



UPPSALA  
UNIVERSITET

# Learning dynamical systems using SMC

---

Thomas Schön, Uppsala University

Bosch center for Artificial Intelligence  
Renningen, Germany  
May 30, 2018.

# What we do in the team

We automate the extraction of knowledge and understanding from data.

Both basic research **and** applied research (with companies).



Create **probabilistic models** of dynamical systems and their surroundings.

Develop methods to **learn** models from data.

The models can then be used by machines (or humans) to **understand** or **take decisions** about what will happen next.

# Nonlinear state space model (SSM)

The state space model (SSM) is a **Markov** chain that makes use of a **latent** variable representation to describe dynamical phenomena.

---

It consists of two stochastic processes:

1. unobserved (state) process  $\{x_t\}_{t \geq 0}$  modelling the dynamics,
2. observed process  $\{y_t\}_{t \geq 1}$  modelling the measurements and their relationship to the unobserved state process.

$$x_t = f(x_{t-1}, \theta) + v_t,$$

$$y_t = g(x_t, \theta) + e_t,$$

where  $\theta \in \mathbb{R}^{n_\theta}$  denotes static model parameters.

The SSM offers a practical representation not only for **modelling**, but also for **reasoning** and **inference**.

## Ex) “what are $x_t$ , $\theta$ and $y_t$ ”?

**Aim (motion capture):** Compute  $x_t$  (position and orientation of the different body segments) of a person ( $\theta$  describes the body shape) moving around indoors using measurements  $y_t$  (accelerometers, gyroscopes and ultrawideband).



**Show movie!**

---

Manon Kok, Jeroen D. Hol and Thomas B. Schön. Using inertial sensors for position and orientation estimation, *Foundations and Trends of Signal Processing*, 11(1–2):1–153, 2017.



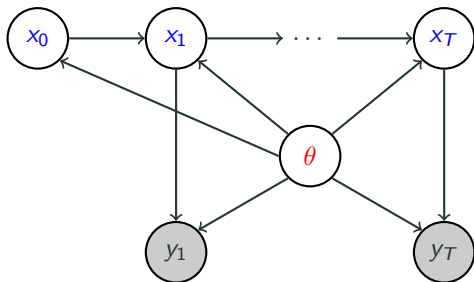
# Three different representations of the SSM

Three alternative representations, using

1. graphical models,
2. probability distributions or
3. probabilistic programs.

---

1. Representing the SSM using a graphical model:



# Representations using distributions or probabilistic programs

## 2. Representation using probability distributions

$$x_t \mid (x_{t-1}, \theta) \sim p(x_t \mid x_{t-1}, \theta),$$

$$y_t \mid (x_t, \theta) \sim p(y_t \mid x_t, \theta),$$

$$x_0 \sim p(x_0 \mid \theta).$$

---

## 3. Representing the SSM using a probabilistic program

```
x[1] ~ Gaussian(0.0, 1.0);           p(x1)
y[1] ~ Gaussian(x[1], 1.0);        p(y1 | x1)
for (t in 2..T) {
  x[t] ~ Gaussian(a*x[t - 1], 1.0); p(xt | xt-1)
  y[t] ~ Gaussian(x[t], 1.0);      p(yt | xt)
}
```

A **probabilistic program** encodes a **probabilistic model** (here an LG-SSM) according to the semantics of a particular probabilistic programming language (here Birch).

# SSM – full probabilistic model

The **full probabilistic model** is given by

$$p(x_{0:T}, \theta, y_{1:T}) = \underbrace{p(y_{1:T} | x_{0:T}, \theta)}_{\text{data distribution}} \underbrace{p(x_{0:T}, \theta)}_{\text{prior}}$$

Distribution describing a parametric nonlinear SSM

$$p(x_{0:T}, \theta, y_{1:T}) = \underbrace{\prod_{t=1}^T \underbrace{p(y_t | x_t, \theta)}_{\text{observation}}}_{\text{data distribution}} \underbrace{\prod_{t=1}^T \underbrace{p(x_t | x_{t-1}, \theta)}_{\text{dynamics}} \underbrace{p(x_0 | \theta)}_{\text{state}} \underbrace{p(\theta)}_{\text{param.}}}_{\text{prior}}$$

**Model = probability distribution!**

# Learning the states and the parameters

Based on our generative model, compute the **posterior distribution**

$$p(x_{0:T}, \theta | y_{1:T}) = \underbrace{p(x_{0:T} | \theta, y_{1:T})}_{\text{state inf.}} \underbrace{p(\theta | y_{1:T})}_{\text{param. inf.}}.$$

**Bayesian** formulation – model the unknown parameters as a random variable  $\theta \sim p(\theta)$  and compute

$$p(\theta | y_{1:T}) = \frac{p(y_{1:T} | \theta)p(\theta)}{p(y_{1:T})}$$

**Maximum likelihood** formulation – model the unknown parameters as a deterministic variable and solve

$$\hat{\theta} = \arg \max_{\theta \in \Theta} p(y_{1:T} | \theta).$$

# Central object – the likelihood

The likelihood is computed by marginalizing

$$p(\mathbf{x}_{0:T}, \mathbf{y}_{1:T} | \theta) = p(\mathbf{x}_0 | \theta) \prod_{t=1}^T p(y_t | \mathbf{x}_t, \theta) \prod_{t=1}^T p(\mathbf{x}_t | \mathbf{x}_{t-1}, \theta),$$

w.r.t the state sequence  $\mathbf{x}_{0:T}$ ,

$$p(\mathbf{y}_{1:T} | \theta) = \int p(\mathbf{x}_{0:T}, \mathbf{y}_{1:T} | \theta) d\mathbf{x}_{0:T}.$$

(We are averaging  $p(\mathbf{x}_{0:T}, \mathbf{y}_{1:T} | \theta)$  over all possible state sequences.)

Equivalently we have

$$p(\mathbf{y}_{1:T} | \theta) = \prod_{t=1}^T p(y_t | \mathbf{y}_{1:t-1}, \theta) = \prod_{t=1}^T \int p(y_t | \mathbf{x}_t, \theta) \underbrace{p(\mathbf{x}_t | \mathbf{y}_{1:t-1}, \theta)}_{\text{key challenge}} d\mathbf{x}_t.$$

# State inference – nonlinear filtering problem

The nonlinear filtering problem involves the **measurement update**

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

and the **time update**

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

- Aim:**
1. Give a (hopefully) intuitive explanation of **sequential Monte Carlo (SMC)** for probabilistic modelling of dynamical systems.
  2. Derive a new **stochastic optimization** method that can for example be used to learn unknown parameters in nonlinear SSMs.

1. Probabilistic modelling of dynamical systems
2. **Sequential Monte Carlo (SMC)**
3. Stochastic optimization
4. Some ongoing research snapshots (if there is time)

# Sequential Monte Carlo

The need for computational methods, such as SMC, is tightly coupled to the intractability of the integrals on the previous slides.

SMC provide approximate solutions to **integration** problems where there is a **sequential structure** present.

The **particle filter** approximates  $p(x_t | y_{1:t})$  for

$$\begin{aligned}x_t &= f(x_{t-1}) + v_t, \\y_t &= g(x_t) + e_t,\end{aligned}$$

by maintaining an empirical distribution made up of  $N$  samples (particles)  $\{x_t^i\}_{i=1}^N$  and the corresponding weights  $\{w_t^i\}_{i=1}^N$

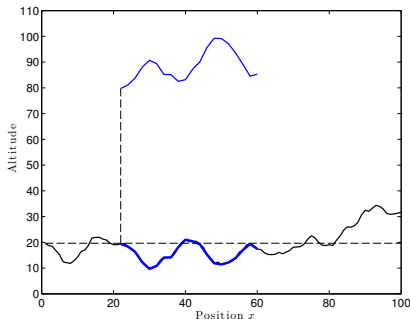
$$\underbrace{\hat{p}(x_t | y_{1:t})}_{\hat{\pi}(x_t)} = \sum_{i=1}^N \frac{w_t^i}{\sum_{j=1}^N w_t^j} \delta_{x_t^i}(x_t).$$



# Particle filter – introductory example (I/II)

Consider a toy 1D localization problem.

## Data



## Model

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  $x_t$  (position) based on the measurements  $y_{1:t} \triangleq \{y_1, \dots, y_t\}$  by computing the filter density  $p(x_t | y_{1:t})$ .

## Particle filter – introductory example (II/II)

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

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

# Sequential Monte Carlo – particle filter



**SMC = sequential importance sampling + resampling**

1. **Propagation:**  $x_t^i \sim p(x_t | x_{1:t-1}^{a_t^i})$  and  $x_{1:t}^i = \{x_{1:t-1}^{a_t^i}, x_t^i\}$ .
2. **Weighting:**  $\bar{w}_t^i = W_t(x_t^i) = p(y_t | x_t^i)$ .
3. **Resampling:**  $\mathbb{P}(a_t^i = j) = \bar{w}_{t-1}^j / \sum_l \bar{w}_{t-1}^l$ .

---

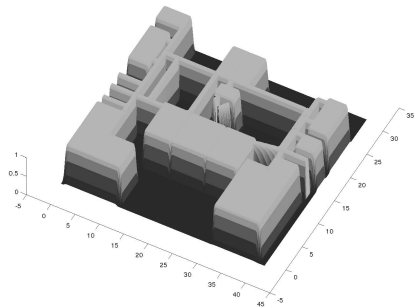
The **ancestor indices**  $\{a_t^i\}_{i=1}^N$  are very **useful** auxiliary variables!  
They make the stochasticity of the resampling step explicit.

## Application – indoor localization (I/III)

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



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



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

# Application – indoor localization (II/III)



Show movie

Johan Kihlberg, Simon Tegelid, Manon Kok and Thomas B. Schön. **Map aided indoor positioning using particle filters.** *Reglermöte (Swedish Control Conference)*, Linköping, Sweden, June 2014.

# Application – indoor localization (III/III)

**Aim:** Compute the **position** using variations in the ambient magnetic field and the motion of the person (acceleration and angular velocities). All of this observed using sensors in a standard smartphone.



Movie – map making:

[www.youtube.com/watch?v=enLMiUqPVJo](http://www.youtube.com/watch?v=enLMiUqPVJo)

Movie – indoor positioning result

Arno Solin, Simo Särkkä, Juho Kannala and Esa Rahtu. **Terrain navigation in the magnetic landscape: Particle filtering for indoor positioning.** In *Proceedings of the European Navigation Conference Helsinki*, Finland, June, 2016.

Arno Solin, Manon Kok, Niklas Wahlström, Thomas B. Schön and Simo Särkkä. **Modeling and interpolation of the ambient magnetic field by Gaussian processes.** *IEEE Transactions on Robotics*, 2018 (to appear).

# Particle MCMC = SMC + MCMC

A systematic way of combining SMC and MCMC.

Builds on an extended target construction.

**Intuitively:** SMC is used as a high-dimensional proposal mechanism on the space of state trajectories  $\mathcal{X}^T$ .

**A bit more precise:** Construct a Markov chain with  $p(\theta, \mathbf{x}_{1:T} | y_{1:T})$  (or one of its marginals) as its stationary distribution. Also used for parameter learning.

## Exact approximations

Pioneered by the work:

Christophe Andrieu, Arnaud Doucet and Roman Holenstein, **Particle Markov chain Monte Carlo methods**, *Journal of the Royal Statistical Society: Series B*, 72:269-342, 2010.

**Aim:** 1. Give a (hopefully) intuitive explanation of **sequential Monte Carlo (SMC)** for probabilistic modelling of dynamical systems. 2. Derive a new **stochastic optimization** method that can for example be used to learn unknown parameters in nonlinear SSMs.

1. Probabilistic modelling of dynamical systems
2. Sequential Monte Carlo (SMC)
- 3. Stochastic optimization**
4. Some ongoing research snapshots (if there is time)



## Quasi-Newton — A non-standard take

Our problem is of the form (note change of notation...)

$$\max_{\mathbf{x}} f(\mathbf{x})$$

**Idea underlying (quasi-)Newton methods:** Learn a local quadratic model  $q(\mathbf{x}_k, \delta)$  of the cost function  $f(\mathbf{x})$  around the current iterate  $\mathbf{x}_k$

$$q(\mathbf{x}_k, \delta) = f(\mathbf{x}_k) + \mathbf{g}(\mathbf{x}_k)^\top \delta + \frac{1}{2} \delta^\top \mathbf{H}(\mathbf{x}_k) \delta$$

A second-order Taylor expansion around  $\mathbf{x}_k$ , where

$$\mathbf{g}(\mathbf{x}_k) = \nabla f(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_k},$$

$$\mathbf{H}(\mathbf{x}_k) = \nabla^2 f(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_k},$$

$$\delta = \mathbf{x} - \mathbf{x}_k.$$

We have measurements of the

- cost function  $f_k = f(\mathbf{x}_k)$ ,
- and its gradient  $\mathbf{g}_k = \mathbf{g}(\mathbf{x}_k)$ .

**Question:** How do we update the Hessian model?

---

Line segment connecting two adjacent iterates  $\mathbf{x}_k$  and  $\mathbf{x}_{k+1}$ :

$$\mathbf{r}_k(\tau) = \mathbf{x}_k + \tau(\mathbf{x}_{k+1} - \mathbf{x}_k), \quad \tau \in [0, 1].$$

## Useful basic facts

The fundamental theorem of calculus states that

$$\int_0^1 \frac{\partial}{\partial \tau} \nabla f(r_k(\tau)) d\tau = \nabla f(r_k(1)) - \nabla f(r_k(0)) = \underbrace{\nabla f(x_{k+1})}_{g_{k+1}} - \underbrace{\nabla f(x_k)}_{g_k}$$

and the chain rule tells us that

$$\frac{\partial}{\partial \tau} \nabla f(r_k(\tau)) = \nabla^2 f(r_k(\tau)) \frac{\partial r_k(\tau)}{\partial \tau} = \nabla^2 f(r_k(\tau)) (x_{k+1} - x_k).$$

$$\underbrace{g_{k+1} - g_k}_{=y_k} = \int_0^1 \frac{\partial}{\partial \tau} \nabla f(r_k(\tau)) d\tau = \int_0^1 \nabla^2 f(r_k(\tau)) d\tau \underbrace{(x_{k+1} - x_k)}_{s_k}.$$

## Result — the quasi-Newton integral

With the definitions  $y_k \triangleq g_{k+1} - g_k$  and  $s_k \triangleq x_{k+1} - x_k$  we have

$$y_k = \int_0^1 \nabla^2 f(r_k(\tau)) d\tau s_k.$$

**Interpretation:** The difference between two consecutive gradients ( $y_k$ ) constitute a *line integral observation of the Hessian*.

**Problem:** Since the Hessian is unknown there is no functional form available for it.

## Solution 1 — recovering existing quasi-Newton algorithms

Existing quasi-Newton algorithms (e.g. BFGS, DFP, Broyden's method) assume the Hessian to be constant

$$\nabla^2 f(r_k(\tau)) \approx H_{k+1}, \quad \tau \in [0, 1],$$

implying the following approximation of the integral (**secant condition**)

$$y_k = H_{k+1} s_k.$$

---

Find  $H_{k+1}$  by **regularizing**  $H$ :

$$\begin{aligned} H_{k+1} &= \min_H \|H - H_k\|_W^2, \\ \text{s.t. } & H = H^\top, \quad H s_k = y_k, \end{aligned}$$

Equivalently, the existing quasi-Newton methods can be interpreted as **particular instances of Bayesian linear regression**.

## Solution 2 — use a flexible nonlinear model

The approach used here is fundamentally different.

Recall that the problem is **stochastic** and **nonlinear**.

Hence, we need a model that can deal with such a problem.

**Idea:** Represent the Hessian using a **Gaussian process** learnt from data.

Two of the remaining challenges:

1. Can we use line integral observations when learning a GP?
2. How do we ensure that the resulting GP represents a Hessian?

## Stochastic quasi-Newton integral

$$y_k = \int_0^1 \underbrace{B(r_k(\tau))}_{=\nabla^2 f(r_k(\tau))} s_k d\tau + e_k,$$

corresponds to noisy ( $e_k$ ) gradient observations.

Since  $B(\mathbf{x})s_k$  is a column vector, the integrand is given by

$$\text{vec}(B(\mathbf{x})s_k) = (s_k^T \otimes I) \text{vec}(B(\mathbf{x})) = (s_k^T \otimes I) \text{vec}(B(\mathbf{x})),$$

where  $\text{vec}(B(\mathbf{x})) = D \underbrace{\text{vech}(B(\mathbf{x}))}_{\tilde{B}(\mathbf{x})}$ .

---

Let us use a GP model for the unique elements of the Hessian

$$\tilde{B}(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}')).$$

# Resulting stochastic qN integral and Hessian model

**Summary:** resulting stochastic quasi-Newton integral:

$$y_k = \underbrace{(s_k^T \otimes I)D}_{=\bar{D}_k} \int_0^1 \tilde{B}(r_k(\tau))d\tau + e_k,$$

with the following model for the Hessian

$$\tilde{B}(x) \sim \mathcal{GP}(\mu(x), \kappa(x, x')).$$

The Hessian can now be estimated using tailored GP regression.

Linear transformations (such as an integral or a derivative) of a GP results in a new GP.



# Resulting stochastic optimization algorithm

Standard non-convex numerical optimization loop with **non-standard components**.

---

## Algorithm 1 Stochastic optimization

---

1. **Initialization** ( $k = 1$ )
  2. **while** *not terminated* **do**
    - (a) Compute a search direction  $p_k$  using the current approximation of the gradient  $g_k$  and Hessian  $B_k$ .
    - (b) Stochastic line search to find a step length  $\alpha_k$  and set
$$x_{k+1} = x_k + \alpha_k p_k.$$
    - (c) Set  $k := k + 1$
    - (d) Update the Hessian estimate (tailored GP regression)
  3. **end while**
-

## ex) Simple linear toy problem

Identify the parameters  $\theta = (a, c, q, r)^\top$  in

$$x_{t+1} = ax_t + w_t,$$

$$y_t = cx_t + e_t,$$

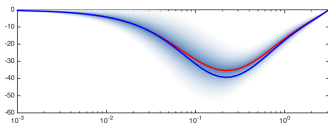
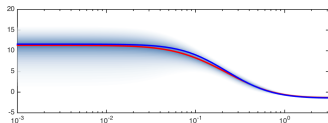
$$w_t \sim \mathcal{N}(0, q),$$

$$e_t \sim \mathcal{N}(0, r).$$

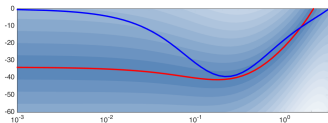
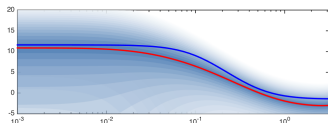
Observations:

- The likelihood  $L(\theta) = p(y_{1:T} | \theta)$  and its gradient  $\nabla_{\theta} L(\theta)$  are available in closed form via standard Kalman filter equations.
- Standard gradient-based search algorithms applies.
- Deterministic optimization problem  $(L(\theta), \nabla_{\theta} L(\theta))$  noise-free).

# ex) Simple linear toy problem



Both alg. in the noise-free case.



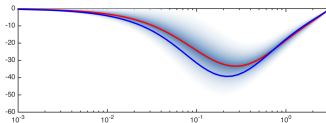
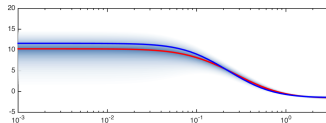
Classical BFGS alg. for noisy observations of  $L(\theta)$  and  $\nabla L(\theta)$ .

100 independent datasets.

Clear blue – True system

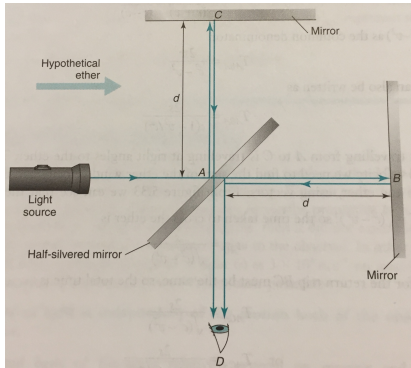
Red – Mean value of estimate

Shaded blue – individual results



GP-based BFGS alg. with noisy observations of  $L(\theta)$  and  $\nabla L(\theta)$ . 30/41

## ex) laser interferometry



The classic Michelson-Morley experiment from 1887.

**Idea:** Merge two light sources to create an interference pattern by superposition.

---

Two cases:

1. Mirror B and C at the **same** distance from mirror A.
2. Mirror B and C at **different** distances from mirror A.

## ex) laser interferometry

Dynamics: constant velocity model (with unknown force  $w$ )

$$\begin{pmatrix} \dot{p} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} p \\ v \end{pmatrix} + \begin{pmatrix} 0 \\ w \end{pmatrix}.$$

Measurements: generated using two detectors

$$y_1 = \alpha_0 + \alpha_1 \cos(\kappa p) + e_1, \quad e_1 \sim \mathcal{N}(0, \sigma^2),$$

$$y_2 = \beta_0 + \beta_1 \sin(\kappa p + \gamma) + e_2, \quad e_2 \sim \mathcal{N}(0, \sigma^2).$$

Unknown parameters:  $\theta = (\alpha_0 \quad \alpha_1 \quad \beta_0 \quad \beta_1 \quad \gamma \quad \sigma)^T$ .

---

Resulting maximum likelihood system identification problem

$$\max_{\theta} p(y_{1:T} | \theta)$$



## Research snapshots

---

## Snapshot 1 – scaling up to large problems

What is the key limitation of our GP-based optimization algorithm?

It **does not** scale to large-scale problems!

Still highly useful and competitive for **small to medium** sized problems.

---

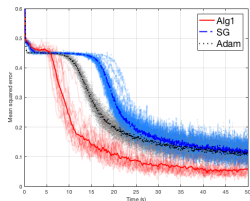
We have developed a **new** technique that scales to **very large** problems.



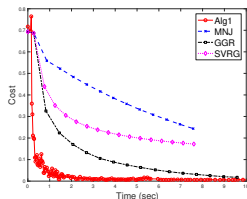
# Snapshot 1 – scaling up to large problems

## Key innovations:

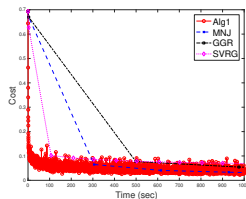
- Replace the GP with a matrix updated using fast Cholesky routines.
- Exploit a receding history of iterates and gradients akin to L-BFGS.
- An auxiliary variable Markov chain construction.



Training a deep CNN for MNIST data.



Logistic loss function with an L2 regularizer, gisette, 6 000 observations and 5 000 unknown variables.



Logistic loss function with an L2 regularizer, URL, 2 396 130 observations and 3 231 961 unknown variables.

## Snapshot 2 – GP-based nonlinear state space model

“Inspired by the Gaussian process, enabled by the particle filter”

$$\begin{aligned}x_{t+1} &= f(x_t) + w_t, & \text{s.t. } f(x) &\sim \mathcal{GP}(0, \kappa_{\eta, f}(x, x')), \\y_t &= g(x_t) + e_t, & \text{s.t. } g(x) &\sim \mathcal{GP}(0, \kappa_{\eta, g}(x, x')).\end{aligned}$$

Results in a **flexible** non-parametric model where the GP prior takes on the **role of a regularizer**.

We can now find the posterior distribution

$$p(f, g, Q, R, \eta \mid y_{1:T}),$$

via some approximation (we use **particle MCMC**).

---

Frigola, Roger, Fredrik Lindsten, Thomas B. Schön, and Carl Rasmussen. **Bayesian inference and learning in Gaussian process state-space models with particle MCMC**. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

Andreas Svensson and Thomas B. Schön. **A flexible state space model for learning nonlinear dynamical systems**, *Automatica*, 80:189-199, June, 2017.

# Snapshot 3 – The ASSEMBLE project and Birch

**Aim:** Automate probabilistic modeling of dynamical systems (and their surroundings) via a formally defined **probabilistic modeling language**.



SWEDISH FOUNDATION for  
STRATEGIC RESEARCH

---

Keep the model and the learning algorithms **separated**.

Create a **market place** for SMC-based learning algorithms (think CVX).

**Birch** — Our prototype probabilistic programming language.

Lawrence M. Murray, Daniel Lundén, Jan Kudlicka, David Broman and Thomas B. Schön. **Delayed sampling and automatic Rao-Blackwellization of probabilistic programs**. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, Lanzarote, Spain, April, 2018.

# Birch - our prototype probabilistic programming language

1. The basic idea of **probabilistic programming** is to equate probabilistic models with the programs that implement them.
  2. Just as we can think of doing inference over models, we can think of doing **inference over programs**.
- 

The particular PPL used here is **Birch**, which is currently being developed at Uppsala University.

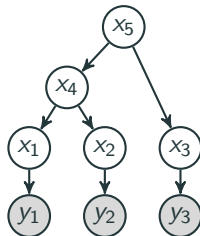
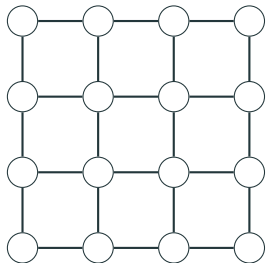
Probabilistic and object-oriented language.

An early pre-release of Birch is available

[birch-lang.org](http://birch-lang.org)

## Snapshot 4 – The nonlinear SSM is just a special case...

Constructing an artificial sequence of intermediate target distributions for an SMC sampler is a powerful (**quite possibly underutilized**) idea.



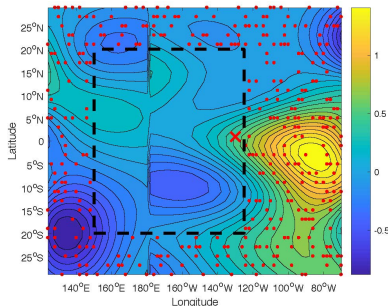
Christian A. Naesseth, Fredrik Lindsten and Thomas B. Schön, **Sequential Monte Carlo methods for graphical models**. *Advances in Neural Information Processing Systems (NIPS) 27*, Montreal, Canada, December, 2014.

Fredrik Lindsten, Adam M. Johansen, Christian A. Naesseth, Bonnie Kirkpatrick, Thomas B. Schön, John Aston and Alexandre Bouchard-Côté. **Divide-and-Conquer with Sequential Monte Carlo**. *Journal of Computational and Graphical Statistics (JCGS)*, 2017.

# Snapshot 5 – Spatio-temporal modelling

**Problem:** predicting **spatio-temporal** processes with temporal patterns varying across spatial regions when data is obtained as a **stream**.

A **localized** spatio-temporal covariance model.



The predictor can be **updated sequentially** with each new data point.

---

Muhammad Osama, Dave Zachariah and Thomas B. Schön. **Learning localized spatio-temporal models from streaming data.** In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, Stockholm, Sweden, July, 2018.

# Conclusion

Probabilistic modelling of nonlinear dynamical systems

$$p(x_{0:T}, \theta, y_{1:T}) = \underbrace{\prod_{t=1}^T \underbrace{p(y_t | x_t, \theta)}_{\text{observation}}}_{\text{data distribution}} \underbrace{\prod_{t=1}^T \underbrace{p(x_t | x_{t-1}, \theta)}_{\text{dynamics}} \underbrace{p(x_0 | \theta)}_{\text{state}} \underbrace{p(\theta)}_{\text{param.}}}_{\text{prior}}$$

SMC provide approximate solutions to **integration** problems where there is a **sequential structure** present.

Stochastic optimization:

- Non-standard interpretation of quasi-Newton.
- Represent the Hessian using a Gaussian process.
- We can scale up to larg(er) problems.