An introduction to particle filters

Andreas Svensson

Department of Information Technology
Uppsala University

June 10, 2014

Outline

- Motivation and ideas
- Algorithm
  - High-level
  - Matlab code
- Practical aspects
  - Resampling
  - Computational complexity
  - Software
  - Terminology
- Advanced topics
  - Convergence
  - Extensions
- References

State Space Models

State Space Models

- Linear Gaussian state space model

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

with w_t and e_t Gaussian and $\mathbb{E}\left[w_t w_t^T\right] = Q, \ \mathbb{E}\left[e_t e_t^T\right] = R.$

State Space Models

- Linear Gaussian state space model

$$x_{t+1} = Ax_t + Bu_t + w_t$$
$$y_t = Cx_t + Du_t + e_t$$

with w_t and e_t Gaussian and $\mathbb{E}\left[w_t w_t^T\right] = Q, \; \mathbb{E}\left[e_t e_t^T\right] = R.$

- A more general state space model in different notation

$$x_{t+1} \sim f(x_{t+1} | x_t, u_t)$$
$$y_t \sim g(y_t | x_t, u_t)$$

Filtering - Find $p(x_t|y_{1:t})$

Filtering - Find $p(x_t|y_{1:t})$

Linear Gaussian problems:

Filtering - Find $p(x_t | y_{1:t})$

Linear Gaussian problems:

- Kalman filter!

Filtering - Find $p(x_t | y_{1:t})$

Linear Gaussian problems:

- Kalman filter!

Nonlinear problems:

Filtering - Find $p(x_t | y_{1:t})$

Linear Gaussian problems:

- Kalman filter!

Nonlinear problems:

- EKF, UKF, ...
- Particle filter

Filtering - Find $p(x_t | y_{1:t})$

Linear Gaussian problems:

- Kalman filter! *"Exact solution to the right problem"*

Nonlinear problems:

- EKF, UKF, ...
- Particle filter

Filtering - Find $p(x_t|y_{1:t})$

Linear Gaussian problems:

- Kalman filter! *"Exact solution to the right problem"*

Nonlinear problems:

- EKF, UKF, … *"Exact solution to the wrong problem"*
- Particle filter

Filtering - Find $p(x_t|y_{1:t})$

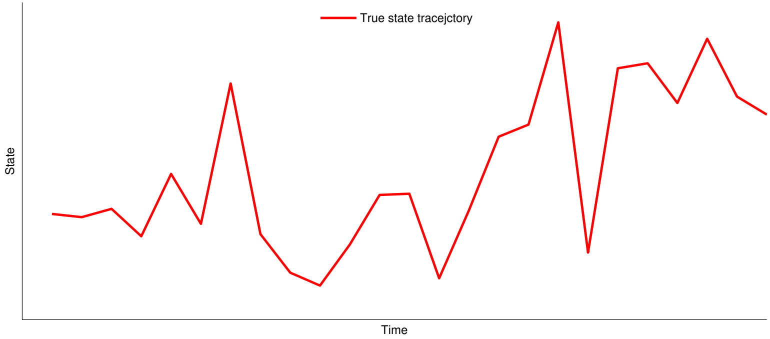
Linear Gaussian problems:

- Kalman filter! *"Exact solution to the right problem"*

Nonlinear problems:

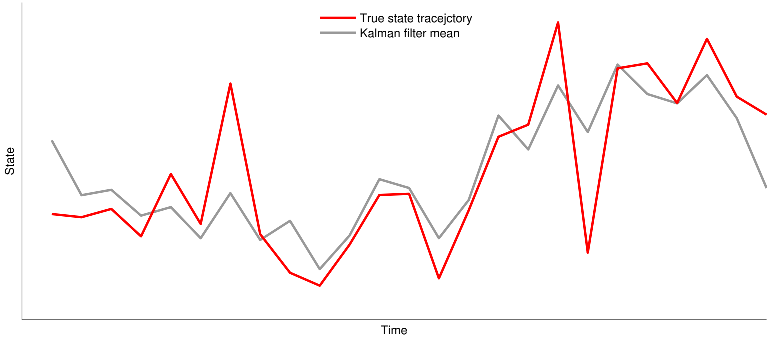
- EKF, UKF, ... *"Exact solution to the wrong problem"*
- Particle filter *"Approximate solution to the right problem"*

Idea



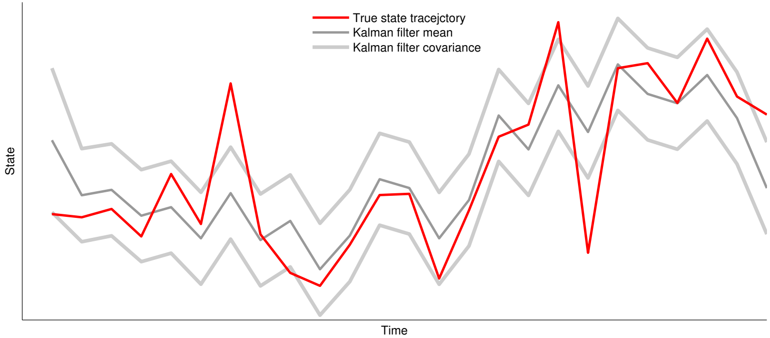
A linear system - Find $p(x_t|y_{1:t})$ (true x_t shown)!

Idea



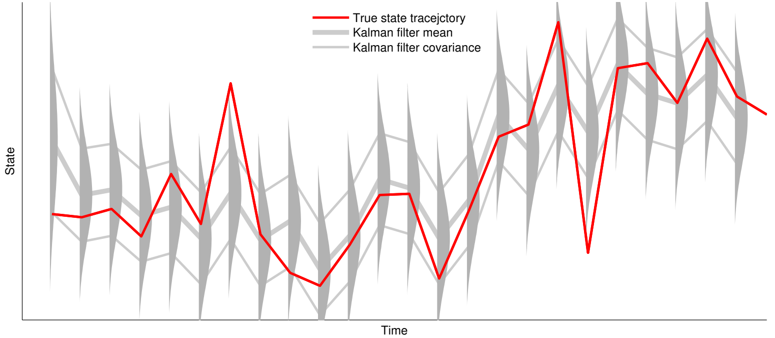
Kalman filter mean

Idea



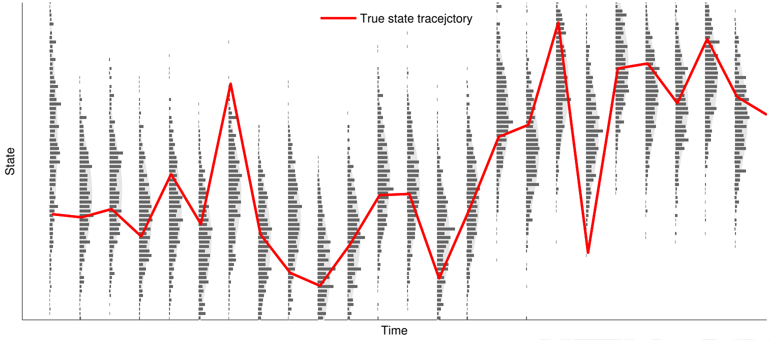
Kalman filter mean and covariance

Idea



Kalman filter mean and covariance defines a Gaussian distribution at each t

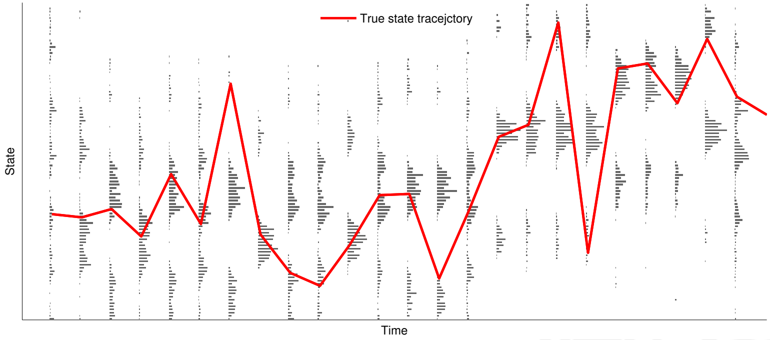
Idea



A numerical approximation can be used to describe the distribution - Particle Filter

UPPSALA UNIVERSITET

Idea



The Particle Filter can easily handle, e.g., non-gaussian multimodal hypotheses

Idea

Generate a lot of hypotheses about x_t, keep the most likely ones and propagate them further to x_{t+1}. Keep the likely hypotheses about x_{t+1}, propagate them again to x_{t+2}, etc.

Algorithm

0 Initialize $x_1^i \sim p(x_1)$ and $w_i = \frac{1}{N}$ for $i = 1, \dots, N$

for t = 1 to T

1. Evaluate $w_t^i = g(y_t | x_t^i, u_t)$ for $i = 1, \dots, N$
2. Resample $\{x_t^i\}_{i=1}^N$ from $\{x_t^i, w_t^i\}_{i=1}^N$
3. Propagate x_t^i by sampling x_{t+1}^i from $f(\cdot | x_t^i, u_t)$ for $i = 1, \dots, N$

end

N Number of particles,

x_t^i Particles,

w_t^i Particle weights

Algorithm

0 Initialize $x_1^i \sim p(x_1)$ and $w_i = \frac{1}{N}$ for $i = 1, \dots, N$

for t = 1 to T

1 Evaluate $w_t^i = g(y_t | x_t^i, u_t)$ for $i = 1, \dots, N$

2 Resample $\{x_t^i\}_{i=1}^N$ from $\{x_t^i, w_t^i\}_{i=1}^N$

3 Propagate x_t^i by sampling x_{t+1}^i from $f(\cdot | x_t^i, u_t)$ for $i = 1, \dots, N$

end

N Number of particles,
x_t^i Particles,
w_t^i Particle weights

```
0 x(:,1) = random(i_dist,N,1);
  w(:,1) = ones(N,1)/N;

  for t = 1:T

    1 w(:,t) =
      pdf(m_dist,y(t)-g(x(:,t)));
      w(:,t) =
      w(:,t)/sum(w(:,t));
    2 Resample x(:,t)
    3 x(:,t+1) = f(x(:,t),u(t))
      + random(t_dist,N,1);

  end
```

Algorithm

→0 Initialize $x_1^i \sim p(x_1)$ and $w_i = \frac{1}{N}$ for $i = 1, \ldots, N$

for t = 1 to T

1 Evaluate $w_t^i = g(y_t | x_t^i, u_t)$ for $i = 1, \ldots, N$

2 Resample $\{x_t^i\}_{i=1}^N$ from $\{x_t^i, w_t^i\}_{i=1}^N$

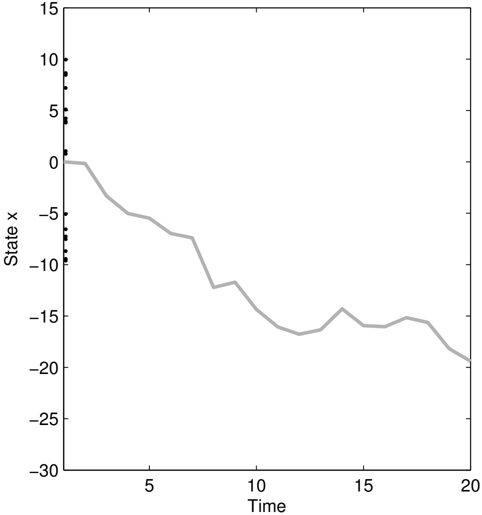
3 Propagate x_t^i by sampling from $f(\cdot | x_{t-1}^i, u_{t-1})$ for $i = 1, \ldots, N$

end

N Number of particles,

x_t^i Particles,

w_t^i Particle weights



Algorithm

0 Initialize $x_1^i \sim p(x_1)$ and $w_i = \frac{1}{N}$ for $i = 1, \ldots, N$

for t = 1 to T

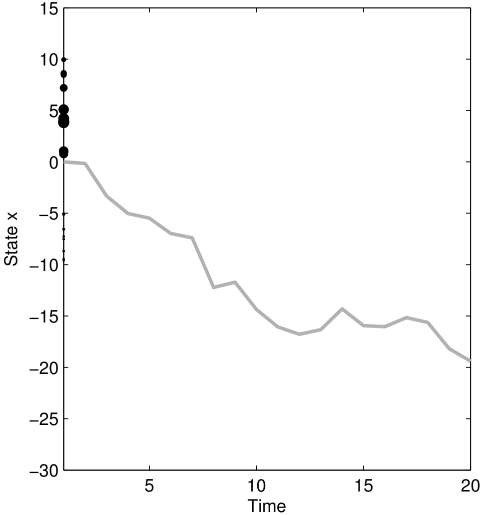
→ **1** Evaluate $w_t^i = g(y_t | x_t^i, u_t)$ for $i = 1, \ldots, N$

2 Resample $\{x_t^i\}_{i=1}^N$ from $\{x_t^i, w_t^i\}_{i=1}^N$

3 Propagate x_t^i by sampling from $f(\cdot | x_{t-1}^i, u_{t-1})$ for $i = 1, \ldots, N$

end

N Number of particles,
x_t^i Particles,
w_t^i Particle weights



Algorithm

0 Initialize $x_1^i \sim p(x_1)$ and $w_i = \frac{1}{N}$ for $i = 1, \ldots, N$

for t = 1 to T

1 Evaluate $w_t^i = g(y_t | x_t^i, u_t)$ for $i = 1, \ldots, N$

$\rightarrow$2 Resample $\{x_t^i\}_{i=1}^N$ from $\{x_t^i, w_t^i\}_{i=1}^N$

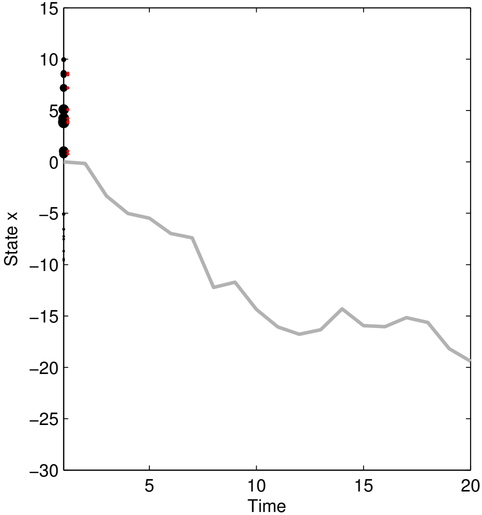
3 Propagate x_t^i by sampling from $f(\cdot | x_{t-1}^i, u_{t-1})$ for $i = 1, \ldots, N$

end

N Number of particles,

x_t^i Particles,

w_t^i Particle weights



Algorithm

0 Initialize $x_1^i \sim p(x_1)$ and $w_i = \frac{1}{N}$ for $i = 1, \ldots, N$

for t = 1 to T

1 Evaluate $w_t^i = g(y_t | x_t^i, u_t)$ for $i = 1, \ldots, N$

2 Resample $\{x_t^i\}_{i=1}^N$ from $\{x_t^i, w_t^i\}_{i=1}^N$

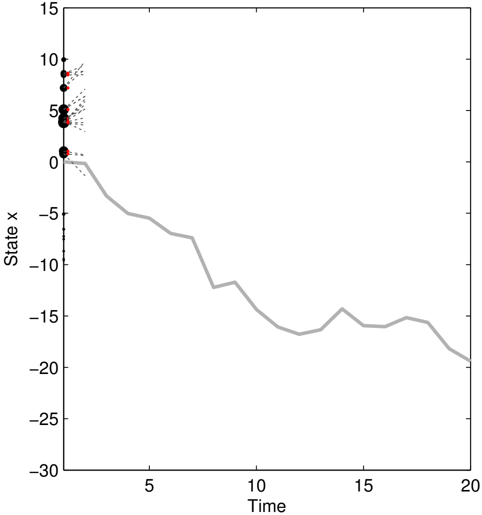
$\rightarrow$3 Propagate x_t^i by sampling from $f(\cdot | x_{t-1}^i, u_{t-1})$ for $i = 1, \ldots, N$

end

N Number of particles,

x_t^i Particles,

w_t^i Particle weights



UPPSALA UNIVERSITET

Algorithm

0 Initialize $x_1^i \sim p(x_1)$ and $w_i = \frac{1}{N}$ for $i = 1, \ldots, N$

for t = 1 to T

→**1** Evaluate $w_t^i = g(y_t | x_t^i, u_t)$ for $i = 1, \ldots, N$

2 Resample $\{x_t^i\}_{i=1}^N$ from $\{x_t^i, w_t^i\}_{i=1}^N$

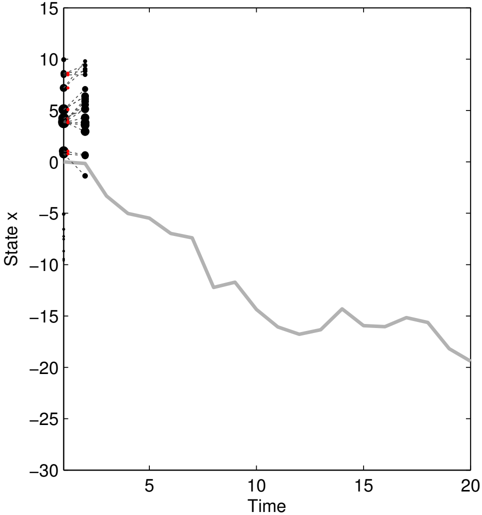
3 Propagate x_t^i by sampling from $f(\cdot | x_{t-1}^i, u_{t-1})$ for $i = 1, \ldots, N$

end

N Number of particles,

x_t^i Particles,

w_t^i Particle weights



UPPSALA UNIVERSITET

Algorithm

0 Initialize $x_1^i \sim p(x_1)$ and $w_i = \frac{1}{N}$ for $i = 1, \ldots, N$

for t = 1 to T

1 Evaluate $w_t^i = g(y_t | x_t^i, u_t)$ for $i = 1, \ldots, N$

$\rightarrow$**2** Resample $\{x_t^i\}_{i=1}^N$ from $\{x_t^i, w_t^i\}_{i=1}^N$

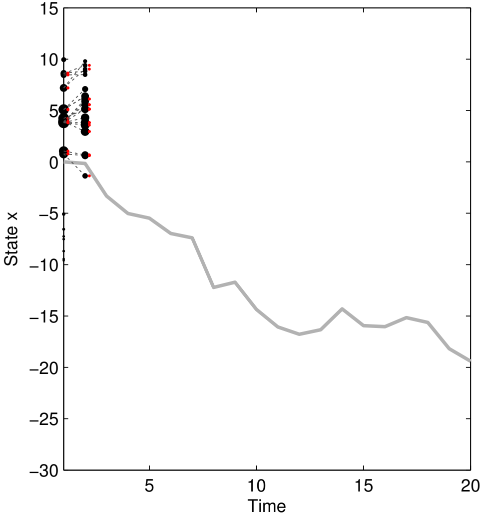
3 Propagate x_t^i by sampling from $f(\cdot | x_{t-1}^i, u_{t-1})$ for $i = 1, \ldots, N$

end

N Number of particles,

x_t^i Particles,

w_t^i Particle weights



Algorithm

0 Initialize $x_1^i \sim p(x_1)$ and $w_i = \frac{1}{N}$ for $i = 1, \ldots, N$

for t = 1 to T

1 Evaluate $w_t^i = g(y_t | x_t^i, u_t)$ for $i = 1, \ldots, N$

2 Resample $\{x_t^i\}_{i=1}^N$ from $\{x_t^i, w_t^i\}_{i=1}^N$

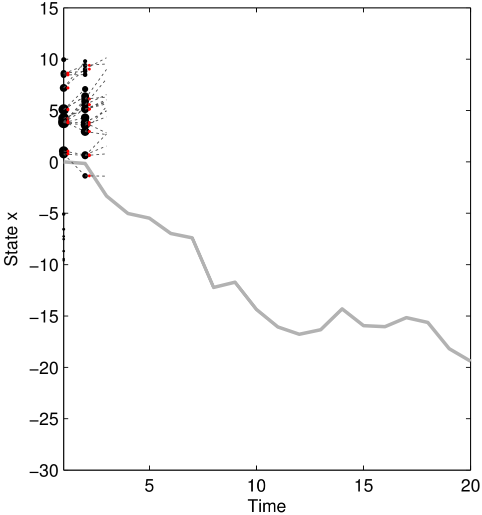
$\rightarrow$3 Propagate x_t^i by sampling from $f(\cdot | x_{t-1}^i, u_{t-1})$ for $i = 1, \ldots, N$

end

N Number of particles,

x_t^i Particles,

w_t^i Particle weights



Algorithm

0 Initialize $x_1^i \sim p(x_1)$ and $w_i = \frac{1}{N}$ for $i = 1, \ldots, N$

for t = 1 to T

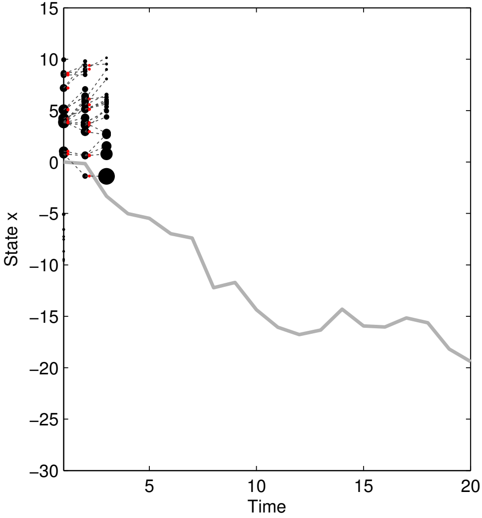
$\rightarrow$ **1** Evaluate $w_t^i = g(y_t | x_t^i, u_t)$ for $i = 1, \ldots, N$

2 Resample $\{x_t^i\}_{i=1}^N$ from $\{x_t^i, w_t^i\}_{i=1}^N$

3 Propagate x_t^i by sampling from $f(\cdot | x_{t-1}^i, u_{t-1})$ for $i = 1, \ldots, N$

end

N Number of particles,
x_t^i Particles,
w_t^i Particle weights



Algorithm

0 Initialize $x_1^i \sim p(x_1)$ and $w_i = \frac{1}{N}$ for $i = 1, \dots, N$

for t = 1 to T

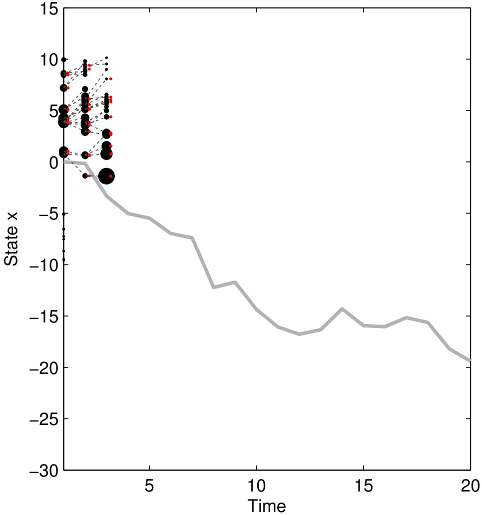
1 Evaluate $w_t^i = g(y_t | x_t^i, u_t)$ for $i = 1, \dots, N$

$\rightarrow$2 Resample $\{x_t^i\}_{i=1}^N$ from $\{x_t^i, w_t^i\}_{i=1}^N$

3 Propagate x_t^i by sampling from $f(\cdot | x_{t-1}^i, u_{t-1})$ for $i = 1, \dots, N$

end

N Number of particles,
x_t^i Particles,
w_t^i Particle weights



Algorithm

0 Initialize $x_1^i \sim p(x_1)$ and $w_i = \frac{1}{N}$ for $i = 1, \dots, N$

for t = 1 to T

1 Evaluate $w_t^i = g(y_t | x_t^i, u_t)$ for $i = 1, \dots, N$

2 Resample $\{x_t^i\}_{i=1}^N$ from $\{x_t^i, w_t^i\}_{i=1}^N$

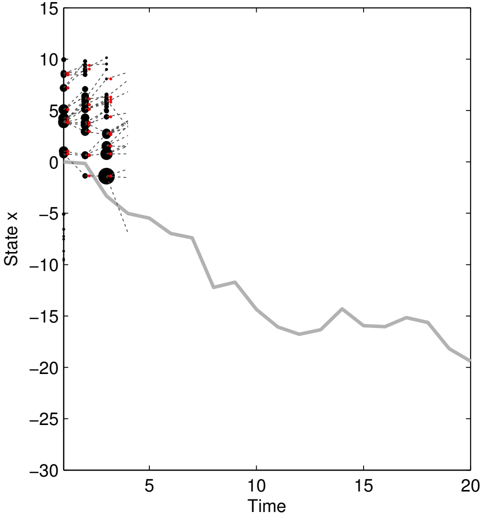
$\rightarrow$3 Propagate x_t^i by sampling from $f(\cdot | x_{t-1}^i, u_{t-1})$ for $i = 1, \dots, N$

end

N Number of particles,

x_t^i Particles,

w_t^i Particle weights



Algorithm

0 Initialize $x_1^i \sim p(x_1)$ and $w_i = \frac{1}{N}$ for $i = 1, \dots, N$

for t = 1 to T

$\rightarrow$1 Evaluate $w_t^i = g(y_t | x_t^i, u_t)$ for $i = 1, \dots, N$

2 Resample $\{x_t^i\}_{i=1}^N$ from $\{x_t^i, w_t^i\}_{i=1}^N$

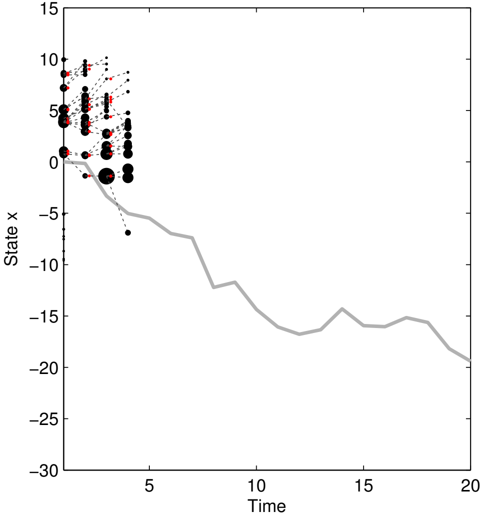
3 Propagate x_t^i by sampling from $f(\cdot | x_{t-1}^i, u_{t-1})$ for $i = 1, \dots, N$

end

N Number of particles,

x_t^i Particles,

w_t^i Particle weights



UPPSALA UNIVERSITET

Algorithm

0 Initialize $x_1^i \sim p(x_1)$ and $w_i = \frac{1}{N}$ for $i = 1, \dots, N$

for t = 1 to T

1 Evaluate $w_t^i = g(y_t | x_t^i, u_t)$ for $i = 1, \dots, N$

$\rightarrow$2 Resample $\{x_t^i\}_{i=1}^N$ from $\{x_t^i, w_t^i\}_{i=1}^N$

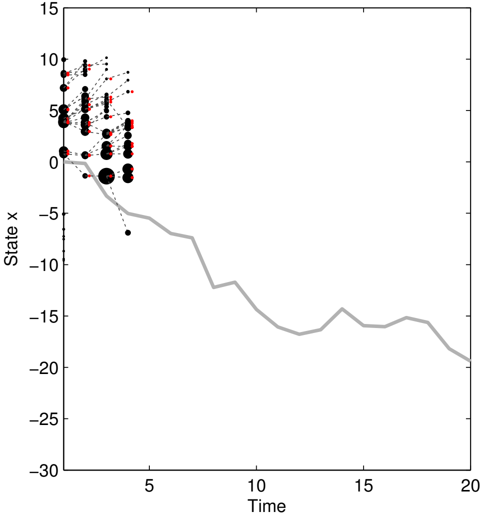
3 Propagate x_t^i by sampling from $f(\cdot | x_{t-1}^i, u_{t-1})$ for $i = 1, \dots, N$

end

N Number of particles,

x_t^i Particles,

w_t^i Particle weights



Algorithm

0 Initialize $x_1^i \sim p(x_1)$ and $w_i = \frac{1}{N}$ for $i = 1, \dots, N$

for t = 1 to T

1 Evaluate $w_t^i = g(y_t | x_t^i, u_t)$ for $i = 1, \dots, N$

2 Resample $\{x_t^i\}_{i=1}^N$ from $\{x_t^i, w_t^i\}_{i=1}^N$

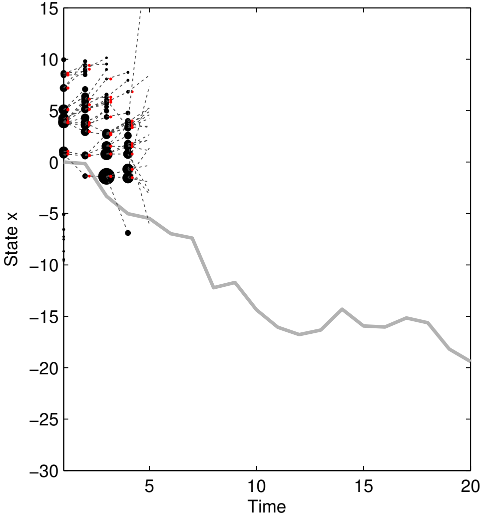
→3 Propagate x_t^i by sampling from $f(\cdot | x_{t-1}^i, u_{t-1})$ for $i = 1, \dots, N$

end

N Number of particles,

x_t^i Particles,

w_t^i Particle weights



Algorithm

0 Initialize $x_1^i \sim p(x_1)$ and $w_i = \frac{1}{N}$ for $i = 1, \ldots, N$

for t = 1 to T

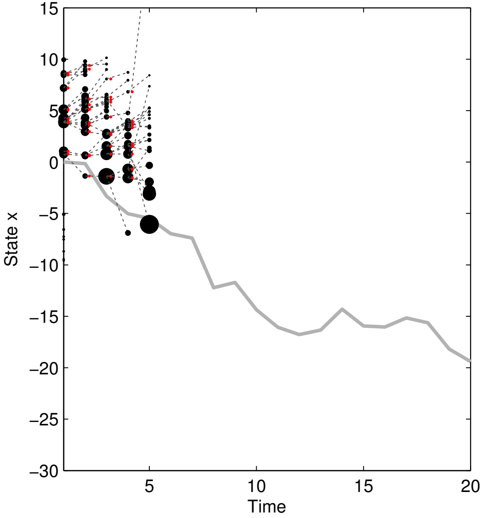
→ **1** Evaluate $w_t^i = g(y_t | x_t^i, u_t)$ for $i = 1, \ldots, N$

2 Resample $\{x_t^i\}_{i=1}^N$ from $\{x_t^i, w_t^i\}_{i=1}^N$

3 Propagate x_t^i by sampling from $f(\cdot | x_{t-1}^i, u_{t-1})$ for $i = 1, \ldots, N$

end

N Number of particles,
x_t^i Particles,
w_t^i Particle weights



UPPSALA UNIVERSITET

Algorithm

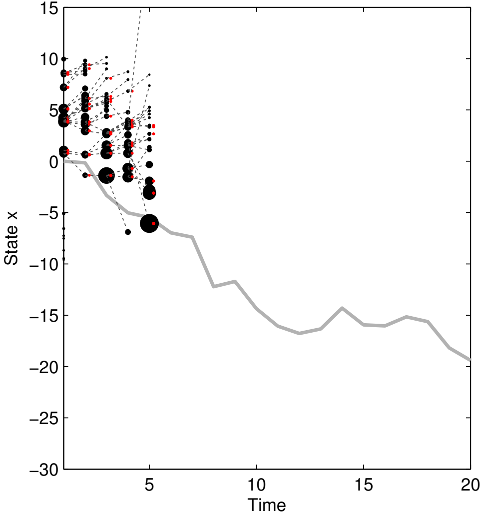
0 Initialize $x_1^i \sim p(x_1)$ and $w_i = \frac{1}{N}$ for $i = 1, \ldots, N$

for t = 1 to T

1. Evaluate $w_t^i = g(y_t | x_t^i, u_t)$ for $i = 1, \ldots, N$
2. → Resample $\{x_t^i\}_{i=1}^N$ from $\{x_t^i, w_t^i\}_{i=1}^N$
3. Propagate x_t^i by sampling from $f(\cdot | x_{t-1}^i, u_{t-1})$ for $i = 1, \ldots, N$

end

N Number of particles,
x_t^i Particles,
w_t^i Particle weights



Algorithm

0 Initialize $x_1^i \sim p(x_1)$ and $w_i = \frac{1}{N}$ for $i = 1, \dots, N$

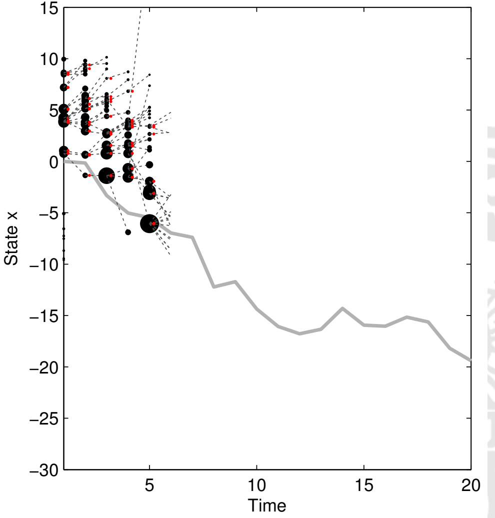
for t = 1 to T

1. Evaluate $w_t^i = g(y_t | x_t^i, u_t)$ for $i = 1, \dots, N$
2. Resample $\{x_t^i\}_{i=1}^N$ from $\{x_t^i, w_t^i\}_{i=1}^N$

$\rightarrow$3 Propagate x_t^i by sampling from $f(\cdot | x_{t-1}^i, u_{t-1})$ for $i = 1, \dots, N$

end

N Number of particles,
x_t^i Particles,
w_t^i Particle weights



Algorithm

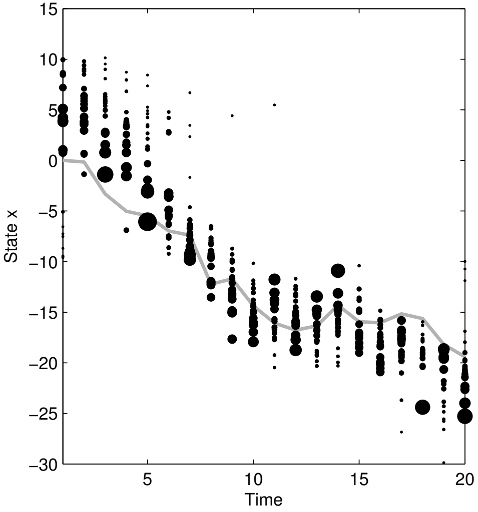
0 Initialize $x_1^i \sim p(x_1)$ and $w_i = \frac{1}{N}$ for $i = 1, \ldots, N$

for t = 1 to T

- **1** Evaluate $w_t^i = g(y_t | x_t^i, u_t)$ for $i = 1, \ldots, N$
- **2** Resample $\{x_t^i\}_{i=1}^N$ from $\{x_t^i, w_t^i\}_{i=1}^N$
- **3** Propagate x_t^i by sampling from $f(\cdot | x_{t-1}^i, u_{t-1})$ for $i = 1, \ldots, N$

end

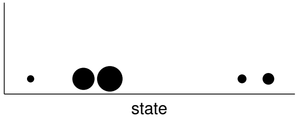
N Number of particles,
x_t^i Particles,
w_t^i Particle weights



Resampling

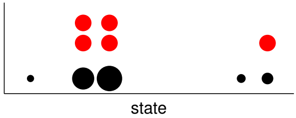
Resampling

- Represent the information contained in the N black dots (of different sizes) …



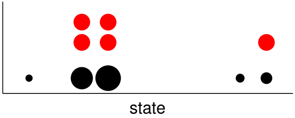
Resampling

- Represent the information contained in the N black dots (of different sizes) ... with the N red dots (of equal sizes)



Resampling

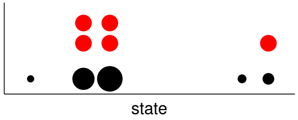
- Represent the information contained in the N black dots (of different sizes) … with the N red dots (of equal sizes)



- Can be seen as sampling from a cathegorical distribution

Resampling

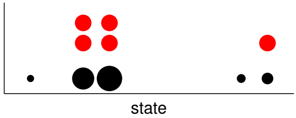
- Represent the information contained in the N black dots (of different sizes) ... with the N red dots (of equal sizes)



- Can be seen as sampling from a cathegorical distribution
- To avoid particle depletion ("a lot of 0-weights particles")

Resampling

- Represent the information contained in the N black dots (of different sizes) ... with the N red dots (of equal sizes)



- Can be seen as sampling from a cathegorical distribution
- To avoid particle depletion ("a lot of 0-weights particles")
- Some Matlab code:

```
v = rand(N,1); wc = cumsum(w(:,t)); [ ,ind1] = sort([v;wc]);
ind=find(ind1<=N)-(0:N-1)'; x(:,t)=x(ind,t);
w(:,t)=ones(N,1)./N;
```

Resampling cont'd

Resampling cont'd

- Crucial step in the development in the 90's for making particle filters useful in practice

Resampling cont'd

- Crucial step in the development in the 90's for making particle filters useful in practice
- Many different techniques exists with different properties and effiency (the Matlab code shown was only one way of doing it)

Resampling cont'd

- Crucial step in the development in the 90's for making particle filters useful in practice
- Many different techniques exists with different properties and effiency (the Matlab code shown was only one way of doing it)
- Computationally heavy. Not necessary to do in every iteration - a "depletion measure" can be introduced, and the resampling only performed when it reaches a certain threshold.

UPPSALA UNIVERSITET

Resampling cont'd

- Crucial step in the development in the 90's for making particle filters useful in practice
- Many different techniques exists with different properties and effiency (the Matlab code shown was only one way of doing it)
- Computationally heavy. Not necessary to do in every iteration - a "depletion measure" can be introduced, and the resampling only performed when it reaches a certain threshold.
- It is sometimes preferred to use the logarithms of the weights for numerical reasons.

Computational complexity

Computational complexity

In theory $\mathcal{O}(NTn_x^2)$ *(n_x is the number of states)*

Computational complexity

In theory $\mathcal{O}(NTn_x^2)$ *(n_x is the number of states)*

(Kalman filter is approx. of order $\mathcal{O}(Tn_x^3)$)

Computational complexity

In theory $\mathcal{O}(NTn_x^2)$ *(n_x is the number of states)*

(Kalman filter is approx. of order $\mathcal{O}(Tn_x^3)$)

Possible bottlenecks:

- Resampling step
- Likelihood evaluation (for the weight evaluation)
- Sampling from f for the propagation (trick: use proposal distributions!)

Software

Software

Lot of software packages exists. However, to my knowledge, no package that "everybody" uses. (In our research, we write our own code.)

Software

Lot of software packages exists. However, to my knowledge, no package that "everybody" uses. (In our research, we write our own code.)

A few relevant existing packages:

- Python: pyParticleEst
- Matlab: PFToolbox, PFLib
- C++: Particle++

Disclaimer: I have not used any of these packages myself

Terminology

Terminology

Bootstrap particle filter $\approx$ "Standard" particle filter

Terminology

Bootstrap particle filter $\approx$ "Standard" particle filter

Sequential Monte Carlo (SMC) $\approx$ Particle filter

Convergence

Convergence

The particle filter is "exact for $N = \infty$"

Convergence

The particle filter is "exact for $N = \infty$"

Consider a function $g(x_t)$ of interest. How well can it be approximated as $\hat{g}(x_t)$ when x_t is estimated in a particle filter?

Convergence

The particle filter is "exact for $N = \infty$"

Consider a function $g(x_t)$ of interest. How well can it be approximated as $\hat{g}(x_t)$ when x_t is estimated in a particle filter?

$\mathbb{E}\left[\hat{g}(x_t) - \mathbb{E}\left[g(x_t)\right]\right]^2 \leq \frac{p_t \|g(x_t)\|_{sup}}{N}$

UPPSALA UNIVERSITET

Convergence

The particle filter is "exact for $N = \infty$"

Consider a function $g(x_t)$ of interest. How well can it be approximated as $\hat{g}(x_t)$ when x_t is estimated in a particle filter?

$\mathbb{E}\left[\hat{g}(x_t) - \mathbb{E}\left[g(x_t)\right]\right]^2 \leq \frac{p_t \|g(x_t)\|_{sup}}{N}$

If the system "forgets exponentially fast" (e.g. linear systems), and some additional weak assumptions, $p_t = p < \infty$, i.e.,

$\mathbb{E}\left[\hat{g}(x_t) - \mathbb{E}\left[g(x_t)\right]\right]^2 \leq \frac{C}{N}$

Extensions

UPPSALA UNIVERSITET

Extensions

- Smoothing: Finding $p(x_t|y_{1:T})$ (marginal smoothing) or $p(x_{1:t}|y_{1:T})$ (joint smoothing) instead of $p(x_t|y_{1:t})$ (filtering). Offline ($y_{1:T}$ has to be available). Increased computational load.

UPPSALA UNIVERSITET

Extensions

- Smoothing: Finding $p(x_t|y_{1:T})$ (marginal smoothing) or $p(x_{1:t}|y_{1:T})$ (joint smoothing) instead of $p(x_t|y_{1:t})$ (filtering). Offline ($y_{1:T}$ has to be available). Increased computational load.
- If $f(x_{t+1}|x_t, u_t)$ is not suitable to sample from, **proposal distributions** can be used. In fact, an optimal proposal (w.r.t. reduced variance) exists.

Extensions

- Smoothing: Finding $p(x_t|y_{1:T})$ (marginal smoothing) or $p(x_{1:t}|y_{1:T})$ (joint smoothing) instead of $p(x_t|y_{1:t})$ (filtering). Offline ($y_{1:T}$ has to be available). Increased computational load.
- If $f(x_{t+1}|x_t, u_t)$ is not suitable to sample from, **proposal distributions** can be used. In fact, an optimal proposal (w.r.t. reduced variance) exists.
- Rao-Blackwellization for mixed linear/nonlinear models.

Extensions

- Smoothing: Finding $p(x_t|y_{1:T})$ (marginal smoothing) or $p(x_{1:t}|y_{1:T})$ (joint smoothing) instead of $p(x_t|y_{1:t})$ (filtering). Offline ($y_{1:T}$ has to be available). Increased computational load.
- If $f(x_{t+1}|x_t, u_t)$ is not suitable to sample from, **proposal distributions** can be used. In fact, an optimal proposal (w.r.t. reduced variance) exists.
- Rao-Blackwellization for mixed linear/nonlinear models.
- System identification: PMCMC, SMC^2.

UPPSALA UNIVERSITET

References

- Gustafsson, F. (2010). **Particle filter theory and practice with positioning applications.** *Aerospace and Electronic Systems Magazine, IEEE,* 25(7), 53-82.
- Schön, T.B., & Lindsten, F. **Learning of dynamical systems - Particle Filters and Markov chain methods.** *Draft available.*
- Doucet, A., & Johansen, A. M. (2009). **A tutorial on particle filtering and smoothing: Fifteen years later.** *Handbook of Nonlinear Filtering*, 12, 656-704.

Homework: Implement your own Particle Filter for any (simple) problem of your choice!