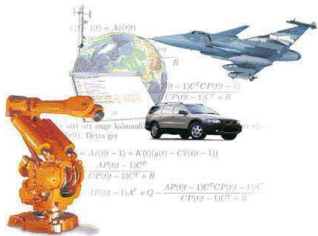


DREAMS Tutorial – The particle filter



Thomas Schön

Division of Automatic Control
Linköping University
Sweden



The **goal of this tutorial** is to derive the particle filter (PF) so that you can start implementing (and deriving) your own PF algorithms to solve problems.

Given the computational tools that we have today it can be rewarding to resist the linear Gaussian convenience!!





In solving problems we have to make assumptions and a **model** will to a large extent capture many of these assumptions.

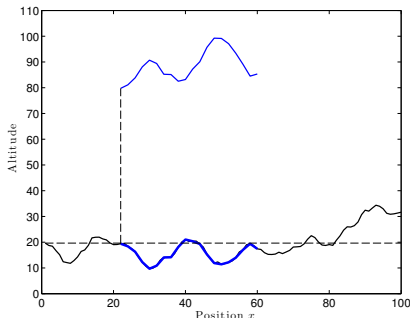
A **model** is a compact and interpretable representation of the data that is observed.

Using models to solve problems requires three key ingredients;

1. **Data:** Measurements from the system we are interested in.
2. **Model:** We use probabilistic models, allowing us to employ probability theory to **represent and systematically work with the uncertainty** that is inherent in most data.
3. **Inference algorithm:** The topic of this tutorial is the particle filter.



Consider a toy 1D localization problem.



Dynamic model:

$$x_{t+1} = x_t + u_t + v_t,$$

where x_t denotes position, u_t denotes velocity (known), $v_t \sim \mathcal{N}(0, 5)$ denotes an unknown disturbance.

Measurements:

$$y_t = h(x_t) + e_t.$$

where $h(\cdot)$ denotes the world model (here the terrain height) and $e_t \sim \mathcal{N}(0, 1)$ denotes an unknown disturbance.



Task: Find the state x_t based on a set of measurements $y_{1:t} \triangleq \{y_1, \dots, y_t\}$. Do this by computing the filter PDF $p(x_t | y_{1:t})$.

The particle filter maintains an approximation according to

$$p(x_t | y_{1:t}) = \sum_{i=1}^N w_t^i \delta_{x_t^i}(x_t),$$

where each sample x_t^i is referred to as a **particle**.

For intuition: Think of each particle as one simulation of the system state (in this example the horizontal position) and only keep the good ones.



Highlights two **key capabilities** of the PF:

1. Automatically handles an unknown and dynamically changing number of hypotheses.
2. Work with nonlinear/non-Gaussian models.



1. Problem formulation

- Probabilistic modeling of dynamical systems
- Strategies for state inference

2. Monte Carlo methods

- The idea
- Importance sampling (IS)
- Trying to use IS in solving the filtering problem
- The particle filter
- Particle filtering examples

3. Conclusions and outlook



Definition (State space model (SSM))

A state space model (SSM) consists of a Markov process $\{x_t\}_{t \geq 1}$ and a measurement process $\{y_t\}_{t \geq 1}$, related according to

$$\begin{aligned}x_{t+1} \mid x_t &\sim f_{\theta,t}(x_{t+1} \mid x_t, u_t), \\y_t \mid x_t &\sim g_{\theta,t}(y_t \mid x_t, u_t), \\x_1 &\sim \mu_{\theta}(x_1),\end{aligned}$$

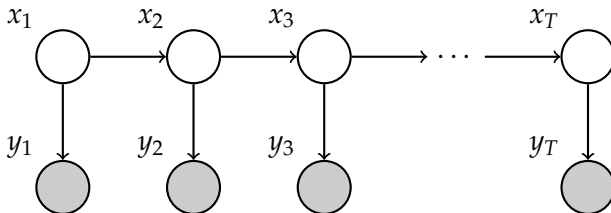
where $x_t \in \mathbb{R}^{n_x}$ denotes the state, $u_t \in \mathbb{R}^{n_u}$ denotes a known deterministic input signal, $y_t \in \mathbb{R}^{n_y}$ denotes the observed measurement and $\theta \in \Theta \subseteq \mathbb{R}^{n_{\theta}}$ denotes any unknown (static) parameters.



In engineering literature, the SSM is often written in terms of a difference equation and an accompanying measurement equation,

$$\begin{aligned}x_{t+1} &= \bar{f}_{\theta,t}(x_t, u_t) + v_{\theta,t}, \\ y_t &= \bar{g}_{\theta,t}(x_t, u_t) + e_{\theta,t}.\end{aligned}$$





The SSM is an instance of a graphical model called **Bayesian network**, or **belief network**.



Definition (Linear Gaussian State Space (LGSS) model)

The time-invariant LGSS model is given by

$$\begin{aligned}x_{t+1} &= Ax_t + Bu_t + v_t, \\y_t &= Cx_t + Du_t + e_t,\end{aligned}$$

where $x_t \in \mathbb{R}^{n_x}$ denotes the state, $u_t \in \mathbb{R}^{n_u}$ denotes the known input signal and $y_t \in \mathbb{R}^{n_y}$ denotes the observed measurement. The initial state and the noise are distributed according to

$$\begin{pmatrix} x_1 \\ v_t \\ e_t \end{pmatrix} \sim \mathcal{N} \left(\begin{pmatrix} \mu \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} P_1 & 0 & 0 \\ 0 & Q & S \\ 0 & S^T & R \end{pmatrix} \right).$$



State inference refers to the problem of finding information about the state(s) $x_{k:l}$ based on the available measurements $y_{1:t}$.

We will represent this information using **PDFs**.

Name	PDF
Filtering	$p(x_t y_{1:t})$
Prediction	$p(x_{t+1} y_{1:t})$
k -step prediction	$p(x_{t+k} y_{1:t})$
Joint smoothing	$p(x_{1:T} y_{1:T})$
Marginal smoothing	$p(x_t y_{1:T}), t \leq T$
Fixed-lag smoothing	$p(x_{t-l+1:t} y_{1:t}), l > 0$
Fixed-interval smoothing	$p(x_{r:t} y_{1:T}), r < t \leq T$

Notation $y_{1:t} \triangleq \{y_1, y_2, \dots, y_t\}$.



State filtering problem: Find x_t based on $\{u_{1:T}, y_{1:T}\}$ when the model is given by,

$$\begin{aligned}x_{t+1} \mid x_t &\sim f(x_{t+1} \mid x_t, u_t), \\y_t \mid x_t &\sim g(y_t \mid x_t, u_t), \\x_1 &\sim \mu(x_1), \quad (\theta \sim p(\theta)).\end{aligned}$$

Strategy: Compute the filter PDF $p(x_t \mid y_{1:t})$.



Let a and b be continuous random variables.

- Conditional probability: $p(a, b) = p(a | b)p(b)$.
- Marginalization: $p(a) = \int p(a, b)db$.
- Bayes' rule:

$$p(a | b) = \frac{p(b | a)p(a)}{p(b)}$$

- Markov property: $p(x_t | x_1, \dots, x_{t-1}) = p(x_t | x_{t-1})$.



Summarizing this development, we have the **measurement update**

$$p(x_t | y_{1:t}) = \frac{\overbrace{g(y_t | x_t)}^{\text{measurement}} \overbrace{p(x_t | y_{1:t-1})}^{\text{prediction pdf}}}{p(y_t | y_{1:t-1})},$$

and the **time update**

$$p(x_t | y_{1:t-1}) = \int \underbrace{f(x_t | x_{t-1})}_{\text{dynamics}} \underbrace{p(x_{t-1} | y_{1:t-1})}_{\text{filtering pdf}} dx_{t-1}.$$



Monte Carlo methods



In solving inference problems we are typically faced with various integration problems, which tend to live in large dimensional spaces.

As an example we mention **expectation**, which is for example used to obtain a point estimate. A commonly used point estimate is the conditional mean

$$\hat{x}_{t|t} = \mathbb{E} [x_t | y_{1:t}] = \int x_t p(x_t | y_{1:t}) dx_t.$$

Monte Carlo methods provides **computational solutions**, where the obtained accuracy is limited by our computational resources.

Monte Carlo methods respects the model and the expressions we are trying to approximate.



(Very) restrictive assumption: Assume that we have N samples $\{z^i\}_{i=1}^N$ from the target density $\pi(z)$,

$$\hat{\pi}(z) = \sum_{i=1}^N \frac{1}{N} \delta_{z^i}(z)$$

Allows for the following approximation of the integral,

$$\mathbb{E}[g(z)] = \int g(z) \pi(z) dz \approx \int g(z) \sum_{i=1}^N \frac{1}{N} \delta_{z^i}(z) dz = \frac{1}{N} \sum_{i=1}^N g(z^i)$$

$$'' \int + \delta \rightarrow \sum ''$$



The integral

$$I(g(z)) \triangleq \mathbb{E} [g(z)] = \int g(z) \pi(z) dz.$$

is approximated by

$$\hat{I}_N(g(z)) = \frac{1}{N} \sum_{i=1}^N g(z^i).$$

The strong law of large numbers tells us that

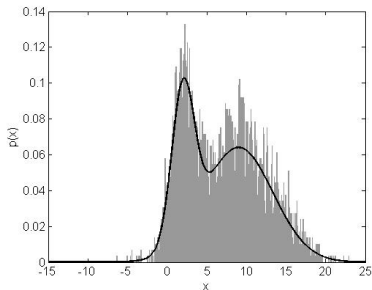
$$\hat{I}_N(g(z)) \xrightarrow{\text{a.s.}} I(g(z)), \quad N \rightarrow \infty,$$

and the central limit theorem state that

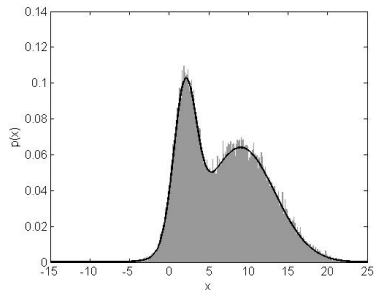
$$\frac{\sqrt{N} \left(\hat{I}_N(g(z)) - I(g(z)) \right)}{\sigma_g} \xrightarrow{d} \mathcal{N}(0, 1), \quad N \rightarrow \infty.$$



$$\pi(z) = 0.3\mathcal{N}(z | 2, 2) + 0.7\mathcal{N}(z | 9, 19)$$



5 000 samples



50 000 samples

Obvious problem: In general we are **not** able to directly sample from the density we are interested in.



1. Problem formulation

- Probabilistic modeling of dynamical systems
- Strategies for state inference

2. Monte Carlo methods

- The idea
- Importance sampling (IS)
- Trying to use IS in solving the filtering problem
- The particle filter
- Particle filtering examples

3. Conclusions and outlook



Algorithm 1 Importance sampler (IS)

1. Sample $z^i \sim q(z)$.
 2. Compute the weights $\tilde{w}^i = \tilde{\pi}(z^i) / q(z^i)$.
 3. Normalize the weights $w^i = \tilde{w}^i / \sum_{j=1}^N \tilde{w}^j$.
-

Each step is carried out for $i = 1, \dots, N$.



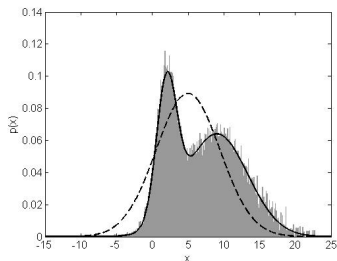
IS does not provide samples from the target density, but the samples $\{z^i\}_{i=1}^N$ together with the normalized weights $\{w^i\}_{i=1}^N$ provides an **empirical approximation** of the target density,

$$\hat{\pi}(z) = \sum_{i=1}^N w^i \delta_{z^i}(z).$$

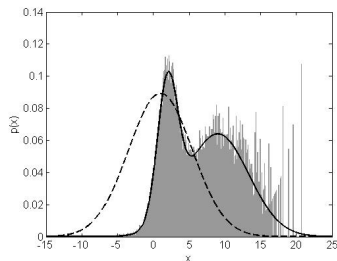
When this approximation is inserted into $I(g(z)) = \int g(z)\pi(z)dz$ the resulting estimate is

$$\hat{I}^N(g(z)) = \sum_{i=1}^N w^i g(z^i).$$





$$q_1(x) = \mathcal{N}(5, 20) \text{ (dashed curve)}$$



$$q_2(x) = \mathcal{N}(1, 20) \text{ (dashed curve)}$$

50 000 samples used in both simulations.

Lesson learned: It is important to be careful in selecting the importance density.



Recall that the nonlinear filtering problem amounts to computing the filter PDF $p(x_t | y_{1:t})$ when the model is given by

$$\begin{aligned}x_{t+1} | x_t &\sim f(x_{t+1} | x_t), \\y_t | x_t &\sim g(y_t | x_t), \\x_1 &\sim \mu(x_1).\end{aligned}$$

We have showed that the solution is

$$\begin{aligned}p(x_t | y_{1:t}) &= \frac{g(y_t | x_t)p(x_t | y_{1:t-1})}{p(y_t | y_{1:t-1})}, \\p(x_t | y_{1:t-1}) &= \int f(x_t | x_{t-1})p(x_{t-1} | y_{1:t-1})dx_{t-1}.\end{aligned}$$

Relevant idea: Try to solve this using importance sampling!!



We have just motivated the following proposal (which is just one of many possible choices)

$$q(x_{1:t}) = \mu(x_1) \prod_{s=2}^t f(x_s | x_{s-1})$$

In practice this means:

- At time $t = 1$ we sample $x_1 \sim \mu(x_1)$.
- At each time $s = 2, \dots, t$ we sample $x_s^i \sim f(x_s | x_{s-1}^i)$.

This completes step one of the importance sampler. What about sequential computation of the importance weights?



Algorithm 2 SIS targeting $p(x_{1:t} \mid y_{1:t})$

1. Sample $x_1^i \sim \mu(x_1)$ and initialize the weights, $w_0^i = 1/N$.
 2. **for** $t = 1, 2 \dots$ **do**
 - (a) Compute the unnormalized weights $\tilde{w}_t^i = p(y_t \mid x_t^i)w_{t-1}^i$.
 - (b) Normalize the weights $w_t^i = \tilde{w}_t^i / \sum_{j=1}^N \tilde{w}_t^j$.
 - (c) Sample $x_{t+1}^i \sim f(x_{t+1} \mid x_t^i)$ and store $x_{1:t+1}^i = \{x_{1:t}^i, x_{t+1}^i\}$.
-

Each step is carried out for $i = 1, \dots, N$.



Consider the following LGSS model

$$\begin{aligned}x_{t+1} &= 0.7x_t + v_t, & v_t &\sim \mathcal{N}(0, 0.1), \\y_t &= 0.5x_t + e_t, & e_t &\sim \mathcal{N}(0, 0.1), \\p(x_1) &= \mathcal{N}(x_1 \mid 0, 0.1),\end{aligned}$$

We will now make use of the **SIS algorithm** to compute an approximation of the filtering density

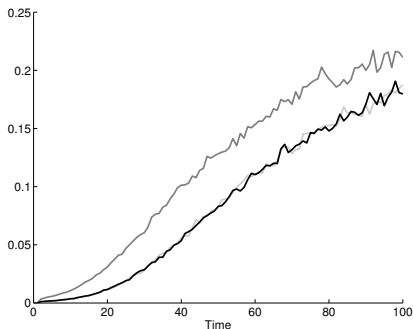
$$\hat{p}(x_t \mid y_{1:t}) = \sum_{i=1}^N w_t^i \delta_{x_t^i}(x_t).$$

Study

- Point estimate $\hat{x}_{t|t} = \int x_t \hat{p}(x_t \mid y_{1:t}) dx_t = \sum_{i=1}^N w_t^i x_t^i$.
- The weights w_t^i .



Use $T = 100$ samples, 1 000 realisations of data and $N = 500$,
 $N = 5\,000$ and $N = 50\,000$, respectively.



Compare with the true filter density (from KF). $\text{RMSE}(\hat{x}_{t|t}^{\text{SIS}} - \hat{x}_{t|t}^{\text{KF}})$

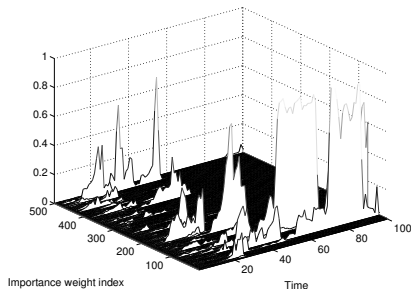
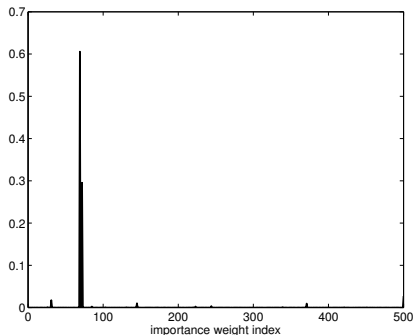
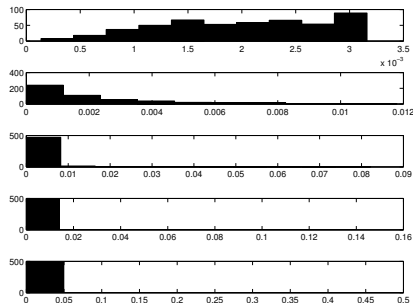


Illustration of the problem with the weights.





The 500 importance weights at $t = 100$.



Histograms of the weights for $t = 2, 5, 10, 20$ and $t = 50$, respectively.

Very important question: How do we resolve this weight degeneracy problem?



Idea: Remove the weights from the representation!

This of course leads us to the next question, **How?**



The SIS representation of the target density is

$$\hat{\pi}^1(z) = \sum_{i=1}^N w^i \delta_{z^i}(z).$$

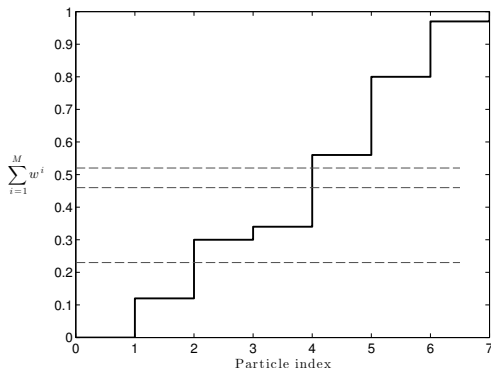
An unweighted representation of the target density can be created by **resampling with replacement**. This is done by generating a new sample z^i for each $i = 1, \dots, N$, where

$$\mathbb{P}(z^i = \tilde{z}^j) = w^j, \quad j = 1, \dots, N.$$

The resulting unweighted representation is

$$\hat{\pi}^2(z) = \sum_{i=1}^N \frac{1}{N} \delta_{z^i}(z).$$





Illustrating how resampling with replacement works (using 7 particles).

1. Compute the cumulative sum of the weights.
2. Generate $u \sim \mathcal{U}[0, 1]$.

Three new samples are generated in the figure above, corresponding to sample 2, 4 and 4.



Algorithm 3 Sampling Importance Resampler (SIR)

1. Sample $z^i \sim q(z)$.
 2. Compute the weights $\tilde{w}^i = \tilde{\pi}(z^i) / q(z^i)$.
 3. Normalize the weights $w^i = \tilde{w}^i / \sum_{j=1}^N \tilde{w}^j$.
 4. Resample $\{w_t^i, z^i\}$ to obtain equally weighted samples $\{1/N, \tilde{z}^j\}$.
-

Each step is carried out for $i = 1, \dots, N$.

Note that step 1 – 3 corresponds to the importance sampler.



Algorithm 4 Particle filter (SIS and resampling)

1. Sample $x_1^i \sim \mu(x_1)$ and initialize the weights, $\tilde{w}_0^i = 1/N$.
 2. **for** $t = 1, 2 \dots$ **do**
 - (a) Compute the unnormalized weights $\tilde{w}_t^i = p(y_t | x_t^i)w_{t-1}^i$.
 - (b) Normalize the weights $w_t^i = \tilde{w}_t^i / \sum_{j=1}^N \tilde{w}_t^j$.
 - (c) Resample $\{w_t^i, x_t^i\}$ to obtain equally weighted samples $\{1/N, \tilde{x}_t^j\}$.
 - (d) Sample $x_{t+1}^i \sim f(x_{t+1} | x_t^i)$ and store $x_{1:t+1}^i = \{x_{1:t}^i, x_{t+1}^i\}$.
-

Each step is carried out for $i = 1, \dots, N$.



Consider the same LGSS model used in illustrating the SIS algorithm,

$$\begin{aligned}x_{t+1} &= 0.7x_t + v_t, & v_t &\sim \mathcal{N}(0, 0.1), \\y_t &= 0.5x_t + e_t, & e_t &\sim \mathcal{N}(0, 0.1), \\p(x_1) &= \mathcal{N}(x_1 \mid 0, 0.1).\end{aligned}$$

We will now make use of SIS and resampling (**particle filter**) to compute an approximation of the filtering density

$$\hat{p}(x_t \mid y_{1:t}) = \sum_{i=1}^N w_t^i \delta_{x_t}(\tilde{x}_t^i).$$

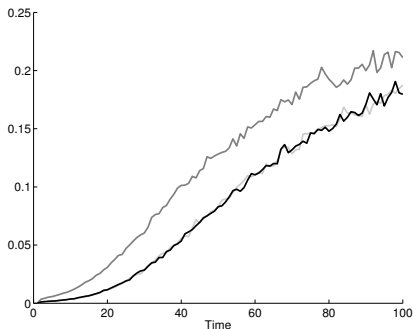
Study

- Point estimate $\hat{x}_{t|t} = \int x_t \hat{p}(x_t \mid y_{1:t}) dx_t = \sum_{i=1}^N w_t^i \tilde{x}_t^i$.
- The weights w_t^i .

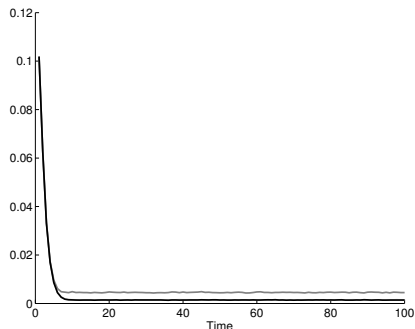


Same setting as before, exactly the same data.

Compare with the true filter density (from KF), $\text{RMSE}(\hat{x}_{t|t}^{\text{PF}} - \hat{x}_{t|t}^{\text{KF}})$

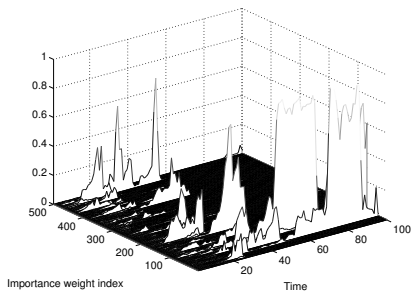


SIS

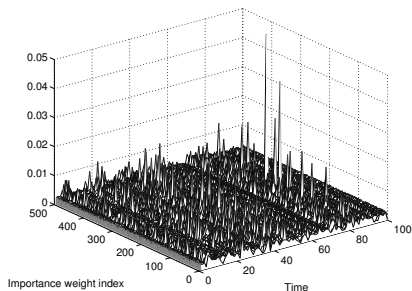


PF





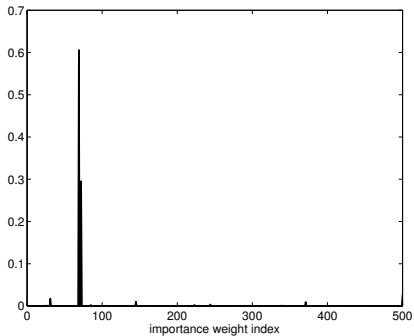
SIS



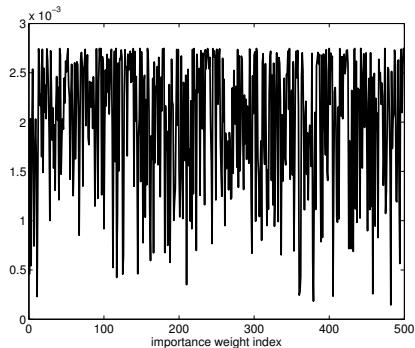
PF

Note the different scaling!





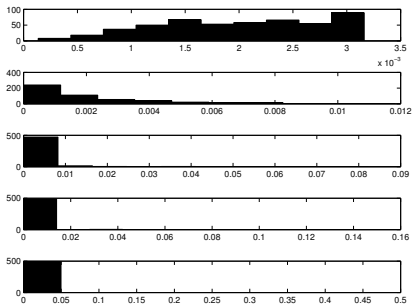
SIS



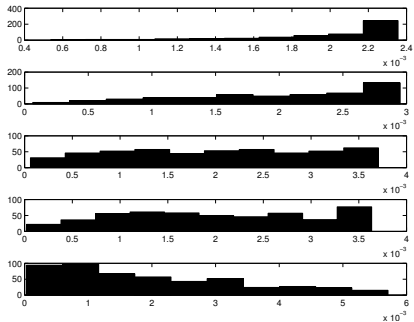
PF

Note the different scaling!





SIS



PF



“Whenever you are working on a nonlinear inference method, always make sure that it solves the linear special case first.”

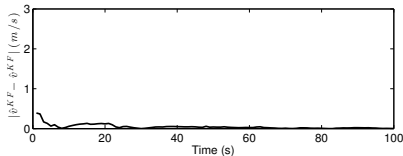
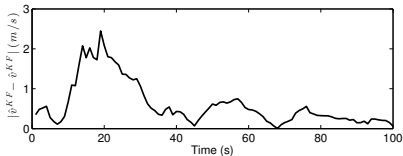
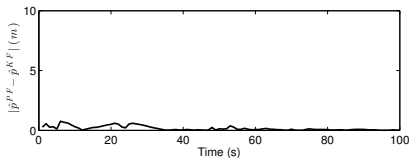
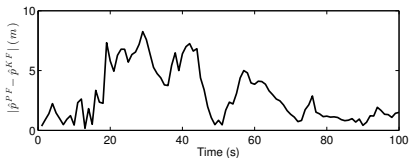
Consider the following LGSS model (simple one dimensional positioning example)

$$\begin{pmatrix} p_{t+1} \\ v_{t+1} \\ a_{t+1} \end{pmatrix} = \begin{pmatrix} 1 & T_s & T_s^2/2 \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_t \\ v_t \\ a_t \end{pmatrix} + \begin{pmatrix} T_s^3/6 \\ T_s^2/2T_s \end{pmatrix} v_t, \quad v_t \sim \mathcal{N}(0, Q),$$

$$y_t = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_t \\ v_t \\ a_t \end{pmatrix} + e_t, \quad e_t \sim \mathcal{N}(0, R).$$

The KF provides the true filtering density, which implies that we can compare the PF to the truth in this case.





Using 200 particles.

Using 20 000 particles.

The PF estimate converge as the number of particles tends to infinity.

Xiao-Li Hu, Thomas B. Schön and Lennart Ljung. **A Basic Convergence Result for Particle Filtering.** *IEEE Transactions on Signal Processing*, 56(4):1337-1348, April 2008. [\[pdf\]](#)

D. Crisan and A. Doucet, **A survey of convergence results on particle filtering methods for practitioners,** *IEEE Transactions on Signal Processing*, vol. 50, no. 3, pp. 736-746, 2002. [\[pdf\]](#)



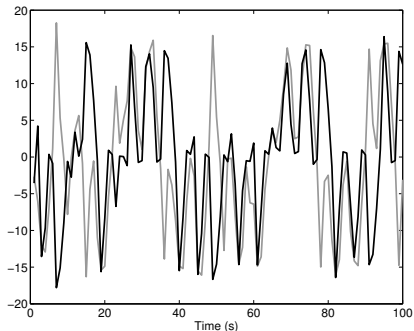
Consider the following SSM (standard example in PF literature)

$$x_{t+1} = \frac{x_t}{2} + \frac{25x_t}{1+x_t^2} + 8 \cos(1.2t) + v_t, \quad v_t \sim \mathcal{N}(0, 0.5),$$
$$y_t = \frac{x_t^2}{20} + e_t, \quad e_t \sim \mathcal{N}(0, 0.5).$$

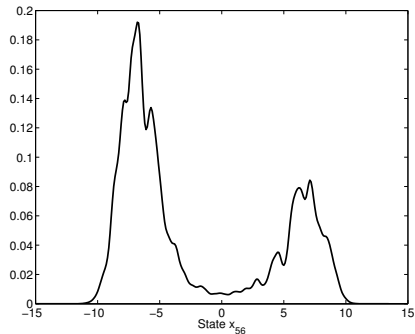
What is tricky with this model?

The best (only?) way of really understanding something is to implement it yourself.





True state (gray) and PF conditional mean estimate (black).



PF estimate of the filtering pdf $\hat{p}(x_{56} | y_{1:56})$.

Another indication that the conditional mean point estimate is dangerous.





This implies that if we are interested in the smoothing density

$$p(x_{1:T} \mid y_{1:T})$$

or some of its marginals we are **forced** to use different algorithms, which leads us to **particle smoothers**.

However, the algorithms derived in this tutorial are perfectly valid for solving the filtering problem, i.e., estimating $p(x_t \mid y_{1:t})!$



Conclusion

- Goal: Derive the PF so that you can start implementing (and deriving) your own PF algorithms to solve problems.
 - Details and references are available the manuscript.
-

Outlook

- Particle smoothers (PS)
- Rao-Blackwellized PF (RBPF) and RBPS
- Using particle methods to infer static parameters
 - Frequentist approach: e.g., via EM based approaches
 - Bayesian approach: e.g., Particle MCMC
- and much more...

Should you find this interesting I have a PhD course – *Computational inference in dynamical systems* – covering this material, see

users.isy.liu.se/rt/schon/course_CIDS.html

