Nonlinear data assimilation:

Particle filters from a Bayesian perspective

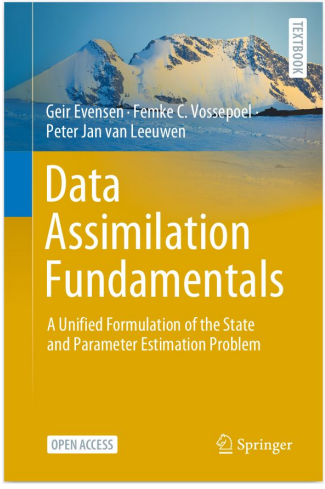
Femke C. Vossepoel

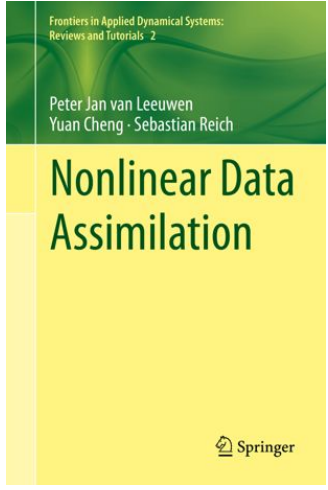
TU Delft

Based on the book available from

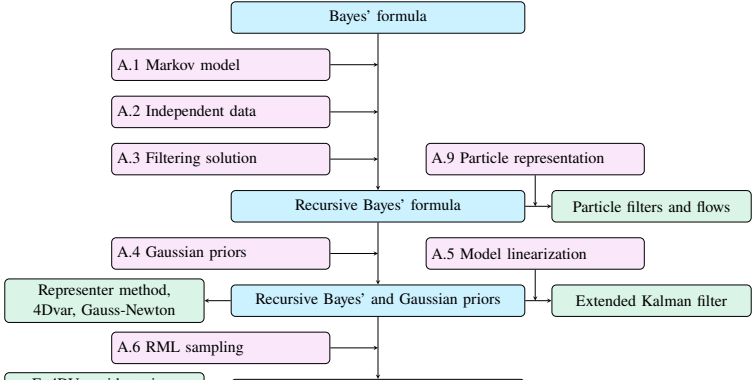
https://github.com/geirev/Data-Assimilation-Fundamentals.git

Full details in:

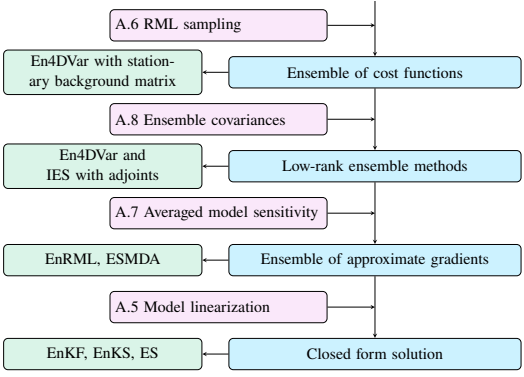




Overview of approximations and methods



Overview of approximations and methods



Approximation 9 (Particle representation of the pdfs)

It is possible to approximate a probability density function by a finite ensemble of N model states (or particles) as

$$f(\mathbf{z}) \approx \sum_{j=1}^{N} \frac{1}{N} \delta(\mathbf{z} - \mathbf{z}_j), \tag{1}$$

where $\delta(\cdot)$ denotes the Dirac-delta function.

Importance sampling Monte Carlo

We can now use what we derived about the probability of $\mathcal{Z}$ given the data, and use Monte Carlo samples to approximate the probability. This involves:

- generate N pseudo-random realisations z_j from $f(\mathbf{z}|\mathbf{d})$ with $j = 1, \ldots, N$.
- evaluate for each realisation the outcome of the forward model and compute the arithmetic average of the results.

Importance sampling Monte Carlo

So, we can approximate the distribution $f(\mathbf{z}|\mathbf{d})$ by:

$$f(\mathbf{z}|\mathbf{d}) = \sum_{j=1}^{N} w_j \delta(\mathbf{z} - \mathbf{z}_j), \tag{2}$$

where $\delta_{\mathbf{z}_j}$ is a Dirac delta, and the so-called likelihood weights w_j given by

$$w_j = \frac{f(\mathbf{d}|\mathbf{z}_j)}{f(\mathbf{d})} = \frac{f(\mathbf{d}|\mathbf{z}_j)}{\sum_{j=1}^{N} f(\mathbf{d}|\mathbf{z}_j))}. \tag{3}$$

denominator: standard self normalization to ensure the weights add up to one, $f(\mathbf{d}) = \int f(\mathbf{d}|\mathbf{z}) f(\mathbf{z})\, d\mathbf{z} \approx \sum_{j=1}^{N} f(\mathbf{d}|\mathbf{z}_j)$.

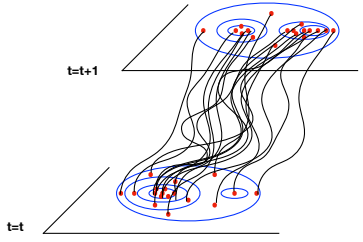
Basic Importance Sampling

In practice, sequential scheme

1. Sample N particles $\mathbf{z}_j$ from initial model probability density $f(\mathbf{z}(t=0))$
2. Integrate all particles forward to measurement time: $f(\mathbf{z}_t|\mathbf{z}_{t-1,j})$ for time t, for each j.
3. Calculate weights w_j, for each sample, and normalize
4. Increase t by 1, and repeat steps 2 and 3

Note: particles are not modified, only their weights.

Concept of particle filtering



Ensemble of N realisations (particles) to estimate pdf evolution (figure courtesy Peter Jan van Leeuwen)

Posterior is proportional to prior times likelihood

Prior:

$$f(\mathbf{z}) = \Sigma_{j=1}^{N} \frac{1}{N} \delta(\mathbf{z} - \mathbf{z}_j)$$

Likelihood:

$f(\mathbf{d}|\mathbf{z}) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-} \frac{1}{2}(\frac{\mathbf{d}-\mathbf{z}}{\sigma})^2$

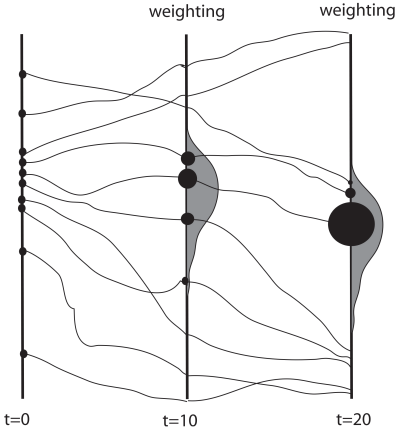
(Gaussian, can also be Lorentz function)

Posterior:

$f(\mathbf{z}) \propto f(\mathbf{d}|\mathbf{z}) f(\mathbf{z})$

Ensemble degeneracy in particle filtering

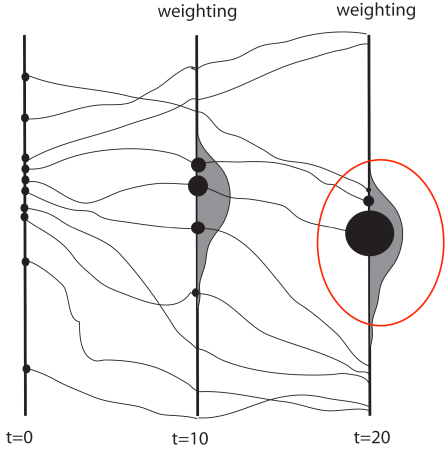
Size of circles indicates weight (This and following 2 figures from van Leeuwen et al., 2015)



Samples that are closest to observations obtain largest weight.

Degeneracy

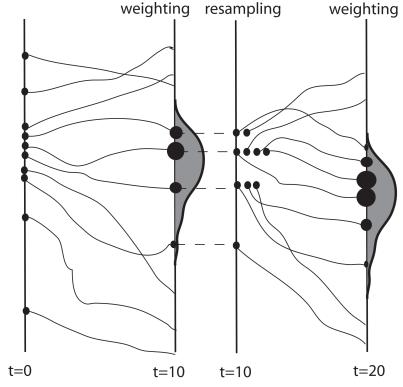
Size of circles indicates weight



Some samples move very far from the observations and obtain a low weight. This means that effectively, there are only few samples left!

Importance Resampling

Size of circles indicates weight



Samples that are closest to observations obtain largest weight and are being duplicated. Low weight samples are removed from the ensemble.

Pseudo algorithm

1: Input: $\mathbf{Z} \in \Re^{n \times N}$ ▷ Initial model state-vector ensemble
2: Input: $\mathbf{d} \in \Re^{m}$ ▷ Measurements vector
3: **for** $j = 1, N$ **do** ▷ Log Particle weights
4: $\tilde{w}_j = \ln f(\mathbf{d}|\mathbf{z}_j)$
5: **end for**
6: $\tilde{w}_{\max} = \max_j(\tilde{w}_j)$ ▷ Max of log weights
7: **for** $j = 1, N$ **do** ▷ Unnormalized Particle weights
8: $\tilde{w}_j = \exp(\tilde{w}_{\max} - \tilde{w}_j)$
9: **end for**
10: **for** $j = 1, N$ **do** ▷ Particle weights
11: $w_j = \frac{\tilde{w}_j}{\sum_i \tilde{w}_i}$
12: **end for**
13: **call Resample**$(\mathbf{w}, \mathbf{I})$ ▷ Resampling step
14: **for** $j = 1, N$ **do** ▷ Resample ensemble
15: $\mathbf{z}_j = \mathbf{z}_{I_j}$
16: **end for**

When to resample? Effective ensemble size

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} w_i^2}, \tag{4}$$

Effective ensemble size: measure for particle divergence.
w_i: normalized weights
resampling: typically when $N_{\text{eff}} \leq 0.8N$

Resampling in practice

1. Sample N particles $\mathbf{z}_j$ from initial model probability density $f(\mathbf{z}(t=0))$
2. Integrate all particles forward to measurement time: $f(\mathbf{z}_t|\mathbf{z}_{t-1,j})$ for time t, for each j.
3. Calculate weights w_j, for each sample, and normalize
4. Resample, while choosing for each particle a weight equal to $1/N$
5. Repeat steps 2, 3, 4 until all observations have been processed

Note: some particles are lost: analysis not smooth anymore!

Methods of resampling

For step 4, the resampling, there are three common approaches:

- **probabilistic resampling:** sample from distribution of ensemble members. Straightforward, but introducing sampling noise.
- **residual resampling:** multiply weights by N, take integer part of resulting weight (call this n) and take n copies of these particles. Subtract all integer parts of Nw_j, and to complete the set of N samples, sample randomly from the resulting distribution. Lower sampling noise than probabilistic resampling.
- **stochastic universal resampling:** Place all weights after each other between $[0, 1]$. Draw a random number between $[0, 1/N]$ and start with a sample at this place. Then sample again at each $1/N$ distance from the previous sample. Samples with a large weight will have a higher chance of being 'hit'. Lowest sampling noise of three options.

Stochastic Universal Resampling

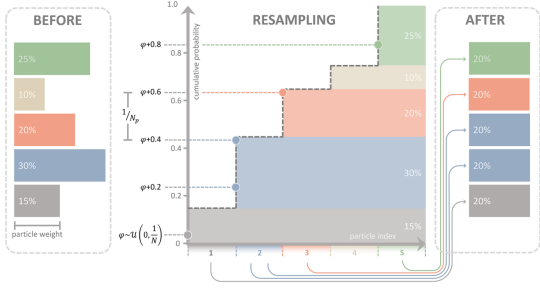


Figure 1. Schematic illustration of stochastic universal resampling for $N = 5$. Starting with an ensemble of nonuniformly weighted particles (left), we construct a cumulative probability function (dashed gray line, center). After drawing a random offset φ from a uniform distribution $\mathcal{U}(min{=}0, max{=}1/N)$, we obtain resampled particle indices by sampling this function in additive increments of $1/N$. Finally, each particle slot inherits the variables of its respective resampled particle index ($1 \rightarrow 1{,}2 \rightarrow 2{,}3 \rightarrow 2{,}4 \rightarrow 3{,}5 \rightarrow 5$) and the weights are reset (right, equation 15).

Ramgraber et al. 2019, 10.1029/2019WR026777

Why would we apply resampling?

What factors influence degeneracy?

- **Size of state vector**, the larger the size of the state vector, the higher the chance of degeneracy, see next slide on asymptotic limit of *Snyder et al, 2018*.
- **Number of independent observations**. Hypothetical example of one observation at 0.1σ, and another at 0.2σ from model, and a total of N_y observations with variance 1σ. Using a Gaussian likelihood, it can be shown that the weight of the second observation is $3 \cdot 10^{-7}$ times smaller than weight of first observation when number of observations is 1000. This becomes $0,22$ when N_y is 100. So the larger the number of independent observations, the higher the chance of degeneracy, even with inaccurate observations.
- **Proposal densities**. Instead of drawing samples from our prior, we can also draw samples from a density function that minimises degeneracy: the proposal density.

Condition for collapse - asymptotic limit of Snyder et al (2008)

$$log_{10}N_e = 0.05N_x + 0.78$$

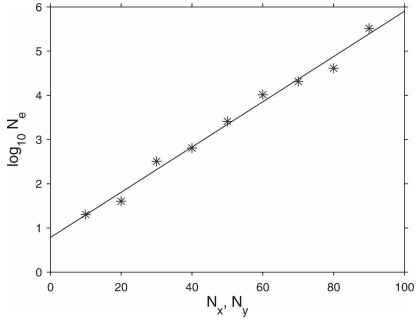


FIG. 2. The ensemble size N_e as a function of N_x (or N_y) required if the posterior mean estimated by the particle filter is to have average squared error less than the prior or observations, in the simple example considered in the text. Asterisks show the simulation results, averaged over 400 realizations. The best-fit line is given by $\log_{10}N_e = 0.05N_x + 0.78$.

Snyder et al. 2008, 10.1175/2008MWR2529.1

Proposal densities

Instead of sampling from the prior, we sample form another pdf, the *proposal pdf* $q(\mathbf{z})$. Rewriting Bayes:

$$f(\mathbf{z}|\mathbf{d}) = \frac{f(\mathbf{d}|\mathbf{z})}{f(\mathbf{d})} f(\mathbf{z}) = \frac{f(\mathbf{d}|\mathbf{z})}{f(\mathbf{d})} \frac{f(\mathbf{z})}{q(\mathbf{z})} q(\mathbf{z}). \tag{5}$$

Assume we have samples from $q(\mathbf{z})$ we can write

$$q(\mathbf{z}) = \sum_{i=1}^{N} \frac{1}{N} \delta(\mathbf{z} - \mathbf{z}_i), \tag{6}$$

Proposal density

Again, we write:

$$f(\mathbf{z}|\mathbf{d}) = \sum_{i=1}^{N} w_i \delta(\mathbf{z} - \mathbf{z}_i), \tag{7}$$

but now with weights

$$w_i = \frac{f(\mathbf{d}|\mathbf{z}_i)}{N \sum_{j=1}^{N} f(\mathbf{d}|\mathbf{z}_j)} \frac{f(\mathbf{z}_i)}{q(\mathbf{z}_i)}. \tag{8}$$

likelihood weights are multiplied by *proposal weights*.

Moving prior to posterior: Particle Flow Filter

$$\frac{\partial \mathbf{z}_j}{\partial s} = \mathbf{m}_s(\mathbf{z}_j) = -\mathbf{B}\frac{1}{N}\sum_{l=1}^{N}\Big(\mathcal{K}(\mathbf{z}_j, \mathbf{z}_l) \cdot \nabla_{\mathbf{z}_l} \ln f(\mathbf{z}_l|\mathbf{d}) + \nabla_{\mathbf{z}_l} \cdot \mathcal{K}(\mathbf{z}_j, \mathbf{z}_l)\Big). \tag{9}$$

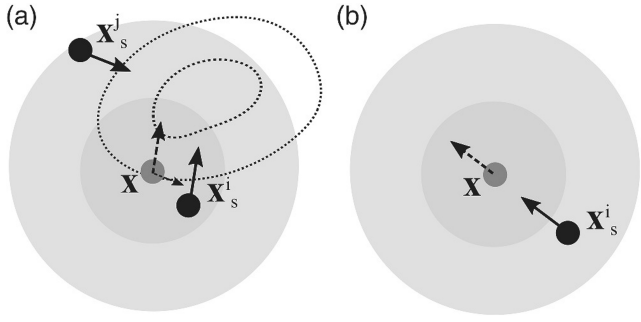


Illustration of particle flow: black dots are the states of the particles, gray dot is the state $\mathbf{x}$ ($\mathbf{z}$ in Eq. 9) at which the particle flow is being evaluated. (a): weighted average of the gradient of the logarithm of the posterior (first term in right-hand side, attracting force), kernel indicated in gray, posterior indicated in dashed contours. (b) divergence from the kernel (second term in right-hand side, repulsive force). Figure and description from Hu and van Leeuwen (2021).

Summary

- Representing the pdf with particles allows us to handle non-Gaussian distributions
- The importance sampling particle filter suffers from filter degeneracy
- Resampling the particles can help to avoid this
- For higher dimensions, proposal densities can help
- The particle flow filter uses a transport from prior to posterior distribution for efficient estimation of the posterior

Questions?