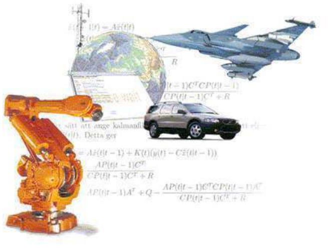
An introduction to the theory and practice of particle filters



Thomas Schön

Division of Automatic Control
Linköping University
Sweden

The goal of this talk

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

Dynamical systems are everywhere

Model – data – inference algorithm

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

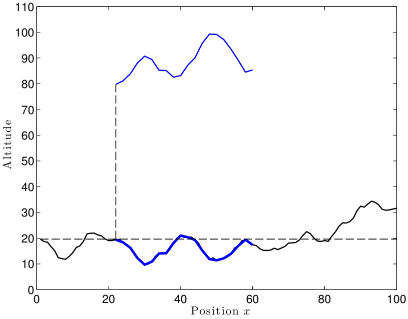
A **model** is a compact and interpretable representation of the data that is observed.

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.

Particle filter – introductory example (I/IV)

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.

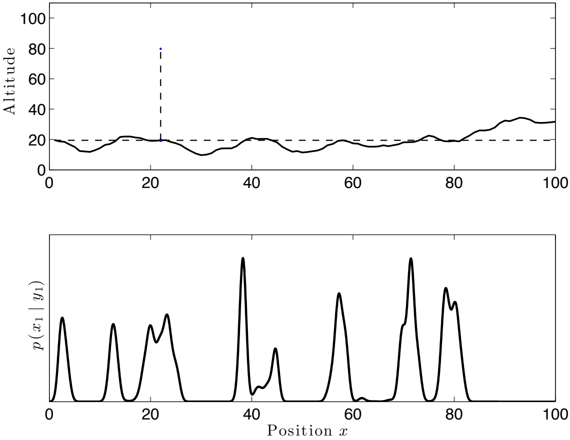
Task: Find the state x_t based on a set of measurements $y_{1:t} \triangleq \{y_1, \ldots, y_t\}$. Do this by computing the filter PDF $p(x_t \mid y_{1:t})$.

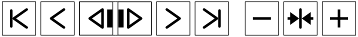
The particle filter maintains an approximation according to

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

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.


100
80
60
40
20
0
Altitude
0
20
40
60
80
100
$p(x_1 \mid y_1)$
0
20
40
60
80
100
Position x



The simple 1D localization example is an illustration of a problem involving a multimodal filter PDF.

The example also highlights the **key capabilities** of the PF:

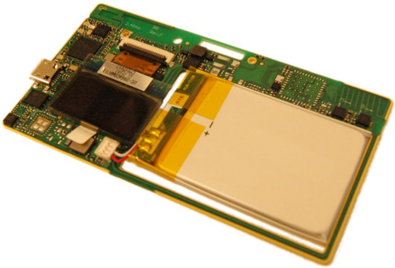
1. To automatically handle an unknown and dynamically changing number of hypotheses.
2. Work with nonlinear/non-Gaussian models.

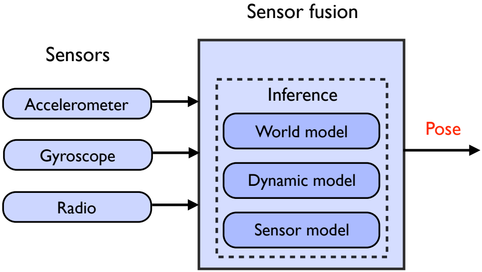
We have implemented a similar localization solution for this aircraft (Gripen).

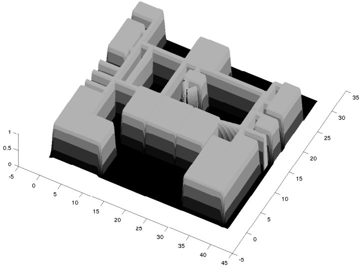
Industrial partner: Saab

Aim: Compute the position of a person moving around indoors using sensors (inertial, magnetometer and radio) located in an ID badge and a map.

Industrial partner: Xdin




Sensor fusion
Sensors
Accelerometer
Gyroscope
Radio
Inference
World model
Dynamic model
Sensor model
Pose


1
0.5
0
-5
0
5
10
15
20
25
30
35
40
45
-5
0
5
10
15
20
25
30
35

Show movie

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))

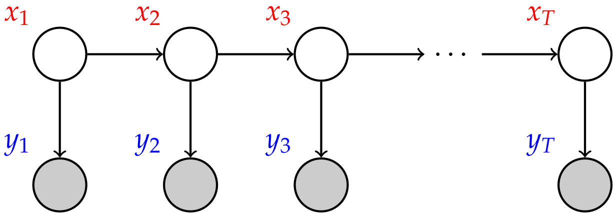
A state space model (SSM) consists of a Markov process $\{x_t\}_{t\geq 1}$ and a measurement process $\{y_t\}_{t\geq 1}$, related according to

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

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

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

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: $p(a, b) = p(a \mid b)p(b)$.
- Marginalization: $p(a) = \int p(a, b) \mathrm{d}b$.
- Bayes' rule:

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

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

From application of Bayes' theorem we have

$$p(x_t \mid y_{1:t}) = p(x_t \mid y_t, y_{1:t-1}) = \frac{p(y_t \mid x_t, y_{1:t-1})p(x_t \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})}.$$

The measurements from an SSM are conditionally independent, which results in

$$p(x_t \mid y_{1:t}) = \frac{g(y_t \mid x_t)p(x_t \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})},$$

commonly referred to as the **measurement update**.

To find the prediction density we start by considering the joint PDF

$$\begin{aligned} p(x_t, x_{t-1} \mid y_{1:t-1}) &= p(x_t \mid x_{t-1}, y_{1:t-1}) p(x_{t-1} \mid y_{1:t-1}) \\ &= f(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{1:t-1}). \end{aligned}$$

Let us now marginalize w.r.t.

$$\begin{aligned} p(x_t \mid y_{1:t-1}) &= \int p(x_t, x_{t-1} \mid y_{1:t-1}) \mathrm{d}x_{t-1} \\ &= \int f(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{1:t-1}) \mathrm{d}x_{t-1}, \end{aligned}$$

which is referred to as the **time update**.

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

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

and the **time update**

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

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

$$\widehat{x}_{t|t} = \mathrm{E}\left[x_t \mid y_{1:t}\right] = \int x_t p(x_t \mid y_{1:t}) \mathrm{d}x_t.$$

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 $\pi(z)$,

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

Allows for the following approximation of the integral,

$$\mathrm{E}\left[\varphi(z)\right] = \int \varphi(z)\pi(z)\mathrm{d}z \approx \int \varphi(z) \sum_{i=1}^{N} \frac{1}{N}\delta_{z^i}(z)\mathrm{d}z = \frac{1}{N}\sum_{i=1}^{N}\varphi(z^i)$$

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

The Monte Carlo idea (II/II)

The integral

$$I(\varphi(z)) \triangleq \mathrm{E}\left[\varphi(z)\right] = \int \varphi(z)\pi(z)\mathrm{d}z.$$

is approximated by

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

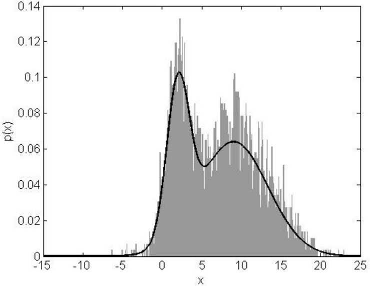
The strong law of large numbers tells us that

$$\widehat{I}_N(\varphi(z)) \xrightarrow{\text{a.s.}} I(\varphi(z)), \qquad N \to \infty,$$

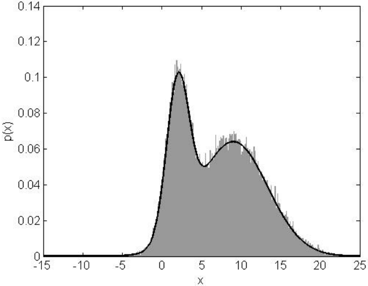
and the central limit theorem state that

$$\frac{\sqrt{N}\left(\widehat{I}_N(\varphi(z)) - I(\varphi(z))\right)}{\sigma_\varphi} \xrightarrow{\text{d}} \mathcal{N}\left(0, 1\right), \qquad N \to \infty.$$

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



5 000 samples



50 000 samples

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

Outline

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

Consider the integral

$$I(\varphi(z)) \triangleq \mathrm{E}\left[\varphi(z)\right] = \int \varphi(z)\pi(z)\mathrm{d}z.$$

and introduce a proposal distribution $q(z)$ and let $z' \sim q(z)$.

$$\begin{aligned}\mathrm{E}\left[\varphi(z)\right] &= \int \varphi(z)\pi(z)\mathrm{d}z = \int \varphi(z)\frac{\pi(z)}{q(z)}q(z)\mathrm{d}z \\ &= \int \varphi(z)W(z)q(z)\mathrm{d}z = \mathrm{E}\left[\varphi(z')W(z')\right],\end{aligned}$$

where we have introduced the **weight function** $W(z) \triangleq \pi(z)/q(z)$.

By construction, it is easy to generate samples from $q(z)$. We can construct a Monte Carlo estimator for the integral by sampling independently $z^i \sim q(z)$ for $i = 1, \ldots, N$ and setting,

$$\widetilde{I}^N_{\text{IS}}(\varphi) = \frac{1}{N} \sum_{i=1}^{N} W(z^i)\varphi(z^i).$$

where the importance weights $W(z^i = \pi(z^i)/q(z^i))$ accounts for the discrepancy between the proposal and the target densities.

Importance sampling - normalization

We often work with the **normalized** importance sampling estimator,

$$\widehat{I}^{N}_{\mathsf{IS}}(\varphi) = \sum_{i=1}^{N} w^i \varphi(z^i),$$

where $\{w^i\}_{i=1}^N$ denote the normalized importance weights, defined according to

$$w^i \triangleq \frac{\widetilde{w}^i}{\sum_{j=1}^{N} \widetilde{w}^j}.$$

Algorithm 1 Importance sampler (IS)

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

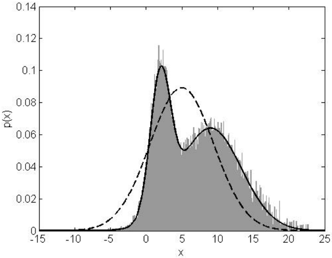
Each step is carried out for $i = 1, \ldots N$.

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,

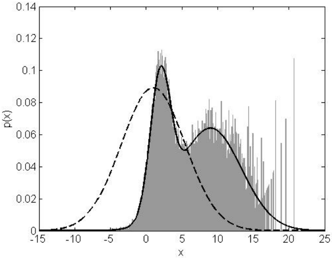
$$\widehat{\pi}(z) = \sum_{i=1}^{N} w^i \delta_{z^i}(z).$$

When this approximation is inserted into $I(\varphi(z)) = \int \varphi(z)\pi(z)\mathrm{d}z$ the resulting estimate is

$$\widehat{I}_{\mathsf{IS}}^N(\varphi) = \sum_{i=1}^{N} w^i \varphi(z^i).$$



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



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

$50\,000$ samples used in booth 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 $p(x_t \mid y_{1:t})$ when the model is given by

$$x_{t+1} \mid x_t \sim f_t(x_{t+1} \mid x_t),$$
$$y_t \mid x_t \sim g_t(y_t \mid x_t),$$
$$x_1 \sim \mu(x_1).$$

We have showed that the solution is

$$p(x_t \mid y_{1:t}) = \frac{g(y_t \mid x_t)p(x_t \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})},$$
$$p(x_t \mid y_{1:t-1}) = \int f(x_t \mid x_{t-1})p(x_{t-1} \mid y_{1:t-1})\mathrm{d}x_{t-1}.$$

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

Sequential sampling from the proposal

Let us use the following proposal (which is just one of many possible choices)

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

In practice this means:

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

This completes step one of the importance sampler. We can now show that the importance weights are given by

$$\widetilde{w}_t^i = g(y_t \mid x_t^i) w_{t-1}^i.$$

A sequential importance sampler (SIS)

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** $t = 1, 2 \ldots$ **do**
   (a) Compute the unnormalized weights $\widetilde{w}_t^i = g(y_t \mid x_t^i) w_{t-1}^i$.
   (b) Normalize the weights $w_t^i = \widetilde{w}_t^i / \sum_{j=1}^N \widetilde{w}_t^j$.
   (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 $i = 1, \ldots N$.

Consider the following LGSS model

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

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

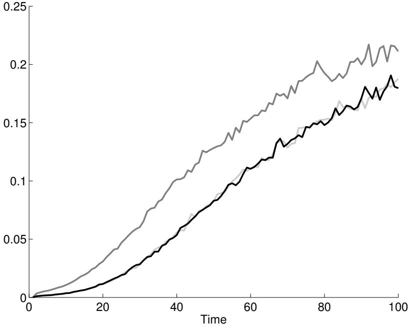
$$\widehat{p}(x_t \mid y_{1:t}) = \sum_{i=1}^{N} w_t^i \delta_{x_t^i}(x_t).$$

Study

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

SIS example (II/III)

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). RMSE($\widehat{x}_{t|t}^{\text{SIS}} - \widehat{x}_{t|t}^{\text{KF}}$)

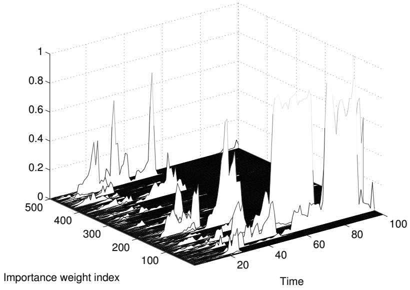
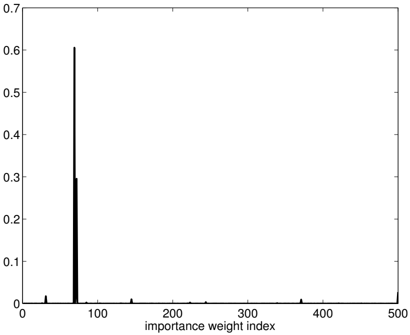


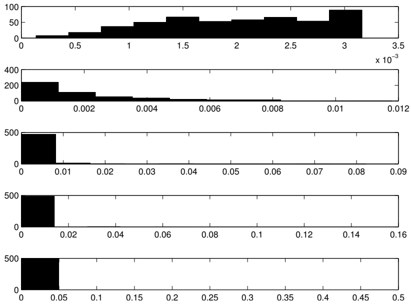
Illustration of the problem with the weights.

AUTOMATIC CONTROL
REGLERTEKNIK
LINKÖPINGS UNIVERSITET

SIS example (III/III)



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

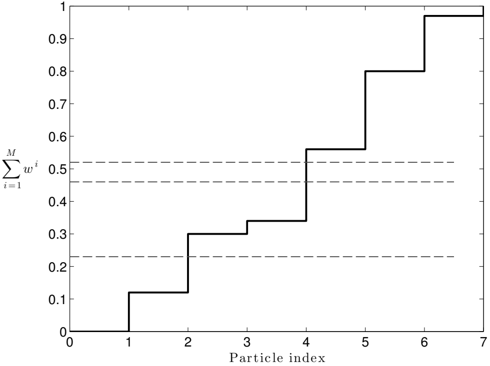
$$\widehat{\pi}^1(z) = \sum_{i=1}^{N} w^i \delta_{\widetilde{z}^i}(z).$$

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

$$\mathbb{P}\left(z^i = \widetilde{z}^j\right) = w^j, \qquad j = 1, \dots, N.$$

The resulting unweighted representation is

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



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

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, $\widetilde{w}_0^i = 1/N$.
2. **for** $t = 1, 2 \ldots$ **do**
   (a) Compute the unnormalized weights $\widetilde{w}_t^i = g(y_t \mid x_t^i)$.
   (b) Normalize the weights $w_t^i = \widetilde{w}_t^i / \sum_{j=1}^N \widetilde{w}_t^j$.
   (c) Resample $\{w_t^i, x_t^i\}$ to obtain equally weighted samples $\{1/N, \widetilde{x}_t^j\}$.
   (d) 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 $i = 1, \ldots N$.

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

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

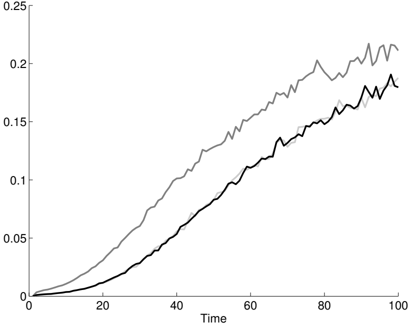
Study

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

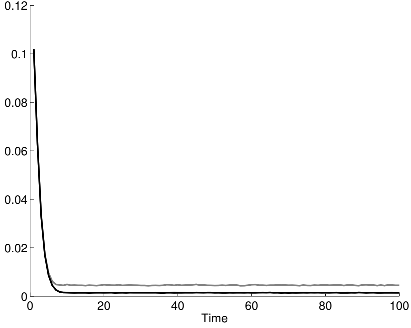
PF example (II/V)

Same setting as before, exactly the same data.

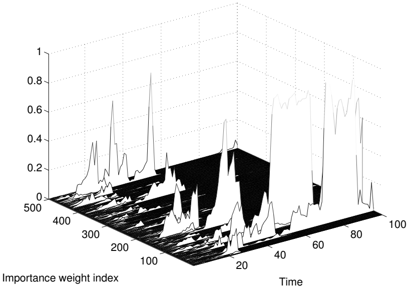
Compare with the true filter density (from KF), RMSE($\widehat{x}^{\text{PF}}_{t|t} - \widehat{x}^{\text{KF}}_{t|t}$)



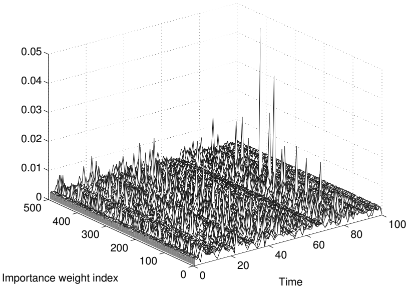
SIS



PF

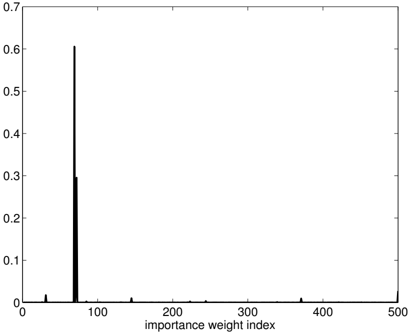


SIS

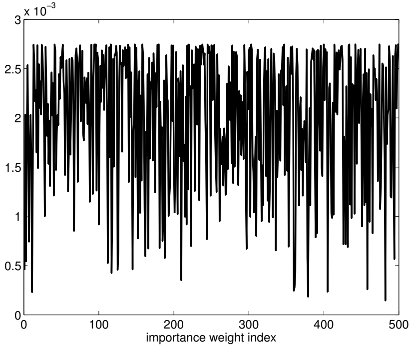


PF

Note the different scaling!

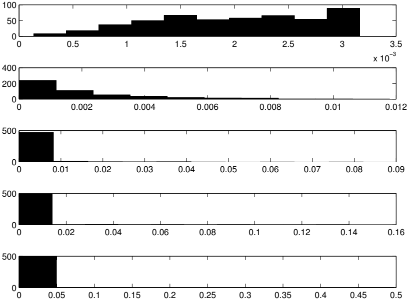


SIS



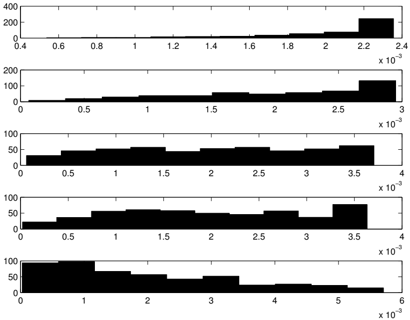
PF

Note the different scaling!



100
50
0
0 0.5 1 1.5 2 2.5 3 3.5
x 10^-3
400
200
0
0 0.002 0.004 0.006 0.008 0.01 0.012
500
0
0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09
500
0
0 0.02 0.04 0.06 0.08 0.1 0.12 0.14 0.16
500
0
0 0.05 0.1 0.15 0.2 0.25 0.3 0.35 0.4 0.45 0.5

SIS



400
200
0
0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2 2.2 2.4
x 10^-3
200
100
0
0 0.5 1 1.5 2 2.5 3
x 10^-3
100
50
0
0 0.5 1 1.5 2 2.5 3 3.5 4
x 10^-3
100
50
0
0 0.5 1 1.5 2 2.5 3 3.5 4
x 10^-3
100
50
0
0 1 2 3 4 5 6
x 10^-3

PF

An LGSS example (I/II)

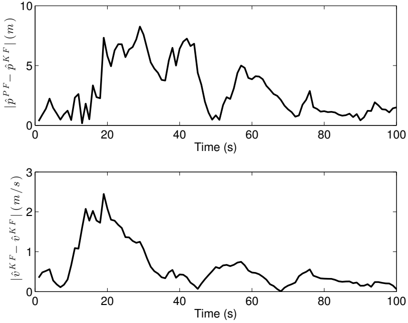
"Whenever you are working on a nonlinear inference method, always make sure that it solves the linear special case first."

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

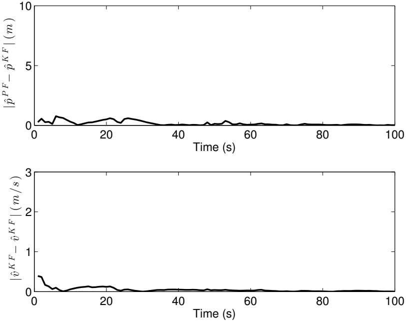
$$\begin{pmatrix} p_{t+1} \\ v_{t+1} \\ a_{t+1} \end{pmatrix} = \begin{pmatrix} 1 & T_s & T_s^2/2 \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_t \\ v_t \\ a_t \end{pmatrix} + \begin{pmatrix} T_s^3/6 \\ T_s^2/2T_s \end{pmatrix} v_t, \qquad v_t \sim \mathcal{N}(0, Q),$$

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

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]

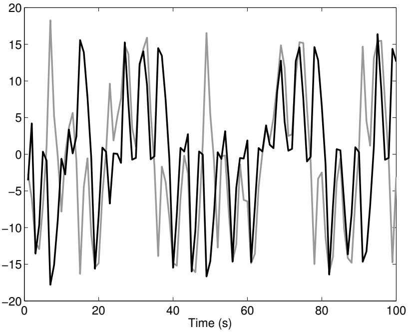
D. Crisan and A. Doucet, **A survey of convergence results on particle filtering methods for practitioners**, *IEEE Transactions on Signal Processing*, vol. 50, no. 3, pp. 736-746, 2002. [pdf]

Consider the following SSM (standard example in PF literature)

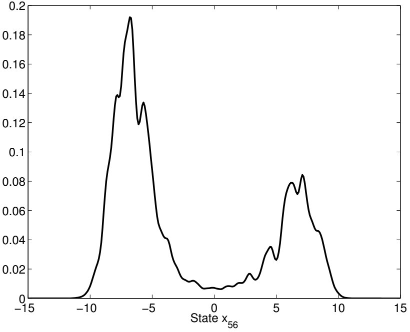
$$
\begin{aligned}
x_{t+1} &= \frac{x_t}{2} + \frac{25x_t}{1+x_t^2} + 8\cos(1.2t) + v_t, & v_t &\sim \mathcal{N}(0, 0.5),\\
y_t &= \frac{x_t^2}{20} + e_t, & e_t &\sim \mathcal{N}(0, 0.5).
\end{aligned}
$$

What it tricky with this model?

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



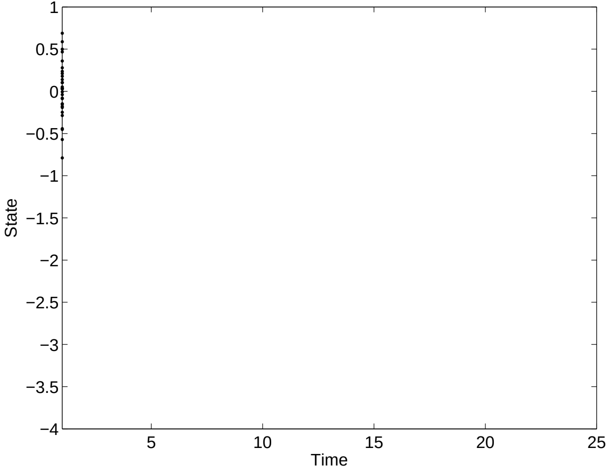
True state (gray) and PF conditional mean estimate (black).



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

Another indication that the conditional mean point estimate is dangerous.

Illustration of the particle degeneracy problem



Implication of the particle degeneracy problem

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.

Outlook

- Particle smoothers (PS)
- Rao-Blackwellized PF (RBPF) and RBPS
- Using particle methods to infere 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`