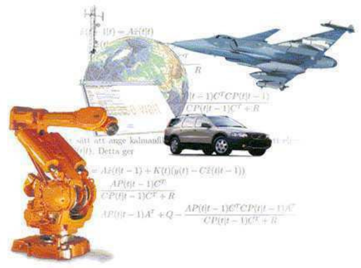
Part 3 - Nonlinear state inference using sequential Monte Carlo



Thomas Schön

Division of Automatic Control
Linköping University
Sweden

The aim – Part 3

The **aim in part 3** is to introduce the particle filter and the particle smoother.

This will be done by first introducing the importance sampler and then show how this foundation can be used to derive a first working particle filter.

Outline

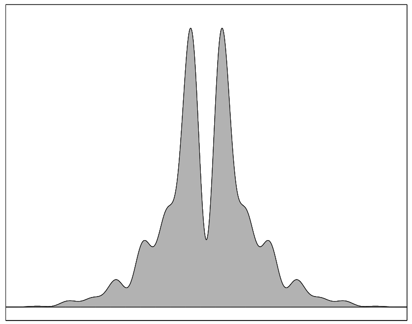
1. Basic sampling methods
   a) Rejection sampling
   b) Importance sampling
2. Particle filtering
3. Particle smoothing
   a) Forward filtering backward smoothing (FFBSm)
   b) Forward filtering backward simulation (FFBSi)

Rejection sampling (I/VII)

Rejection sampling is a Monte Carlo method that produce i.i.d. samples from a target distribution

$$\pi(z) = \frac{\widetilde{\pi}(z)}{C_\pi},$$

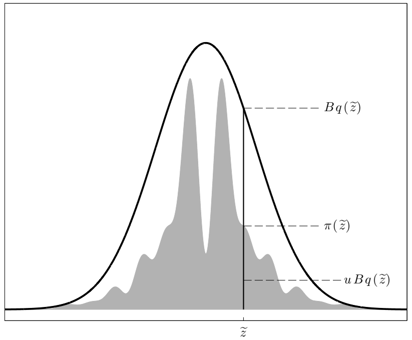
where $\widetilde{\pi}(z)$ can be evaluated and C_π is a normalization constant.



Key idea: Generate random numbers uniformly from the area under the graph of the target distribution $\pi(z)$.

Just as hard as the original problem, but what if...

Rejection sampling (II/VII)



Generate a sample $\tilde{z}$ from a proposal distribution $q(z)$ and a sample $u \sim \mathcal{U}[0,1]$.

The sample $\tilde{z}$ is then an i.i.d. sample from the target if

$$u \le \frac{\pi(\tilde{z})}{Bq(\tilde{z})}$$

Rejection sampling (III/VII)

Assumptions:

1. It is easy to sample from $q(z)$.
2. There exists a constant B such that $\pi(z) \le Bq(z), \forall z \in \mathcal{Z}$.
3. The support of $q(z)$ includes the support of $\pi(z)$, i.e., $q(z) > 0$ when $\pi(z) > 0$.

Algorithm 1 Rejection sampling (RS)

1. Sample $\tilde{z} \sim q(z)$.
2. Sample $u \sim \mathcal{U}[0,1]$.
3. If $u \le \frac{\pi(\tilde{z})}{Bq(\tilde{z})}$ accept $\tilde{z}$ as a sample from $\pi(z)$ and go to 1.
4. Otherwise, reject $\tilde{z}$ and go to 1.

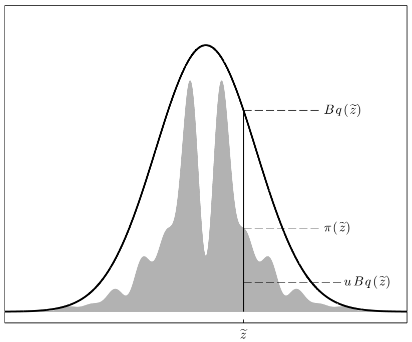
Rejection sampling (IV/VII)

- The procedure can be used with multivariate densities in the same way.
- The rejection rate depends on B, choose B as small as possible, while still satisfying $\pi(z) \le Bq(z), \forall z \in \mathcal{Z}$.
- Choosing a good proposal distribution $q(z)$ is very important.
- Rejection sampling is used to construct fast **particle smoothers** via backward simulation.

Rejection sampling (V/VII)

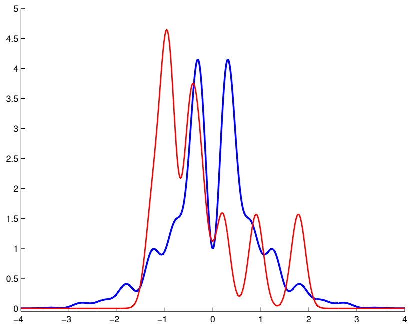
Task: Generate N i.i.d. samples from the following distribution,

$$\pi(z) = \frac{1}{C_\pi} e^{-\frac{1}{2}z^2} \left(\sin(6z)^2 + 3\cos(z)^2 \sin(4z)^2 + 1 \right),$$

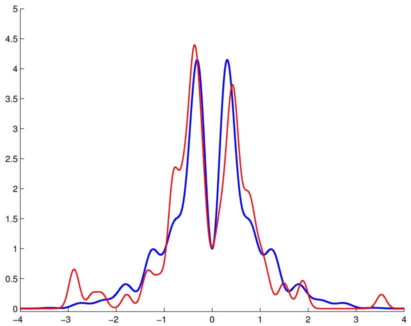


Solution: Use rejection sampling where $q(z) = \mathcal{N}(z \mid 0, 1)$ and $B = 12$.

Rejection sampling (VI/VII)

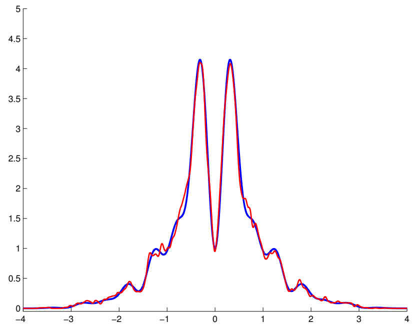


(a) 10 samples

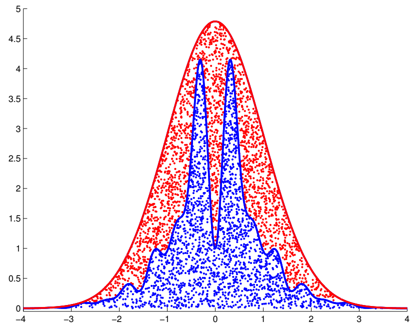


(b) 100 samples

Rejection sampling (VII/VII)



(c) 10000 samples



(d) Showing rejected (red) and accepted (blue) samples

Importance sampling (I/VIII)

Algorithm 2 Importance sampler (IS)

1. Generate N i.i.d. samples $\{z^i\}_{i=1}^N$ from the proposal distribution $q(z)$.
2. Compute the importance weights

$$\widetilde{w}^i = \frac{\widetilde{\pi}(z^i)}{\widetilde{q}(z^i)}, \qquad i = 1, \dots, N.$$

3. Normalize the importance weights

$$w^i = \frac{\widetilde{w}^i}{\sum_{j=1}^N \widetilde{w}^j}, \qquad i = 1, \dots, N.$$

Importance sampling – (II/VIII)

An alternative interpretation of IS

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

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

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

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

Let us revisit the same problem (scalar LGSS) used in illustrating the EM algorithm,

$$\begin{aligned} x_{t+1} &= \theta x_t + v_t, \\ y_t &= \frac{1}{2}x_t + e_t, \end{aligned} \qquad \begin{pmatrix} v_t \\ e_t \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix} \right).$$

$p(x_1) = \mathcal{N}(x_1 \mid 0, 0.1)$. The true parameter value for θ is given by $\theta^\star = 0.9$. We use $p(\theta) = \mathcal{N}(\theta \mid \mu_\theta, \sigma_\theta^2)$ as prior distribution for θ.

The identification problem is now to determine the parameter θ on the basis of the observations $y_{1:T}$ and the above model, using the IS algorithm. The result will be an estimate of the posterior distribution $p(\theta \mid y_{1:T})$.

The importance sampler will target

$$\pi(\theta) = p(\theta \mid y_{1:T}) = \frac{p(y_{1:T} \mid \theta)p(\theta)}{p(y_{1:T})} \propto p(y_{1:T} \mid \theta)p(\theta).$$

Chose the proposal distribution to be the same as the prior,

$$q(\theta) = \mathcal{N}\left(\theta \mid \mu_\theta, \sigma_\theta^2\right).$$

The importance weights are then computed according to

$$\widetilde{w}^i = \frac{\widetilde{\pi}(\theta^i)}{\widetilde{q}(\theta^i)} = p(y_{1:T} \mid \theta^i), \qquad i = 1, \dots, N,$$

i.e., the likelihood.

The Kalman filter straightforwardly allows us to evaluate the importance weights $\widetilde{w}^i = p(y_{1:T} \mid \theta^i)$,

$$p(y_{1:T} \mid \theta^i) = \prod_{t=1}^{T} p(y_t \mid y_{1:t-1}, \theta^i) = \prod_{t=1}^{T} \mathcal{N}\left(y_t \mid \widehat{y}_{t|t-1}(\theta^i), S_{t|t-1}(\theta^i)\right),$$

$$\begin{aligned} \widehat{y}_{t|t-1}(\theta^i) &= 0.5\widehat{x}_{t|t-1}(\theta^i), \\ S_{t|t-1}(\theta^i) &= 0.5^2 P_{t|t-1}(\theta^i) + 0.1, \end{aligned}$$

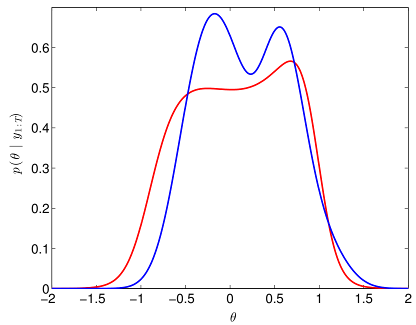
where $\widehat{x}_{t|t-1}(\theta^i)$ and $P_{t|t-1}(\theta^i)$ are provided by the Kalman filter.

Algorithm 3 Importance sampler targeting $p(\theta \mid y_{1:T})$

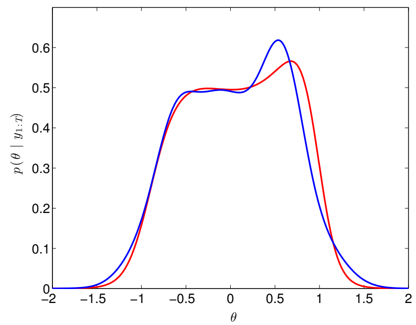
1. Generate N i.i.d. samples from the proposal distribution, i.e. $\theta^i \sim \mathcal{N}(\theta \mid \mu_\theta, \sigma_\theta^2)$ for $i = 1, \dots, N$.
2. Compute importance weights $\widetilde{w}^i = p(y_{1:T} \mid \theta^i)$ for $i = 1, \dots, N$.
3. Normalize the importance weights $w^i = \frac{\widetilde{w}^i}{\sum_{j=1}^N \widetilde{w}^j}$ for $i = 1, \dots, N$.

Importance sampling – (VII/VIII)

Using $T = 15$ measurements, $y_{1:15}$.

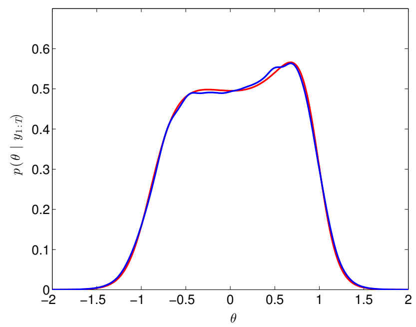


10 samples,
$q(\theta) = \mathcal{N}\left(\theta \mid 0, 1.5^2\right)$

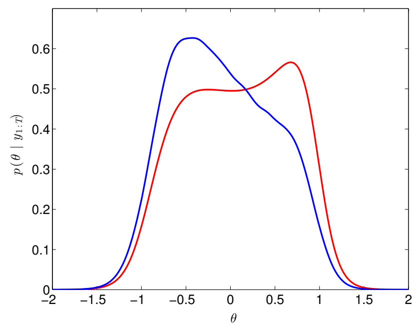


100 samples,
$q(\theta) = \mathcal{N}\left(\theta \mid 0, 1.5^2\right)$

Importance sampling – (VIII/VIII)



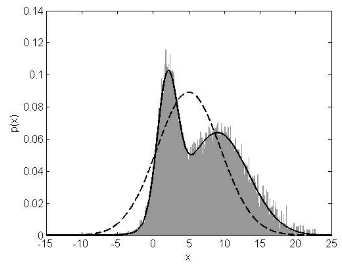
50000 samples,
$q(\theta) = \mathcal{N}\left(\theta \mid 0, 1.5^2\right)$



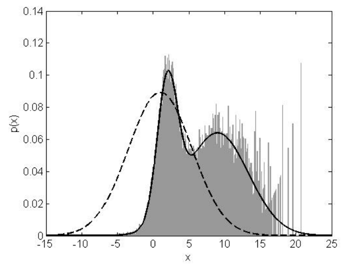
100000 samples,
$q(\theta) = \mathcal{N}\left(\theta \mid -0.5, 1^2\right)$

Note the different proposal distributions used above.

The importance of a good proposal density



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



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

50 000 samples used in booth experiments.

Lesson learned (again): It is very important to be careful in selecting the importance density.

Using IS for our purposes

A first attempt to solve the nonlinear filtering problem, which amounts to compute $p(x_t \mid y_{1:t})$ for

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

The solution is given by

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

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

IS algorithm again

Algorithm 4 Importance sampler (IS)

1. Generate N i.i.d. samples $\{z^i\}_{i=1}^N$ from the proposal distribution $q(z)$.
2. Compute the importance weights

$$\widetilde{w}^i = \frac{\widetilde{\pi}(z^i)}{\widetilde{q}(z^i)}, \qquad i = 1, \dots, N.$$

3. Normalize the importance weights

$$w^i = \frac{\widetilde{w}^i}{\sum_{j=1}^N \widetilde{w}^j}, \qquad i = 1, \dots, N.$$

Sequential sampling from the proposal

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

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

In practice this means:

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

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

A sequential importance sampler (SIS)

Algorithm 5 SIS targeting $p(x_{1:t} \mid y_{1:t})$

1. Generate N initial samples $x_1^i \sim \mu(x_1)$, set the importance weights, $\widetilde{w}_0^i = 1/N, i = 1, \dots, N$ and set $k = 0$.
2. **for** $k = 1$ **to** t **do**
   (a) Compute the importance weights $\widetilde{w}_k^i = p(y_k \mid x_k^i)\widetilde{w}_{k-1}^i$.
   (b) Normalize the importance weights $w_k^i = \frac{\widetilde{w}_k^i}{\sum_{j=1}^N \widetilde{w}_k^j}$ and store the new weights $\{w_{1:k}^i\}_{i=1}^N = \{w_{1:k-1}^i, w_k^i\}_{i=1}^N$.
   (c) Generate N i.i.d. samples from the proposal distribution, $x_{k+1}^i \sim f(x_{k+1} \mid x_k^i)$ and store the new samples $\{x_{1:k+1}^i\}_{i=1}^N = \{x_{1:k}^i, x_{k+1}^i\}_{i=1}^N$.

SIS example (I/III)

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 distribution

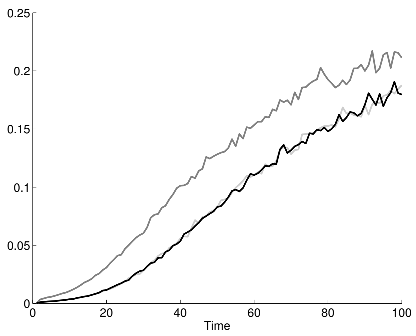
$$\widehat{p}(x_t \mid y_{1:t}) = \sum_{i=1}^{N} w_t^i \delta_{x_t^i}(x_t).$$

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, 1000 realisations of data and $N = 500$, $N = 5000$ and $N = 50000$, respectively.



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

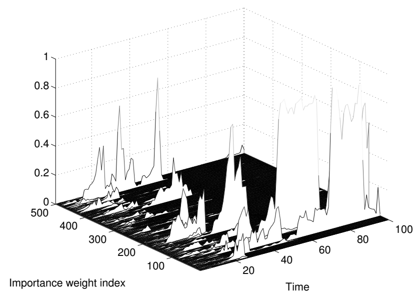
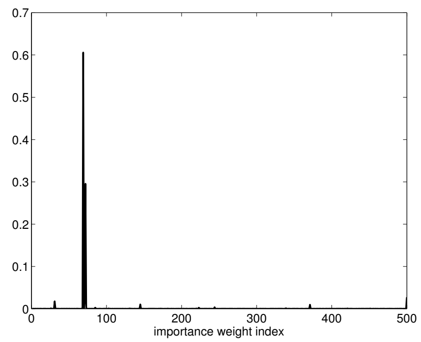
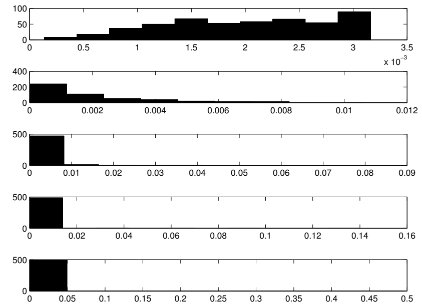


Illustration of the problem in the weights.

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?

Solving the weight degeneracy problem

Idea: Remove the weights from the representation!

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

Resampling (I/II)

The SIS representation of the target density is

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

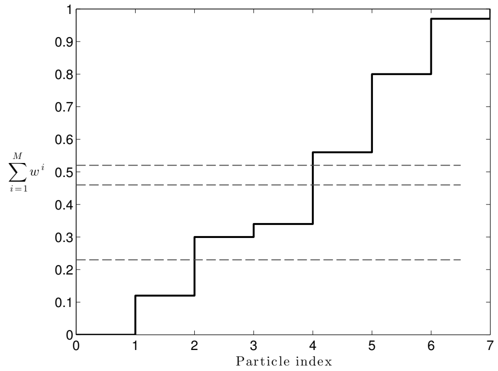
An unweighted representation of the target density can be created by **resampling with replacement**. This is done by generating a new sample z^i for each $i = 1, \dots, N$, where

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

The resulting unweighted representation is

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

Resampling (II/II)



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.

Sampling importance resampling (SIR)

Algorithm 6 Sampling Importance Resampler (SIR)

1. Generate N i.i.d. samples $\{\widetilde{z}^i\}_{i=1}^N$ from the proposal $q(z)$.
2. Compute the importance weights $\widetilde{w}^i = \frac{\widetilde{\pi}(\widetilde{z}^i)}{\widetilde{q}(\widetilde{z}^i)}, i = 1, \dots, N$.
3. Normalize the importance weights $w^i = \frac{\widetilde{w}^i}{\sum_{j=1}^N \widetilde{w}^j}, i = 1, \dots, N$.
4. For each $i = 1, \dots, N$ generate a new sample z^i, where

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

Note that step $1 - 3$ corresponds to the importance sampler.

Revisit the nonlinear filtering problem

Algorithm 7 A first particle filter targeting $p(x_{1:t} \mid y_{1:t})$

1. Generate N initial samples $\widetilde{x}_1^i \sim \mu(x_1)$, set the importance weights, $\widetilde{w}_0^i = 1/N, i = 1, \dots, N$ and set $k = 0$.
2. **for** $k = 1$ **to** t **do**
   - (a) Compute the importance weights $\widetilde{w}_k^i = p(y_k \mid \widetilde{x}_k^i)\widetilde{w}_{k-1}^i$.
   - (b) Normalize the importance weights $w_k^i = \frac{\widetilde{w}_k^i}{\sum_{j=1}^N \widetilde{w}_k^j}$
   - (c) For each $i = 1, \dots, N$ generate a new sample x_t^i, where $\mathbb{P}\left(x_t^i = \widetilde{x}_t^j\right) = w^j, \qquad j = 1, \dots, N.$
   - (d) Generate N i.i.d. samples from the proposal distribution, $\widetilde{x}_{k+1}^i \sim f(x_{k+1} \mid x_k^i)$.

PF example (I/V)

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 + resampling **(particle filter)** to compute an approximation of the filtering distribution

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

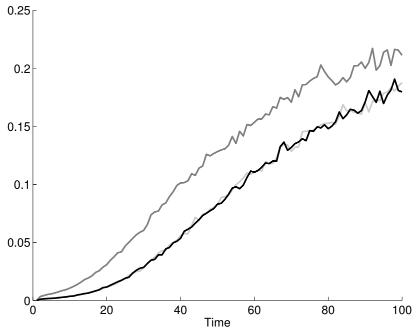
Study

- 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.

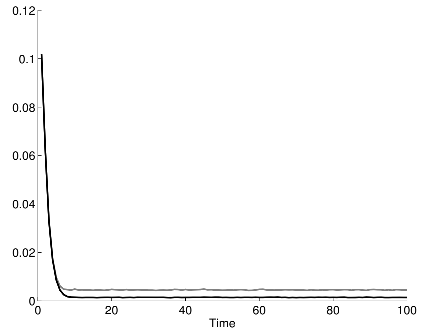
PF example (II/V)

Same setting as before, exactly the same data.

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

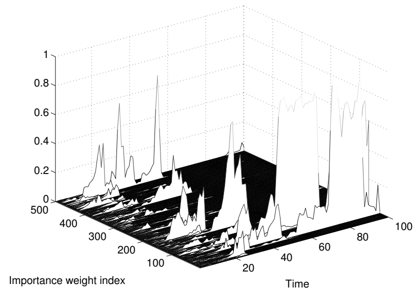


SIS

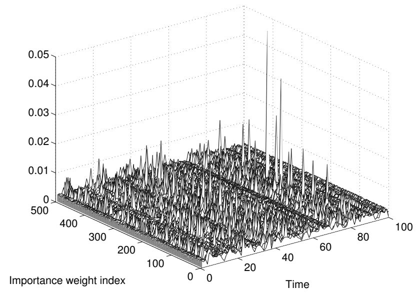


PF

PF example (III/V)



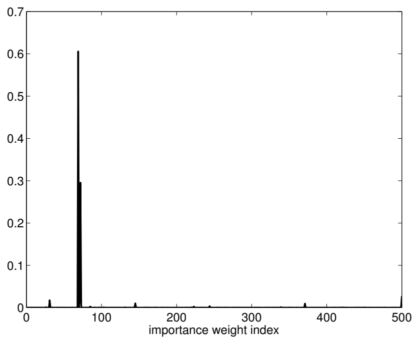
SIS



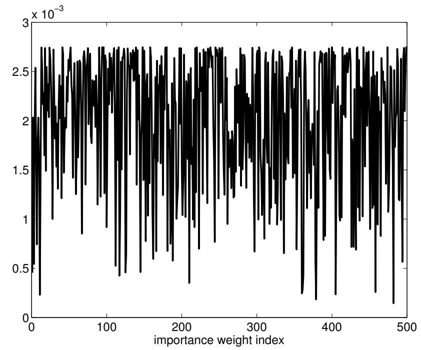
PF

Note the different scaling!

PF example (IV/V)



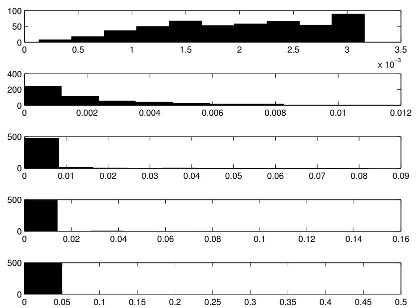
SIS



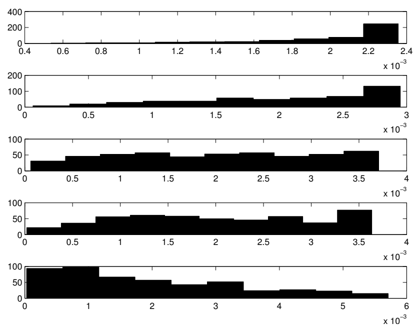
PF

Note the different scaling!

PF example (V/V)



SIS



PF

An LGSS example (I/II)

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

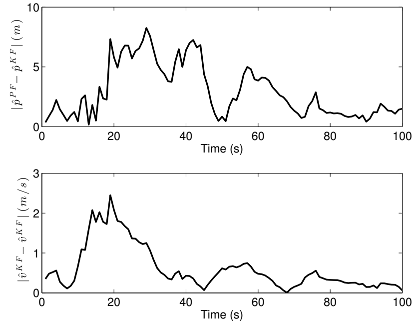
Consider the following LGSS model (simple one dimensional positioning example)

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

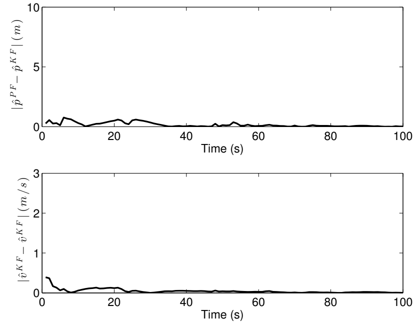
$$y_t = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_t \\ v_t \\ a_t \end{pmatrix} + e_t, \qquad e_t \sim \mathcal{N}(0, R).$$

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

An LGSS example (II/II)



Using 200 particles.



Using 20000 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]

A nonlinear example (I/II)

Consider the following SSM (standard example in PF literature)

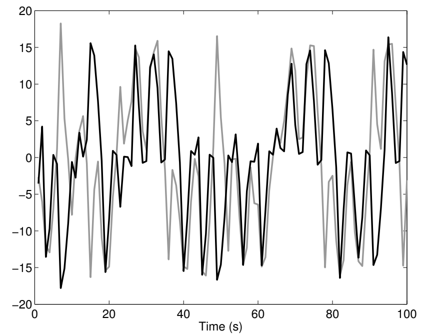
$$x_{t+1} = \frac{x_t}{2} + \frac{25 x_t}{1 + x_t^2} + 8\cos(1.2t) + v_t, \qquad v_t \sim \mathcal{N}(0, 0.5),$$

$$y_t = \frac{x_t^2}{20} + e_t, \qquad e_t \sim \mathcal{N}(0, 0.5).$$

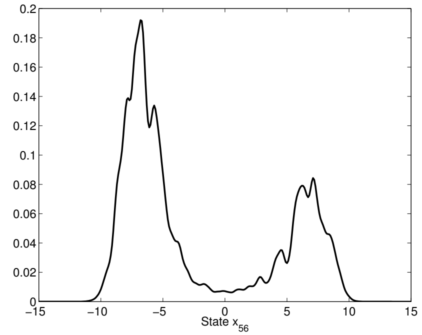
What it tricky with this model?

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

A nonlinear example (II/II)



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

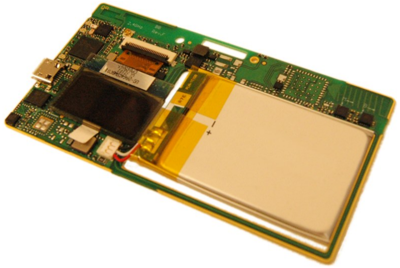


PF estimate of the filtering pdf $\widehat{p}(x_{56} \mid y_{1:56})$.

Nonlinear example – indoor positioning (I/III)

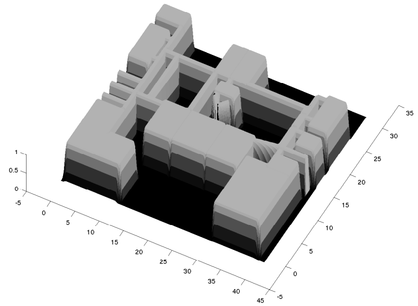
Aim: Compute the position of a person moving around indoors using sensors located in an ID badge and a map.

The sensors (IMU and radio) and the DSP are mounted inside an ID badge.

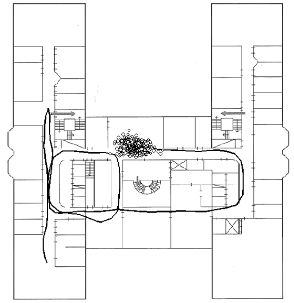


The inside of the ID badge.

Nonlinear example – indoor positioning (I/III)



pdf for an office environment, the bright areas are rooms and corridors (i.e., walkable space).



An estimated trajectory and the particle cloud visualized at a particular instance.

Nonlinear example – indoor positioning (III/III)

Show movies

This is work by Johan Kihlberg and Simon Tegelid in their MSc thesis entitled **Map Aided Indoor Positioning**, which is available for download. [pdf]

Sequential Monte Carlo (SMC) – more general

Assume that at time t we have a particle system $\{x_{1:t}^i, w_t^i\}_{i=1}^N$ approximating the target distribution $p(x_{1:t} \mid y_{1:t})$ according to

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

The three steps in the SMC algorithm are

1. **Resampling**: $\{x_{1:t}, w_t^i\}_{i=1}^N \to \{\widetilde{x}_{1:t}, 1/N\}_{i=1}^N$
2. **Propose new samples**: $x_{t+1}^i \sim Q_t(x_{t+1} \mid \widetilde{x}_{1:t}^i)$ for $i = 1, \dots, N$.
3. **Weighting**: $w_{t+1}^i = W_t(x_{t+1}^i, \widetilde{x}_{1:t}^i)$ for $i = 1, \dots, N$.

SMC is a combination of SIS and resampling.

Algorithm 8 Sequential Monte Carlo (SMC)

1. Initialise by sampling $x_1^i \sim Q_1(x_1)$, setting $w_1^i = W_1(x_1^i)$ and setting $t = 1$.
2. **for** $t = 1$ **to** T **do**
   (a) Resample: $\{x_{1:t}^i, w_t^i\}_{i=1}^N \to \{\widetilde{x}_{1:t}^i, 1/N\}_{i=1}^N$.
   (b) Propose new samples: $x_{t+1}^i \sim Q_t(x_{t+1} \mid \widetilde{x}_{1:t}^i)$.
   (c) Weighting: $w_{t+1}^i = W_t(x_{t+1}^i, \widetilde{x}_{1:t}^i)$.

There is a myriad of possible design choices available, some of them carrying their own name. The important thing is to understand the basic principles in action, then the rest is just design choices.

Notation: $\mathbf{w}_t \triangleq \{w_t^1, \dots, w_t^N\}$.

Algorithm 9 Sequential Monte Carlo (SMC)

1. **Initialise:** Sample $x_1^i \sim Q_1(x_1)$ and set $w_1^i = W_1(x_1^i)$. Set $t = 1$.
2. **For** $t = 2 : T$ **do:**
   (a) Resampling: $a_t^i \sim R(a_t \mid \mathbf{w}_{t-1})$.
   (b) Sample from the proposal kernel: $x_t^i \sim Q_t(x_t \mid x_{1:t-1}^{a_t^i})$ and set $x_{1:t}^i = \{x_{1:t-1}^{a_t^i}, x_t^i\}$.
   (c) Weighting: $w_t^i = W_t(x_t^i, \widetilde{x}_{1:t-1}^{a_t^i})$.

Here: a_t^i to denote the **index** of the parent/ancestor at time $t-1$ of particle x_t^i.

Just like any algorithm that is used to generate random numbers there is an **underlying distribution** also for the SMC sampler that encodes the probabilistic properties of the involved stochastic variables.

The SMC sampler generates a realisation of the random variables $\mathbf{X_{1:T}} = \{X_{1:T}^1, \dots, X_{1:T}^N\}$ and $\mathbf{A_{2:T}} = \{A_{2:T}^1, \dots, A_{2:T}^N\}$, where the pdf is

$$\psi(\mathbf{x}_{1:T}, \mathbf{a}_{2:T}) = \underbrace{\left\{\prod_{i=1}^N Q_1(x_1^i)\right\}}_{\text{initialisation}} \left\{\prod_{t=2}^T \prod_{i=1}^N \underbrace{R(a_t^i \mid \mathbf{w}_{t-1})}_{\text{Resampling}} \underbrace{Q_t\left(x_t^i \mid x_{1:t-1}^{a_t^i}\right)}_{\text{Proposing new particles}}\right\}$$

and it is defined on the space $\mathcal{X}^{TN} \times \{1, \dots, N\}^{(T-1)N}$.

The SMC algorithm produce the following approximation of the target distribution

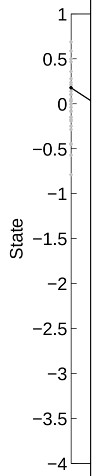
$$\widehat{\pi}^N(x_{1:T}) = \sum_{i=1}^N W_T^i \delta_{X_{1:T}^i}(x_{1:T}).$$

We can use $\widehat{\pi}^N(x_{1:T})$ to produce samples of the state trajectory by sampling from

$$q^N(x_{1:T}) = \mathrm{E}_\psi\left[\widehat{\pi}^N(x_{1:T})\right]$$
$$= \int \sum_{i=1}^N W_T^i \delta_{X_{1:T}^i}(x_{1:T}) \psi(\mathbf{X}_{1:T}, \mathbf{A}_{1:T-1}) \mathrm{d}\mathbf{X}_{1:T} \mathrm{d}\mathbf{A}_{1:T-1}.$$

Note that this integration can never be carried out explicitly. Hence, we cannot evaluate this distribution for a specific sample, but it can be used to generate samples from it.

Illustration of the particle degeneracy problem



Implication of the particle degeneracy problem

This implies that if we are interested in the smoothing distribution

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

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

Particle smoother

Algorithm 10 Forward filtering/Backward smoothing (FFBSm)

1. Run the PF and store the particles and their weights $\{(w_t^i, x_t^i)\}_{i=1}^N$ for $t = 1, \dots, T$.
2. Initialise the smoothed weights $w_{T|T}^i = w_T^i$, for $i = 1, \dots, N$.
3. **for** $t = T-1$ **to** 1 **do**
   (a) Compute the smoothed weights

$$w_{t|T}^i = w_t^i \sum_{k=1}^{N} w_{t+1|T}^i \frac{f(x_{t+1}^k \mid x_t^i)}{v_t^k}, \qquad v_t^k = \sum_{i=1}^{N} w_t^i f(x_{t+1}^k \mid x_t^i).$$

The resulting approximation $\widehat{p}(x_t \mid y_{1:T}) = \sum_{i=1}^N w_{t|T}^i \delta_{x_t^i}(x_t)$ does not suffer from degeneracy.

Sampling from the backward kernel

Key idea: Generate a backward trajectory by sampling from the following approximation

$$\widehat{p}(x_t \mid \widetilde{x}_{t+1}^j, y_{1:t}) = \sum_{i=1}^{N} \underbrace{\frac{w_t^i f(\widetilde{x}_{t+1}^j \mid x_t^i)}{\sum_{k=1}^N w_t^k f(\widetilde{x}_{t+1}^j \mid x_t^k)}}_{\triangleq \widetilde{w}_{t|T}^{i,j}} \delta_{x_t^i}(x_t)$$

of the backward kernel. The resulting sample trajectory is a sample from the joint smoothing density $p(x_{1:T} \mid y_{1:T})$.

$$\widetilde{x}_t^j \sim \widehat{p}(x_t \mid \widetilde{x}_{t+1}^j, y_{1:t}),$$
$$\widetilde{x}_{t:T}^j = \{\widetilde{x}_t^j, \widetilde{x}_{t+1:T}^j\}.$$

Still computationally very costly!

Sampling from the backward kernel

Algorithm 11 Backward simulator (BSi)

1. **Initialise:** Set $b_T = m$ with probability $w_T^m / \sum_{l=1}^N w_T^l$.
2. **For** $t = T-1 : -1 : 1$ **do:**
   (a) Given $x_{t+1}^{b_{t+1}}$, compute the smoothing weights,

$$w_{t|T}^m = \frac{w_t^m f(x_{t+1}^{b_{t+1}} \mid x_t^m)}{\sum_{l=1}^N w_t^l f(x_{t+1}^{b_{t+1}} \mid x_t^l)}, \qquad m = 1, \dots, N.$$

   (b) Set $b_t = m$ with probability $w_{t|T}^m$.

Sampling from the backward kernel - in words

The BSi samples a trajectory from the empirical joint smoothing density by first choosing a particle x_T^m with probability proportional to w_T^m and then generating the **ancestral path** $b_{1:T}$ according to the backward kernel.

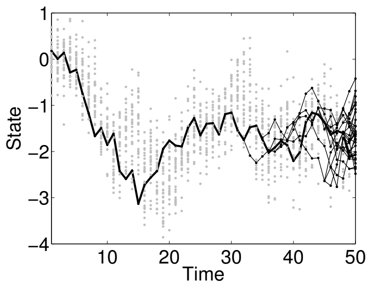
The resulting trajectory is

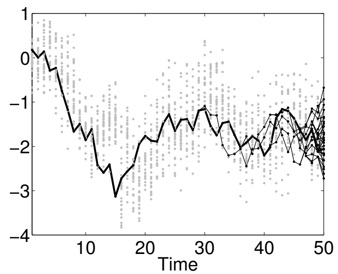
$$x_{1:T}^\star = \{x_1^{b_1}, \dots, x_T^{b_T}\}.$$

The computational cost is $\mathcal{O}(N^2)$, which is too expensive to be practical.

Sampling from the backward kernel - in pictures

Recall that the **main problem** with using the PF to sample from $p(x_{1:T} \mid y_{1:T})$ is the particle degeneracy problem.

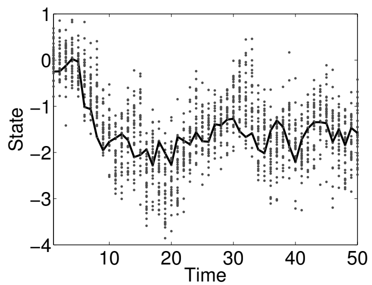


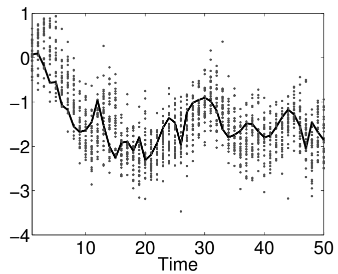


In the figures we show two samples from $p(x_{1:T} \mid y_{1:T})$ generated using the PF. The diversity is completely gone from time ≈ 30 and backwards.

Sampling from the backward kernel - in pictures

The backward simulator **resolves the particle degeneracy problem**.





Note that in the above samples there is no loss of diversity.

Forward filtering/Backward simulation (FFBSi)

Algorithm 12 Forward filtering/Backward simulation (FFBSi)

1. Run the PF and store $\{(w_t^m, x_t^m)\}_{m=1}^N$ for $t = 1, \dots, T$.
2. Initialise: Set the smoothed weights $w_{T|T}^m = w_T^m$ and set $t = T - 1$.
3. **For** $t = T - 1 : -1 : 1$ **do:**
4. **For** $j = 1 : M$ **do:**
   (a) Set $v_t^j = \sum_{k=1}^N w_t^k f(\tilde{x}_{t+1}^j \mid x_t^k)$
   (b) For $m = 1, \dots, N$, compute $\tilde{w}_{t|T}^{m,j} = w_t^m \frac{f(\tilde{x}_{t+1}^j | x_t^m)}{v_t^k}$.
   (c) Sample from the empirical backward kernel,
   $I(j) \sim \text{Cat}\left(\{\tilde{w}_{t|T}^{m,j}\}_{m=1}^N\right), \qquad \tilde{x}_t^j = x_t^{I(j)},$
   and append the sample $\tilde{x}_{t:T}^j = \{\tilde{x}_t^j, \tilde{x}_{t+1:T}^j\}$

Towards a practical particle smoother

To obtain a practical algorithm we have to reduce the computational complexity, how?!

Key insight: We do not have to compute all the smoothing weights $\{\tilde{w}_{t|T}^{m,j}\}$ to be able to sample from the empirical backward kernel.

How: Use rejection sampling to **sample the indices**! This means that we should not have to compute N indices just to draw one sample.

For details and MATLAB code, see Algorithm 5.4 (Fast FFBSi) in

F. Lindsten, **Rao-Blackwellised particle methods for inference and identification**. Licentiate thesis, Linköping University, LiU-TEK-LIC-2011:19, 2011. [pdf]

Original reference:

R. Douc, A. Garivier, E. Moulines, and J. Olsson. **Sequential Monte Carlo smoothing for general state space hidden Markov models**. *Annals of Applied Probability*, 21(6):2109–2145, 2011. [pdf]