Time Series and Sequence Learning

Lecture 7 – Learning of State Space Models

Johan Alenlöv, Linköping University

2020-09-15

Summary of Lecture 6: Structural time series

A general structural time series model

$$y_t = \mu_t + \gamma_t + \varepsilon_t$$

can be written in state space form using **block matrices**.

State vector:

$$\alpha_t = \begin{bmatrix} \mu_t & \mu_{t-1} & \cdots & \mu_{t-k+1} & \gamma_t & \gamma_{t-1} & \cdots & \gamma_{t-s+2} \end{bmatrix}^\top$$

State space model:

$$\alpha_t = \begin{bmatrix} T_{[\mu]} & \\ & T_{[\gamma]} \end{bmatrix} \alpha_{t-1} + \begin{bmatrix} R_{[\mu]} & \\ & R_{[\gamma]} \end{bmatrix} \eta_t, \quad \eta_t \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_\zeta^2 & 0 \\ 0 & \sigma_\omega^2 \end{bmatrix} \right),$$

$$y_t = \begin{bmatrix} Z_{[\mu]} & Z_{[\gamma]} \end{bmatrix} \alpha_t + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, \sigma_\varepsilon^2).$$

Summary of Lecture 6: Trend component

A $k-1$th order polynomial trend model $\Delta^k \mu_t = \zeta_t$ can be written as

$$\alpha_t = \begin{bmatrix} c_1 & c_2 & \cdots & c_{k-1} & c_k \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \alpha_{t-1} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ \cdots \\ 0 \end{bmatrix} \zeta_t,$$

$$\mu_t = \begin{bmatrix} 1 & 0 & 0 & \vdots & 0 \end{bmatrix} \alpha_t,$$

where the **state vector** is

$$\alpha_t = \begin{bmatrix} \mu_t & \mu_{t-1} & \cdots & \mu_{t-k+1} \end{bmatrix}^\mathsf{T}$$

and $c_i = (-1)^{i+1} \binom{k}{i}$.

Summary of Lecture 6: Seasonal component

A s period seasonal model, $\sum_{j=0}^{s-1} \gamma_{t-j} = \omega_t$, can be written a

$$\alpha_t = \begin{bmatrix} -1 & -1 & \cdots & -1 & -1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \alpha_{t-1} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \omega_t$$

$$\gamma_t = \begin{bmatrix} 1 & 0 & 0 & \vdots & 0 \end{bmatrix} \alpha_t,$$

where the **state vector** is

$$\alpha_t = \begin{bmatrix} \gamma_t & \gamma_{t-1} & \cdots & \gamma_{t-s+2} \end{bmatrix}^{\mathsf{T}}.$$

Summary of Lecture 6: The Kalman filter

For any s, t, denote by $\hat{\alpha}_{t|s} = \mathbb{E}[\alpha_t \mid y_{1:s}]$ and $P_{t|s} = \mathrm{Cov}(\alpha_t \mid y_{1:s})$.

Thm. For an LGSS model, $p(\alpha_t \mid y_{1:s}) = \mathcal{N}(\alpha_t \mid \hat{\alpha}_{t|s}, P_{t|s})$.

Of particular interest are:

- Filtering distribution,

$$p(\alpha_t \mid y_{1:t}) = \mathcal{N}(\alpha_t \mid \hat{\alpha}_{t|t}, P_{t|t}).$$

- (1-step) Predictive distributions,

$$p(\alpha_t \mid y_{1:t-1}) = \mathcal{N}(\alpha_t \mid \hat{\alpha}_{t|t-1}, P_{t|t-1}),$$
$$p(y_t \mid y_{1:t-1}) = \mathcal{N}(y_t \mid \hat{y}_{t|t-1}, F_{t|t-1}).$$

Summary of Lecture 6: Parameter Estimation

- The **log-likelihood** for a LGSSM is calculated using the **Kalman filter**

$$\ell(\theta) = \text{const} - \frac{1}{2}\sum_{t=1}^{n}\Big(\log|F_{t|t-1}| + (y_t - \hat{y}_{t|t-1})^{\mathsf{T}} F_{t|t-1}^{-1}(y_t - \hat{y}_{t|t-1})\Big)$$

- Difficult to take derivatives $\Rightarrow$ **direct maximization difficult**.
- For few parameters, grid the parameters and calculate the log-likelihood.

Summary of Lecture 6: Expectation-Maximization

In the **Expectation Maximization** (EM) algorithm we alternate two steps,

1. E-step: Calculate $\mathcal{Q}(\theta, \tilde{\theta}) = \mathbb{E}[\log p_\theta(\alpha_{1:n}, y_{1:n}) \mid y_{1:n}, \tilde{\theta}]$
2. M-step: Find $\theta^\star$ that maximizes $\mathcal{Q}(\theta, \tilde{\theta})$.

We have that,

$$\mathcal{Q}(\theta, \tilde{\theta}) = \text{const.} - \frac{1}{2} \sum_{t=1}^{n} \Big[\log |\sigma_\epsilon^2| + \log |Q|$$
$$+ \{\hat{\varepsilon}_{t|n}^2 + \mathrm{Var}[\varepsilon_t \mid y_{1:n}]\}\sigma_\varepsilon^{-2} + \mathrm{tr}[\{\hat{\eta}_{t|n}\hat{\eta}_{t|n}^\mathsf{T} + \mathrm{Var}[\eta \mid y_{1:n}]\}Q^{-1}]\Big],$$

where $\hat{\varepsilon}_{t|n}, \mathrm{Var}[\varepsilon_t \mid y_{1:n}], \hat{\eta}_{t|n}$, and $\mathrm{Var}[\eta \mid y_{1:n}]$ are the **smoothed** mean and variances of ε_t and η_t.

To find $\theta^\star$ maximize $\mathcal{Q}(\theta, \tilde{\theta})$ by taking the derivative and set the derivative to zero.

Non-Linear State-Space Models

Limits of Linear Gaussian State-Space Models

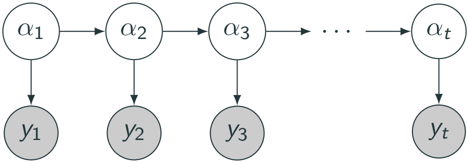
- Only allows for Gaussian observations.
  - What if we have discrete data?
- Only allows for Linear transformations.
  - What if the state moves in a non-linear fashion?
  - What if variance of the noise depends on the state?
  - What if the observation in a non-linear transformation of the states?
- To expend our models we need to work with non-linear and/or non-Gaussian models.

From LGSS model to general state-space model

Def. A **Linear Gaussian State-Space (LGSS)** model is given by:

$$\alpha_t = T\alpha_{t-1} + R\eta_t, \qquad \eta_t \sim \mathcal{N}(0, Q),$$
$$y_t = Z\alpha_t + \varepsilon_t \qquad \varepsilon_t \sim \mathcal{N}(0, \sigma_\epsilon^2),$$

and initial distribution $\alpha_1 \sim \mathcal{N}(a_1, P_1)$.

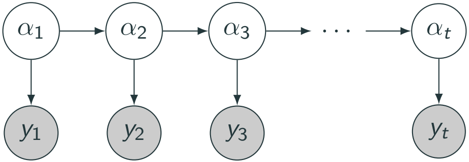


From LGSS model to general state-space model

Def. A **General State-Space** model is given by:

$$\alpha_t \,|\, \alpha_{t-1} \sim q(\alpha_t \,|\, \alpha_{t-1})$$
$$y_t \,|\, \alpha_t \sim g(y_t \,|\, \alpha_t)$$

and initial distribution $\alpha_1 \sim q(\alpha_1)$.



From LGSS model to general state-space model

Def. A **General State-Space** model is given by:

$$\alpha_t \,|\, \alpha_{t-1} \sim q(\alpha_t \,|\, \alpha_{t-1})$$
$$y_t \,|\, \alpha_t \sim g(y_t \,|\, \alpha_t)$$

and initial distribution $\alpha_1 \sim q(\alpha_1)$.

Thm. The **joint-smoothing distribution** is given by

$$p(\alpha_{1:n} \,|\, y_{1:n}) = \frac{q(\alpha_1) g(y_1 \,|\, \alpha_1) \prod_{i=2}^{n} q(\alpha_i \,|\, \alpha_{i-1}) g(y_i \,|\, \alpha_i)}{L_n(y_{1:n})},$$

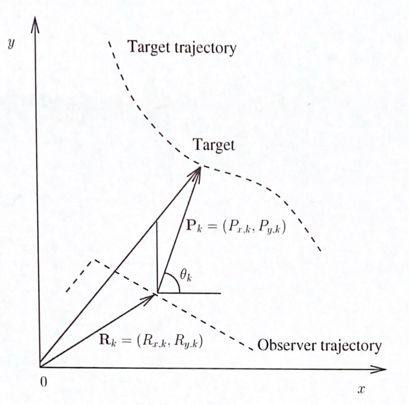
where $L_n(y_{1:n}) = \int q(\alpha_1) g(y_1 \,|\, \alpha_1) \prod_{i=2}^{n} q(\alpha_i \,|\, \alpha_{i-1}) g(y_i \,|\, \alpha_i) \mathrm{d}y_{1:n}$ is the **likelihood**.

Example: Bearings-only Tracking

$$\alpha_k = A\alpha_{k-1} + R\eta_k,$$
$$y_k = \arctan\left(\frac{P_{y,k} - R_{y,k}}{P_{x,k} - R_{x,k}}\right) + \sigma_\varepsilon \varepsilon_k,$$

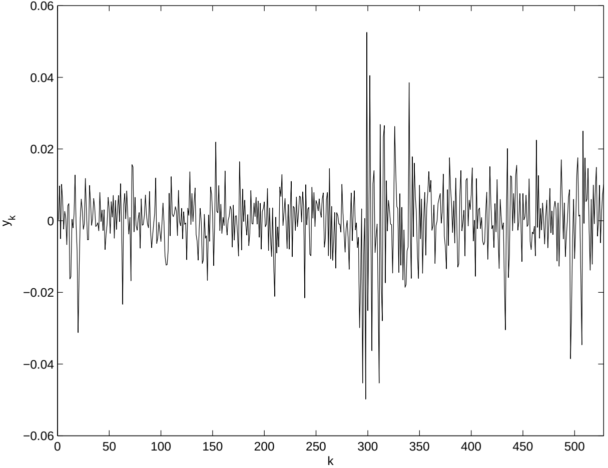
where $\alpha_k = (P_{x,k}, \dot{P}_{x,k}, P_{y,k}, \dot{P}_{y,k})^\mathsf{T}$
and

$$A = \begin{pmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad R = \begin{pmatrix} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \end{pmatrix}$$



Example: Stochastic Volatility

$$
\begin{aligned}
\alpha_k &= a\alpha_{k-1} + \sigma_\eta \eta_k, & \eta_k &\sim \mathcal{N}(0, \sigma_\eta^2) \\
y_k &= b \exp(\alpha_k/2)\varepsilon_k, & \varepsilon_k &\sim \mathcal{N}(0, \sigma_\varepsilon^2)
\end{aligned}
$$



Our Goal

- Given a **time-series** $y_{1:n}$ we are interested in:
  - Estimate the **filter distributions**.
  - Estimate the **parameters**.
- For the non-Linear models our aim is to estimate the expected values

$$\mathbb{E}[h(\alpha_t) \mid y_{1:t}] = \int h(\alpha_t) p(\alpha_t \mid y_{1:t}) \mathrm{d}\alpha_t,$$

where h is some function of interest.

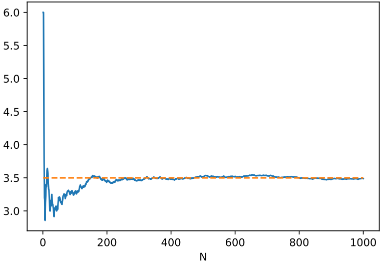
Monte Carlo and Importance Sampling

The Monte Carlo Method

- For non-linear / non-Gaussian models **exact** calculations of the filter distribution is **not possible**.
- Instead we use the Monte Carlo method:
  - Sample $x^i \sim p(x)$ for $i = 1, \dots, N$
  - Then $\hat{h} = \frac{1}{N} \sum_{i=1}^{N} h(x^i)$ is an estimate of $\mathbb{E}[h(x)] = \int h(x)p(x)\mathrm{d}x$

The Monte Carlo Method

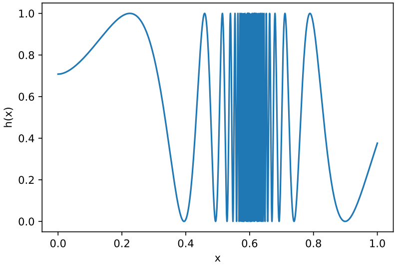
- For non-linear / non-Gaussian models **exact** calculations of the filter distribution is **not possible**.
- Instead we use the Monte Carlo method:
  - Sample $x^i \sim p(x)$ for $i = 1, \ldots, N$
  - Then $\hat{h} = \frac{1}{N} \sum_{i=1}^{N} h(x^i)$ is an estimate of $\mathbb{E}[h(x)] = \int h(x)p(x)\mathrm{d}x$



1. `x = np.random.randint(1,7,size = N)`
2. `est = np.mean(x)`

The Monte Carlo Method

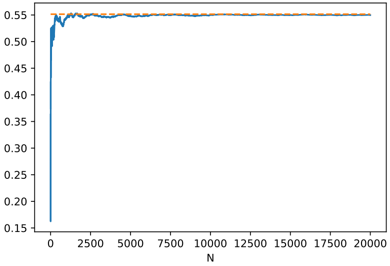
- For non-linear / non-Gaussian models **exact** calculations of the filter distribution is **not possible**.
- Instead we use the Monte Carlo method:
  - Sample $x^i \sim p(x)$ for $i = 1, \ldots, N$
  - Then $\hat{h} = \frac{1}{N} \sum_{i=1}^{N} h(x^i)$ is an estimate of $\mathbb{E}[h(x)] = \int h(x)p(x)\mathrm{d}x$



$$h(x) = \sin(1/\cos(\log(1 + 2\pi x)))^2$$

The Monte Carlo Method

- For non-linear / non-Gaussian models **exact** calculations of the filter distribution is **not possible**.
- Instead we use the Monte Carlo method:
  - Sample $x^i \sim p(x)$ for $i = 1, \dots, N$
  - Then $\hat{h} = \frac{1}{N}\sum_{i=1}^{N} h(x^i)$ is an estimate of $\mathbb{E}[h(x)] = \int h(x)p(x)\mathrm{d}x$



1. `x = np.random.uniform(0,1,size = N)`
2. `hx = np.sin(1/np.cos(np.log(1 + 2*np.pi*x)))**2`
3. `est = np.mean(hx)`

Importance Sampling

- Sometimes **sampling** from $p(x)$ is **difficult/impossible**.
- It might be easier to sample form a distribution $f(x)$.
- Assume that $f(x) = 0 \Rightarrow p(x) = 0$.
- We have that

$$\mathbb{E}_p[h(x)] = \int h(x)p(x)\mathrm{d}x = \int h(x)p(x)\frac{f(x)}{f(x)}\mathrm{d}x = \mathbb{E}_f\left[h(x)\frac{p(x)}{f(x)}\right]$$

- A Monte Carlo estimator would then become:
  1. Draw $x^i \sim f(x)$, for $i = 1, \ldots, N$
  2. Calculate $\omega^i = p(x^i)/f(x^i)$
  3. Estimate $\hat{h} = \frac{1}{N}\sum_{i=1}^{N} \omega^i h(x^i)$
- Known as **importance sampling**

Importance Sampling

- Often $p(x)$ is only known up to a normalizing constant $p(x) = z(x)/c$ where the constant $c = \int z(x)\mathrm{d}x$ is **unknown**.
- We can stil perform **importance sampling** in the following way:
  1. Draw $x^i \sim f(x)$, for $i = 1, \dots, N$.
  2. Calculate $\omega^i = z(x^i)/f(x^i)$
  3. Estimate $\hat{h} = \Omega^{-1} \sum_{i=1}^{N} \omega^i h(x^i)$, where $\Omega = \sum_{i=1}^{N} \omega^i$.
- Notice that:

$$\frac{1}{N} \sum_{i=1}^{N} \omega^i h(x^i) \to c \cdot \mathbb{E}_p[h(x)]$$

$$\frac{1}{N} \sum_{i=1}^{N} \omega^i \to c.$$

- We get an estimate of the **normalizing constant**.

Importance Sampling in SSM

Sequential Importance Sampling

Def. A **General State-Space** model is given by:

$$\alpha_t \mid \alpha_{t-1} \sim q(\alpha_t \mid \alpha_{t-1})$$
$$y_t \mid \alpha_t \sim g(y_t \mid \alpha_t)$$

and initial distribution $\alpha_1 \sim q(\alpha_1)$.

Thm. The **joint-smoothing distribution** is given by

$$p(\alpha_{1:n} \mid y_{1:n}) = \frac{q(\alpha_1)g(y_1 \mid \alpha_1)\prod_{i=2}^{n} q(\alpha_i \mid \alpha_{i-1})g(y_i \mid \alpha_i)}{L_n(y_{1:n})},$$

where $L_n(y_{1:n}) = \int q(\alpha_1)g(y_1 \mid \alpha_1)\prod_{i=2}^{n} q(\alpha_i \mid \alpha_{i-1})g(y_i \mid \alpha_i)\mathrm{d}y_{1:n}$ is the **likelihood**.

We wish to sample from $p(\alpha_{1:n} \mid y_{1:n})$ using **importance sampling**.

Sequential Importance Sampling

- Target this using **importance sampling**:
  - Assume that we have generated $(\alpha_{1:n}^i)_{i=1}^N$ from $f(\alpha_{1:n})$ such that

$$\sum_{i=1}^{N} \frac{\omega_n^i}{\Omega_n} h(\alpha_{1:n}^i) \approx \mathbb{E}[h(\alpha_{1:n}) \mid y_{1:n}]$$

  - To go to $n+1$ we do the following for each $i = 1, 2, \ldots, N$:
    - Draw $\alpha_{n+1}^i \sim f(\alpha_{n+1} \mid \alpha_{1:n}^i)$
    - Set $\alpha_{1:n+1}^i = (\alpha_{1:n}^i, \alpha_{n+1}^i)$
    - Set $\omega_{n+1}^i = \frac{q(\alpha_{n+1}^i \mid \alpha_n^i) g(y_{n+1} \mid \alpha_n^i)}{f(\alpha_{n+1}^i \mid \alpha_{1:n}^i)} \times \omega_n^i$.
- This gives us **sequential importance sampling** (SIS) where:

$$\sum_{i=1}^{N} \frac{\omega_{n+1}^i}{\Omega_{n+1}} h(\alpha_{1:n+1}) \approx \mathbb{E}[h(\alpha_{1:n+1}) \mid y_{1:n+1}]$$

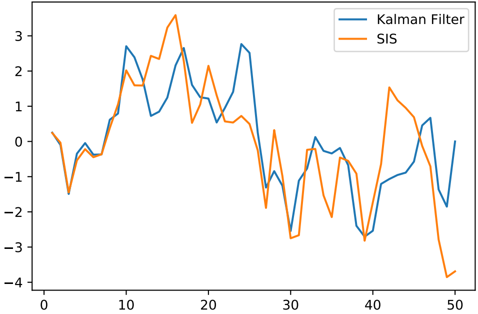
$$\frac{1}{N}\Omega_{n+1} = \frac{1}{N}\sum_{i=1}^{N} \omega_{n+1}^i \approx L(y_{1:n+1}).$$

Example: Linear Gaussian State Space Model

Def. A **Linear Gaussian State-Space (LGSS)** model is given by:

$$\alpha_t = T\alpha_{t-1} + R\eta_t, \qquad \eta_t \sim \mathcal{N}(0, Q),$$
$$y_t = Z\alpha_t + \varepsilon_t \qquad \varepsilon_t \sim \mathcal{N}(0, \sigma_\epsilon^2),$$

and initial distribution $\alpha_1 \sim \mathcal{N}(a_1, P_1)$.

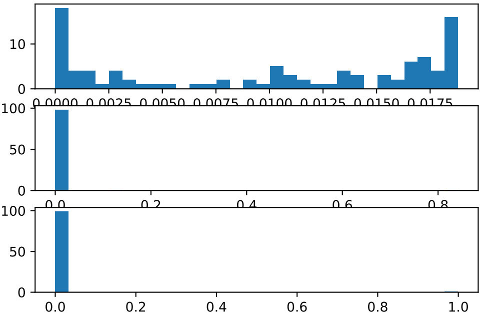


Example: Linear Gaussian State Space Model

Def. A **Linear Gaussian State-Space (LGSS)** model is given by:

$$\alpha_t = T\alpha_{t-1} + R\eta_t, \qquad \eta_t \sim \mathcal{N}(0, Q),$$
$$y_t = Z\alpha_t + \varepsilon_t \qquad \varepsilon_t \sim \mathcal{N}(0, \sigma_\epsilon^2),$$

and initial distribution $\alpha_1 \sim \mathcal{N}(a_1, P_1)$.

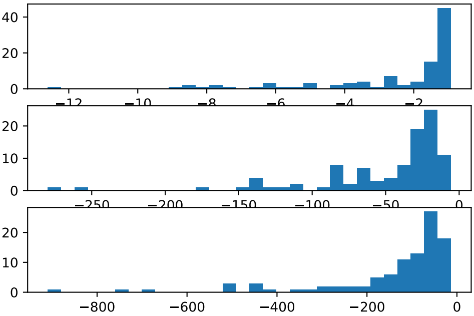


Example: Linear Gaussian State Space Model

Def. A **Linear Gaussian State-Space (LGSS)** model is given by:

$$\alpha_t = T\alpha_{t-1} + R\eta_t, \qquad \eta_t \sim \mathcal{N}(0, Q),$$
$$y_t = Z\alpha_t + \varepsilon_t \qquad \varepsilon_t \sim \mathcal{N}(0, \sigma_\epsilon^2),$$

and initial distribution $\alpha_1 \sim \mathcal{N}(a_1, P_1)$.



Solving the Weight Problem

- Unfortunately, weight degeneracy is a **universal problem** with the sequential importance sampling algorithm.
- The degeneracy is due to the repeated multiplication used to calculate the weights.
- This drawback prevented the sequential importance sampling algorithm from being practically useful during several decades.
- We will now discuss a solution to this problem: **sequential importance sampling with resampling** (SISR)

Interlude: IS with resampling

- Having a weighted sample $(x^i, \omega^i)_{i=1}^N$ approximating p. We can get a **uniformly weighted sample** by **resampling**, with replacement new variables $(\tilde{x}^i)_{i=1}^N$ from $(x^i)_{i=1}^N$ according to the weights $(\omega^i)_{i=1}^N$
- We get that $\tilde{x}^i = x^j$ with probability $\frac{\omega^j}{\Omega}$.
- This does **not** add bias to the estimator.
- The resampled estimator is

$$\frac{1}{N}\sum_{i=1}^{N} h(\tilde{x}^i) \approx \mathbb{E}_p[h(x)]$$

SIS with resampling

- **A simple – but revolutionary! – idea**: duplicate/kill particles with large/small weights!
- The most natural such selection is to draw new particles $(\tilde{\alpha}^i_{1:n})^N_{i=1}$ among the SIS-produced $(\alpha^i_{1:n})^N_{i=1}$ with probabilities by the normalized importance weights.
- Formally, for $i = 1, 2, \ldots, N$

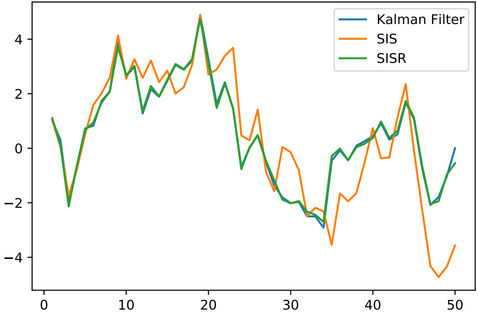
$$\tilde{\alpha}^i_{1:n} = \alpha^j_{1:n} \quad \text{w. pr.} \quad \frac{\omega^j_n}{\Omega_n}$$

Example: Linear Gaussian State Space Model

Def. A **Linear Gaussian State-Space (LGSS)** model is given by:

$$\begin{aligned} \alpha_t &= T\alpha_{t-1} + R\eta_t, & \eta_t &\sim \mathcal{N}(0, Q), \\ y_t &= Z\alpha_t + \varepsilon_t & \varepsilon_t &\sim \mathcal{N}(0, \sigma_\epsilon^2), \end{aligned}$$

and initial distribution $\alpha_1 \sim \mathcal{N}(a_1, P_1)$.



Algorithm: Particle Filter

Particle Filter:

Draw $\alpha_1^i \sim f(\alpha_1)$

Set $\omega_1^i = \frac{q(\alpha_1^i)g(y_1 \mid \alpha_1^i)}{f(\alpha_1^i)}$

Set $\Omega_1 = \sum_{i=1}^N \omega_1^i$

for $t = 2, 3, \ldots, n$ **do**

 Draw $I^i = j$ w. pr. $\frac{\omega_{t-1}^j}{\Omega_{t-1}}$

 Draw $\alpha_t^i \sim f(\alpha_t \,|\, \alpha_{1:t-1}^{I^i})$

 Set $\omega_t^i = \frac{q(\alpha_t^i \mid \alpha_{t-1}^{I^i})g(y_t \mid \alpha_t^i)}{f(\alpha_t^i \mid \alpha_{1:t-1}^{I^i})}$

 Set $\alpha_{1:t}^i = (\alpha_{1:t-1}^{I^i}, \alpha_t^i)$

 Set $\Omega_t = \sum_{i=1}^N \omega_t^i$

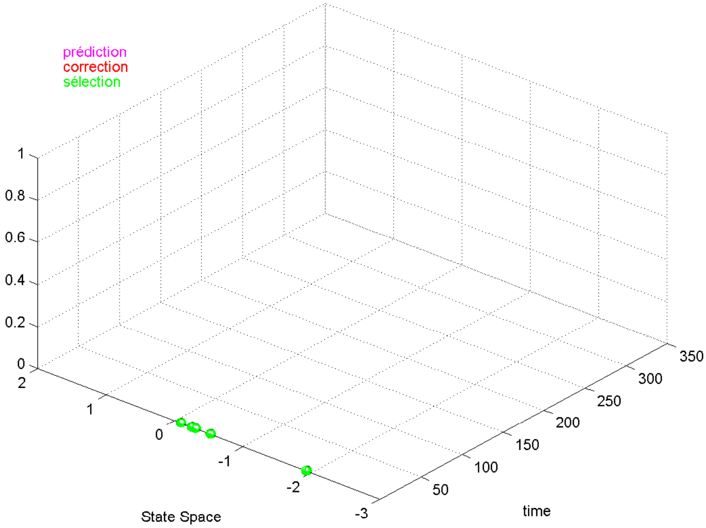
end for

Algorithm: Bootstrap Particle Filter

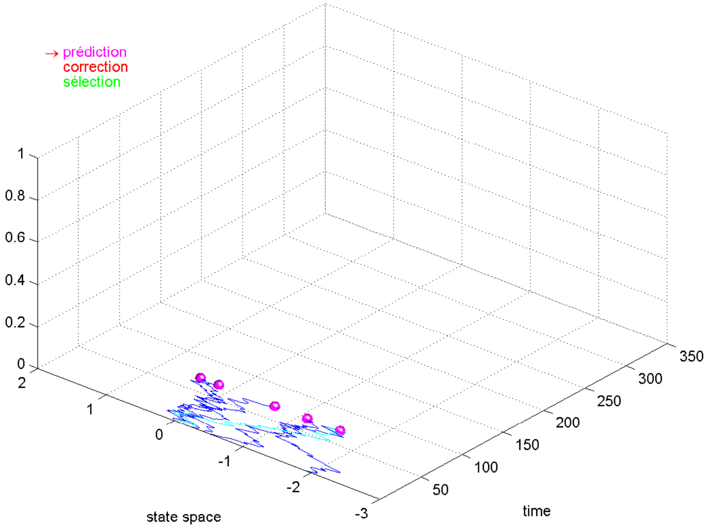
Bootstrap Particle Filter:

Draw $\alpha_1^i \sim q(\alpha_1)$
Set $\omega_1^i = g(y_1 \mid \alpha_1^i)$
Set $\Omega_1 = \sum_{i=1}^N \omega_1^i$
for $t = 2, 3, \ldots, n$ **do**
    Draw $I^i = j$ w. pr. $\frac{\omega_{t-1}^j}{\Omega_{t-1}}$
    Draw $\alpha_t^i \sim q(\alpha_t \mid \alpha_{t-1}^{I^i})$
    Set $\omega_t^i = g(y_t \mid \alpha_t^i)$
    Set $\alpha_{1:t}^i = (\alpha_{1:t-1}^{I^i}, \alpha_t^i)$
    Set $\Omega_t = \sum_{i=1}^N \omega_t^i$
end for

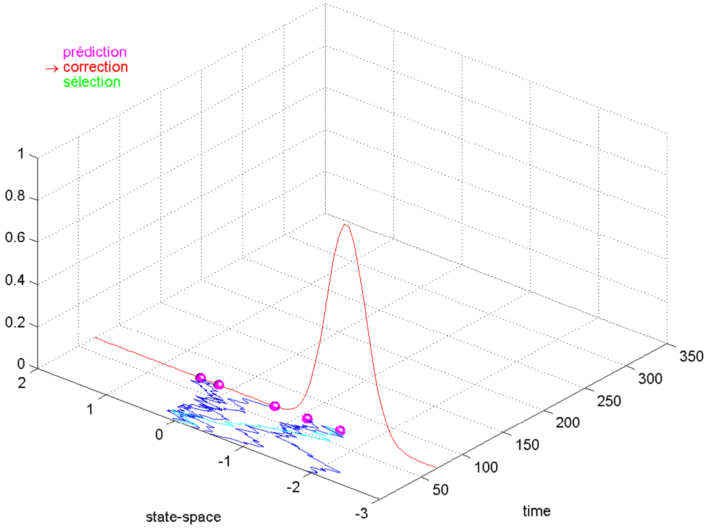
Particle Filter Movie



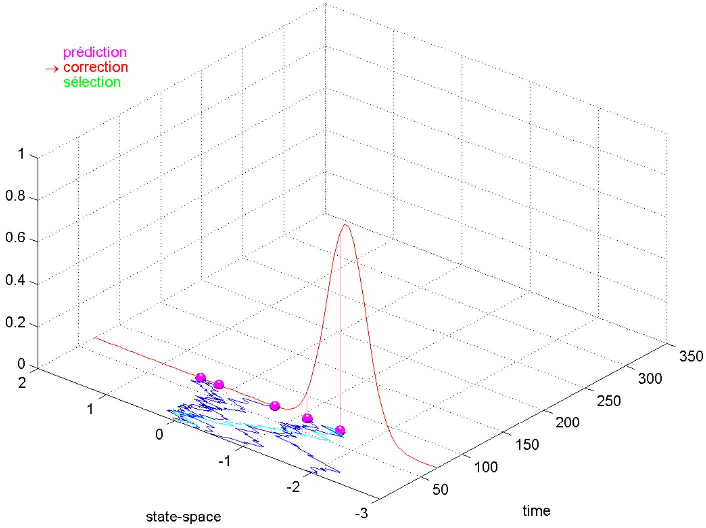
Particle Filter Movie



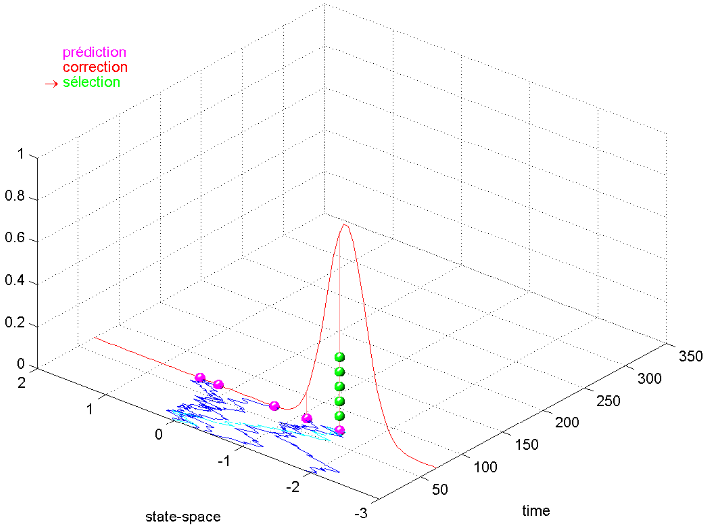
Particle Filter Movie



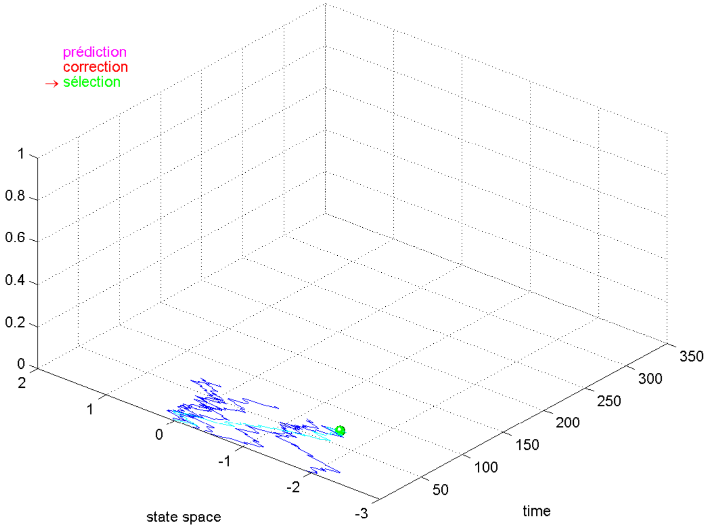
Particle Filter Movie



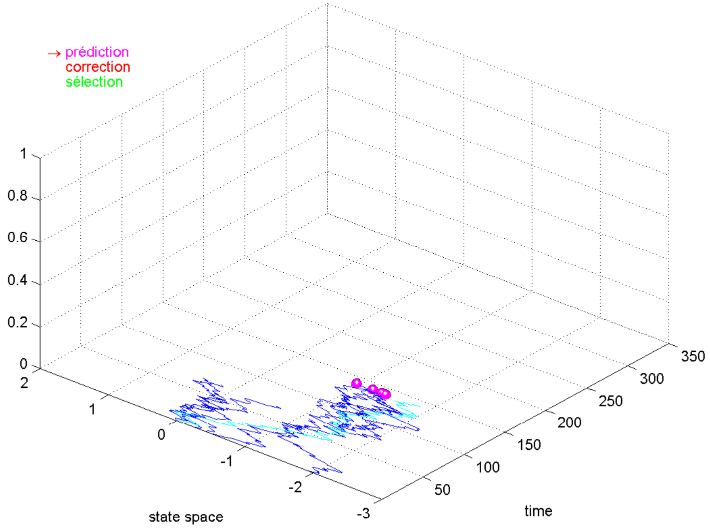
Particle Filter Movie



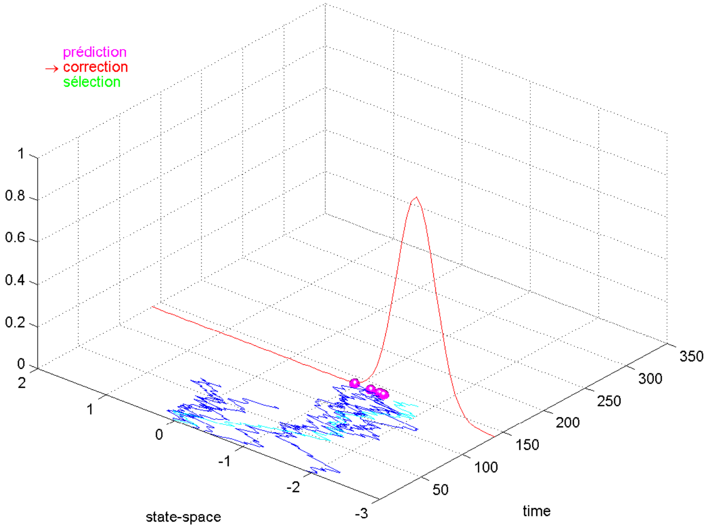
Particle Filter Movie



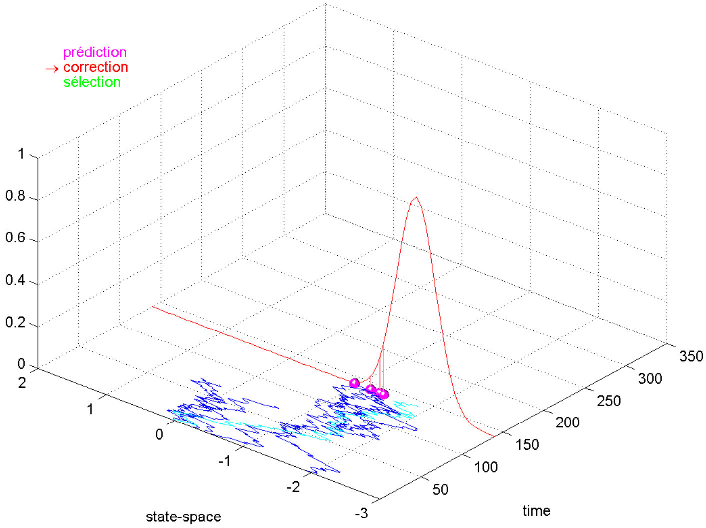
Particle Filter Movie



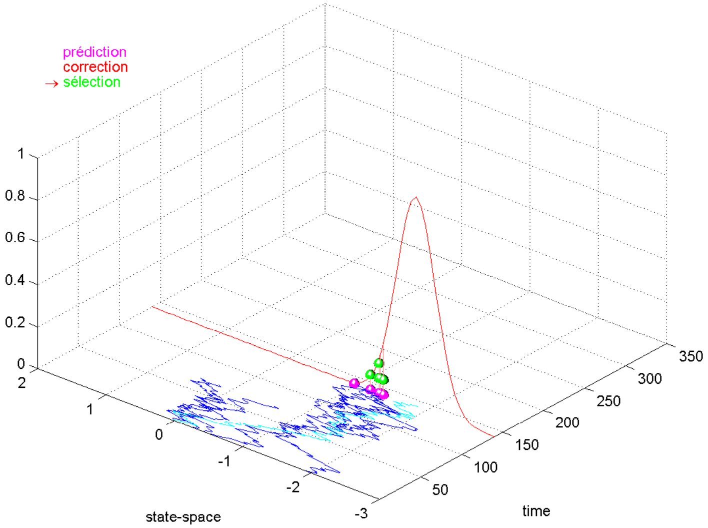
Particle Filter Movie



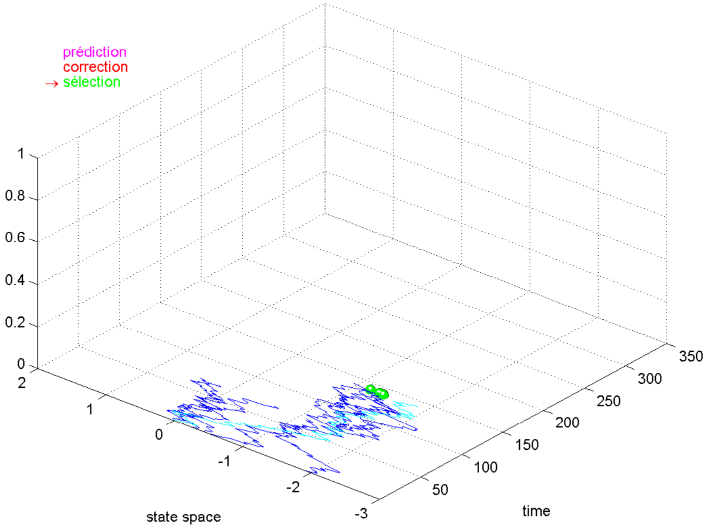
Particle Filter Movie



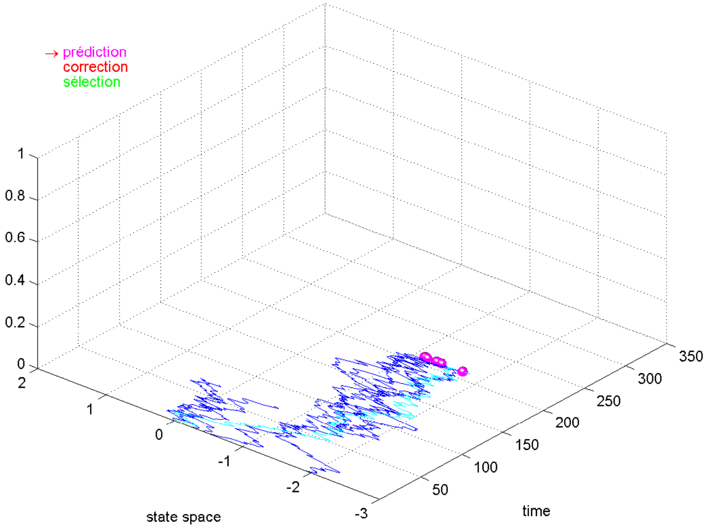
Particle Filter Movie



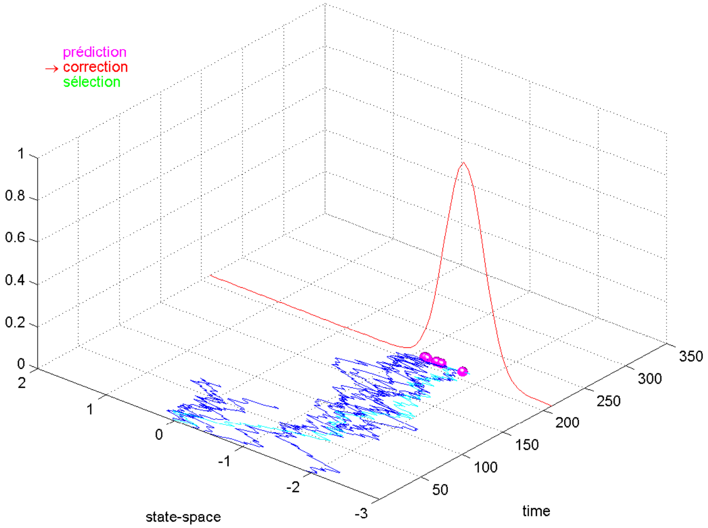
Particle Filter Movie



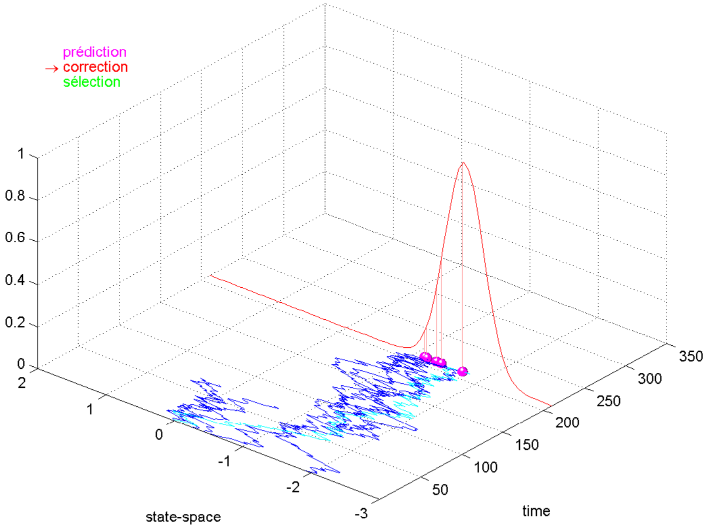
Particle Filter Movie



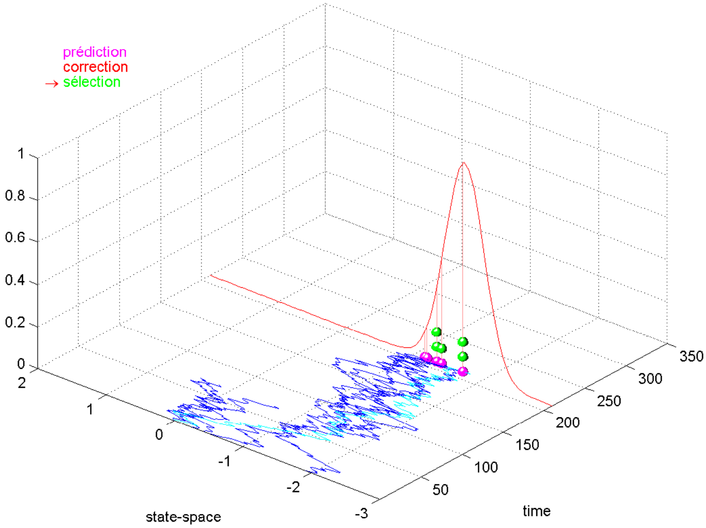
Particle Filter Movie



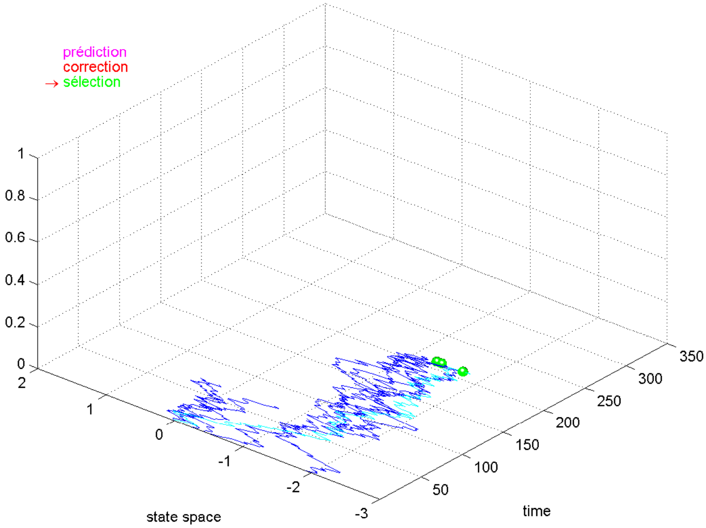
Particle Filter Movie



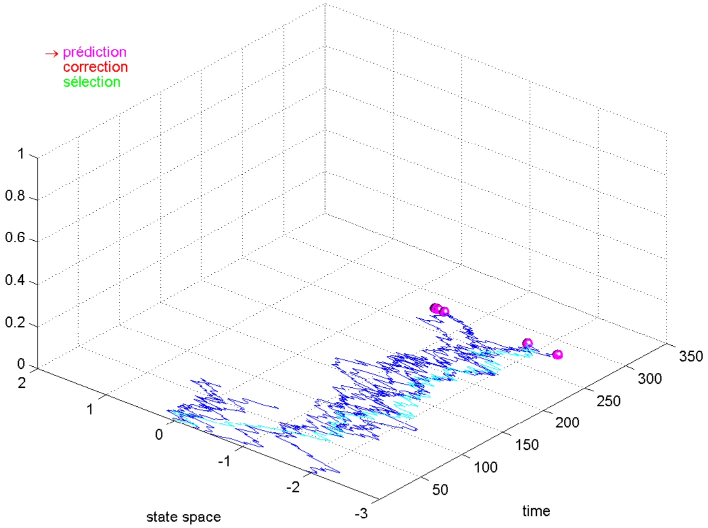
Particle Filter Movie



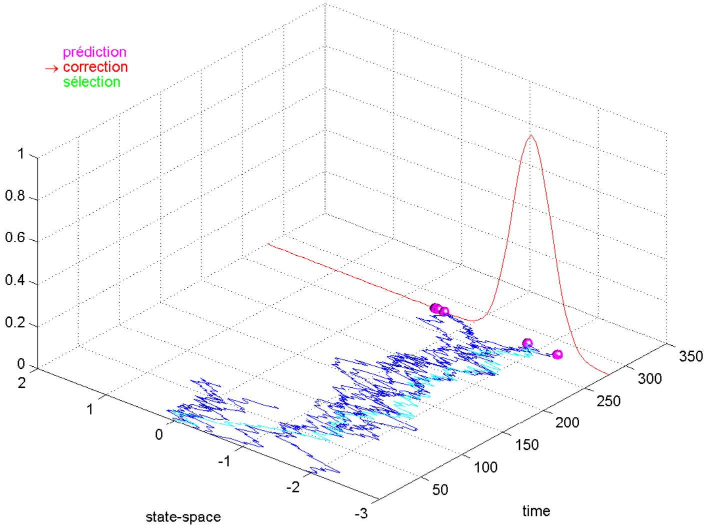
Particle Filter Movie



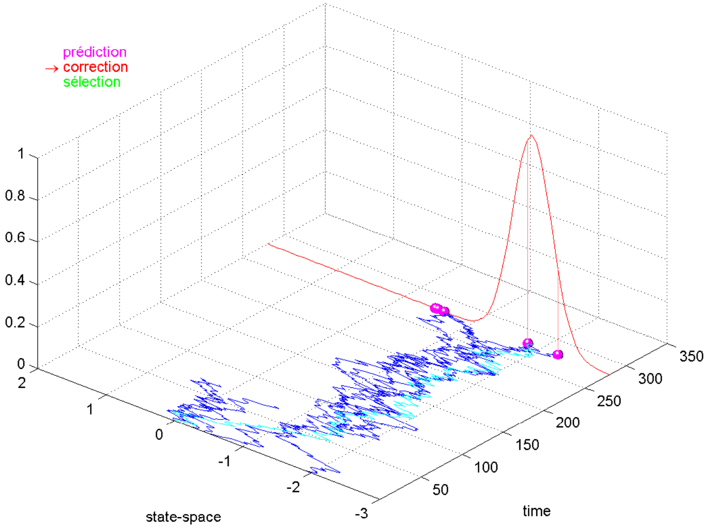
Particle Filter Movie



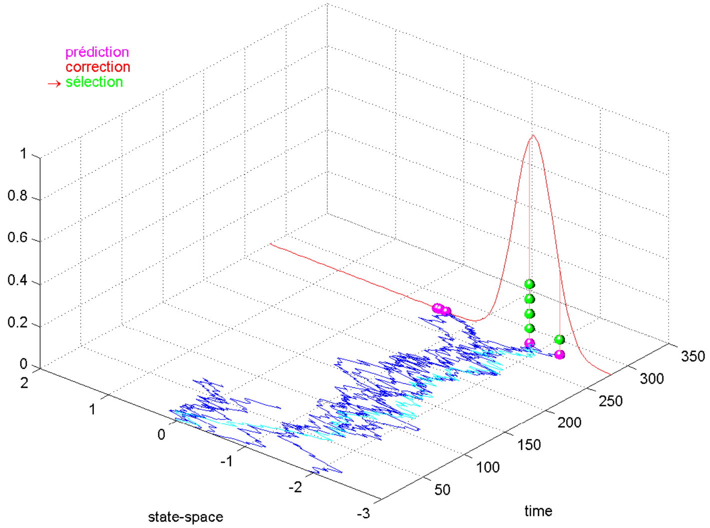
Particle Filter Movie



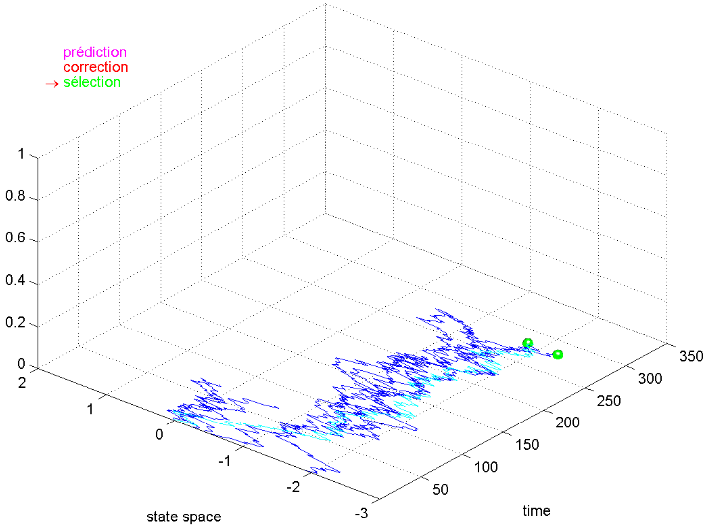
Particle Filter Movie



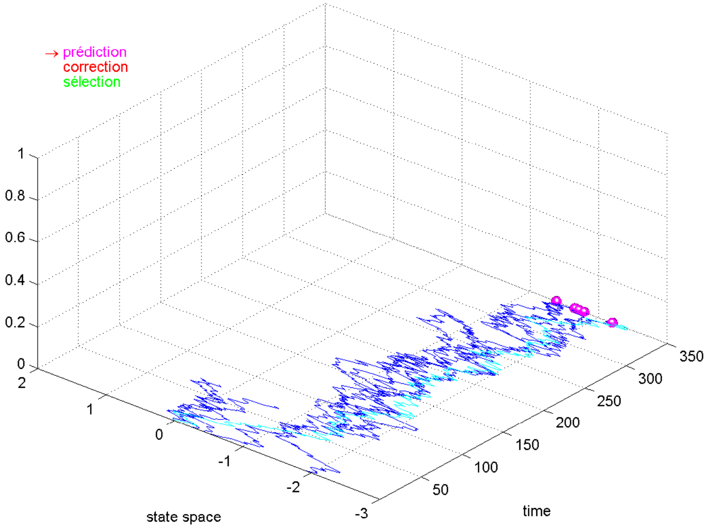
Particle Filter Movie



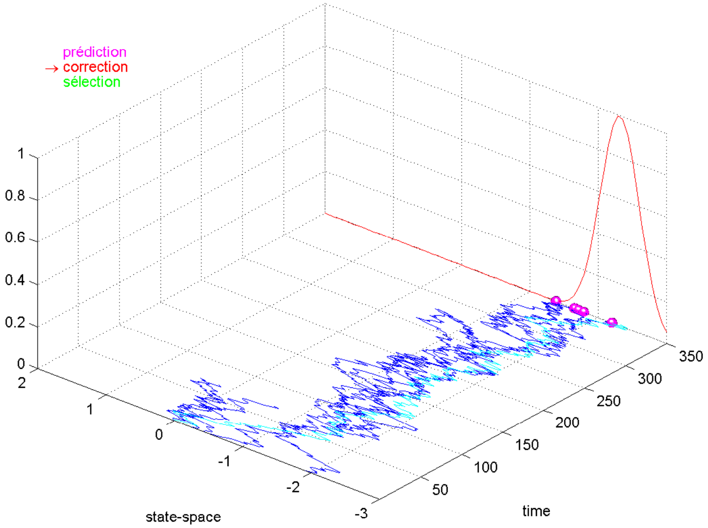
Particle Filter Movie



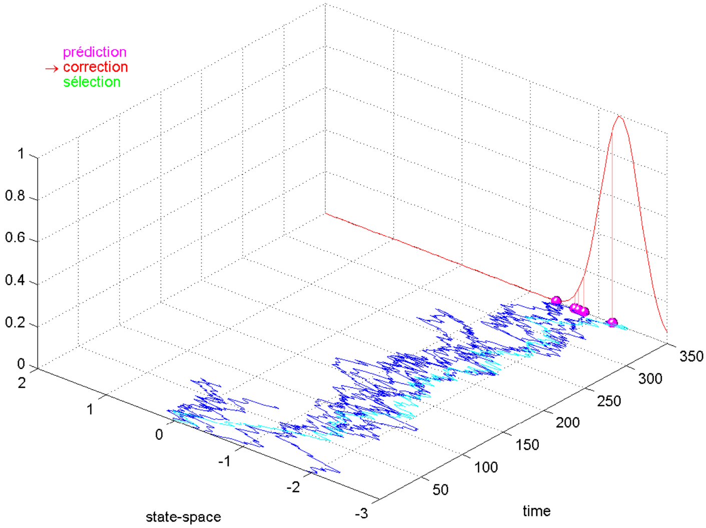
Particle Filter Movie



Particle Filter Movie



Particle Filter Movie



Summary

- Non-linear/non-Gaussian components in state-space models allows for more complicated models.
- Exact calculations is **not possible**, we have to approximate instead.
- The **Monte Carlo** method is a way of approximating distribution by random numbers.
- **Importance sampling** can be used when it is hard/impossible to sample directly from the distribution.
- **Sequential importance sampling**, repeatedly applied importance sampling to a state-space model.
  - Does not work in practice.
  - Weights degenerate.
- Solution is to **resample** the particles.
- The **particle filter** works by combining the sequential importance sampler with a resampling step.
  - The **bootstrap particle filter** is the simplest version, where the particles move according to the dynamics.