UPPSALA
UNIVERSITET

GP-based probabilistic modelling of dynamical systems

Thomas Schön, Uppsala University

The University of Sheffield, Gaussian Process Summer School,
September 4, 2018.

What we do in the team — who we are

We automate the extraction of knowledge and understanding from data.

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



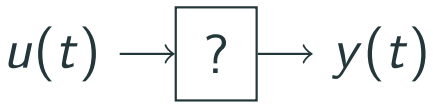

Create **probabilistic models** of dynamical systems and their surroundings.

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.

What is a dynamical system?

A dynamical system **evolves over time** and it has a **memory**.



Ex. 1 Linear time-invariant dynamical system described by

$$y(t) = \int_0^\infty g(\tau)u(t-\tau)d\tau + e(t).$$

Ex. 2 Nonlinear autoregressive model with exogenous (NARX) inputs

$$y_t = \varphi(y_{t-1}, \ldots, y_{t-n_y}, u_t, \ldots, u_{t-n_u}) + e_t.$$

Ex. 3 State space model (using **latent** variables x_t)

$$x_{t+1} = f(x_t, u_t, \theta) + v_t,$$
$$y_t = g(x_t, u_t, \theta) + e_t.$$

Ex) "What are x_t, θ and y_t"?

Aim (motion capture): Compute x_t (position and orientation of the different body segments) of a person (θ 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.

Key lesson from contemporary Machine Learning

Flexible models often give the best performance.

How can we build and work with these flexible models?

1. Models that use a large (but fixed) number of parameters. (**parametric**, ex. deep learning)

   LeCun, Y., Bengio, Y., and Hinton, G. **Deep learning**, *Nature*, Vol 521, 436–444, 2015.

2. Models that use more parameters as we get access to more data. (**non-parametric**, ex. Gaussian process)

   Ghahramani, Z. **Bayesian nonparametrics and the probabilistic approach to modeling**. *Phil. Trans. R. Soc. A* 371, 2013.

   Ghahramani, Z. **Probabilistic machine learning and artificial intelligence**. *Nature* 521:452-459, 2015.

Today we will focus on using the **Gaussian process in modelling dynamical systems**.

What is a Gaussian process (GP)?

The Gaussian process is a **non-parametric** and **probabilistic** model of a nonlinear function.

- **Non-parametric** means that it does not rely on any particular parametric functional form to be postulated.
- **Probabilistic** means that it takes uncertainty into account in every aspect of the model.

Definition: (Gaussian Process, GP) A GP is a (potentially infinite) collection of random variables such that any finite subset of it is jointly distributed according to a Gaussian.

Aim and outline

Aim: Provide some useful answers to the following two questions:

1. **General:** How can we mathematically construct probabilistic models of dynamical systems?
2. **Specific:** How can the GP be used to model dynamical systems?

1. Introduction – What is a dynamical system?
2. **Linear dynamical systems**
   a) Impulse response estimation
   b) Autoregressive (AR)
   c) Linear state space model (SSM)
3. Nonlinear dynamical systems
   a) Nonlinear AR
   b) Nonlinear SSM (GP-SSM)
4. Snapshots of some ongoing research (if there is time)

Comment on 44 and μ.

Part 2 – Linear dynamical systems

μ: A fundamental concept from systems theory and control

A fundamental concept: The **impulse response** $g(\tau)$ provides knowledge about **everything** there is to know about a linear system.

$$u(t) \longrightarrow \boxed{?} \longrightarrow y(t) \qquad\qquad y(t) = \int_0^\infty g(\tau)u(t-\tau)d\tau.$$

The impulse response of a dynamical system is its **output** when presented with an "impulse" input signal.

This **impulse** (the Dirac delta function) models the density of an idealized point mass as a function equal to zero everywhere except for zero and whose integral over the entire real line is equal to one.

The impulse function **contains all frequencies**, which means that the impulse response defines the response of a linear time-invariant system for all frequencies.

GP-based linear impulse response estimation

Consider a linear time-invariant dynamical system described by

$$y(t) = \int_0^\infty g(\tau)u(t-\tau)d\tau + e(t).$$

Task: Learn a model of the true underlying impulse response $g(\tau)$.

Placing a GP prior on the impulse response offers better performance than the "classical" system identification approach.

Gianluigi Pillonetto and Giuseppe De Nicolao. **A new kernel-based approach for linear system identification**. *Automatica*, 46(1):81–93, 2010.

Gianluigi Pillonetto, Francesco Dinuzzo, Tianshi Chen, Giuseppe De Nicolao and Lennart Ljung. **Kernel methods in system identification, machine learning and function estimation: A survey**. *Automatica*, 50(3):657–682, 2014.

Note that the integral of a GP is also a GP, so this is rather natural.

The GP offers a **data-driven model flexibility tuning**, an automatic **regularization** striking a bias-variance trade-off that is "just right".

μ: Two common parametric representations

1. One of the simplest parametric rep. is provided by the AR(X) model.

$$y_t = \varphi(y_{t-1}, \ldots, y_{t-n}, u_t, \ldots, u_{t-n_u}) + e_t.$$

These "classic" parametric approaches and the GP-based impulse response approach are linked via a **decision-theoretic** formulation.

Johan Wågberg, Dave Zachariah and TS. **Regularized parametric system identification: a decision-theoretic formulation**. In *Proceedings of the American Control Conference (ACC)*, Milwaukee, WI, USA, June, 2018.

---

2. Another very useful parametric representation is offered by introducing **latent variables**, which results in the so-called **state space model**.

μ: Bayesian autoregressive model

An autoregressive model of order n is given by

$$\begin{aligned} y_t &= a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_n y_{t-n} + e_t, \quad e_t \sim \mathcal{N}(\mu, \tau^{-1}) \\ &= \underbrace{\theta^{\mathsf{T}} z_t}_{f(z_t)} + e_t, \end{aligned}$$

where μ and τ are known explanatory variables ($\mu = 0, \tau \neq 0$).

The unknown model variables are collected as

$$\theta = (a_1, a_2, \ldots, a_n)^{\mathsf{T}}$$

with the prior

$$\theta \sim \mathcal{N}(0, \rho^{-1} I_n), \qquad \text{where } \rho \text{ assumed to be known.}$$

Task: Compute the posterior $p(\theta \mid y_{1:T})$.

μ: Bayesian autoregressive model

Full probabilistic model $p(\theta, y_{1:T}) = p(y_{1:T} \mid \theta)p(\theta)$, where the data distribution is given by

$$p(y_{1:T} \mid \theta) = p(y_T \mid y_{1:T-1}, \theta)p(y_{1:T-1} \mid \theta) = \cdots = \prod_{t=1}^{T} p(y_t \mid y_{1:t-1}, \theta).$$

From the model we have that

$$p(y_t \mid y_{1:t-1}, \theta) = \mathcal{N}(y_t \mid \theta^{\mathsf{T}} z_t, \tau^{-1}),$$

where $z_t = (y_{t-1}, y_{t-2}, \ldots, y_{t-n})^{\mathsf{T}}$. Hence,

$$p(y_{1:T} \mid \theta) = \prod_{t=1}^{T} \mathcal{N}\left(y_t \mid \theta^{\mathsf{T}} z_t, \tau^{-1}\right) = \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{z}\theta, \tau^{-1} I_T\right),$$

where we have made use of $\boldsymbol{y} = (y_1, y_2, \ldots, y_T)^{\mathsf{T}}$ and $\boldsymbol{z} = (z_1, z_2, \ldots, z_T)^{\mathsf{T}}$.

μ: Bayesian autoregressive model

$$p(\theta, \boldsymbol{y}) = \underbrace{\mathcal{N}\left(\boldsymbol{y} \,\middle|\, \boldsymbol{z}\theta, \tau^{-1} I_T\right)}_{p(\boldsymbol{y} \mid \theta)} \underbrace{\mathcal{N}\left(\theta \,\middle|\, 0, \rho^{-1} I_n\right)}_{p(\theta)}$$

$$= \mathcal{N}\left(\begin{pmatrix} \theta \\ \boldsymbol{y} \end{pmatrix} \middle| \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \rho^{-1} I_2 & \rho^{-1} \boldsymbol{z}^{\mathsf{T}} \\ \rho^{-1} \boldsymbol{z} & \tau^{-1} I_T + \rho^{-1} \boldsymbol{z}\boldsymbol{z}^{\mathsf{T}} \end{pmatrix} \right).$$

The posterior is given by

$$p(\theta \mid \boldsymbol{y}) = \mathcal{N}(\theta \mid m_T, S_T),$$

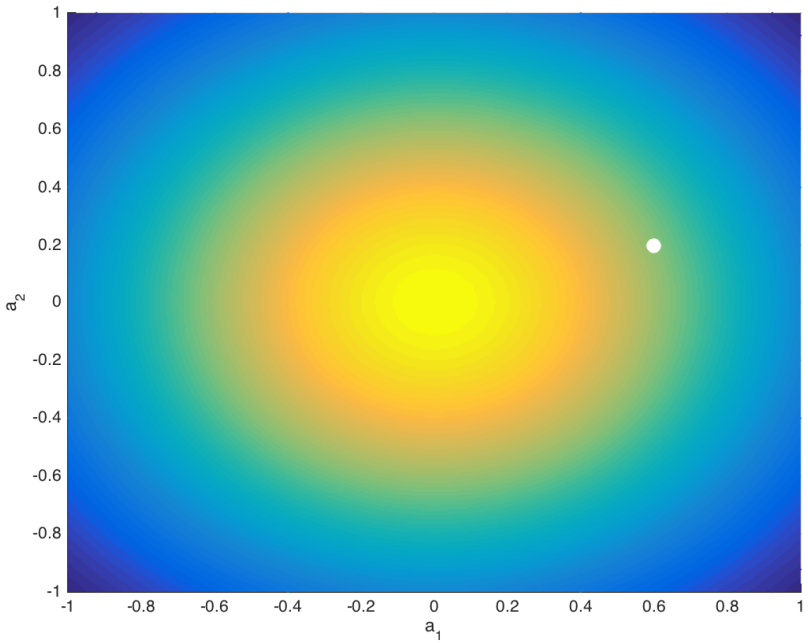
where

$$m_T = \tau S_T \boldsymbol{z}^{\mathsf{T}} \boldsymbol{y},$$

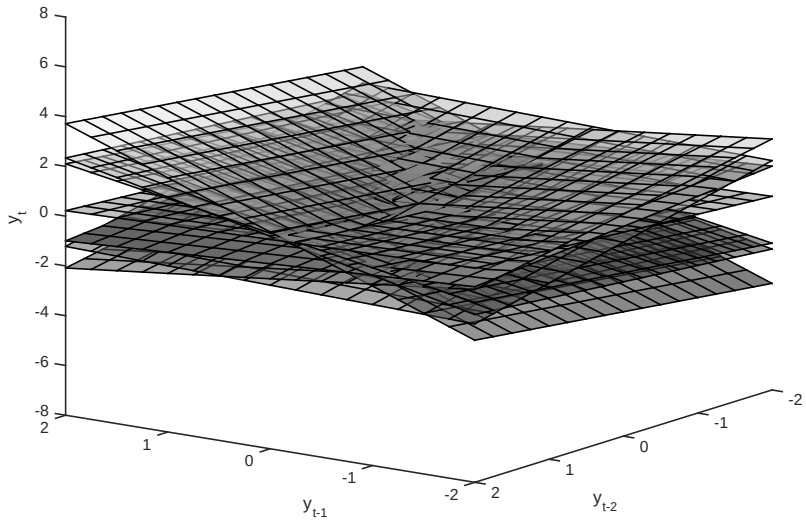
$$S_T = \left(\rho^{-1} I_2 + \sigma \boldsymbol{z}^{\mathsf{T}} \boldsymbol{z}\right)^{\mathsf{T}}.$$

μ: Ex) Situation before any data is used

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + e_t, \qquad e_t \sim \mathcal{N}(0, 0.2).$$



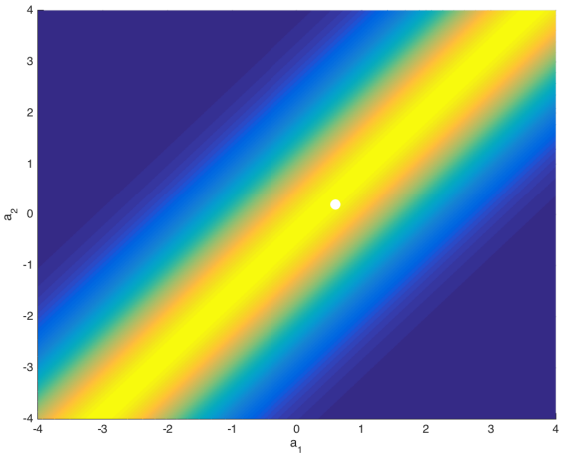
Prior



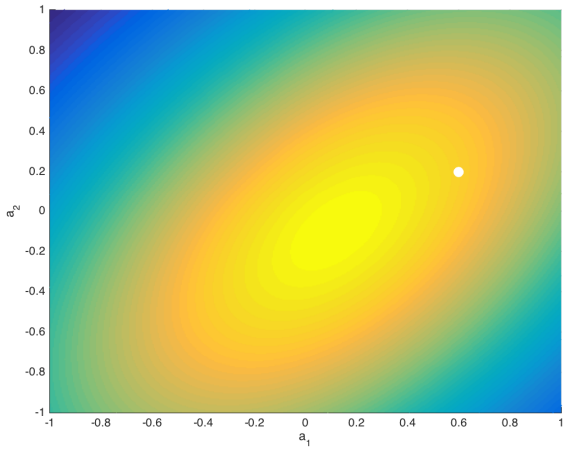
7 samples from the prior

White dot – true value for $\theta = (0.6, 0.2)$.

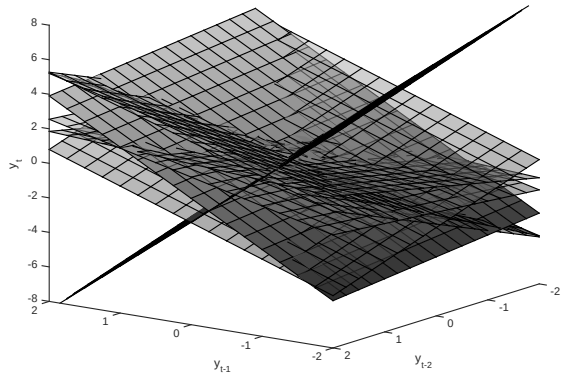
μ: Ex) Situation after y_1 is obtained



Likelihood

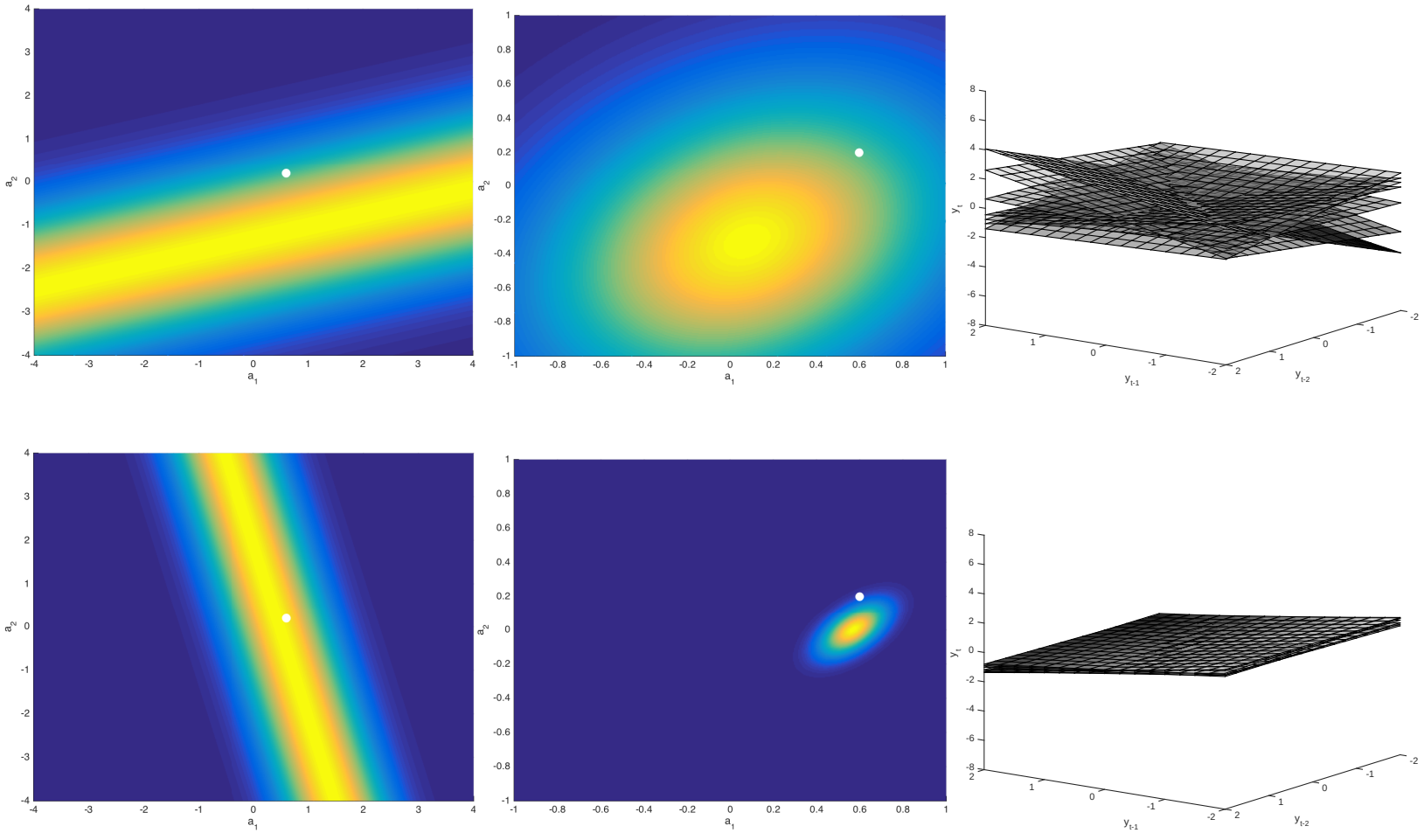


Posterior



7 samples from the posterior

μ: Ex) Situation after $y_{1:2}$ and $y_{1:20}$



Likelihood

Posterior

7 samples from the posterior

An abstract idea

In Bayesian linear regression

$$y_t = \underbrace{\theta^\mathsf{T} z_t}_{f(z_t)} + e_t, \qquad e_t \sim \mathcal{N}(0, \sigma^2),$$

we place a prior on θ, e.g. $\theta \sim \mathcal{N}(0, \alpha^2 I)$.

(Abstract) idea: What if we instead place a prior directly on the function $f(\cdot)$

$$f \sim p(f)$$

and look for $p(f \mid y_{1:T})$ rather than $p(\theta \mid y_{1:T})$?!

GP-AR model

An autoregressive model with exogenous (ARX) inputs

$$y_t = \varphi(y_{t-1}, \ldots, y_{t-n}, u_t, \ldots, u_{t-n_u}) + e_t.$$

Place a GP prior over φ with the following input

$$x_t = \begin{pmatrix} y_{t-1} & \ldots & y_{t-n} & u_t & \ldots & u_{t-n_u} \end{pmatrix}^{\mathsf{T}}$$

There is of course no reason to limit ourself to linear models when we are modelling φ using a GP. (a bit more about this later)

A. Girard, C. E. Rasmussen, J. Q. Candela, and R. Murray-Smith, **Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting**. in *Advances in neural information processing systems (NIPS)*, 2003.

J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith, **Dynamic systems identification 24 with Gaussian processes**. *Mathematical and Computer Modelling of Dynamical Systems*, 11(4):411–424, 2005.

μ: Latent variable model

Model variables that are not observed are called **latent** (a.k.a. hidden, missing and unobserved) variables.

The idea of introducing latent variables into models is probably one of the most **powerful concepts** in probabilistic modelling.

Latent variables provide **more expressive** models that can capture **hidden structures** in data that would otherwise not be possible.

Cost: Learning the model often becomes (significantly) harder.

Standard use within dynamical systems: **State space models**.

μ: Markov chain

The Markov chain is a probabilistic model that is used for modelling a sequence of states $(x_0, x_1, \ldots, x_T)$.

Definition (Markov chain)

A stochastic process $\{x_t\}_{t \geq 0}$ is referred to as a Markov chain if, for every $k > 0$ and t,

$$p(x_{t+k} \mid x_0, x_1, \ldots, x_t) = p(x_{t+k} \mid x_t).$$

A **Markov chain** is completely specified by:

1. An initial value x_0 and
2. a transition model (kernel) $\kappa(x_{t+1} \mid x_t)$ describing the transition from state x_t to state x_{t+1}, according to $x_{t+1} \mid (x_t = x_t) \sim \kappa(x_{t+1} \mid x_t)$.

μ: Markov chain

The **state** of the Markov chain acts as a **memory** containing all information there is to know about the phenomenon at a particular point in time.

Two important applications of Markov chains:

1. The Markov model is used in the **state space model (SSM)** where we can only observe the state indirectly via a measurement that is related to the state.
2. The Markov chain constitutes the basic ingredient in the **Markov chain Monte Carlo (MCMC)** methods.

Guess I have to mention a third application of the Markov chain as well:

3. Stochastic gradient methods

μ: Linear Gaussian state space model (LG-SSM)

The linear Gaussian state space model (LG-SSM) is given by

$$x_t = Ax_{t-1} + Bu_t + v_t,$$
$$y_t = Cx_t + Du_t + e_t,$$

where $x_t \in \mathbb{R}^{n_x}$ denotes the state, $u_t \in \mathbb{R}^{n_u}$ denotes an explanatory variable (known signal) and $y_t \in \mathbb{R}^{n_y}$ denotes the measurement (data).

The initial state and the noise are distributed according to

$$\begin{pmatrix} x_0 \\ v_t \\ e_t \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} P_0 & 0 & 0 \\ 0 & Q & S \\ 0 & S^{\mathsf{T}} & R \end{pmatrix} \right)$$

Remark: There is a connection between the SSM and the GP, where the GP can sometimes be reformulated as LG-SSMs. Opens up for linear complexity inference via the Kalman filter.

J. Hartikainen and S. Särkkä. **Kalman Filtering and Smoothing Solutions to Temporal Gaussian Process Regression Models**. *Proceedings of IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2010.

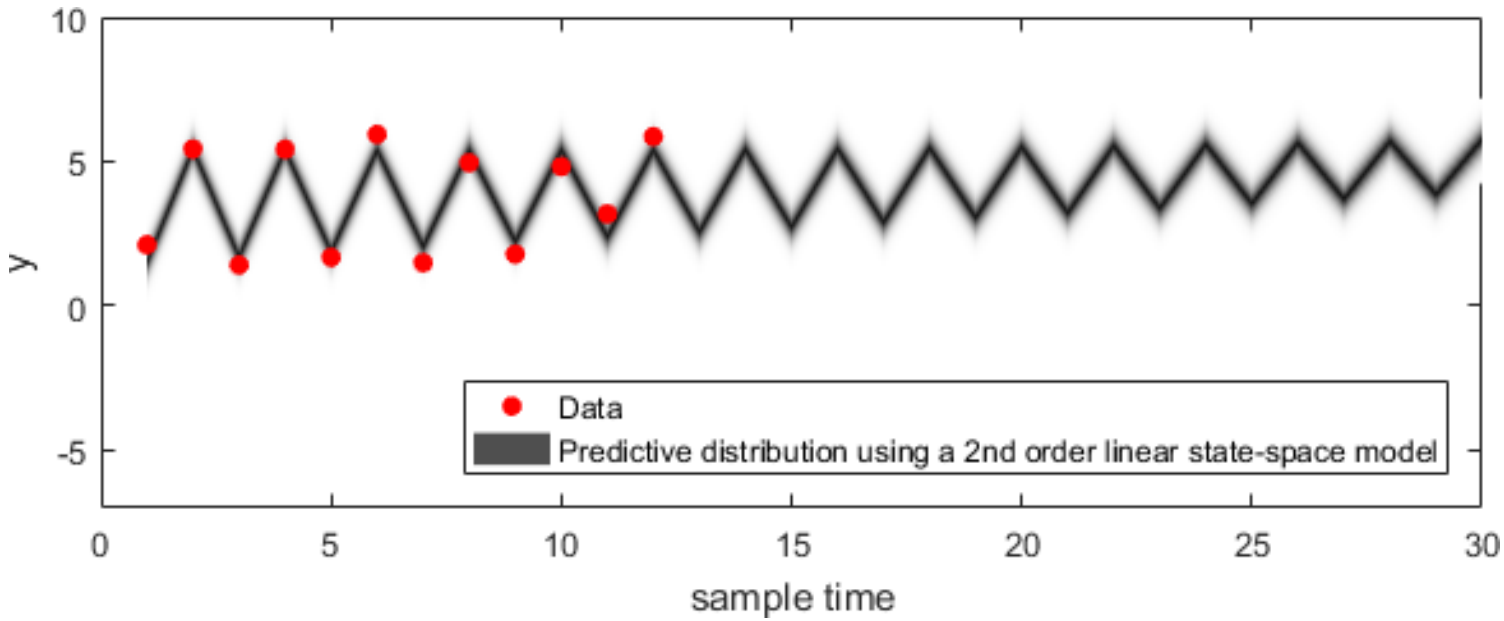
An attempt to illustrate why SSM + GP might make sense

Again, the linear Gaussian state space model

$$x_{t+1} = Ax_t + v_t,$$
$$y_t = Cx_t + e_t.$$

is designed to model dynamical behaviour.

However, it is limited in its expressiveness and uncertainty modelling.

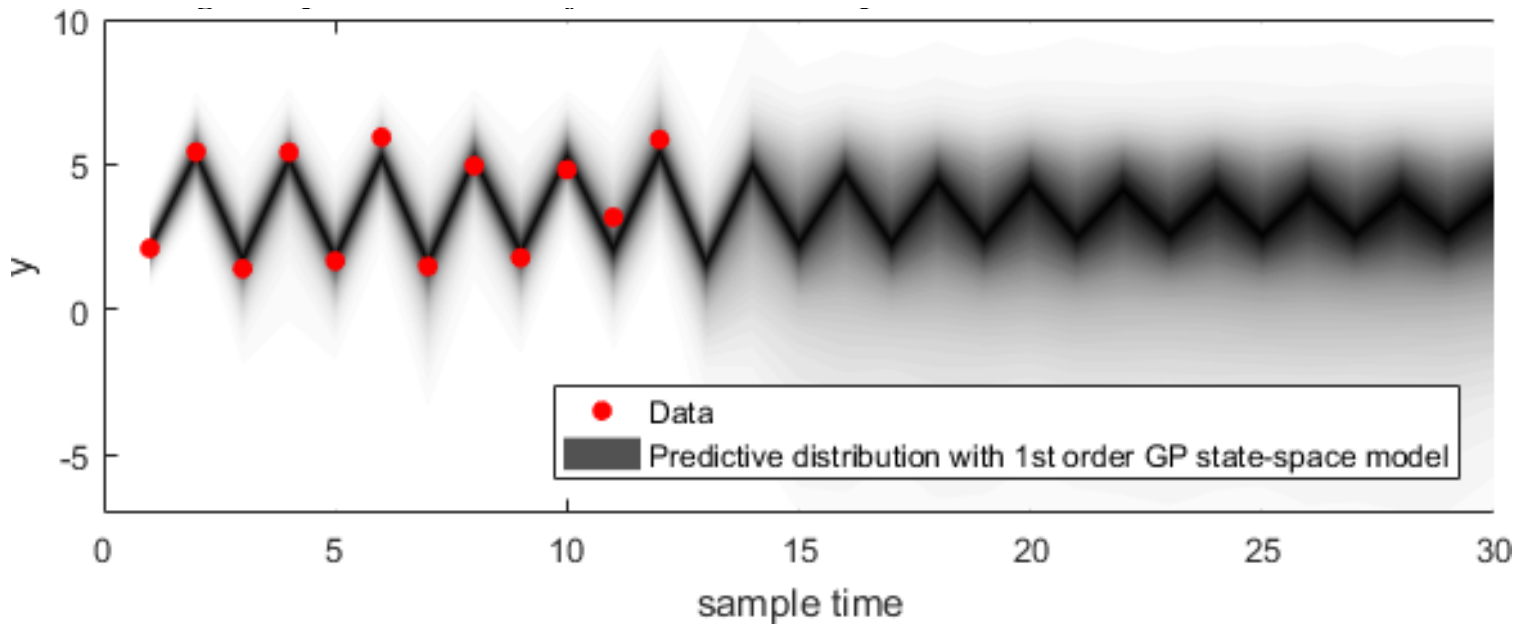


An attempt to illustrate why SSM + GP might make sense

The Gaussian process state space model (GP-SSM)

$$x_{t+1} = f(x_t) + v_t, \qquad \text{where} \quad f \sim \mathcal{GP},$$
$$y_t = g(x_t) + e_t, \qquad \text{where} \quad g \sim \mathcal{GP}.$$

combines the non-parametric flexibility and the uncertainty representation of the GP with the dynamical modeling capabilities of the SSM.



Aim and outline

Aim: Provide some useful answers to the following two questions:

1. **General:** How can we mathematically construct probabilistic models of dynamical systems?
2. **Specific:** How can the GP be used to model dynamical systems?

1. Introduction – What is a dynamical system?
2. Linear dynamical systems
   a) Impulse response estimation
   b) Autoregressive (AR)
   c) Linear state space model (SSM)
3. **Nonlinear dynamical systems**
   a) Nonlinear AR model
   b) Nonlinear SSM (GP-SSM)
4. Snapshots of some ongoing research (if there is time)

Part 3 – Nonlinear dynamical systems

Nonlinear ARX model — GP style

An autoregressive model with exogenous (ARX) inputs

$$y_t = \varphi(y_{t-1}, \ldots, y_{t-n}, u_t, \ldots, u_{t-n_u}) + e_t.$$

Place a GP prior over φ with the following input

$$x_t = \begin{pmatrix} y_{t-1} & \ldots & y_{t-n} & u_t & \ldots & u_{t-n_u} \end{pmatrix}^{\mathsf{T}}$$

Challenges (standard challenges with the basic GP):

1. **Computationally** too expensive.
2. It cannot efficiently make use of new measurements **online**.
3. Cannot deal with **stochastic (noisy) inputs**.

As you have heard by now there are ways around all of these.

Hildo Bijl, TS, Jan-Willem van Wingerden and Michel Verhaegen. **System identification through online sparse Gaussian process regression with input noise**. *IFAC Journal of Systems and Control*, 2:1-11, 2017.

μ: 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.

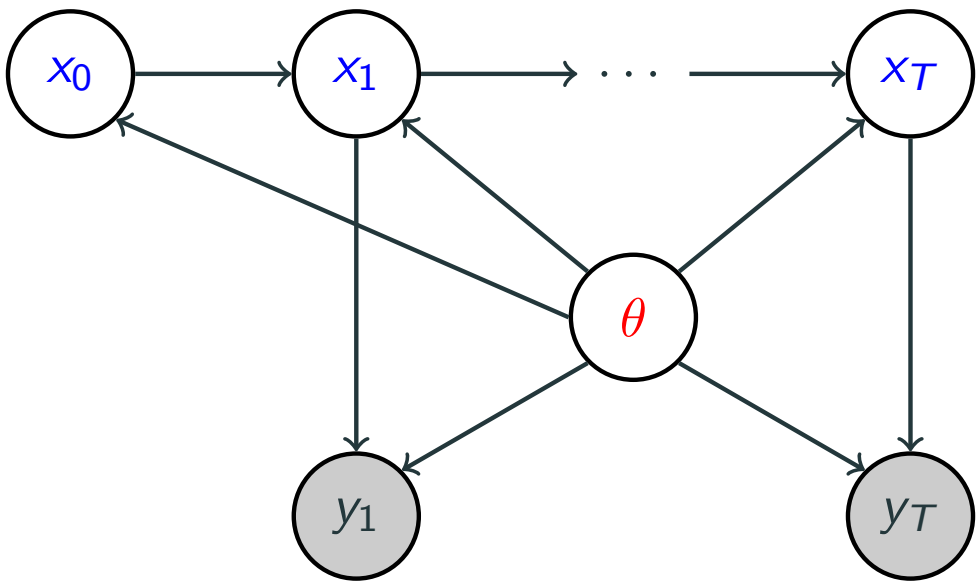
μ: Three different representations of the SSM

Three alternative representations, using

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

---

1. Representing the SSM using a graphical model:



μ: 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);
y[1] ~ Gaussian(x[1], 1.0);
for (t in 2..T) {
  x[t] ~ Gaussian(a*x[t - 1], 1.0);
  y[t] ~ Gaussian(x[t], 1.0);
}
```

$p(x_1)$
$p(y_1 \mid x_1)$

$p(x_t \mid x_{t-1})$
$p(y_t \mid x_t)$

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

μ: Parametric SSM – full probabilistic model

The **full probabilistic model** is given by

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

Distribution describing a parametric nonlinear SSM

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

Model = probability distribution!

μ: Finding the states and the parameters

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

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

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

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

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

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

μ: Central object – the likelihood

The likelihood is computed by marginalizing

$$p(x_{0:T}, y_{1:T} \mid \theta) = p(x_0 \mid \theta) \prod_{t=1}^{T} p(y_t \mid x_t, \theta) \prod_{t=1}^{T} p(x_t \mid x_{t-1}, \theta),$$

w.r.t the state sequence $x_{0:T}$,

$$p(y_{1:T} \mid \theta) = \int p(x_{0:T}, y_{1:T} \mid \theta) \mathrm{d}x_{0:T}.$$

(We are averaging $p(x_{0:T}, y_{1:T} \mid \theta)$ over all possible state sequences.)

Equivalently we have

$$p(y_{1:T} \mid \theta) = \prod_{t=1}^{T} p(y_t \mid y_{1:t-1}, \theta) = \prod_{t=1}^{T} \int p(y_t \mid x_t, \theta) \underbrace{p(x_t \mid y_{1:t-1}, \theta)}_{\text{key challenge}} \mathrm{d}x_t.$$

TS, Fredrik Lindsten, Johan Dahlin, Johan Wågberg, Christian A. Naesseth, Andreas Svensson and Liang Dai. **Sequential Monte Carlo methods for system identification.** In *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*, Beijing, China, October 2015.

μ: The model – learning relationship

Learning a model based on data leads to computational challenges:

- **Integration:** e.g. the HD integrals arising during marg. (averaging over all possible parameter values z):

$$p(y_{1:T}) = \int p(y_{1:T} \mid z) p(z) \mathrm{d}z.$$

- **Optimization:** e.g. when extracting point estimates, for example by maximizing the likelihood

$$\widehat{z} = \arg\max_{z} \; p(y_{1:T} \mid z)$$

Impossible to compute exactly, approximations are needed:

- Monte Carlo (MC), Markov chain MC, and sequential MC.
- Variational inference (VI).
- Stochastic optimization.

The idea underlying a non-parametric SSM via the GP

$$x_{t+1} = f(x_t) + v_t, \qquad \text{s.t.} \quad f(x) \sim \mathcal{GP}(0, \kappa_{\eta,f}(x, x')),$$
$$y_t = g(x_t) + e_t, \qquad \text{s.t.} \quad g(x) \sim \mathcal{GP}(0, \kappa_{\eta,g}(x, x')).$$

Results in a **flexible** non-parametric model where the GP prior takes on the **role of a regularizer**. Enables regularization also in nonlinear models.

Challenge: Approximate the posterior distribution

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

Mauricio A. Alvarez, David Luengo and Neil D. Lawrence. **Latent force models**. *In Artificial Intelligence and Statistics (AISTATS)*, 2009.

Frigola, Roger, Fredrik Lindsten, TS, 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.

Roger Frigola, Yutian Chen, and Carl E. Rasmussen. **Variational Gaussian process state-space models**. *In Advances in Neural Information Processing Systems (NIPS)*, 2014.

Stefanos Eleftheriadis, Thomas F. W. Nicholson, Marc P. Deisenroth, James Hensman. **Identification of Gaussian process state space models**, *Advances in Neural Information Processing Systems (NIPS)*, 2017

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

Approximate Gaussian processes

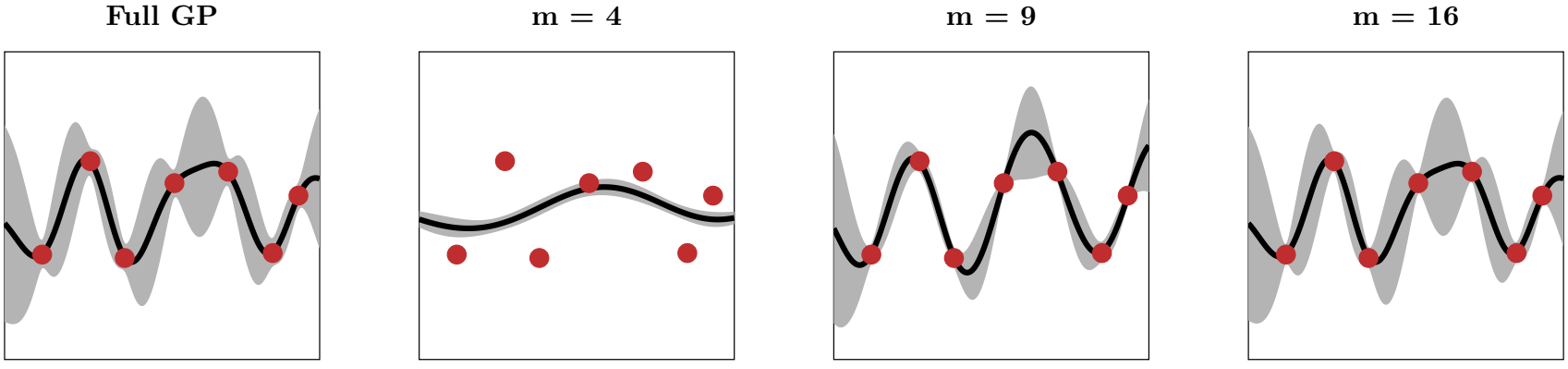
We use a "reduced-rank" GP approximation:

$$f \sim \mathcal{GP}(0, k) \quad \Leftrightarrow \quad f(x) \approx \sum_{j=0}^{m} w^j \phi^j(x)$$

with prior

$$w^j \sim \mathcal{N}(0, S(\lambda^j))$$

For $x \in [-L, L] \subset \mathbb{R}$: $\phi^j(x) = \frac{1}{\sqrt{L}} \sin\left(\frac{\pi j(x+L)}{2L}\right)$.



Arno Solin and Simo Särkkä. **Hilbert Space Methods for Reduced-Rank Gaussian Process Regression.** *arXiv:1401.5508*, 2014.

Computationally feasible GP-SSM

Original formulation:

$$x_{t+1} = f(x_t) + v_t, \qquad v_t \sim \mathcal{N}(0, Q),$$
$$y_t = g(x_t) + e_t, \qquad e_t \sim \mathcal{N}(0, R),$$
$$f(x) \sim \mathcal{GP}(0, \kappa_{\eta,f}(x, x'))$$

Formulation using the reduced-rank GP approximation:

$$x_{t+1} = \sum_{j=0}^{m} w^j \phi^j(x_t) + v_t, \qquad v_t \sim \mathcal{N}(0, Q),$$
$$y_t = g(x_t) + e_t, \qquad e_t \sim \mathcal{N}(0, R),$$
$$w^j \sim \mathcal{N}(0, S(\lambda^j)).$$

Linear in the parameters w^i and nonlinear in the states x_t.

The learning problem (dynamical systems)

Compute the posterior distribution

$$p(x_{1:T}, \theta \mid y_{1:T}) = \underbrace{p(x_{1:T} \mid \theta, y_{1:T})}_{\text{state}} \underbrace{p(\theta \mid y_{1:T})}_{\text{parameter}}.$$

HD integration/optimization problems without analytical solution.

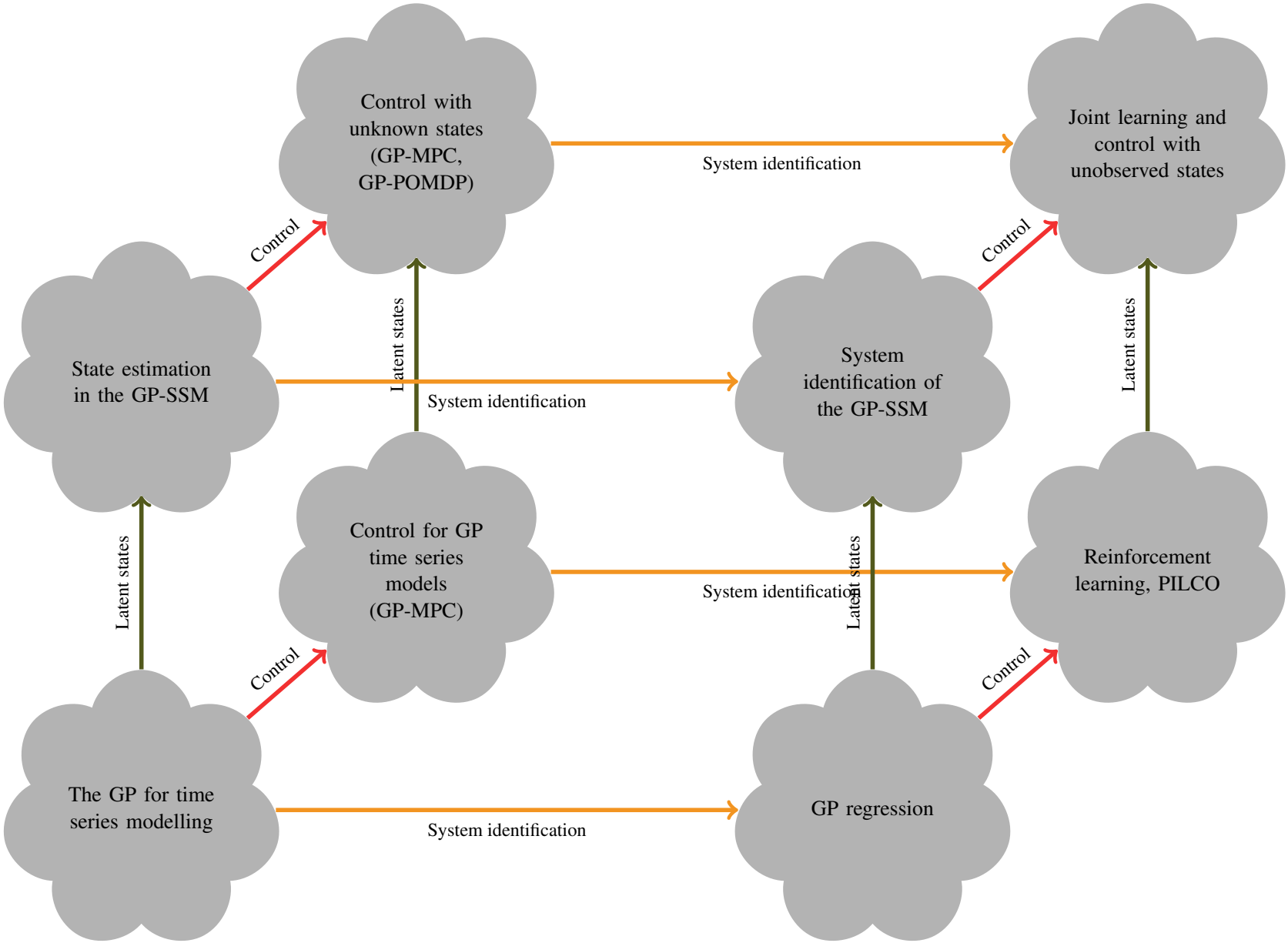
Sequential Monte Carlo provide approximations to integration problems where there is a **sequential structure** present.

Learning the parameters θ is rather straightforward in this GP-SSM.

The states $x_{1:T}$ are still challenging. We use a combination of SMC and MCMC.

Andreas Svensson and TS. **A flexible state space model for learning nonlinear dynamical systems**, *Automatica*, 80:189–199, 2017.

GP-SSM — A Zoubin cube to describe it



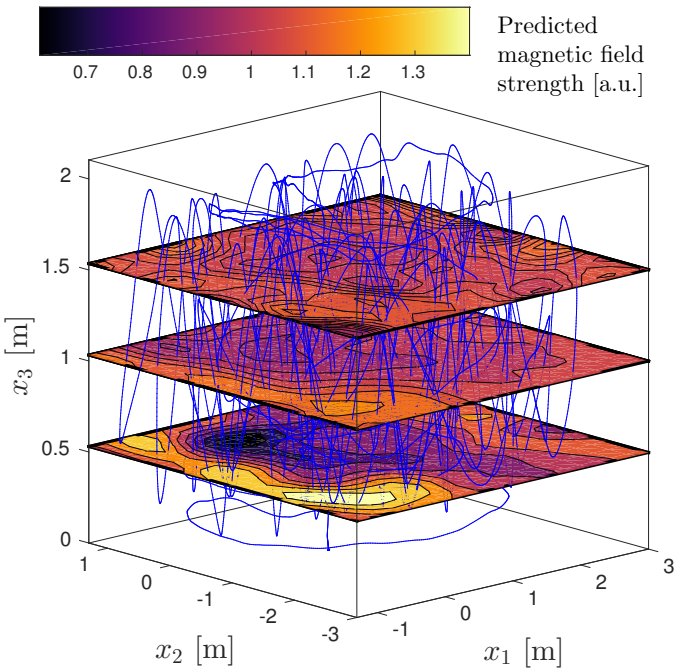
Part 4 – Snapshots of some ongoing research and message

Snapshot 1 — A linearly constrained GP

Innovation: Modification of the covariance function in a GP to correctly account for **known linear operator** constraints.

Contribution:

1. A probabilistic model that is **guaranteed** to fulfil known linear operator constraints.
2. A **constructive procedure** for designing the transformation.



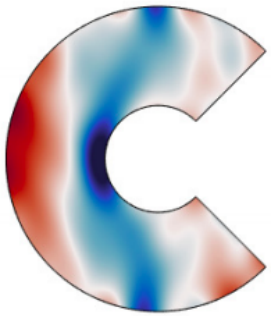
Carl Jidling, Niklas Wahlström, Adrian Wills and TS. **Linearly constrained Gaussian processes**. *Advances in Neural Information Processing Systems (NIPS)*, Long Beach, CA, USA, December, 2017.

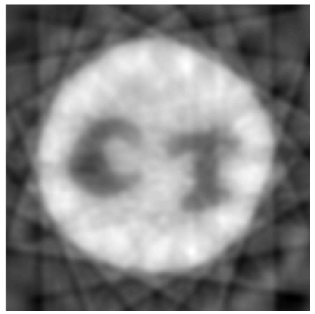
Snapshot 2 — Constrained GP for tomographic reconstruction

Tomographic reconstruction goal: Build a map of an unknown quantity within an object using information from irradiation experiments.

Ex1) Modelling and reconstruction of strain fields.

Ex2) Reconstructing the internal structure from limited x-ray projections.





Carl Jidling, Johannes Hendriks, Niklas Wahlström, Alexander Gregg, TS, Chris Wensrich and Adrian Wills. **Probabilistic modelling and reconstruction of strain**. *Nuclear inst. and methods in physics research: section B*, 2018. (to appear)

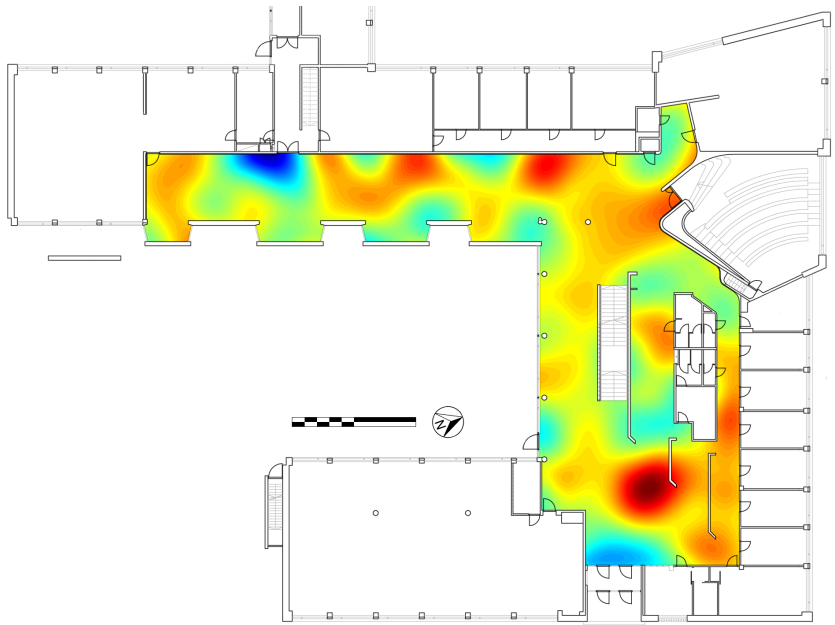
Zenith Purisha, Carl Jidling, Niklas Wahlström, Simo Särkkä and TS. **Probabilistic approach to limited-data computed tomography reconstruction**. *Draft*, 2018

Snapshot 3 — Model of the ambient magnetic field with GPs

The Earth's magnetic field sets a background for the ambient magnetic field. Deviations make the field vary from point to point.

Aim: Build a map (i.e., a model) of the magnetic environment based on magnetometer measurements.

Solution: Customized Gaussian process that obeys Maxwell's equations.



www.youtube.com/watch?v=enlMiUqPVJo

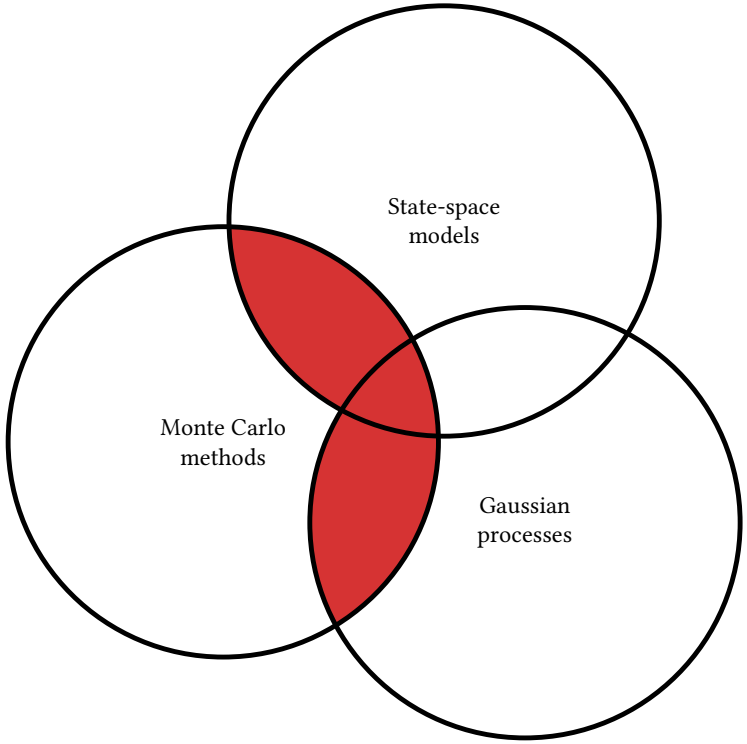
Arno Solin, Manon Kok, Niklas Wahlström, TS and Simo Särkkä. **Modeling and interpolation of the ambient magnetic field by Gaussian processes**. *IEEE Transactions on Robotics*, 34(4):1112–1127, 2018.

Carl Jidling, Niklas Wahlström, Adrian Wills and TS. **Linearly constrained Gaussian processes**. *Advances in Neural Information Processing Systems (NIPS)*, Long Beach, CA, USA, December, 2017.

Snapshot 4 — New PhD thesis on the topic

Machine learning with state-space models, Gaussian processes and Monte Carlo methods.

By Andreas Svensson



Link to the thesis:

www.it.uu.se/katalog/andsv164/main/thesis_andreas_svensson_webb.pdf

Snapshot 5 — Use the GP to learn Hessians

Results in a **stochastic quasi-Newton** method.

Summary: Stochastic quasi-Newton integral:

$$y_k = D_k \int_0^1 \widetilde{B}(r_k(\tau))\mathrm{d}\tau + e_k,$$

with the following model for the Hessian

$$\widetilde{B}(\theta) \sim \mathcal{GP}(\mu(\theta), \kappa(\theta, \theta')).$$

Talk more about this on Thursday.

My message (I/II)

Message: The Gaussian process can be used to construct **useful representations** of dynamical systems.

I have hinted at how we can combine standard dynamical models like

1. Linear – impulse response
2. Nonlinear – Autoregressive (AR)
3. Nonlinear – State-space model (SSM)

and the Gaussian process to achieve useful constructions.

"If you have a new nonlinear construction, **always** *make sure that it "does the right thing" in the linear Gaussian special case."*

My message (II/II)

System identification

the Gaussian process

Remember to talk to people who work on **different problems** with **different tools**!!