UPPSALA
UNIVERSITET

Constructing probabilistic Newton-type algorithms

"bringing curvature information into stochastic optimization"

Thomas Schön, Uppsala University.

Workshop – Learning from a control perspective, Linköping University.
October 23, 2017.

Joint work with Adrian Wills at the University of Newcastle, Australia.

Mindset — Numerical methods are inference algorithms

A numerical method **estimates** a certain **latent** property **given** the result of computations.

Computation is inference meaning that numerical methods can be interpreted as estimation rules.

Basic numerical methods and basic statistical models are **deeply connected in a formal way**!

Poincaré, H. **Calcul des probabilités**. Paris: Gauthier-Villars, 1896.

Diaconis, P. **Bayesian numerical analysis**. *Statistical decision theory and related topics*, IV(1), 163–175, 1988.

O'Hagan, A. **Some Bayesian numerical analysis**. *Bayesian Statistics*, 4, 345–363, 1992.

Hennig, P., Osborne, M. A., and Girolami, M. **Probabilistic numerics and uncertainty in computations**. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 471(2179), 2015.

probabilistic-numerics.org/

Mindset — Numerical methods are inference algorithms

The task of a numerical algorithm is

to estimate unknown quantities from known ones.

Ex) basic algorithms that are equivalent to Gaussian MAP inference:

- Conjugate Gradients for linear algebra
- BFGS for nonlinear optimization
- Gaussian quadrature rules for Integration
- Runge-Kutta solvers for ODEs

The structure of num. algs. is similar to statistical inference where

- The tractable quantities play the role of "data"/"observations".
- The intractable quantities relate to "latent"/"hidden" quantities.

Problem formulation

If computation is inference maybe it is possible to use this in deriving new (and possibly more capable) algorithms.

What? Solve the nonlinear stochastic optimization problem

$$\widehat{x} = \arg\max_{x} f(x)$$

when we only have access to **noisy** evaluations of the cost function and its derivatives.

Why? These stochastic optimization problems are common:

- The cost function cannot be evaluated on the entire dataset.
- When numerical methods approximate $f(x)$ and $\nabla^i f(x)$.
- The "reparameterization trick" makes variational inference applicable to almost any differentiable model.
- ...

Outline

Aim: Derive a probabilistic quasi-Newton algorithm.

Spin-off: Combine it with particle filters for maximum likelihood identification in nonlinear state space models.

Outline:

1. Mindset (probabilistic numerics) and problem formulation
2. **A non-standard take on quasi-Newton**
3. μ on the Gaussian process (GP)
4. Assembling a new probabilistic optimization algorithm
   a. Integral formulation of the secant condition
   b. Modeling the Hessian with a GP
5. Testing ground – maximum likelihood in SSMs
6. Some ongoing research (if there is time)

Quasi-Newton — A non-standard take

Our problem is of the form

$$\widehat{x} = \arg\max_{x} f(x)$$

Idea underlying (quasi-)Newton methods: Learn a local quadratic model $q(x_k, \delta)$ of the cost function $f(x)$ around x_k

$$q(x_k, \delta) = f(x_k) + g(x_k)^\mathsf{T}\delta + \frac{1}{2}\delta^\mathsf{T} H(x_k)\delta$$

A second order Taylor expansion around x_k, where

$$g(x_k) = \nabla f(x)\big|_{x=x_k},$$
$$H(x_k) = \nabla^2 f(x)\big|_{x=x_k},$$
$$\delta = x - x_k.$$

Quasi-Newton model

Quasi-Newton methods compute an **estimate** of the Hessian based on the function values and their gradients.

Quasi-Newton model of the cost function:

$$f_q(x_k + \delta) = f(x_k) + g(x_k)^\mathsf{T}\delta + \delta^\mathsf{T} B_k \delta.$$

Note that

$$\nabla_\delta f_q(x_k + \delta) = g(x_k) + B_k \delta.$$

Assumption made by quasi-Newton methods:

$$\nabla_\delta f_q(x_k + \delta)\big|_{\delta = x_{k-1} - x_k} = \nabla_\delta f(x_k) = g(x_{k-1}).$$

Secant condition

The quasi-Newton **assumption**

$$\nabla_\delta f_q(x_k + \delta)\big|_{\delta = x_{k-1} - x_k} = \nabla_\delta f(x_k) = g(x_{k-1}),$$

combined with the gradient of the quasi-Newton **model**

$$\nabla_\delta f_q(x_k + \delta) = g(x_k) + B_k \delta,$$

results in $g(x_{k-1}) = g(x_k) + B_k(x_{k-1} - x_k)$. Let

$$y_k = g(x_k) - g(x_{k-1}),$$

$$s_k = x_k - x_{k-1}.$$

Result: the **secant condition** (quasi-Newton equation)

$$B_k s_k = y_k.$$

We still do not have enough information to define the elements of the Hessian approximation B_k. **Q:** What do you do?

Recovering existing quasi-Newton methods

The existing quasi-Newton methods (e.g. BFGS, DFP, Broyden's method) can all be interpreted as employing some particular form of **regularization** on B_k:

$$\begin{aligned} \min_{B} \quad & \|B - B_{k-1}\|_W^2, \\ \text{s.t.} \quad & B = B^\mathsf{T}, \\ & Bs_k = y_k, \end{aligned}$$

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

This opens up for some **flexibility** in finding new algorithms.
The **Gaussian process** is one way in which we can **use this flexibility**!

Philipp Hennig. **Probabilistic interpretation of linear solvers**, *SIAM Journal on Optimization*, 25(1):234–260, 2015.

μ on the Gaussian process (GP)

The Gaussian process is a model for nonlinear functions

Q: Why is the Gaussian process used everywhere?

It is a **non-parametric** and **probabilistic** model for nonlinear functions.

- **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.

An abstract idea

In probabilistic (Bayesian) linear regression

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

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

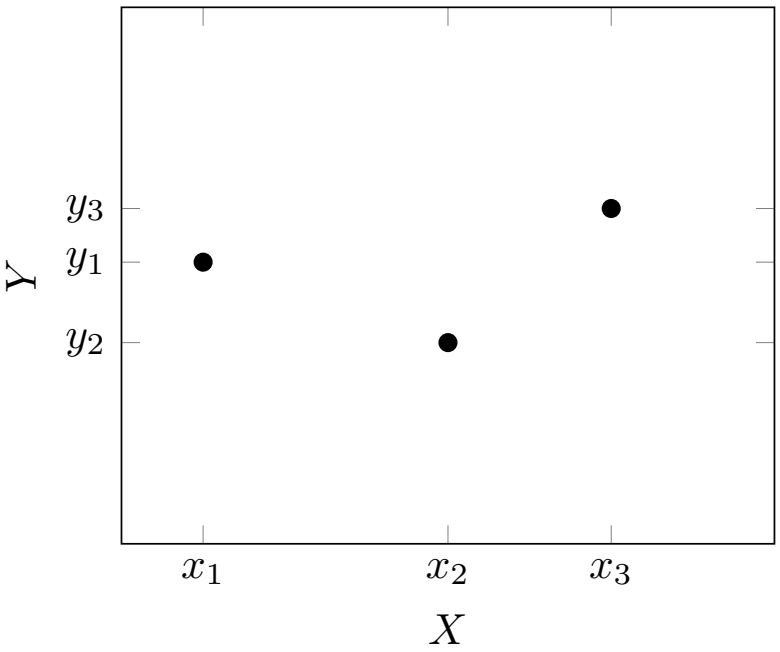
(Abstract) idea: What if we instead place a prior directly on the function $f(\cdot)$

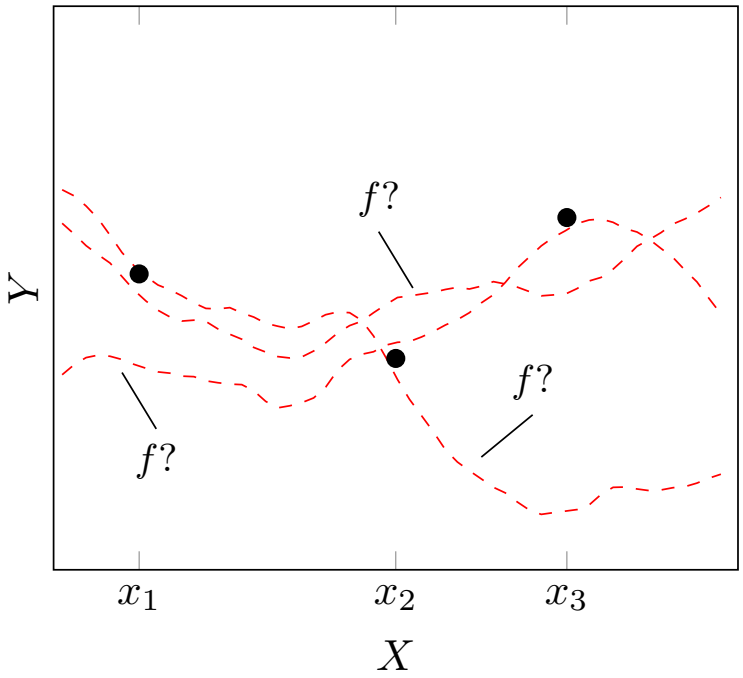
$$f \sim p(f)$$

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

An abstract idea – pictures

What does it actually mean to have a prior over functions?





Can we construct a probabilistic object operating on functions?

One concrete construction

Well, one (arguably simple) idea on how we can reason probabilistically about an unknown function f is by assuming that $f(x)$ and $f(x')$ are jointly Gaussian distributed

$$\begin{pmatrix} f(x) \\ f(x') \end{pmatrix} \sim \mathcal{N}(m, K).$$

If we accept the above idea we can without conceptual problems generalize to any *arbitrary* finite set of input values $\{x_1, x_2, \ldots, x_T\}$.

$$\begin{pmatrix} f(x_1) \\ \vdots \\ f(x_T) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(x_1) \\ \vdots \\ m(x_N) \end{pmatrix}, \begin{pmatrix} k(x_1, x_1) & \ldots & k(x_1, x_T) \\ \vdots & \ddots & \vdots \\ k(x_T, x_1) & \ldots & k(x_T, x_T) \end{pmatrix} \right)$$

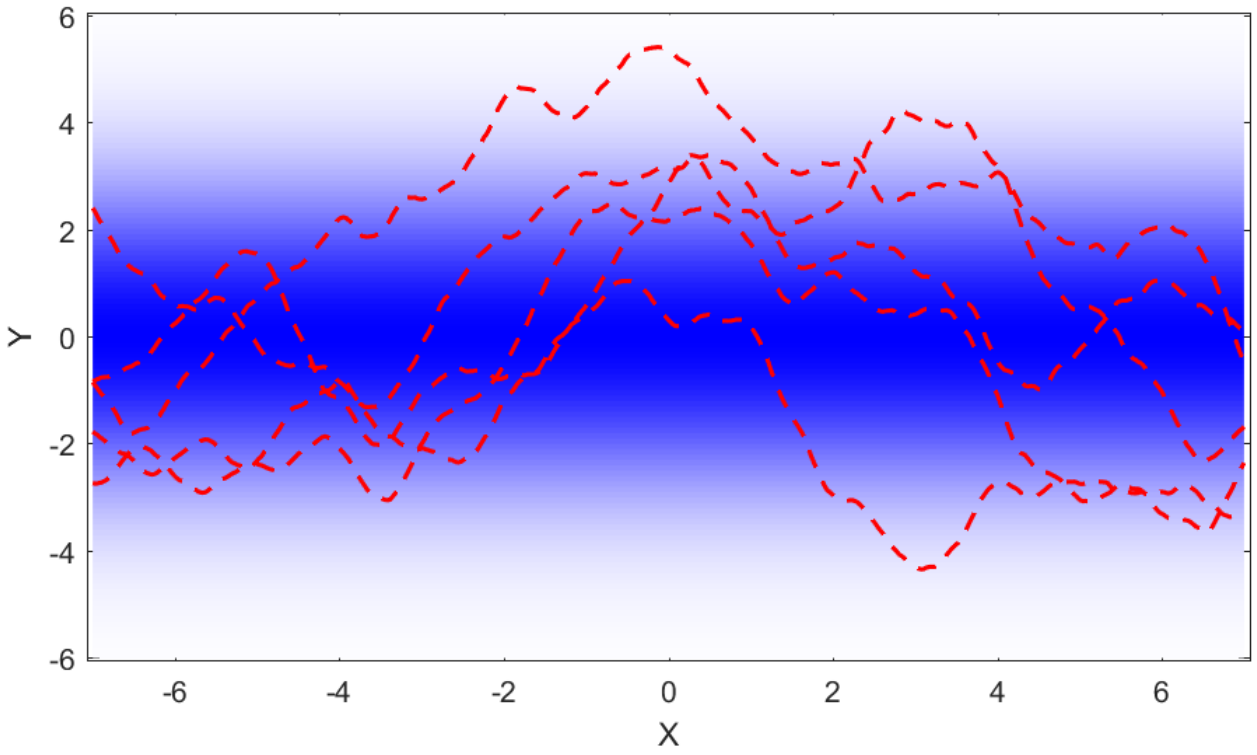
Definition

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.

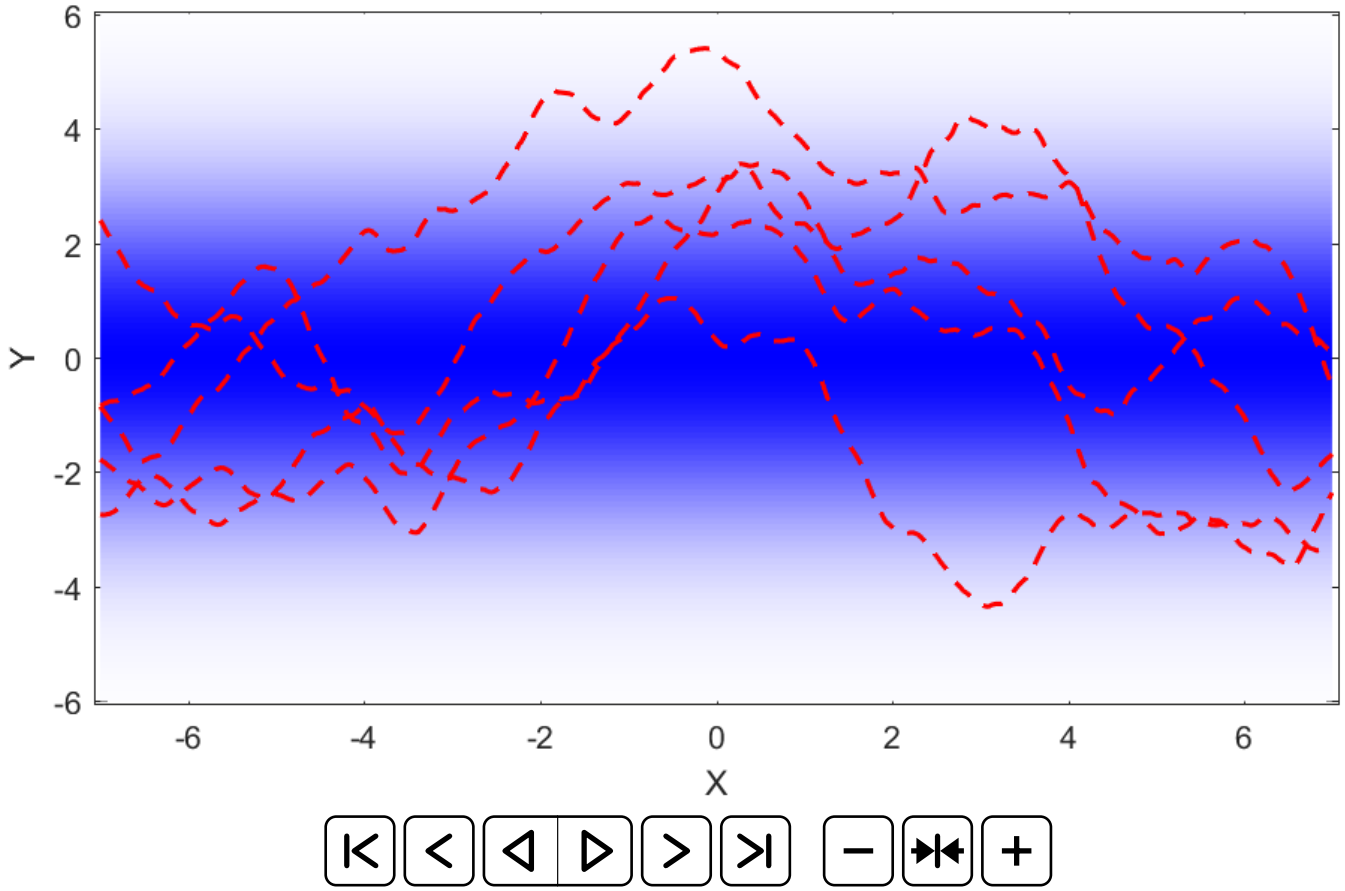
We now have a prior!

$$f \sim \mathcal{GP}(m, k)$$

The GP is a **generative** model so let us first sample from the prior.



GP regression – illustration



Probabilistic optimization

Probabilistic optimization algorithm

Two different ways of using the GP for optimization:

1. Model the cost function as a GP

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')),$$

and possibly the gradient and the Hessian as well.

2. Model the Hessian using a GP

$$\widetilde{B}(\tau) \sim \mathcal{GP}(m(\tau), k(\tau, \tau')),$$

The Hessian prior model is then updated using noisy observations of the cost function and its derivatives.

Integral formulation of the secant condition

Recall: the **secant condition** (quasi-Newton equation)

$$B_k \underbrace{(x_k - x_{k-1})}_{s_k} = \underbrace{g(x_k) - g(x_{k-1})}_{y_k}.$$

Reformulate using a line **integral** rather than gradients.

Let

$$r_k(\tau) = x_{k-1} + \tau(x_k - x_{k-1}), \qquad \tau \in [0, 1],$$

denote the line segment between x_{k-1} and x_k.

Note that

$$y_k = \underbrace{\nabla f(x_k)}_{g(x_k)} - \underbrace{\nabla f(x_{k-1})}_{g(x_{k-1})} = \nabla f(r_k(1)) - \nabla f(r_k(0)) = \int_0^1 \frac{\partial}{\partial \tau} \nabla f(r_k(\tau)) \mathrm{d}\tau.$$

Integral formulation of the secant condition

The chain rule tells us that

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

Therefore,

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

An **integral version** of the secant condition $B_k s_k = y_k$ is thus

$$y_k = \int_0^1 B(r_k(\tau))s_k \mathrm{d}\tau.$$

Using a GP for the Hessian

An interpretation of

$$y_k = \int_0^1 B(r_k(\tau))s_k \mathrm{d}\tau.$$

is that the difference between gradients (i.e. y_k) can be considered a line integral observation of the Hessian matrix $B(r_k(\tau))$.

Problem: Since the Hessian is unknown we do not have any functional form for it.

Idea: Use a GP to represent the Hessian and infer it 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?

GP prior for the Hessian

The gradient observations are typically **noisy** (e_k)

$$y_k = \int_0^1 B(r_k(\tau))s_k \mathrm{d}\tau + e_k.$$

Standard GP regression applies.

Assume a GP prior for the Hessian

$$\widetilde{B}(\tau) = \mathrm{vech}\left(B(r_k(\tau))\right),$$

that is given by

$$\widetilde{B} \sim \mathcal{GP}(m, k).$$

The Hessian can now be estimated using tailored GP regression.

Resulting probabilistic optimization algorithm

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

Algorithm 1 Probabilistic 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) Probabilistic line search to find a step length α_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**

Testing ground – nonlinear sys.id.

Probabilistic modelling of dynamical systems

$$
\begin{aligned}
x_t &= f(x_{t-1}, \theta) + w_t, \\
y_t &= g(x_t, \theta) + e_t, \\
x_0 &\sim p(x_0 \mid \theta), \\
(\theta &\sim p(\theta)).
\end{aligned}
\qquad
\begin{aligned}
x_t \mid (x_{t-1}, \theta) &\sim p(x_t \mid x_{t-1}, \theta), \\
y_t \mid (x_t, \theta) &\sim p(y_t \mid x_t, \theta), \\
x_0 &\sim p(x_0 \mid \theta), \\
(\theta &\sim p(\theta)).
\end{aligned}
$$

Corresponding full probabilistic model:

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

Model = probability distribution!

Nonlinear system identification

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

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

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

$$\underbrace{p(\theta \mid y_{1:T})}_{=\pi(\theta) \text{ target}} = \frac{p(y_{1:T} \mid \theta) p(\theta)}{p(y_{1:T})}.$$

Central object – data distribution/likelihood

The likelihood can be computed by marginalizing

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

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

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

Average over all possible values for the state trajectory $x_{0:T}$.

Alternative way of performing the averaging:

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

Using SMC for nonlinear system identification

Sequential Monte Carlo (SMC) can be used to approximately

1. Compute the likelihood and its derivatives.
2. Solve state smoothing problems, e.g. compute $p(x_{0:T} \mid y_{1:T})$.
3. Simulate from the smoothing density $\widetilde{x}_{0:T} \sim p(x_{0:T} \mid y_{1:T})$.

These three capabilities are key components in implementing various nonlinear system identification strategies.

Tutorial paper on the use of SMC for nonlinear sys.id.

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

ex) Simple linear toy problem

Identify the parameters $\theta = (a, c, q, r)^\mathsf{T}$ in

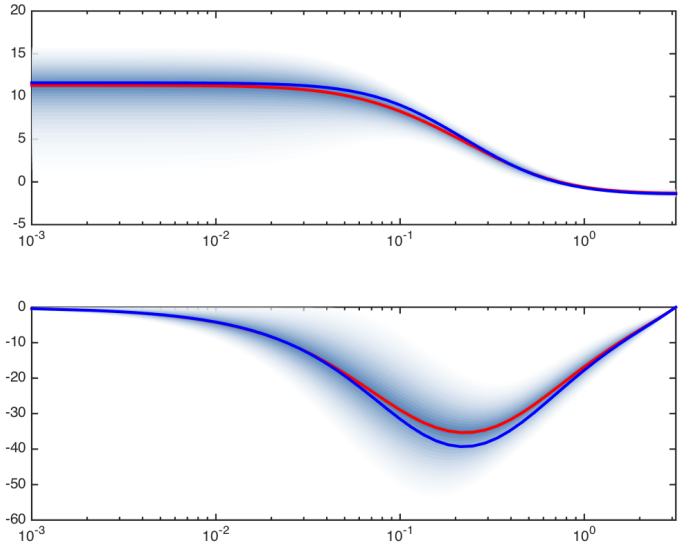
$$x_{t+1} = a x_t + w_t, \qquad w_t \sim \mathcal{N}(0, q),$$
$$y_t = c x_t + e_t, \qquad e_t \sim \mathcal{N}(0, r).$$

Observations:

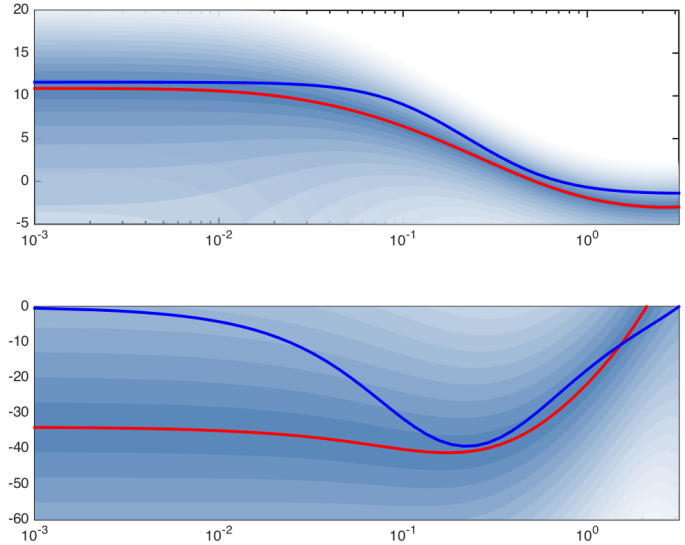
- The likelihood $L(\theta) = p(y_{1:T} \mid \theta)$ and its gradient $\nabla_\theta L(\theta)$ are available in closed form via standard Kalman filter equations.
- Standard gradient-based search algorithms applies.
- Deterministic optimization problem ($L(\theta), \nabla_\theta L(\theta)$ noise-free).

Lennart's rule: Make sure that your new algorithm reproduce the result of standard algorithms on toy problems.

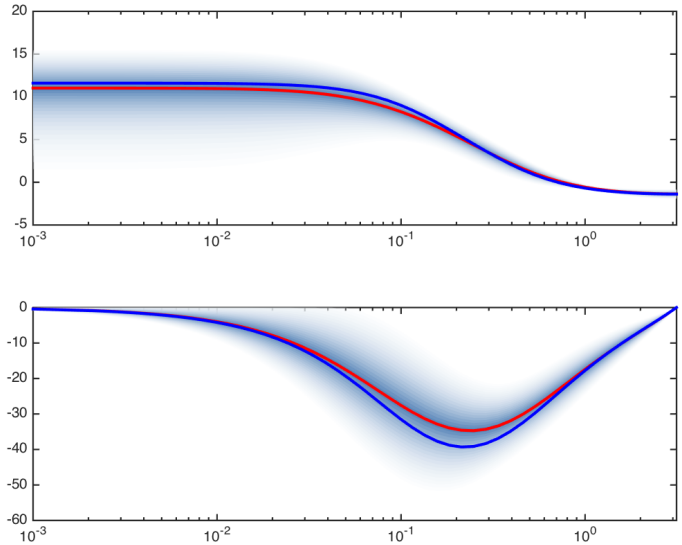
ex) Simple linear toy problem



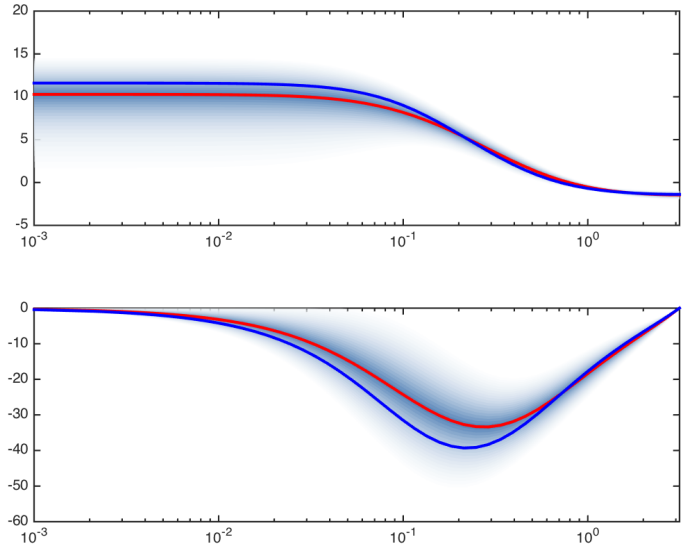
All three alg. for the noise-free case.



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



GP-based Hessian appr. for noisy observations of $L(\theta)$ and $\nabla L(\theta)$.



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

ex) Nonlinear toy problem

Consider the following nonlinear state space model

$$X_t = 0.5X_{t-1} + b\frac{X_{t-1}}{1+X_{t-1}^2} + 8\cos(1.2t) + qW_t, \qquad W_t \sim \mathcal{N}(0,1),$$

$$Y_t = 0.05X_t^2 + E_t, \qquad E_t \sim \mathcal{N}(0,0.1).$$

Generate $T = 100$ measurements using the true value $\theta^\star = (25, \sqrt{0.1})^\mathsf{T}$.

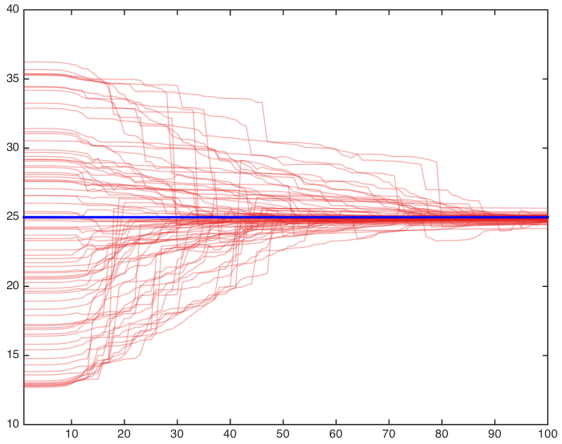
Task: Compute the maximum likelihood estimate of $\theta = (b, q)^\mathsf{T}$.

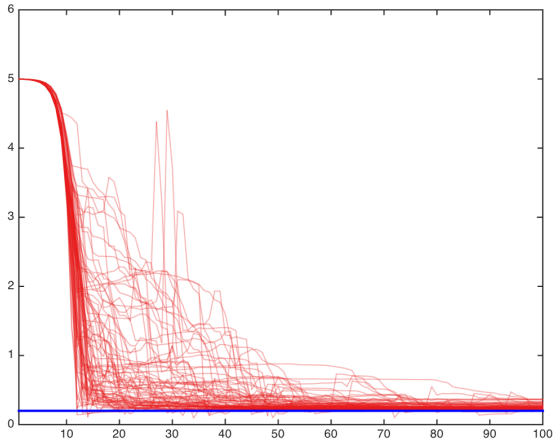
We use $N = 500$ particles in the PF to **estimate the likelihood**.

Fishers' identity together with a particle smoother is used to **estimate the likelihood gradient**.

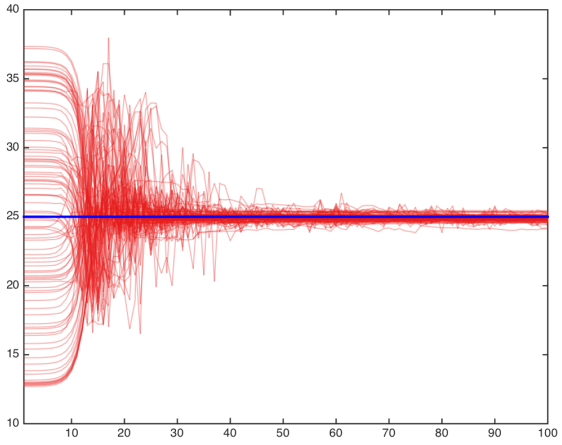
ex) Nonlinear toy problem – result

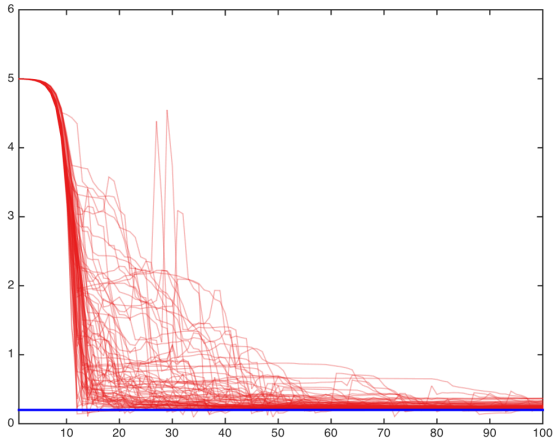
Model the cost function and its gradient as a GP (b left, q right)





Model the Hessian as a GP (b left, q right)





Some ongoing research

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 apart.

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

Birch — Our prototype probabilistic programming language.

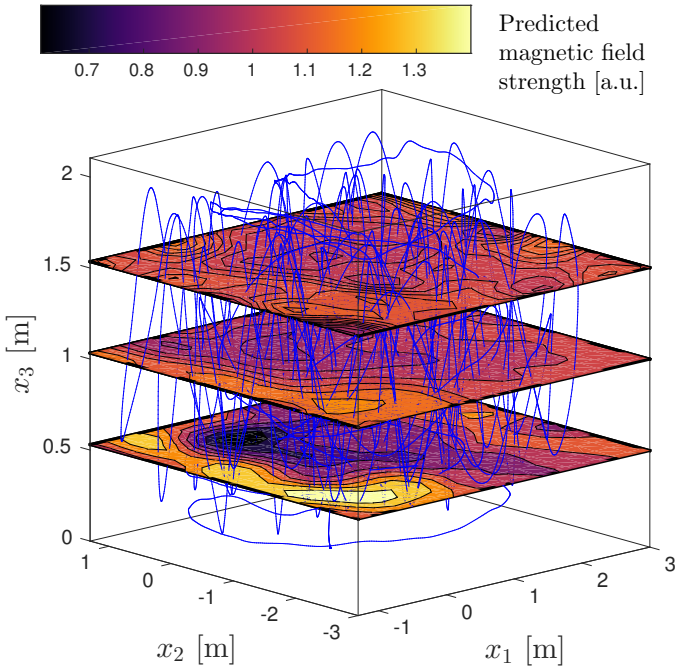
Linearly constrained GPs

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

By modelling the target function as a transformation of another GP the constraints are **explicitly incorporated**.

Result:

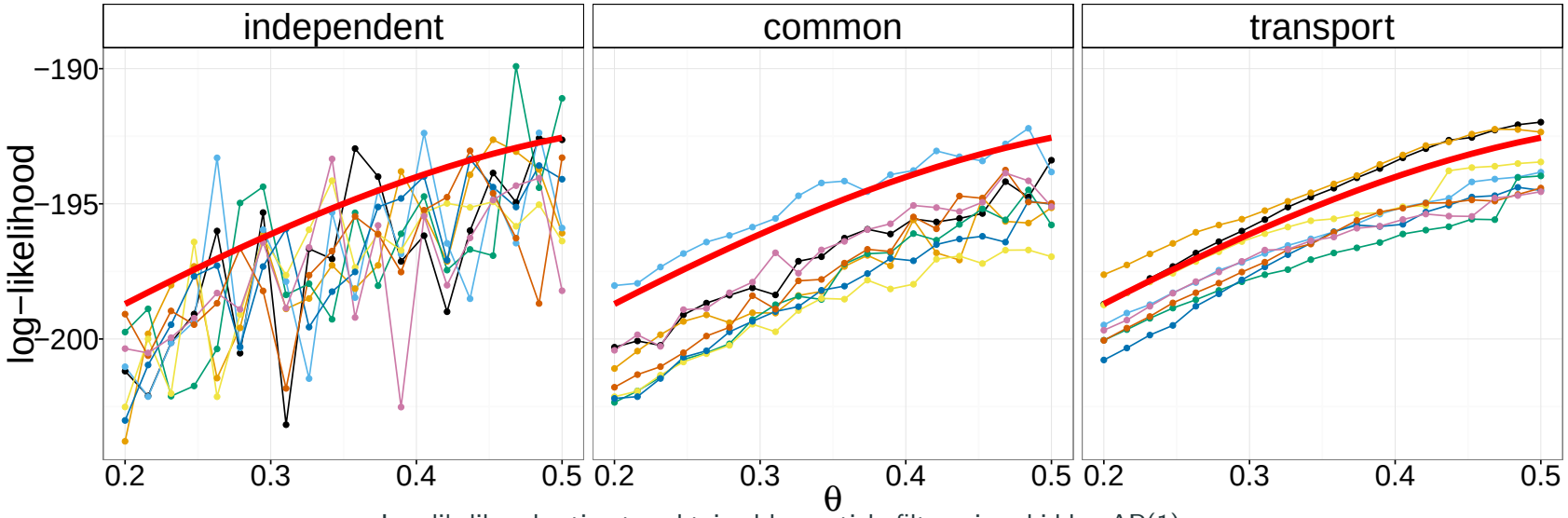
1. A probabilistic model that is **guaranteed** to fulfil known linear operator constraints.
2. A **constructive procedure** for designing the transformation.



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

Coupling of particle filters

New **coupled resampling** schemes can be used to improve a variety of particle-based algorithms.



Log-likelihood estimates obtained by particle filters, in a hidden AR(1).

The **first unbiased estimator** of smoothing expectations,

- The algorithm can be completely parallelized and
- we can compute accurate confidence intervals.

Pierre Jacob, Fredrik Lindsten, Thomas B. Schön, **Smoothing with couplings of conditional particle filters**. *arXiv:1701.02002*, 2017.

Gaussian process state space model

"Inspired by the Gaussian process, enabled by the particle filter"

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

The model functions f and g are assumed to be realizations from Gaussian process priors and $w_t \sim \mathcal{N}(0, Q)$, $e_t \sim \mathcal{N}(0, R)$.

We can now find the posterior distribution

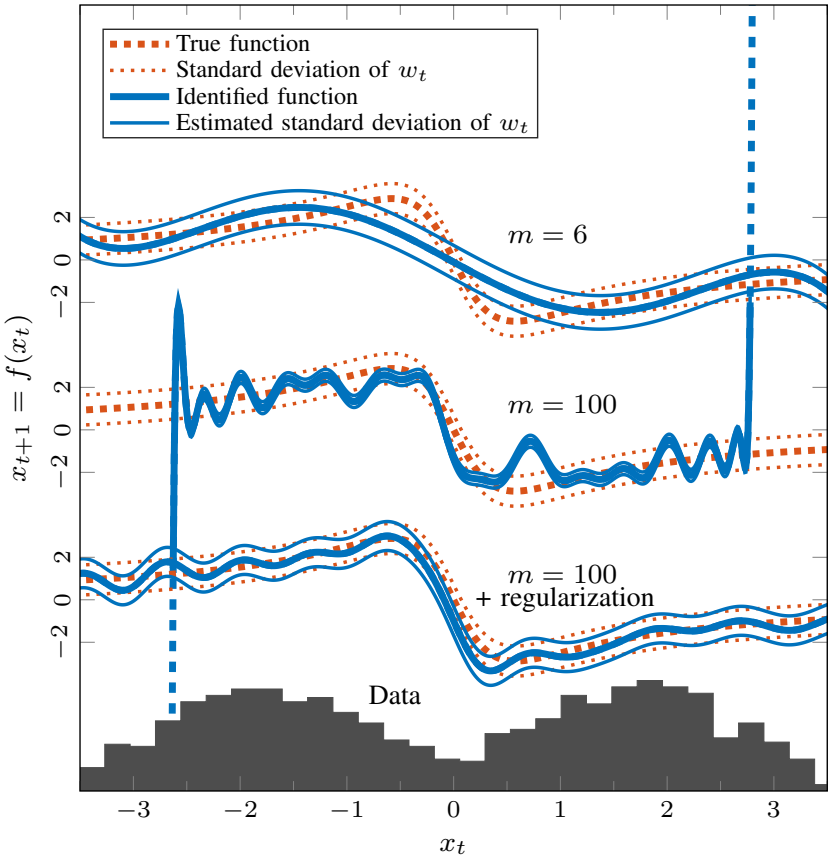
$$p(f, g, Q, R, \eta \mid y_{1:T}),$$

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.

Frigola, Roger. **Bayesian time series learning with Gaussian processes** PhD thesis, University of Cambridge, 2015.

Regularization in nonlinear state spaces



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

Provides a data-driven way of **tuning** the model flexibility.

Toy example:

$$x_{t+1} = -10\frac{x_t}{1+3x_t^2} + w_t,$$

$$y_t = x_t + e_t.$$

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

Conclusions

Derived a **probabilistic** quasi-Newton algorithm that can be used with **noisy** observations of the cost function and its derivatives.

- Integral formulation of the secant condition.
- Gaussian process to model the Hessian.
- Application: Maximum likelihood estimation in nonlinear SSMs.

Adrian G. Wills and Thomas B. Schön. **On the construction of probabilistic Newton-type algorithms**, *Proceedings of the 56th IEEE Conference on Decision and Control (CDC)*, Melbourne, Australia, December 2017.

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

Backup slides

GP regression

Remaining problem: Given training data $\mathcal{T} = \{x_t, y_t\}_{i=1}^{T}$ and our GP prior $f \sim \mathcal{GP}(m, k)$ compute $p(f_\star \mid \mathbf{y})$ for an arbitrary test point $(x_\star, y_\star)$.

$$\begin{pmatrix} \mathbf{y} \\ f_\star \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(\mathbf{x}) \\ m(x_\star) \end{pmatrix}, \begin{pmatrix} k(\mathbf{x}, \mathbf{x}) + \sigma^2 I_T & k(\mathbf{x}, x_\star) \\ k(x_\star, \mathbf{x}) & k(x_\star, x_\star) \end{pmatrix} \right),$$

The conditioning theorem for partitioned Gaussians results in

$$\begin{aligned} f_\star \mid \mathbf{y} &\sim \mathcal{N}\left(\mu_\star, k_\star\right), \\ \mu_\star &= m(x_\star) + \mathbf{s}^\mathsf{T}(\mathbf{y} - m(\mathbf{x})), \\ k_\star &= k(x_\star, x_\star) - \mathbf{s}^\mathsf{T} k(\mathbf{x}, x_\star), \end{aligned}$$

where $\mathbf{s}^\mathsf{T} = k(x_\star, \mathbf{x})(k(\mathbf{x}, \mathbf{x}) + \sigma^2 I_T)^{-1}$.