieve greater accuracy in our approximations.

The second method considered is the *Nested Monte Carlo* algorithm. It divides the spatio-temporal problem into a spatial problem first and then a temporal problem. For each time iteration t and each particle N, an internal approximation with M particles of the distributions in the spatial direction is performed. Once this approximation is done, for each external particle, the best internal particle is selected for each dimension, and the most relevant external particle is created. Weights are adjusted to obtain the estimation at time t. Then, we proceed to the next time step.



(a) DAC vs BPF for different dimensions

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

Figure [3a] presents the RMSE difference between DAC and BPF for different transition matrices and unit variances. We define $A = a \cdot I_D$ and $B = b \cdot I_D$. This graph allows us to see that the results for BPF and DAC are highly dependent on these parameters. For some combinations of values, even in high dimensions, BPF proves to be more advantageous. These cases remain in the minority, confirming that DAC brings a real improvement. This improvement becomes even more significant as the dimension of the problem increases. Similarly, certain noise levels give an advantage to BPF, an advantage that diminishes as the dimension increases.

Figure [3b] presents the performance of Nested SMC compared to BPF as the dimension increases and the number of internal particles increases. The algorithms have the same computational budget : $M \cdot N$ particles for BPF ; and M internal particles and N external particles for Nested SMC. Nested SMC brings a real improvement as soon as the dimension becomes significant, regardless of the model parameters (a, b, variances).

In other words, DAC and Nested SMC are two efficient algorithms for Gaussian linear models as the dimension increases. DAC is also superior in terms of performance and computation time compared to Nested SMC. Therefore, in the following, we will prefer DAC.

Lorenz model

Once the implementations in the Linear Gaussian model are complete, we can proceed to apply them to meteorological data. We consider the Lorenz96 model, which is deterministic, and we introduce Gaussian noise during the simulation to make it stochastic. The model is given by :

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

Let also y_t such as :

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

Since the model is not linear, it is not possible to use the Kalman filter. Nevertheless, we can implement the 'extended' Kalman filter [7], which, by linearizing the equations, provides an approximation. It's important to note that this filter is not an analytical solution ! We are interested in the RMSE difference between the filters and the true values of X. Here, the goal is to assess the relevance of our meteorological predictions by examining how closely our filters approach the true values.

Figure [4] presents the RMSE of the different filters for a number of particles ranging from 20 to 1000, with a sampling step of 10. Once again, DAC proves to be the most accurate filter in terms of precision concerning the values of X, closely followed by the extended Kalman filter. BPF, on the other hand, is the least precise, but its RMSE decreases much more than that of FAPF as the number of particles increases. It's worth noting that while DAC remains the most performant, it is also the most computationally expensive, with a computation time of 310 minutes, compared to about twenty minutes for FAPF and around ten minutes for BPF.

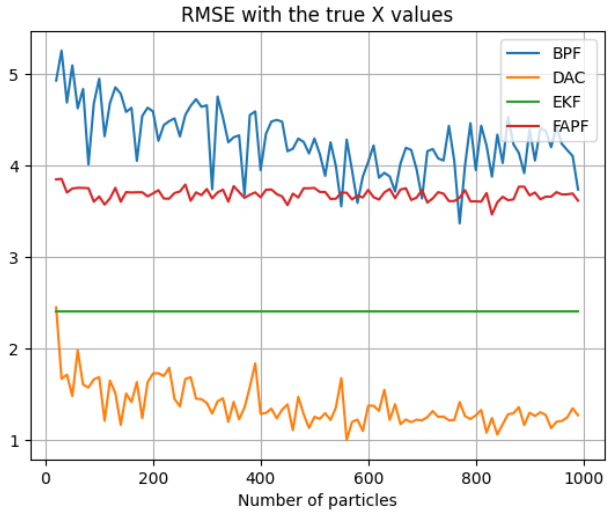


FIGURE 4 – Filters accuracies

Conclusion

In conclusion, particle filters are particularly useful for obtaining information about the distribution of a process from spatio-temporal models. In meteorology, the high dimensionality of data requires the use of more advanced algorithms than basic ones to achieve good results.

When the dimension increases, a mathematical principle is to break down the problem into many smaller-dimensional problems. This is exactly what the Nested SMC and Divide and Conquer SMC algorithms do. In the first case, we filter our observations in the spatial dimension with one set of particles, and then filter in the temporal dimension with a second set of particles derived from the first one. In the second case, we divide our distributions into nearly binary trees and approximate one-dimensional values, which are then grouped together and reweighted.

The performance of these algorithms depends on the specific problem, spatio-temporal dependencies, and knowledge of marginal distributions, but they prove to be much better as the dimension increases. In both the Gaussian linear model and the Lorenz model, DAC is the most efficient of the implemented algorithms, especially as the dimension of the problem increases.

The direct extension of this project is the implementation of these algorithms on real meteorological data using supercomputing centers like the NSC (National Supercomputer Center) at Linköping University, made available to SMHI (Swedish Meteorological and Hydrological Institute).

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] To access the code : `https://github.com/candicebaud/liu_smc`