Particle Metropolis-Hastings

A tutorial on non-linear system identification

Johan Dahlin

`johan.dahlin@liu.se`

`work.johandahlin.com`

Division of Automatic Control,
Linköping University.

LINKÖPING
UNIVERSITY

Some short instructions

- Grab a seat on every second row,
  so that we might move around to help you if needed.
- Download and unzip MATLAB skeleton

  `http://bit.ly/1Xrqw3q`
- Open MATLAB and set working directory to the skeleton.

Particle Metropolis-Hastings

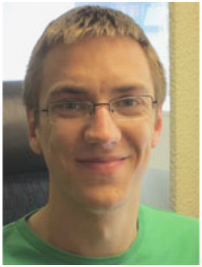
A tutorial on non-linear system identification

Johan Dahlin
`johan.dahlin@liu.se`
`work.johandahlin.com`

Division of Automatic Control,
Linköping University.

This is collaborative work together with

Christian Andersson Naesseth (Linköping University, Sweden).
Fredrik Lindsten (Uppsala University, Sweden).
Thomas Schön (Uppsala University, Sweden).
Andreas Svensson (Uppsala University, Sweden).



Why are we doing this?

Introduce some popular statistical algorithms.
Exemplify applications in system identification.

How will we do this?

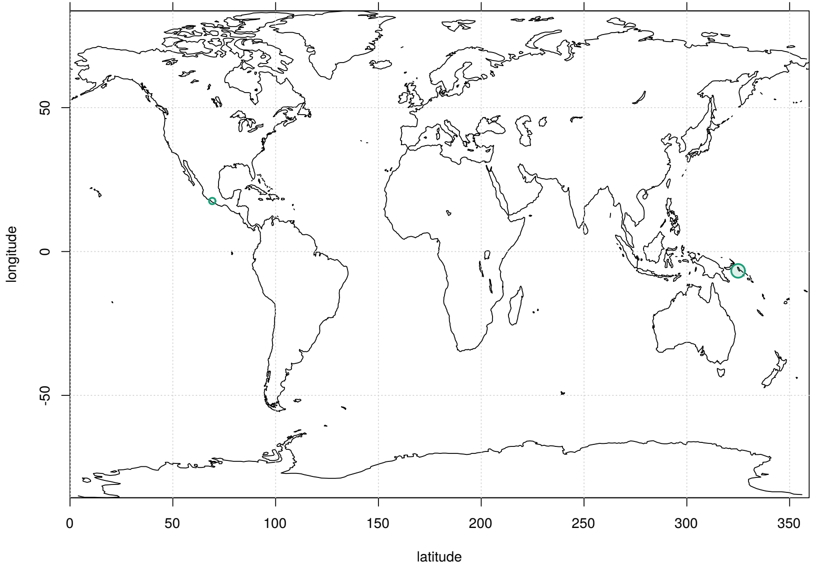
Interactive session!
You implement the algorithm yourself.
Questions and comments are welcome!

What are we going to do?

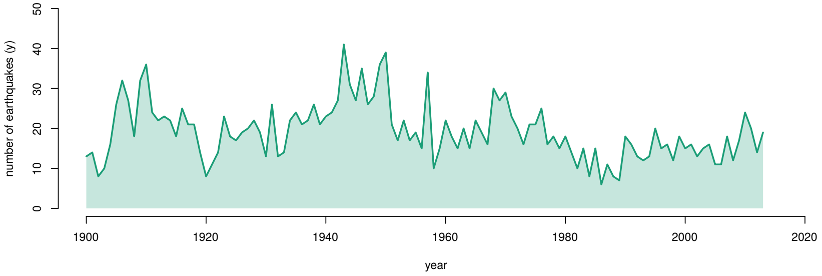
Discuss some underlying theory.
Implement PMH in MATLAB using a code skeleton.
Construct a model of Earthquake counts.

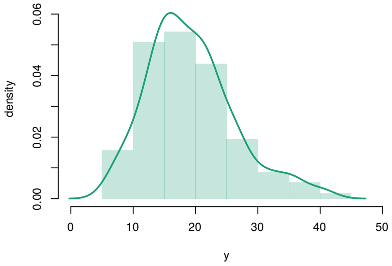
LINKÖPING
UNIVERSITY

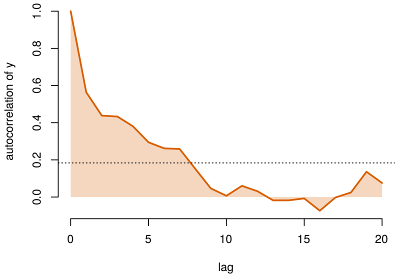
Problem: modelling earthquake counts [I/II]



Problem: modelling earthquake counts [II/II]







Independent and identically distributed (IID) model

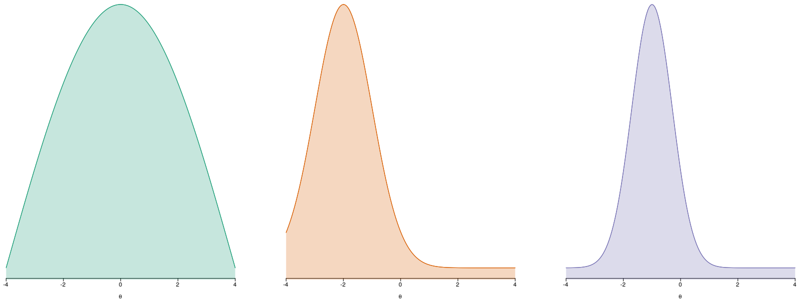
We consider the Poisson IID model

$$y_t \overset{\text{iid}}{\sim} \mathcal{P}\left(y_t; \boldsymbol{\lambda}\right),$$

where $\boldsymbol{\lambda} \in \mathbb{R}^+$ denotes the intensity.

Task: Estimate $\boldsymbol{\theta} = \boldsymbol{\lambda}$.

Bayesian parameter inference [I/II]



$$\pi(\theta) = p(\theta|y) \propto p(y|\theta)p(\theta), \quad \pi[\varphi] = \mathbb{E}_\pi\left[\varphi(\theta)\right] = \int \varphi(\theta')\pi(\theta')\mathrm{d}\theta'.$$

Bayesian parameter inference [II/II]

The log-likelihood is given by the model

$$\begin{aligned}\log p(y|\theta) &= \sum_{t=1}^{T} \log p(y_t|\theta) \\ &= -T\lambda + \sum_{t=1}^{T} \Big[y_t \log(\lambda) - \log(y_t!) \Big].\end{aligned}$$

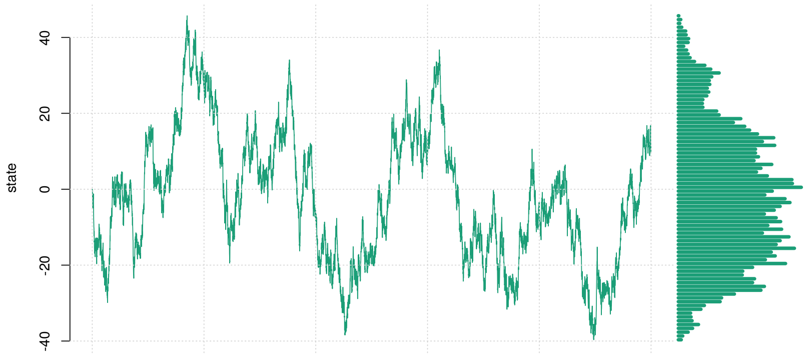
The log-prior is selected by us as

$$\log p(\theta) = \log\left(\mathbb{I}_{\lambda>0}\right).$$

Hence, the log-posterior is given by

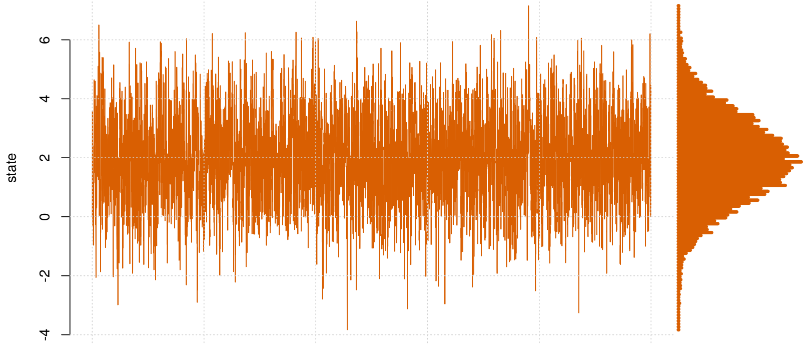
$$\log p(\theta|y) = \log\left(\mathbb{I}_{\lambda>0}\right) - T\lambda + \sum_{t=1}^{T} \Big[y_t \log(\lambda) - \log(y_t!) \Big].$$

Stationary distribution: Gaussian random walk



$$\theta_k | \theta_{k-1} \sim \mathcal{N}\left(\theta_k; \theta_{k-1}, \sigma^2\right).$$

Stationary distribution: Autoregressive process



$$\theta_k|\theta_{k-1} \sim \mathcal{N}\left(\theta_k; \mu + \phi(\theta_{k-1} - \mu), \sigma^2\right).$$

Stationary distribution: Autoregressive process (cont)

$$\theta_k = \mu + \phi(\theta_{k-1} - \mu) + \sigma z_k, \qquad z_k \sim \mathcal{N}(0,1),$$

Mean value of the process

$$\underbrace{\mathbb{E}[\theta_k]}_{\triangleq \bar{\mu}} = \mu + \phi(\underbrace{\mathbb{E}[\theta_{k-1}]}_{\triangleq \bar{\mu}} - \mu) + \sigma \underbrace{\mathbb{E}[z_k]}_{=0},$$

$$(1-\phi)\bar{\mu} = (1-\phi)\mu.$$

Variance of the process

$$\underbrace{\mathbb{V}[\theta_k]}_{\triangleq \bar{\sigma}^2} = \underbrace{\mathbb{V}[\mu]}_{=0} + \phi(\underbrace{\mathbb{V}[\theta_{k-1}]}_{\triangleq \bar{\sigma}^2} - \underbrace{\mathbb{V}[\mu]}_{=0}) + \sigma^2 \underbrace{\mathbb{V}[z_k]}_{=1},$$

$$(1-\phi)\bar{\sigma}^2 = \sigma^2$$

Metropolis-Hastings

We generate samples $\{\theta_k\}_{k=1}^{K}$ from $p(\theta|y)$ by

(i) Sample a candidate parameter θ' from a proposal distribution

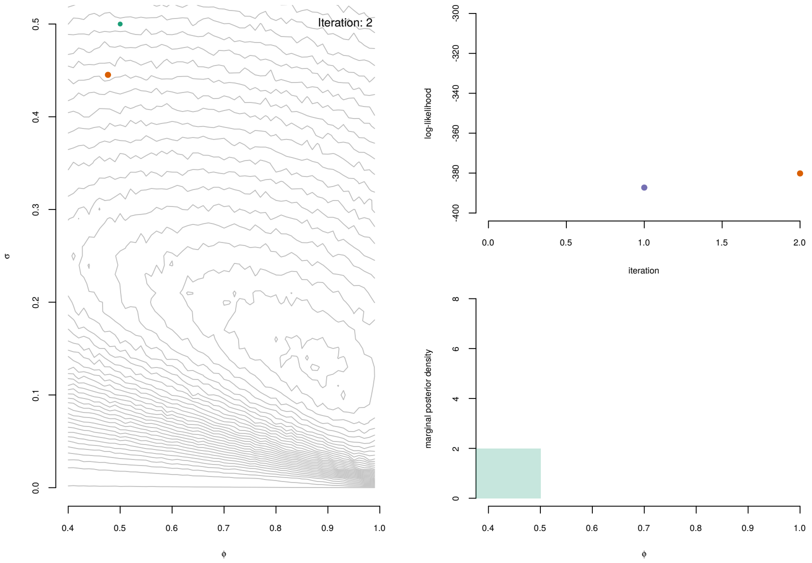
$$\theta' \sim \mathcal{N}(\theta'; \theta_{k-1}, \Sigma).$$

(ii) Accept θ' by setting $\theta_k \leftarrow \theta'$ with probability

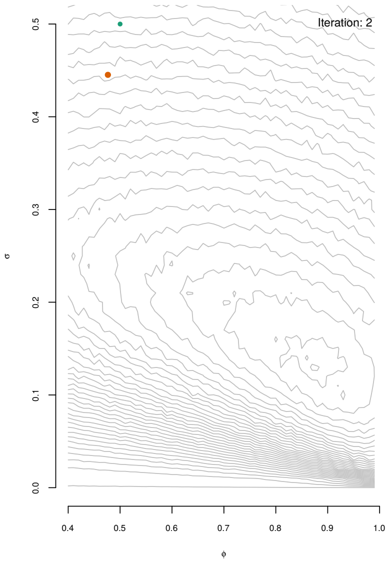
$$\min\left\{1, \frac{p(\theta'|y)}{p(\theta_{k-1}|y)}\right\},$$

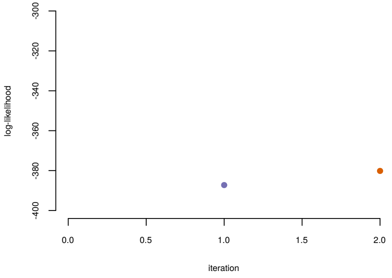
and otherwise reject θ' by setting $\theta_k \leftarrow \theta_{k-1}$.

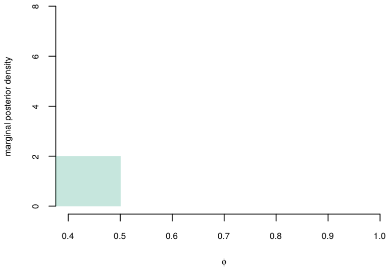
Metropolis-Hastings: toy example



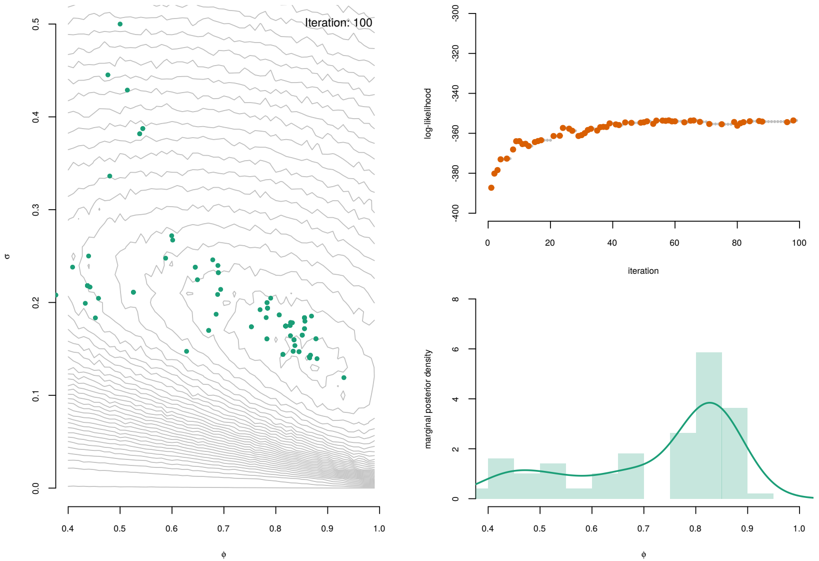
Metropolis-Hastings: toy example







Metropolis-Hastings: toy example



Tailoring Metropolis-Hastings for our problem

Lets have a look at the script

`example1_iid.m`

together with the two helpers

`mh.m` and `dpoisson.m`

Structure of `mh.m`

```
<< Initialise variables >>
<< Set the initial parameter >>
<< Compute the initial log-posterior >>

% Main loop
for kk = 2:nIterations
  <<  Propose a new parameter >>
  <<  Compute the log-posterior >>
  <<  Compute the acceptance probability >>
  <<  Accept / reject step >>
  <<  Write out progress >>
end
```

Implement: the Metropolis-Hastings algorithm

```
%<< Initialise variables >>
%<< Set the initial parameter >>
<< Compute the initial log-posterior >>

% Main loop
for kk = 2:nIterations
  <<  Propose a new parameter >>
  <<  Compute the log-posterior >>
  <<  Compute the acceptance probability >>
  %<<  Accept / reject step >>
  %<<  Write out progress >>
end
```

Implement: compute the log-posterior

Remember that the log-posterior is given by

$$\begin{aligned}\log p(\theta|y) &= \log\left(\mathbb{I}_{\lambda>0}\right) + \sum_{t=1}^{T} \log p(y_t|\theta) \\ &= \begin{cases} -\inf, & \text{if } \lambda \leq 0 \\ \sum_{t=1}^{T} \log p(y_t|\theta), & \text{if } \lambda > 0 \end{cases}.\end{aligned}$$

Hints:

- `dpoisson.m` returns $\log p(y_t|\theta)$.
- `observations` contains $y_{1:T}$.
- `theta` contains θ_k.

Implement: propose a new parameter

Remember that the parameter proposal is given by

$$\theta' \sim \mathcal{N}\left(\theta'; \theta_{k-1}, \Sigma\right).$$

Hints:

- `mvnrnd` implements a draw from $\mathcal{N}(\mu, \Sigma)$.
- `theta_proposed` contains θ'.
- `theta` contains θ_{k-1}.
- `Sigma` contains Σ.

Implement: compute the acceptance probability

Remember that the acceptance probability is given by

$$\min\left\{1, \frac{p(\theta'|y)}{p(\theta_{k-1}|y)}\right\},$$

which can be expressed by

$$\alpha(\theta', \theta_{k-1}) = \exp\big(\log p(\theta'|y) - \log p(\theta_{k-1}|y)\big) .$$

Hints:

- `aprob` contains $\alpha(\theta', \theta_{k-1})$.
- `logposterior` contains $\log p(\theta_{k-1}|y)$.
- `logposterior_proposed` contains $\log p(\theta'|y)$.

Implement: the Metropolis-Hastings algorithm

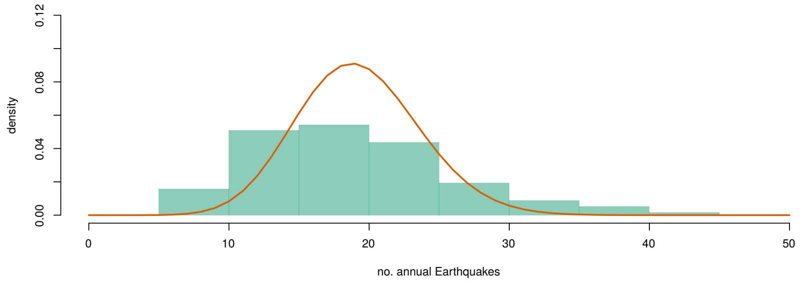
```
% Propose a new parameter
theta_proposed = mvnrnd( theta(kk-1,:), Sigma );

% Compute the log-posterior if the intensity is positive
if ( theta_proposed(1) > 0.0 )
  logposterior_proposed =
      sum( dpoisson(observations, theta_proposed) );
else
  logposterior_proposed = -inf;
end

% Compute the acceptance probability
aprob = exp( logposterior_proposed - logposterior(kk-1) );
```

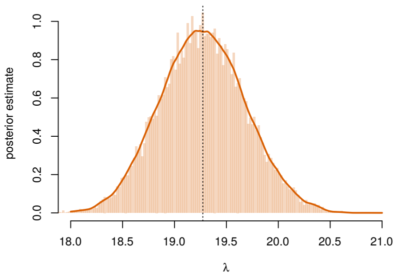
Implement: tuning the algorithm

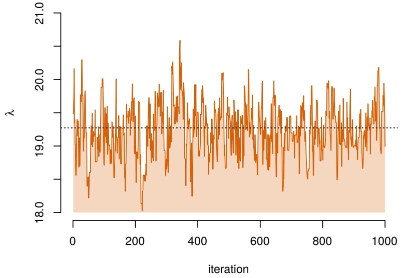
The user needs to decide on:

- The initial parameter value θ_0.
- The covariance matrix of the Gaussian proposal Σ.
- The number of iterations K and the length of the burn-in K_b.

Results for IID model







Time series model

We consider the model

$$x_{t+1}|x_t \sim \mathcal{N}\Big(x_{t+1}; \phi x_t, \boldsymbol{\sigma}^2\Big),$$
$$y_t|x_t \sim \mathcal{P}\Big(y_t; \boldsymbol{\beta} \exp(x_t)\Big),$$

where the parameters describe:

ϕ: persistence of intensity.

$\boldsymbol{\sigma}$: standard deviation of innovation in intensity.

$\boldsymbol{\beta}$: *nominal* number of annual earthquakes.

Task: Estimate $\boldsymbol{\theta} = \{\phi, \boldsymbol{\sigma}, \boldsymbol{\beta}\}$ and $x_{0:T}$ given $y_{1:T}$.

Particle filtering

Particle filters provide an estimate of the log-posterior

$$\log \widehat{p}^N(\theta|y),$$

which is implemented in

```
[ state_estimate, loglikelihood_estimate ]
       = pf( observations, theta, nParticles )
```

The computational cost of the algorithm is $\propto \mathcal{O}(NT) \approx \mathcal{O}(T^2)$.

Implement: the particle Metropolis-Hastings algorithm

```
%<< Initialise variables >>
%<< Set the initial parameter >>
<< Estimate the initial log-posterior using particle filter >>

% Main loop
for kk = 2:nIterations
  %<<  Propose a new parameter >>
  <<  Estimate the log-posterior using particle filter >>
  %<<  Compute the acceptance probability >>
  %<<  Accept / reject step >>
  %<<  Write out progress >>
end
```

Implement: estimate the log-posterior

The log-posterior is given by

$$\log p(\theta|y) = \log\left(\mathbb{I}_{|\phi|<1}\right) + \log\left(\mathbb{I}_{\sigma>0}\right) + \log\left(\mathbb{I}_{\beta>0}\right) + \log \widehat{p}^N(\theta|y).$$

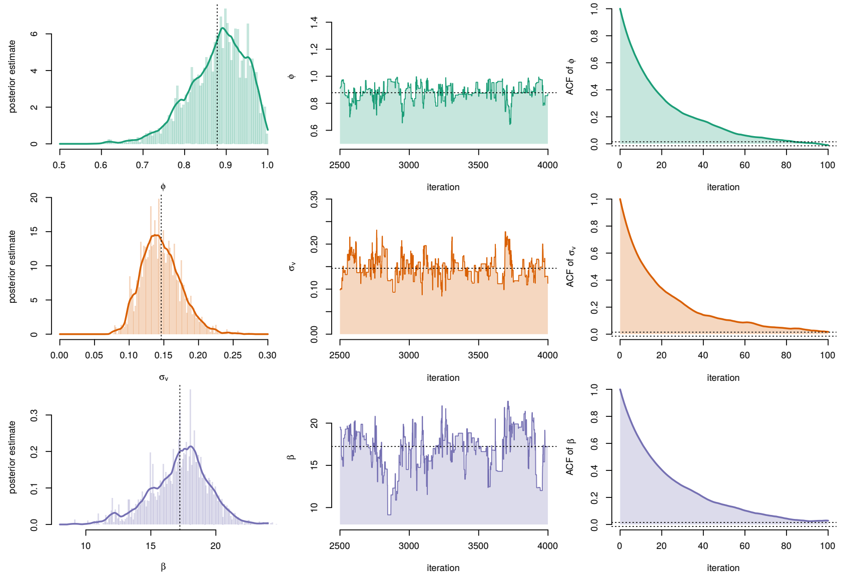
Hints:

- `pf` returns $\log p(y|\theta)$.
- `observations` contains $y_{1:T}$.
- `theta` contains θ_k.

Implement: the particle Metropolis-Hastings algorithm

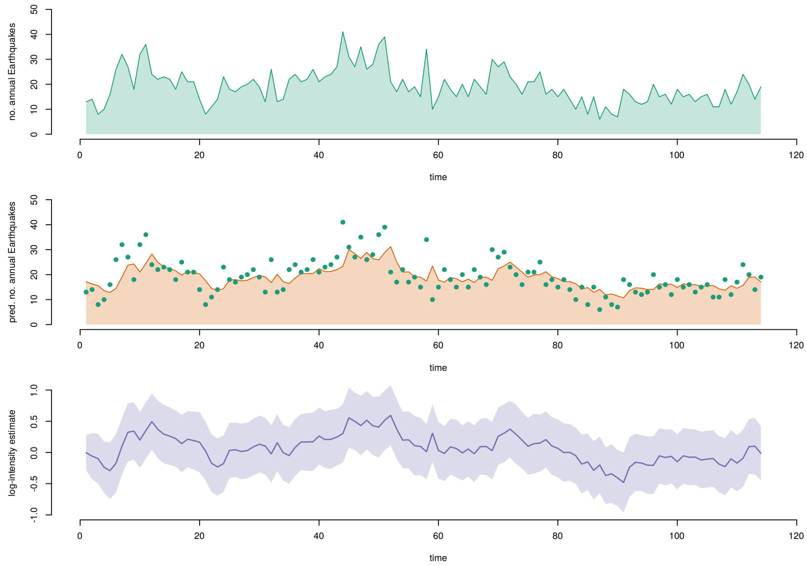
```
% Estimate the log-posterior (don't run if unstable system)
if (      ( abs( theta_proposed(1) ) < 1  ) &&
          ( theta_proposed(2) > 0         ) &&
          ( theta_proposed(3) > 0         ) )

  [ ~, logposterior_proposed ] =
            pf( observations, theta_proposed, nParticles );
else
  logposterior_proposed = -inf;
end
```

Results for the time series model: parameters



LINKÖPING UNIVERSITY

Results for the time series model: state



Some open problems

- Selection of N (no. particles) versus K (no. iterations).

  Correlated particle filters & rules-of-thumb.

  G. Deligiannidis, A. Doucet, and M.K. Pitt, The Correlated Pseudo-Marginal Method. arxiv:1511.04992, 2015.

  J. Dahlin, F. Lindsten, J. Kronander and T.B. Schön, Accelerating pseudo-marginal Metropolis-Hastings by correlating auxiliary variables. arxiv:1511.05483, 2015.

  A. Doucet, M.K. Pitt, G. Deligiannidis, and R. Kohn, Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. Biometrika, 102(2):295–313, 2015.

- Selection of $q(\theta'|\theta)$ or Σ (parameter proposal).

  Including gradients and Hessians & rules-of-thumb.

  J. Dahlin, F. Lindsten and T.B. Schön, Particle Metropolis-Hastings using gradient and Hessian information. Statistics and Computing 25(1), pp. 81-92, Springer, 2015.

  C. Sherlock, A.H. Thiery, G.O. Roberts, and J.S. Rosenthal, On the efficency of pseudo-marginal random walk Metropolis algorithms. The Annals of Statistics, 43(1), pp. 238-275, 2015.

- Better particle filters.

  Variance reduction and scaling in dim(x_t).

Why did we do this?

Introduce some popular statistical algorithms.
Exemplify applications in system identification.

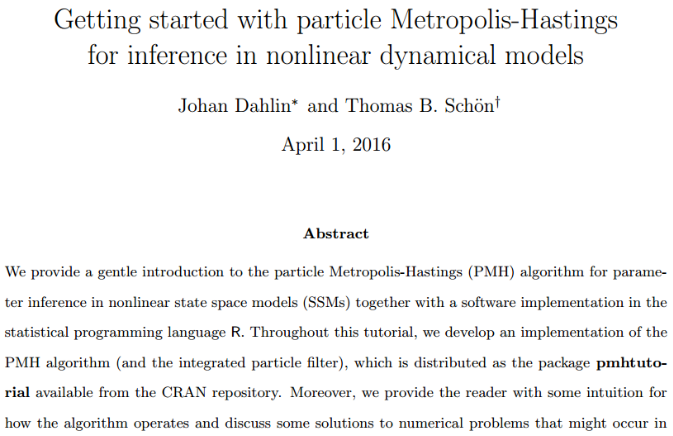
How did we do it?

You implemented the algorithm yourself.
Discussed some theory and challenges.

What did we achieve?

A fairly general implementation of PMH.
Can be used for inference in (any) scalar SSM.
Simple to tailor to your own specific problem.

Would you like to know more?



Getting started with particle Metropolis-Hastings for inference in nonlinear dynamical models

Johan Dahlin* and Thomas B. Schön†

April 1, 2016

Abstract

We provide a gentle introduction to the particle Metropolis-Hastings (PMH) algorithm for parameter inference in nonlinear state space models (SSMs) together with a software implementation in the statistical programming language R. Throughout this tutorial, we develop an implementation of the PMH algorithm (and the integrated particle filter), which is distributed as the package **pmhtutorial** available from the CRAN repository. Moreover, we provide the reader with some intuition for how the algorithm operates and discuss some solutions to numerical problems that might occur in

Complete tutorial is available at arXiv:1511.01707

Thank you for listening

Comments, suggestions and/or questions?

Johan Dahlin

johan.dahlin@liu.se

work.johandahlin.com

Complete tutorial is available at arXiv:1511.01707

LINKÖPING UNIVERSITY

Particle filtering

Three steps to approximate the state and the likelihood:

(i) Resample the particle $x_{t-1}^{(i)}$ using $\{w_{t-1}^{(i)}\}_{i=1}^{N}$ to obtain $\tilde{x}_{t-1}^{(i)}$.

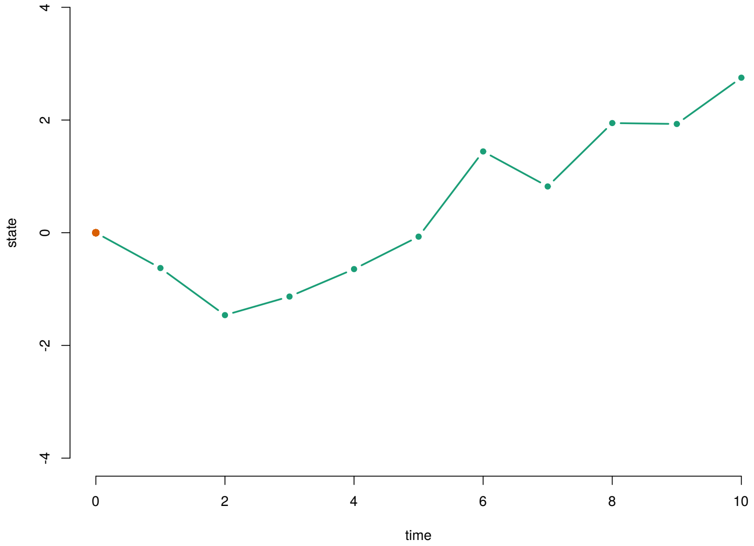
(ii) Propagate the particle by

$$x_t^{(i)} \sim R_\theta\left(x_t^{(i)} | \tilde{x}_{t-1}^{(i)}\right).$$

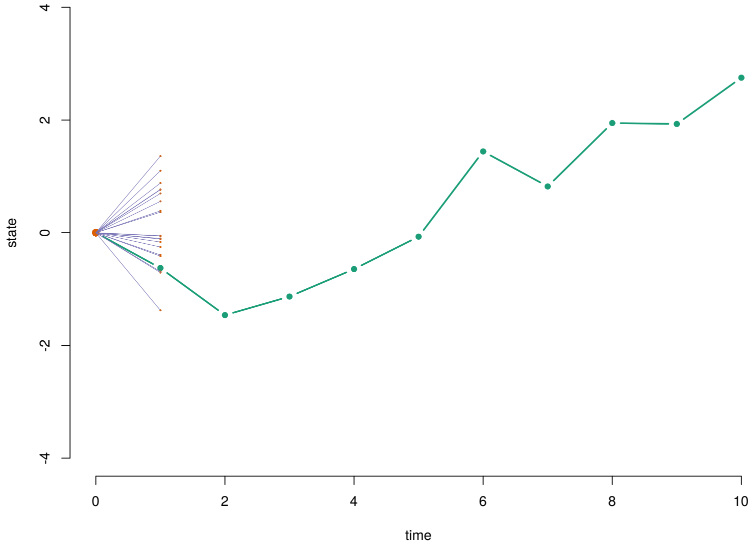
(iii) Compute the weight for the particle by

$$\tilde{w}_t^{(i)} = W\left(x_t^{(i)}\right), \quad w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^{N} \tilde{w}_t^{(j)}}.$$

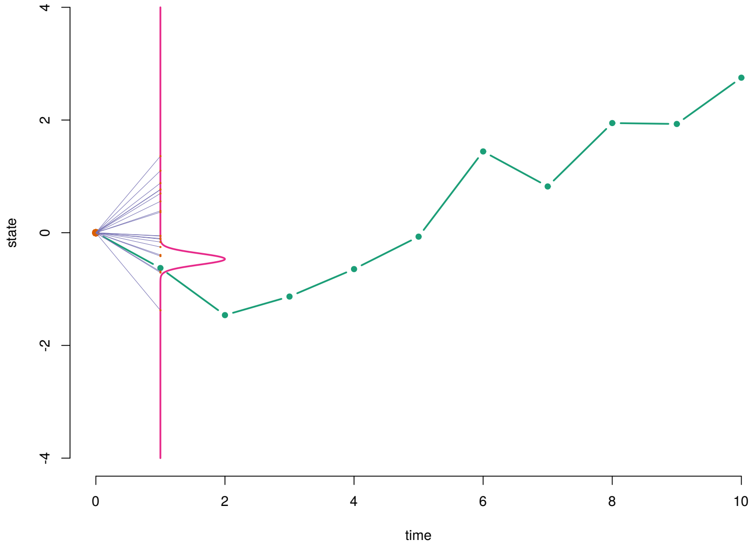
Particle filtering: toy example



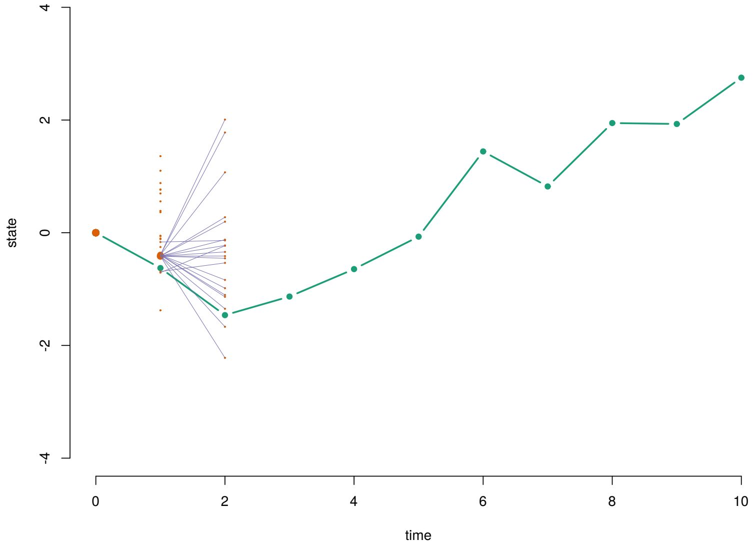
Particle filtering: toy example



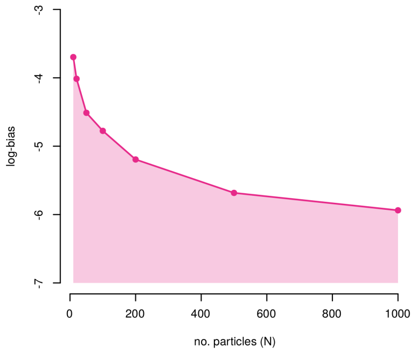
Particle filtering: toy example

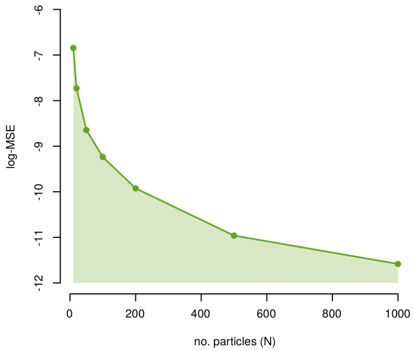


Particle filtering: toy example



The particle filter approximates the optimal filter





Bootstrap particle filtering for the Earthquake model

$$\begin{aligned} x_{t+1}|x_t &\sim \mathcal{N}\Big(x_{t+1}; \phi x_t, \sigma^2\Big), \\ y_t|x_t &\sim \mathcal{P}\Big(y_t; \beta \exp\left(x_t\right)\Big), \end{aligned}$$

We implement the bootstrap version by selecting

$$R_\theta\left(x_t^{(i)}|\tilde{x}_{t-1}^{(i)}\right) = f_\theta\left(x_t^{(i)}|\tilde{x}_{t-1}^{(i)}\right) = \mathcal{N}\left(x_t^{(i)}; \phi \tilde{x}_{t-1}^{(i)}, \sigma^2\right),$$

which results in

$$W\left(x_t^{(i)}\right) = g_\theta\left(y_t|x_t^{(i)}\right) = \mathcal{P}\left(y_t; \beta \exp\left(x_t^{(i)}\right)\right).$$

Some challenges

- How do we choose N, R_θ and W_θ in the particle filter?
- How do we choose K and q in the PMH algorithm?
- Scalability in T and p?

- Rule-of-thumbs (does not always work):
  - Choose N such that

$$\mathbb{V}\left[\log \widehat{p}^N(y|\theta)\right] \approx 1.4.$$

  - Choose q (if target is close to a Gaussian) to be

$$q(\theta'|\theta_{k-1}) = \mathcal{N}\left(\theta'; \theta_{k-1}, \frac{2.562^2}{p}\widehat{\Sigma}\right),$$

    or use adaptive algorithms.