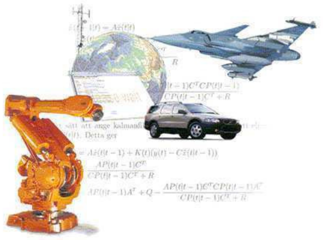
Learning models of dynamical systems

– strategies and concrete examples



Thomas Schön

Division of Automatic Control
Linköping University
Sweden

Joint work with (alphabetical order): **Michael I. Jordan** (UC Berkeley), **Fredrik Lindsten** (Linköping University), **Lennart Ljung** (Linköping University), **Brett Ninness** (University of Newcastle, Australia) and **Adrian Wills** (MRA, Newcastle, Australia).

Problem formulation

A state space model (SSM) consists of a Markov process $\{x_t\}_{t\geq 1}$ and a measurement process $\{y_t\}_{t\geq 1}$, related according to

$$\begin{aligned} x_{t+1} \mid x_t &\sim f_{\theta,t}(x_{t+1} \mid x_t, u_t), \\ y_t \mid x_t &\sim h_{\theta,t}(y_t \mid x_t, u_t), \\ x_1 &\sim \mu_\theta(x_1), \quad (\theta \sim p(\theta)). \end{aligned}$$

Learning problem: Find θ based on $\{u_{1:T}, y_{1:T}\}$.

We will study two different learning problems for finding static parameters θ in SSMs:

1. **Maximum likelihood:** Find the value for θ that maximizes the likelihood function $p_\theta(y_{1:T})$.
2. **Bayesian:** Compute the posterior distribution $p(\theta \mid y_{1:T})$.

Problem formulation – ML learning (I/II)

The three steps of ML learning (applied to SSMs) are:

1. Model the obtained measurements $y_1, \dots, y_T$ as a realisation from the stochastic variables $Y_1, \dots, Y_T$.
2. Assume $y_t \mid x_t \sim h_\theta(y_t \mid x_t)$ and $x_t \mid x_{t-1} \sim f_\theta(x_t \mid x_{t-1})$.
3. Assume that the stochastic variables $Y_1, \dots, Y_T$ are conditionally iid.

ML amounts to solving,

$$\widehat{\theta}^{\text{ML}} = \arg\max_{\theta} \log p_\theta(y_{1:T})$$

where the log-likelihood function is given by

$$\log p_\theta(y_{1:T}) = \sum_{t=1}^{T} \log p_\theta(y_t \mid y_{1:t-1})$$

Problem formulation – ML learning (II/II)

There are at least two challenges with the ML formulation:

1. The one-step prediction pdf $p_\theta(y_t \mid y_{1:t-1})$ has to be computed.
2. In solving the optimization problem

$$\widehat{\theta}^{\text{ML}} = \arg\max_{\theta} \; \log p_\theta(y_{1:T})$$

the derivatives $\frac{\partial}{\partial \theta} p_\theta(y_t \mid y_{1:t-1})$ are useful.

Problem formulation – Bayesian learning

Bayesian model: θ is a random variable with a prior density $p(\theta)$.

The **goal** in Bayesian modeling is to compute the posterior $p(\underbrace{\theta, x_{1:T}}_{\triangleq \eta} \mid y_{1:T}) = p(\eta \mid y_{1:T})$ (or one of its marginals).

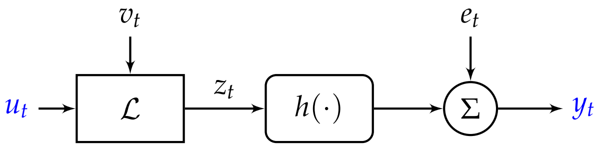
Bayesian modeling/learning amounts to:

1. Find an expression for the likelihood $p(y_{1:T} \mid \eta)$.
2. Assign priors $p(\eta)$ to all unknown stochastic variables η present in the model.
3. Determine the posterior distribution $p(\eta \mid y_{1:T})$.

The **key challenge** is that there is no closed form expression available for the posterior.

Example used – the Wiener model

As an example we will study how to learn a **Wiener model**.


$$\begin{tikzcd} & v_t \arrow[d] & & & e_t \arrow[d] & \\ u_t \arrow[r] & \boxed{\mathcal{L}} \arrow[r, "z_t"] & \boxed{h(\cdot)} \arrow[rr] & & \Sigma \arrow[r] & y_t \end{tikzcd}$$

A Wiener model is a linear dynamical model ($\mathcal{L}$) followed by a static nonlinearity $(h(\cdot))$.

Learning problem: Find $\mathcal{L}$ and $h(\cdot)$ based on $\{u_{1:T}, y_{1:T}\}$.

Wiener model – explicit state space equations

Dynamics: Linear Gaussian state space (LGSS) model:

$$x_{t+1} = \begin{pmatrix} A & B \end{pmatrix} \begin{pmatrix} x_t \\ u_t \end{pmatrix} + v_t, \qquad v_t \sim \mathcal{N}(0, Q),$$
$$z_t = C x_t.$$

Measurements: Static nonlinearity:

- Parametric: $y_t = h(z_t, \beta) + e_t, \quad e_t \sim \mathcal{N}(0, R)$.
- Non-parametric: $y_t = h(z_t) + e_t, \quad e_t \sim \mathcal{N}(0, R)$.

Wiener model – common simplifying assumptions

Most of the existing work deals with special cases of the general problem. Typical restrictions imposed are:

- The nonlinearity $h(\cdot)$ is assumed to be invertible.
- The measurement noise e_t is absent.
- The LGSS model is deterministic (v_t is absent).
- The LGSS model is stochastic, but v_t is assumed white.

In the models and solutions provided here we do **not** have to make any of these restrictive assumptions.

Outline – rest of the talk

Presents solutions to the two problems just formulated. These solutions are illustrated using the special case of the Wiener model.

---

1. Solving the ML problem
    - Expectation maximization combined with a particle smoother
    - Example – blind learning of a Wiener model.
2. Solving the Bayesian problem
    - Markov chain Monte Carlo (MCMC) combined with a particle filter/smoother
    - Examples – learning two semiparametric Wiener models
3. Conclusions

Expectation Maximization (EM) – strategy and idea

The strategy underlying the EM algorithm is to separate the original ML problem into **two closely linked subproblems**, each of which is hopefully more tractable than the original problem.

This separation is accomplished by exploiting the **structure** inherent in the probabilistic model.

The idea is to consider the joint log-likelihood function of the observed variables $y_{1:T}$ and the latent variables $Z \triangleq \{x_1, \dots, x_T\}$,

$$\ell_\theta(x_{1:T}, y_{1:T}) = \log p_\theta(x_{1:T}, y_{1:T}).$$

EM – the algorithm

Algorithm 1 Expectation Maximization (EM)

```
1. Initialise: Set $i = 1$ and choose an initial $\theta^1$.
2. While not converged do:
   (a) Expectation (E) step: Compute
       $$\mathcal{Q}(\theta, \theta^i) = \mathrm{E}_{\theta^i}\left[\log p_\theta(x_{1:T}, y_{1:T}) \mid y_{1:T}\right] = \int \log p_\theta(x_{1:T}, y_{1:T}) p_{\theta^i}(x_{1:T} \mid y_{1:T}) \mathrm{d}x_{1:T}$$
   (b) Maximization (M) step: Compute
       $$\theta^{i+1} = \arg\max_{\theta \in \Theta} \mathcal{Q}(\theta, \theta^i)$$
   (c) $i \leftarrow i + 1$
```

Expectation (E) step – approximating $\mathcal{Q}$

In computing the $\mathcal{Q}$-function

$$\begin{aligned}\mathcal{Q}(\theta, \theta^i) &= \mathrm{E}_{\theta^i}\left[\log p_\theta(x_{1:T}, y_{1:T}) \mid y_{1:T}\right] \\ &= \int \log p_\theta(x_{1:T}, y_{1:T}) p_{\theta^i}(x_{1:T} \mid y_{1:T}) \mathrm{d}x_{1:T},\end{aligned}$$

we start by noting that

$$\begin{aligned}\log p_\theta(x_{1:T}, y_{1:T}) &= \log p_\theta(y_{1:T} \mid x_{1:T}) + \log p_\theta(x_{1:T}) \\ &= \log p_\theta(x_1) + \sum_{t=1}^{T-1} \log p_\theta(x_{t+1} \mid x_t) + \sum_{t=1}^{T} \log p_\theta(y_t \mid x_t)\end{aligned}$$

Approximating the $\mathcal{Q}$-function (I/III)

This results in the following expression for the $\mathcal{Q}$-function

$$\mathcal{Q}(\theta, \theta^i) = I_1 + I_2 + I_3,$$

where

$$I_1 = \int \log p_\theta(x_1) p_{\theta^i}(x_1 \mid y_{1:N}) dx_1,$$
$$I_2 = \sum_{t=1}^{T-1} \int \int \log p_\theta(x_{t+1} \mid x_t) p_{\theta^i}(x_{t+1}, x_t \mid y_{1:N}) dx_t dx_{t+1},$$
$$I_3 = \sum_{t=1}^{T} \int \log p_\theta(y_t \mid x_t) p_{\theta^i}(x_t \mid y_{1:N}) dx_t.$$

This leads us to a nonlinear state smoothing problem, which we can solve using sequential Monte Carlo (here, **particle smoothers**).

Approximating the $\mathcal{Q}$-function (II/III)

Using particle filters and particle smoothers we straightforwardly obtain the following approximations

$$p_{\theta^i}(x_1 \mid y_{1:T}) \approx \widehat{p}^N_{\theta^i}(x_1 \mid y_{1:T}) = \sum_{i=1}^{N} w^i_{1|T} \delta_{x^i_1}(x_1),$$

$$p_{\theta^i}(x_{t:t+1} \mid y_{1:T}) \approx \widehat{p}^N_{\theta^i}(x_{t:t+1} \mid y_{1:T}) = \sum_{i=1}^{N} w^i_{t|T} \delta_{x^i_{t:t+1}}(x_{t:t+1}).$$

The particle smoother employed is the so called forward filtering backward simulation (FFBS) particle smoother derived by

R. Douc, A. Garivier, E. Moulines, and J. Olsson. **Sequential Monte Carlo smoothing for general state space hidden Markov models**. *Annals of Applied Probability*, 21(6):2109 2145, 2011.

Approximating the $\mathcal{Q}$-function (III/III)

Inserting the above approximations into the integrals yields the approximation we are looking for,

$$\begin{aligned}
\widehat{I}_1 &= \int \log p_\theta(x_1) \sum_{i=1}^{N} w_{1|T}^i \delta_{x_1^i}(x_1) \mathrm{d}x_1 \\
&= \sum_{i=1}^{N} w_{1|T}^i \log p_\theta(x_1^i), \\
\widehat{I}_3 &= \sum_{t=1}^{T} \int \log p_\theta(y_t \mid x_t) \sum_{i=1}^{N} w_{t|T}^i \delta_{x_t^i}(x_t) \mathrm{d}x_t \\
&= \sum_{t=1}^{T} \sum_{i=1}^{N} w_{t|T}^i \log p_\theta(y_t \mid x_t^i),
\end{aligned}$$

and similarly for I_2.

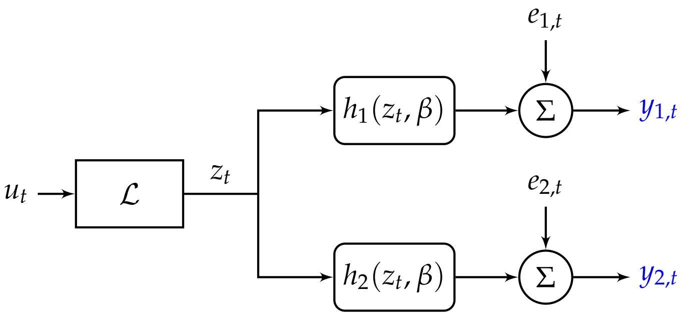
Final ML learning algorithm

Algorithm 2 EM for learning nonlinear systems

```
1. Initialise: Set $i = 1$ and choose an initial $\theta^1$.
2. While not converged do:
   (a) Expectation (E) step: Run a FFBS PS and compute
       $$\widehat{\mathcal{Q}}(\theta, \theta^i) = \widehat{I}_1(\theta, \theta^i) + \widehat{I}_2(\theta, \theta^i) + \widehat{I}_3(\theta, \theta^i)$$
   (b) Maximization (M) step: Compute $\theta^{i+1} = \arg\max_{\theta \in \Theta} \widehat{\mathcal{Q}}(\theta, \theta^i)$
       using an off-the-shelf numerical optimization algorithm.
   (c) $i \leftarrow i + 1$
```

Thomas B. Schön, Adrian Wills and Brett Ninness. **System Identification of Nonlinear State-Space Models**. *Automatica*, 47(1):39-49, January 2011.

Example – blind Wiener learning (I/III)

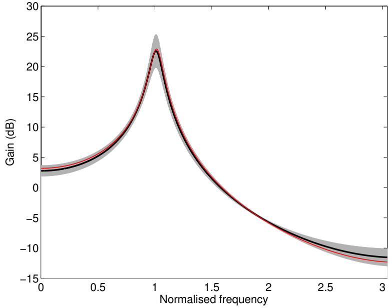


$$x_{t+1} = \begin{pmatrix} A & B \end{pmatrix} \begin{pmatrix} x_t \\ u_t \end{pmatrix}, \qquad u_t \sim \mathcal{N}(0, Q),$$
$$z_t = Cx_t, \qquad y_t = h(z_t, \beta) + e_t, \qquad e_t \sim \mathcal{N}(0, R).$$

Learning problem: Find $\mathcal{L}$, β, r_1, and r_2 based on $\{y_{1,1:T}, y_{2,1:T}\}$.

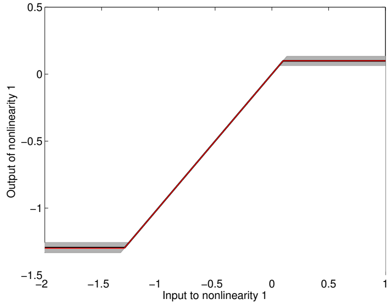
Example – blind Wiener learning (II/III)

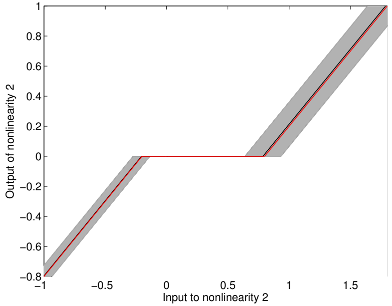
- Second order LGSS model with complex poles.
- Employ the EM-PS with $N = 100$ particles.
- EM-PS was terminated after 100 iterations.
- Results obtained using $T = 1000$ samples.
- The plots are based on 100 realisations of data.
- Nonlinearities (dead zone and saturation) shown on next slide.



Bode plot of estimated mean (black), true system (red) and the result for all 100 realisations (gray).

Example – blind Wiener learning (III/III)





Estimated mean (black), true static nonlinearity (red) and the result for all 100 realisations (gray).

Adrian Wills, Thomas B. Schön, Lennart Ljung and Brett Ninness. **Identification of Hammerstein-Wiener Models**. *Automatica*, 49(1): 70-81, January 2013.

Bayesian problem formulation again

Consider a Bayesian SSM

$$\begin{aligned} x_{t+1} \mid x_t &\sim f_{\theta,t}(x_{t+1} \mid x_t, u_t), \\ y_t \mid x_t &\sim h_{\theta,t}(y_t \mid x_t, u_t), \\ x_1 &\sim \mu_\theta(x_1), \\ \theta &\sim p(\theta). \end{aligned}$$

We observe $D_T \triangleq \{u_{1:T}, y_{1:T}\}$.

Goal: Compute the posterior $p(\theta, x_{1:T} \mid D_T)$.

Solution used here – Gibbs sampler

Markov chain Monte Carlo (MCMC) methods allows us to generate samples from an arbitrary target distribution by simulating a Markov chain.

Gibbs sampling (blocked) for SSMs amounts to iterating

- Draw $\theta[m] \sim p(\theta \mid x_{1:T}[m-1], D_T)$,
- Draw $x_{1:T}[m] \sim p(x_{1:T} \mid \theta[m], D_T)$.

The result is a Markov chain

$$\{\theta[m], x_{1:T}[m]\}_{m \geq 1}$$

with $p(\theta, x_{1:T} \mid D_T)$ as its stationary distribution!

Ex – Gibbs sampler for LGSS learning

Whenever you are working on an algorithm for nonlinear systems, **always** make sure that it solves the simple LGSS systems first!

Consider a fully parameterised LGSS model $(\theta \triangleq \{\Gamma, \Pi\})$.

$$\begin{pmatrix} x_{t+1} \\ y_t \end{pmatrix} \mid x_t \sim \mathcal{N}\left(\begin{pmatrix} x_{t+1} \\ y_t \end{pmatrix} \mid \underbrace{\begin{pmatrix} A & B \\ C & D \end{pmatrix}}_{\Gamma} \begin{pmatrix} x_t \\ u_t \end{pmatrix}, \underbrace{\begin{pmatrix} Q & S \\ S^\mathsf{T} & R \end{pmatrix}}_{\Pi} \right).$$

The posterior distribution $p(\theta \mid D_T)$ is computed using (blocked) Gibbs sampling,

- Draw $\theta[m] \sim p(\theta \mid x_{1:T}[m-1], D_T)$,
- Draw $x_{1:T}[m] \sim p(x_{1:T} \mid \theta[m], D_T)$.

Adrian Wills, Thomas B. Schön, Fredrik Lindsten and Brett Ninness, **Estimation of Linear Systems using a Gibbs Sampler**, *Proceedings of the 16th IFAC Symposium on System Identification (SYSID)*, Brussels, Belgium, July 2012.

Gibbs sampler for a general SSM

What would a Gibbs sampler for a general nonlinear/non-Gaussian SSM look like?

- Draw $\theta[m] \sim p(\theta \mid x_{1:T}[m-1], D_T)$,
- Draw $x_{1:T}[m] \sim p(x_{1:T} \mid \theta[m], D_T)$.

Problem: $p(x_{1:T} \mid \theta, D_T)$ is not available!!

Idea: Approximate $p(x_{1:T} \mid \theta, D_T)$ using a particle smoother (PS).

(Non-trivial) solution: Careful and clever analysis of how to combine MCMC and PF/PS results in the PMCMC family of algorithms.

PMCMC methods

Facts about Particle Markov Chain Monte Carlo (PMCMC) samplers:

- Provides a systematic and provably correct combination of PF/PS and MCMC.
- Standard MCMC samplers on non-standard spaces.
- Constitutes a family of Bayesian inference methods, including
    - Particle Independent Metropolis Hastings (PIMH)
    - Particle Marginal Metropolis Hastings (PMMH)
    - **Particle Gibbs (PG)**

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.

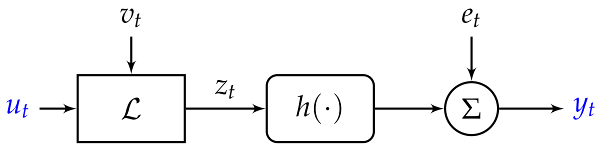
Particle Gibbs with backward simulation (PG-BS)

PG-BS sampler targeting $p(\theta, x_{1:T} \mid D_T)$.

- Conditional particle filter and backward simulation
  - Run a conditional PF, targeting $p(x_{1:T} \mid \theta, D_T)$;
  - Run a backward simulator to sample $x^\star_{1:T}$;
- Draw $\theta^\star \sim p(\theta \mid x^\star_{1:T}, D_T)$.

Powerful and important property of PG-BS: Provably convergent for any $N \geq 2$ particles and it works in practice!

Reminder – semiparametric Wiener model



LGSS and a static nonlinearity:

$$x_{t+1} = \underbrace{\begin{pmatrix} A & B \end{pmatrix}}_{\Gamma} \begin{pmatrix} x_t \\ u_t \end{pmatrix} + v_t, \qquad v_t \sim \mathcal{N}(0, Q),$$

$$z_t = C x_t.$$

$$y_t = h(z_t) + e_t, \qquad e_t \sim \mathcal{N}(0, R).$$

Semiparametric model 1 – known model order

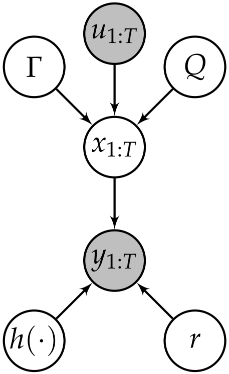
First step towards a fully data driven model (the order of the LGSS model is still assumed known).

Parameters: $\theta = \{\Gamma, Q, r, h(\cdot)\}$.

Bayesian model specified by priors:

- Conjugate priors for $\Gamma = [A\ B]$, Q and r,
  - $p(\Gamma, Q) =$ Matrix-normal inverse-Wishart
  - $p(r) =$ inverse-Wishart
- Gaussian process prior on $h(\cdot)$,

$$h(\cdot) \sim \mathcal{GP}(z, k(z, z')).$$



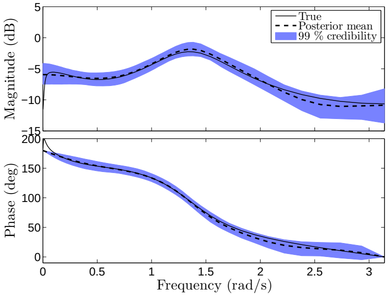
Learning algorithm – PG-BS

Gibbs sampler targeting $p(\theta, x_{1:T} \mid D_T)$.

- Conditional particle filter and backwards simulation
    - Run a conditional PF, targeting $p(x_{1:T} \mid \theta, D_T)$;
    - Run a backward simulator to sample $x^\star_{1:T}$;
- Draw $\{\Gamma^\star, Q^\star, r^\star\} \sim p(\Gamma, Q, r \mid h, x^\star_{1:T}, D_T)$;
- Draw $h^\star \sim p(h \mid r^\star, x^\star_{1:T}, D_T)$.

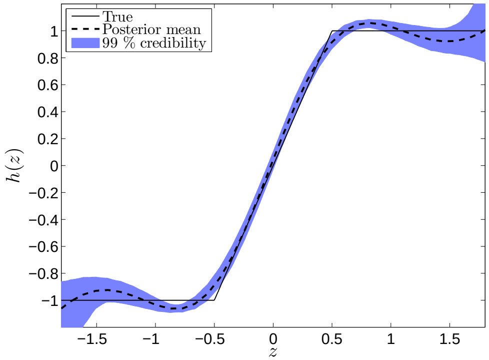
Example – known model order (I/II)

- Bayesian semiparametric model with conjugate prior (MNIW).
- 6^{th} order LGSS model and a saturation.
- Using $T = 1000$ measurements.
- Employ the PG-BS sampler with $N = 15$ particles.
- Run 15000 MCMC iterations, discard 5000 as burn-in.



True Bode diagram of the linear system (solid black), estimated mean (dashed black) and 99% credibility interval (blue).

Example – known model order (II/II)



True static nonlinearity (solid black), estimated posterior mean (dashed black) and 99% credibility interval (blue).

Fredrik Lindsten, Thomas B. Schön and Michael I. Jordan. **A semiparametric Bayesian approach to Wiener system identification**. *Proceedings of the 16th IFAC Symposium on System Identification (SYSID)*, Brussels, Belgium, July 2012.

Semiparametric model 2 – data driven

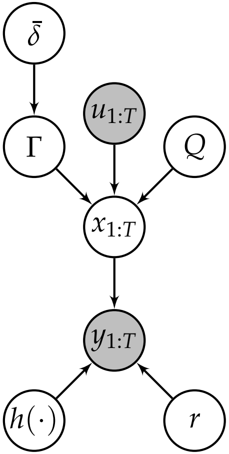
Everything is learned from the data by introducing the possibility to switch specific model components on and off.

Parameters: $\theta = \{\Gamma, Q, \bar{\delta}, r, h(\cdot)\}$.

Bayesian model specified by priors

- Sparseness prior (ARD) on $\Gamma = [A \ B]$,
  - $p(\Gamma \mid \bar{\delta}) = \prod_{j=1}^{n_x+n_u} \mathcal{N}(\gamma_j \mid 0, \delta_j^{-1} I_{n_x+n_u})$
  - $\bar{\delta} = \{\delta_j\}_{j=1}^{n_x+n_u}, \quad p(\delta_j) = \text{Gam}(\delta_j; a, b)$
- Inverse-Wishart prior on Q and r
- Gaussian process prior on $h(\cdot)$,

$$h(\cdot) \sim \mathcal{GP}(z, k(z, z')).$$



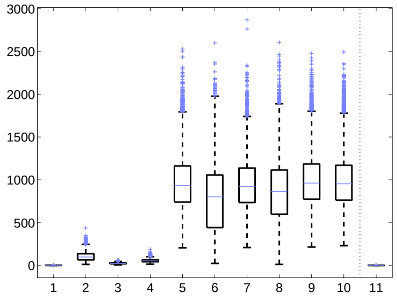
Learning algorithm – PG-BS

Gibbs sampler targeting $p(\theta, x_{1:T} \mid D_T)$.

- Conditional particle filter and backwards simulation
  - Run a conditional PF, targeting $p(x_{1:T} \mid \theta, D_T)$;
  - Run a backward simulator to sample $x^{\star}_{1:T}$;
- Draw $\{\Gamma^{\star}, \bar{\delta}^{\star}, Q^{\star}, r^{\star}\} \sim p(\Gamma, \bar{\delta}, Q, r \mid h, x^{\star}_{1:T}, D_T)$;
- Draw $h^{\star} \sim p(h \mid r^{\star}, x^{\star}_{1:T}, D_T)$.

Data driven learning (I/II)

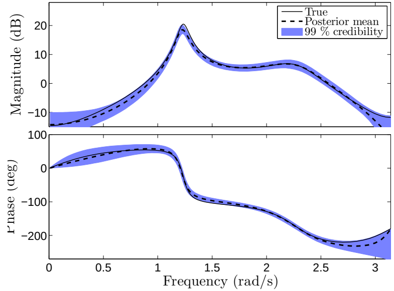
- Bayesian semiparametric model with ARD prior.
- 4^{th} order LGSS model.
- Using $T = 1000$ measurements.
- Employ the PG-BS sampler with $N = 15$ particles.
- Run 15000 MCMC iterations, discard 5000 as burn-in.

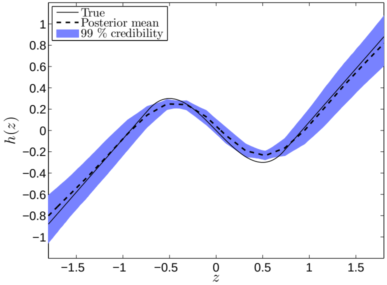


ARD precision parameters $\delta_j = 1, \dots, 11$. The rightmost box plot corresponds to the input signal.

Data driven learning (II/II)

Show movie





Bode diagram of the 4th-order linear system. Estimated mean (dashed black), true system (solid black) and 99% credibility intervals (blue).

Static nonlinearity (non-monotonic), estimated mean (dashed black), true (black) and the 99% credibility intervals (blue).

Observations – two scientific trends

In developing new models and inference/learning tool for building useful models of dynamical systems I see two clear trends:

1. The extreme availability of computational power that we enjoy today is allowing us to attack and solve more and more complex (previously insolvable) problems by employing **computational methods**. The mindset is shifting from looking for closed form solutions to systematically working with computational solutions.
2. New theory is developed where **complex algorithms constitute the basic building blocks**. The use of algorithms as building blocks raises new challenges, both in deriving solutions and in studying the properties of these solutions (analysis).

The work presented in this talk follow these trends.

Conclusions

- Maximum likelihood modeling learning:
  - EM (nonlinear optimization and PS)
- Bayesian modeling and learning:
  - PMCMC = combination of MCMC and PF/PS
  - PMCMC is systematic and provably correct
- Solved various Wiener learning problems for illustration.
- Much interesting research **remains to be done**!!

In this talk I introduced the strategies and showed a few concrete examples. Should you be interested in the details, I am offering a PhD course on this topic.

`users.isy.liu.se/rt/schon/course_CIDS.html`

Some references

- Maximum likelihood modeling learning:

Thomas B. Schön, Adrian Wills and Brett Ninness. **System Identification of Nonlinear State-Space Models**. *Automatica*, 47(1):39-49, January 2011.

Adrian Wills, Thomas B. Schön, Lennart Ljung and Brett Ninness. **Identification of Hammerstein-Wiener Models**. *Automatica*, 49(1): 70-81, January 2013.

Some MATLAB code is available from `users.isy.liu.se/rt/schon/software.html`

- Bayesian modeling and learning:

Fredrik Lindsten, Thomas B. Schön and Michael I. Jordan. **Data driven Wiener system identification**. *Automatica*, 2012 (in revision).

Fredrik Lindsten, Thomas B. Schön and Michael I. Jordan. **A semiparametric Bayesian approach to Wiener system identification**. *Proceedings of the 16th IFAC Symposium on System Identification (SYSID)*, Brussels, Belgium, July 2012.

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.

Fredrik Lindsten, Michael I. Jordan and Thomas B. Schön, **Ancestor sampling for particle Gibbs**, *Proceedings of Neural Information Processing Systems (NIPS)*, Lake Tahoe, NV, US, December, 2012.

MATLAB code is available from `users.isy.liu.se/rt/lindsten/code.html`