Biips: A software for **B**ayesian **i**nference with **i**nteracting **p**article **s**ystems

Compstat 2014

Adrien Todeschini[†], François Caron[*], Pierrick Legrand[†], Pierre Del Moral[‡] and Marc Fuentes[†]

[†]Inria Bordeaux, [*]Univ. Oxford, [‡]UNSW Sydney

Geneva, August 2014

Outline

Summary

Context

BUGS

SMC

Matbiips

Particle MCMC

Context

Biips = **B**ayesian **i**nference with **i**nteracting **p**article **s**ystems

Bayesian inference

- Sample from a posterior distribution $p(X|Y) = \frac{p(X,Y)}{p(Y)}$
- High dimensional, arbitrary complexity
- Stochastic simulation: MCMC, SMC...

Motivation

- Last 20 years: success of SMC in many applications
- No general and easy-to-use software for SMC

Objectives

- Inference in graphical models defined in BUGS language
- Use SMC methods as inference engine instead of MCMC
- User-friendly, "black-box" implementation

Context

Biips $=$ **B**ayesian **i**nference with **i**nteracting **p**article **s**ystems

Bayesian inference

- Sample from a posterior distribution $p(X|Y) = \frac{p(X,Y)}{p(Y)}$
- High dimensional, arbitrary complexity
- Stochastic simulation: MCMC, SMC...

Motivation

- Last 20 years: success of SMC in many applications
- No general and easy-to-use software for SMC

Objectives

- Inference in graphical models defined in BUGS language
- Use SMC methods as inference engine instead of MCMC
- User-friendly, "black-box" implementation

Context

Biips = **B**ayesian **i**nference with **i**nteracting **p**article **s**ystems

Bayesian inference

- Sample from a posterior distribution $p(X|Y) = \frac{p(X,Y)}{p(Y)}$
- High dimensional, arbitrary complexity
- Stochastic simulation: MCMC, SMC...

Motivation

- Last 20 years: success of SMC in many applications
- No general and easy-to-use software for SMC

Objectives

- Inference in graphical models defined in BUGS language
- Use SMC methods as inference engine instead of MCMC
- User-friendly, "black-box" implementation

Summary

BUGS

What is BUGS?

- A language for defining Bayesian graphical models
- A "black-box" inference engine using MCMC

[Gilks et al., 1994]

Bayesian graphical models

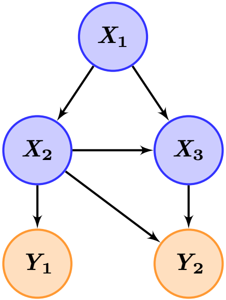


Figure : Directed acyclic graph

The graph displays a factorization of the joint distribution:

$$p(x_{1:3}, y_{1:2})$$

Bayesian graphical models

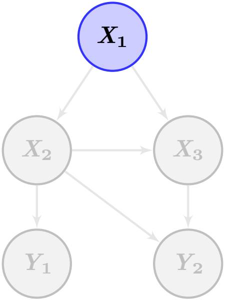


Figure : Directed acyclic graph

The graph displays a factorization of the joint distribution:

$$p(x_{1:3}, y_{1:2}) = p(x_1)\ p(x_2|x_1)\ p(y_1|x_2)\ p(x_3|x_1, x_2)\ p(y_2|x_2, x_3)$$

Bayesian graphical models

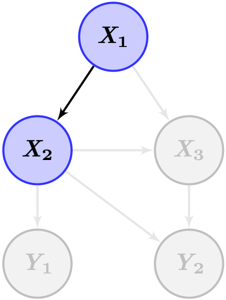


Figure : Directed acyclic graph

The graph displays a factorization of the joint distribution:

$$p(x_{1:3}, y_{1:2}) = p(x_1)\ p(x_2|x_1)\ p(y_1|x_2)\ p(x_3|x_1, x_2)\ p(y_2|x_2, x_3)$$

Bayesian graphical models

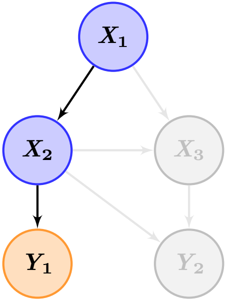


Figure : Directed acyclic graph

The graph displays a factorization of the joint distribution:

$$p(x_{1:3}, y_{1:2}) = p(x_1)\ p(x_2|x_1)\ p(y_1|x_2)\ p(x_3|x_1, x_2)\ p(y_2|x_2, x_3)$$

Bayesian graphical models

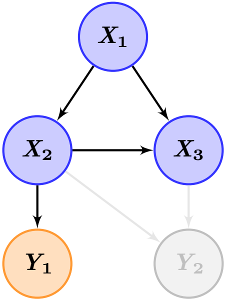


Figure : Directed acyclic graph

The graph displays a factorization of the joint distribution:

$$p(x_{1:3}, y_{1:2}) = p(x_1)\ p(x_2|x_1)\ p(y_1|x_2)\ p(x_3|x_1, x_2)\ p(y_2|x_2, x_3)$$

Bayesian graphical models

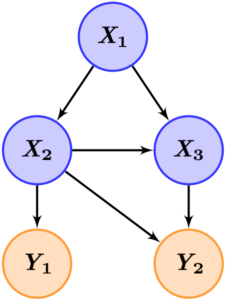


Figure : Directed acyclic graph

The graph displays a factorization of the joint distribution:

$$p(x_{1:3}, y_{1:2}) = p(x_1)\ p(x_2|x_1)\ p(y_1|x_2)\ p(x_3|x_1, x_2)\ p(y_2|x_2, x_3)$$

BUGS language

- S-like declarative language for describing graphical models
- Stochastic relations
- Deterministic relations

BUGS language

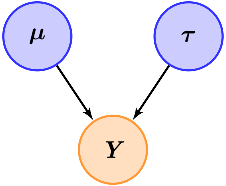
- S-like declarative language for describing graphical models
- Stochastic relations
- Deterministic relations

Linear regression:

```
model {
    Y ~ dnorm(mu, tau)



}
```



BUGS language

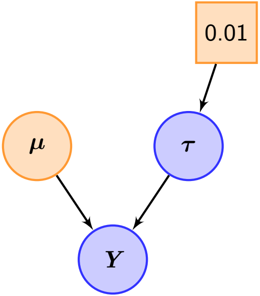
- S-like declarative language for describing graphical models
- Stochastic relations
- Deterministic relations

Linear regression:

```
model {
    Y ~ dnorm(mu, tau)
    tau ~ dgamma(0.01, 0.01)



}
```



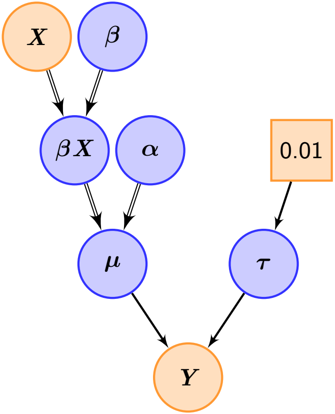
BUGS language

- S-like declarative language for describing graphical models
- Stochastic relations
- Deterministic relations

Linear regression:

```
model {
    Y ~ dnorm(mu, tau)
    tau ~ dgamma(0.01, 0.01)
    mu <- beta * X + alpha


}
```

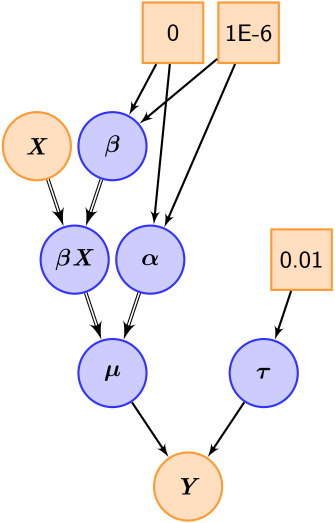


BUGS language

- S-like declarative language for describing graphical models
- Stochastic relations
- Deterministic relations

Linear regression:

```
model {
    Y ~ dnorm(mu, tau)
    tau ~ dgamma(0.01, 0.01)
    mu <- beta * X + alpha
    alpha ~ dnorm(0, 1E-6)
    beta ~ dnorm(0, 1E-6)
}
```

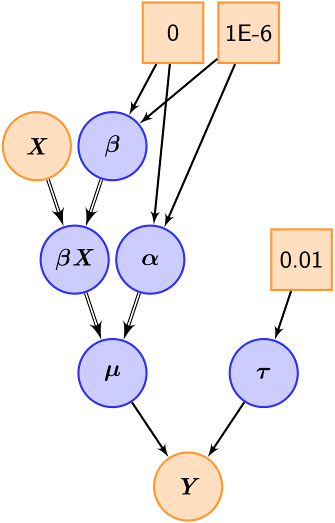


BUGS language

- S-like declarative language for describing graphical models
- Stochastic relations
- Deterministic relations

Linear regression:

```
model {
    Y ∼ dnorm(mu, tau)
    tau ∼ dgamma(0.01, 0.01)
    mu <- beta * X + alpha
    alpha ∼ dnorm(0, 1E-6)
    beta ∼ dnorm(0, 1E-6)
}
```



Goal:

Estimate $p(\alpha, \beta, \tau | X, Y)$

BUGS software

- Expert system automatically derives MCMC methods (Gibbs, Slice, Metropolis, ...) in a 'black-box' fashion
- Very popular among practitioners, applying MCMC methods to a wide range of applications [Lunn et al., 2012]
- Similar software: WinBUGS, OpenBUGS, JAGS [Plummer, 2012], Stan [Stan Development Team, 2013]

Summary

SMC Algorithm

- A.k.a. interacting MCMC, particle filtering, sequential Monte Carlo methods (SMC) ...
- Algorithms designed to sequentially sample from sequence of target distributions of increasing dimension

$$\pi_1(x_1) \to \pi_2(x_{1:2}) \to \dots \to \pi_T(x_{1:T})$$

where, for $t = 1, \dots, T$

$$\pi_t(x_{1:t}) = \pi_{t-1}(x_{1:t-1}) \frac{q_t(x_t | x_{1:t-1}) \; \alpha_t(x_{1:t})}{z_t}$$

Two stochastic mechanisms:

- **Mutation/Exploration**
- **Selection**

[Doucet et al., 2001, Del Moral, 2004, Doucet and Johansen, 2010]

SMC Algorithm

Standard SMC algorithm

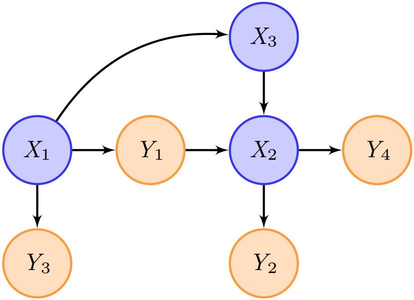
For $t = 1, \ldots, T$:

- For $i = 1, \ldots, N$, sample: $X_t^{(i)} \sim q_t(x_t | \widetilde{X}_{1:t-1}^{(i)})$ and set $X_{1:t}^{(i)} = \{\widetilde{X}_{1:t-1}^{(i)}, X_t^{(i)}\}$
- For $i = 1, \ldots, N$, weight: $w_t^{(i)} = \alpha_t(X_{1:t}^{(i)})$
- For $i = 1, \ldots, N$, normalize: $W_t^{(i)} = \frac{w_t^{(i)}}{\sum_{i=1}^N w_t^{(i)}}$,
- Resample $\{X_{1:t}^{(i)}, W_t^{(i)}\}_{i=1,\ldots,N} \to \{\widetilde{X}_{1:t}^{(i)}, \frac{1}{N}\}_{i=1,\ldots,N}$

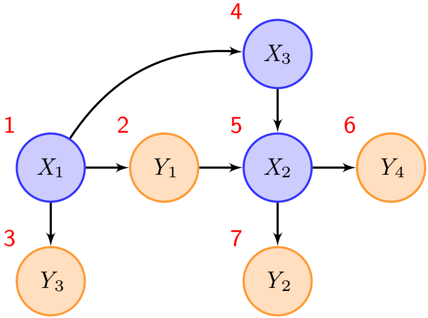
Outputs

- Weighted particles: $\{X_{1:t}^{(i)}, W_t^{(i)}\}_{i=1,\ldots,N}$ for $t = 1, \ldots, T$
- Normalizing constant (unbiased): $\hat{Z}_T = \prod_{t=1}^T \frac{1}{N} \sum_{i=1}^N w_t^{(i)}$

SMC for graphical models

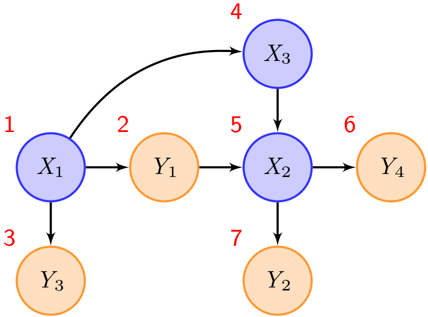


SMC for graphical models

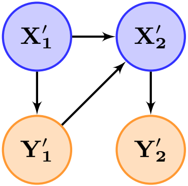


Topological sort (with priority to measurement nodes):
$(X_1, Y_1, Y_3, X_3, X_2, Y_4, Y_2)$

SMC for graphical models



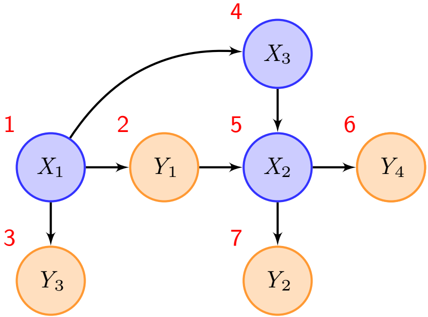
Rearrangement of the directed acyclic graph:



Topological sort (with priority to measurement nodes):

$$(\underbrace{X_1}_{\mathbf{X}'_1}, \underbrace{Y_1, Y_3}_{\mathbf{Y}'_1}, \underbrace{X_3, X_2}_{\mathbf{X}'_2}, \underbrace{Y_4, Y_2}_{\mathbf{Y}'_2})$$

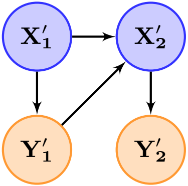
SMC for graphical models



Topological sort (with priority to measurement nodes):

$$(\underbrace{X_1}_{\mathbf{X}_1'}, \underbrace{Y_1, Y_3}_{\mathbf{Y}_1'}, \underbrace{X_3, X_2}_{\mathbf{X}_2'}, \underbrace{Y_4, Y_2}_{\mathbf{Y}_2'})$$

Rearrangement of the directed acyclic graph:



Sequence of conditional distributions:

$$p(\boldsymbol{X}_1' \,|\, \boldsymbol{Y}_1')$$
$$\downarrow$$
$$p(\boldsymbol{X}_1', \boldsymbol{X}_2' \,|\, \boldsymbol{Y}_1', \boldsymbol{Y}_2')$$

SMC for graphical models

Sequence of conditional distributions

$$
\begin{array}{ccccccc}
\pi_1(x_1) & \to & \pi_2(x_{1:2}) & \to \dots \to & \pi_T(x_{1:T}) \\
p(x_1|y_1) & \to & p(x_{1:2}|y_{1:2}) & \to \dots \to & p(x_{1:T}|y_{1:T})
\end{array}
$$

$$
\begin{array}{lccccc}
\text{Filtering:} & p(x_1|y_1) & \to & p(x_2|y_{1:2}) & \to \dots \to & p(x_T|y_{1:T}) \\
\text{Smoothing:} & p(x_1|y_{1:T}) & \to & p(x_2|y_{1:T}) & \to \dots \to & p(x_T|y_{1:T})
\end{array}
$$

where, for $t = 1, \dots, T$

$$
p(x_{1:t}|y_{1:t}) = p(x_{1:t-1}|y_{1:t-1}) \frac{p(x_t|x_{1:t-1}, y_{1:t-1})\, p(y_t|x_{1:t}, y_{1:t-1})}{p(y_t|y_{1:t-1})}
$$

Simplification:

$$
p(x_{1:t}|y_{1:t}) = p(x_{1:t-1}|y_{1:t-1}) \frac{p(x_t|\mathrm{pa}(x_t))\, p(y_t|\mathrm{pa}(y_t))}{p(y_t|y_{1:t-1})}
$$

SMC for graphical models

Sequence of conditional distributions

$$\begin{array}{ccccc} \pi_1(x_1) & \rightarrow & \pi_2(x_{1:2}) & \rightarrow \dots \rightarrow & \pi_T(x_{1:T}) \\ p(x_1|y_1) & \rightarrow & p(x_{1:2}|y_{1:2}) & \rightarrow \dots \rightarrow & p(x_{1:T}|y_{1:T}) \end{array}$$

$$\begin{array}{llcccc} \text{Filtering:} & p(x_1|y_1) & \rightarrow & p(x_2|y_{1:2}) & \rightarrow \dots \rightarrow & p(x_T|y_{1:T}) \\ \text{Smoothing:} & p(x_1|y_{1:T}) & \rightarrow & p(x_2|y_{1:T}) & \rightarrow \dots \rightarrow & p(x_T|y_{1:T}) \end{array}$$

where, for $t = 1, \dots, T$

$$p(x_{1:t}|y_{1:t}) = p(x_{1:t-1}|y_{1:t-1}) \frac{p(x_t|x_{1:t-1}, y_{1:t-1})\ p(y_t|x_{1:t}, y_{1:t-1})}{p(y_t|y_{1:t-1})}$$

Simplification:

$$p(x_{1:t}|y_{1:t}) = p(x_{1:t-1}|y_{1:t-1}) \frac{p(x_t|\text{pa}(x_t))\ p(y_t|\text{pa}(y_t))}{p(y_t|y_{1:t-1})}$$

SMC for graphical models

Sequence of conditional distributions

$$\begin{array}{ccccc} \pi_1(x_1) & \to & \pi_2(x_{1:2}) & \to \dots \to & \pi_T(x_{1:T}) \\ p(x_1|y_1) & \to & p(x_{1:2}|y_{1:2}) & \to \dots \to & p(x_{1:T}|y_{1:T}) \end{array}$$

$$\begin{array}{lcccc} \text{Filtering:} & p(x_1|y_1) & \to & p(x_2|y_{1:2}) & \to \dots \to & p(x_T|y_{1:T}) \\ \text{Smoothing:} & p(x_1|y_{1:T}) & \to & p(x_2|y_{1:T}) & \to \dots \to & p(x_T|y_{1:T}) \end{array}$$

where, for $t = 1, \dots, T$

$$p(x_{1:t}|y_{1:t}) = p(x_{1:t-1}|y_{1:t-1}) \frac{p(x_t|x_{1:t-1}, y_{1:t-1})\; p(y_t|x_{1:t}, y_{1:t-1})}{p(y_t|y_{1:t-1})}$$

Simplification:

$$p(x_{1:t}|y_{1:t}) = p(x_{1:t-1}|y_{1:t-1}) \frac{p(x_t|\mathrm{pa}(x_t))\; p(y_t|\mathrm{pa}(y_t))}{p(y_t|y_{1:t-1})}$$

SMC for graphical models

Sequence of conditional distributions

$$\begin{array}{ccccc} \pi_1(x_1) & \rightarrow & \pi_2(x_{1:2}) & \rightarrow \dots \rightarrow & \pi_T(x_{1:T}) \\ p(x_1|y_1) & \rightarrow & p(x_{1:2}|y_{1:2}) & \rightarrow \dots \rightarrow & p(x_{1:T}|y_{1:T}) \end{array}$$

$$\begin{array}{lcccccc} \text{Filtering:} & p(x_1|y_1) & \rightarrow & p(x_2|y_{1:2}) & \rightarrow \dots \rightarrow & p(x_T|y_{1:T}) \\ \text{Smoothing:} & p(x_1|y_{1:T}) & \rightarrow & p(x_2|y_{1:T}) & \rightarrow \dots \rightarrow & p(x_T|y_{1:T}) \end{array}$$

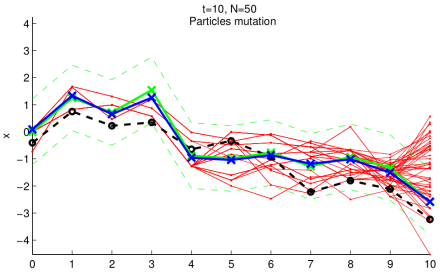
where, for $t = 1, \dots, T$

$$p(x_{1:t}|y_{1:t}) = p(x_{1:t-1}|y_{1:t-1}) \frac{p(x_t|x_{1:t-1}, y_{1:t-1})\ p(y_t|x_{1:t}, y_{1:t-1})}{p(y_t|y_{1:t-1})}$$

Simplification:

$$p(x_{1:t}|y_{1:t}) = p(x_{1:t-1}|y_{1:t-1}) \frac{p(x_t|\mathrm{pa}(x_t))\ p(y_t|\mathrm{pa}(y_t))}{p(y_t|y_{1:t-1})}$$

Limitations of SMC algorithms



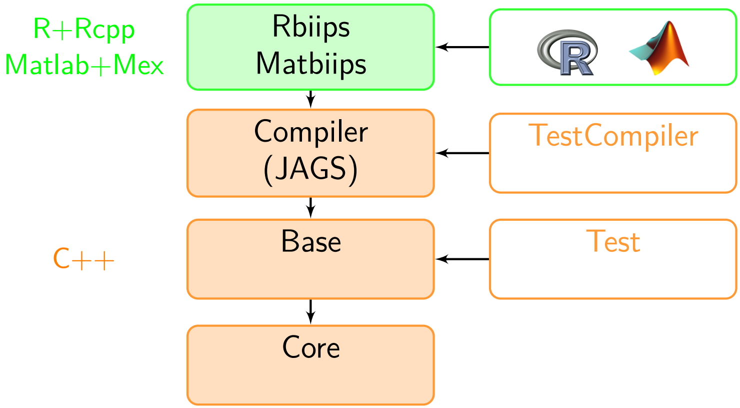
At time $t = 1, ..., T$, for each unique ancestor $X_t'^{(k)}$, $k = 1, \ldots, K_t$, let $W_t'^{(k)} = \sum_{i | X_t^{(i)} = X_t'^{(k)}} W_T^{(i)}$ be its associated total weight.

Smoothing Effective Sample Size (SESS):

$$\mathrm{SESS}_t = \frac{1}{\sum_{k=1}^{K_t} (W_t'^{(k)})^2} \quad \in [1, N]$$

Summary

Technical implementation



- Interfaces: Matlab/Octave, R
- Multi-platform: Windows, Linux, Mac OSX
- Free and open source (GPL)

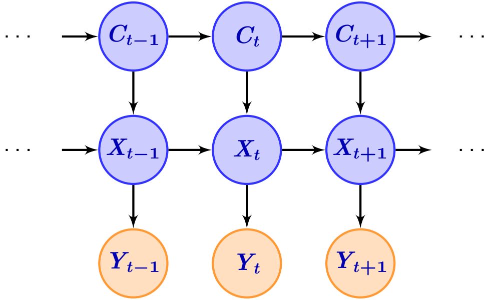
Switching Stochastic Volatility (SSV)

Let Y_t be the response variable and X_t the unobserved log-volatility of Y_t. For $t = 1, \dots, T$

$$X_t | (X_{t-1} = x_{t-1}, C_t = c_t) \sim \mathcal{N}(\alpha_{c_t} + \phi x_{t-1}, \sigma^2)$$
$$Y_t | X_t = x_t \sim \mathcal{N}(0, \exp(x_t))$$

The regime variables C_t follow a two-state Markov process with transition probabilities

$$p_{ij} = \Pr(C_t = j | C_{t-1} = i), \quad \text{for } i, j = 1, 2$$



SSV model in BUGS language

switch_stoch_volatility.bug

```
model
{
  c[1] ~ dcat(pi[c0,])
  mu[1,1] <- alpha[1,1] * (c[1]==1) + alpha[2,1]*(c[1]==2)  + phi*
      x0
  x[1,1] ~ dnorm(mu[1,1], 1/sigma^2)
  y[1,1] ~ dnorm(0, exp(-x[1,1]))
  for (t in 2:t_max)
  {
    c[t] ~ dcat(ifelse(c[t-1]==1, pi[1,], pi[2,]))
    mu[t,1] <- alpha[1,1] * (c[t]==1) + alpha[2,1]*(c[t]==2) + phi*
        x[t-1,1]
    x[t,1] ~ dnorm(mu[t,1], 1/sigma^2)
    y[t,1] ~ dnorm(0, exp(-x[t,1]))
  }
}
```

Model compilation

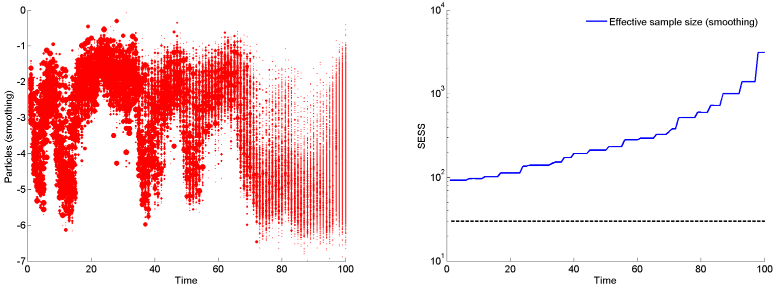
Matbiips

```
% Model parameters
sigma = .4;alpha = [-2.5; -1]; phi = .5; c0 = 1; x0 = 0; t_max =
    200;
pi = [.9, .1; .1, .9];
data = struct('t_max', t_max, 'sigma', sigma,...
        'alpha', alpha, 'phi', phi, 'pi', pi, 'c0', c0, 'x0', x0);
model_filename = 'switch_stoch_volatility.bug'; % BUGS model
    filename
% Parse and compile BUGS model, and sample data
model = biips_model(model_filename, data, 'sample_data', true);
data = model.data;
```

SMC samples

Matbiips

```
n_part = 5000; % Number of particles
variables = {'x'}; % Variables be monitored
% Run SMC
out_smc = biips_smc_samples(model, variables, n_part);
% Diagnostic on the SMC output
diag = biips_diagnostic(out_smc);
```



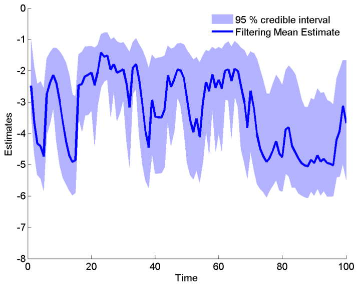
(a) Set of weighted particles of the posterior distribution for the switching stochastic volatility model.

(b) Smoothing Effective sample size with respect to t.

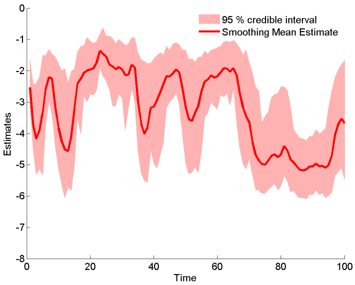
Summary statistics

Matbiips

```
summary = biips_summary(out_smc, 'probs', [.025, .975]); % Summary
    statistics
x_f_mean = summary.x.f.mean; x_f_quant = summary.x.f.quant;
x_s_mean = summary.x.s.mean; x_s_quant = summary.x.s.quant;
```



(c) Filtering

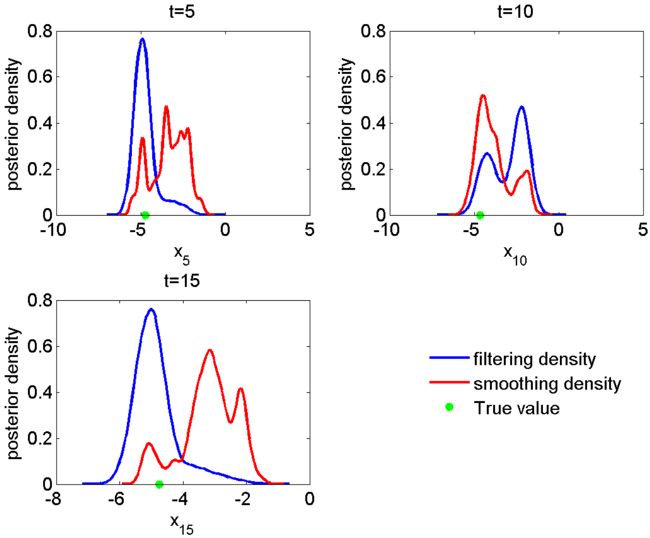


(d) Smoothing

Kernel density estimates

Matbiips

```
kde_estimates = biips_density(out_smc);% kernel density estimates
```



Summary

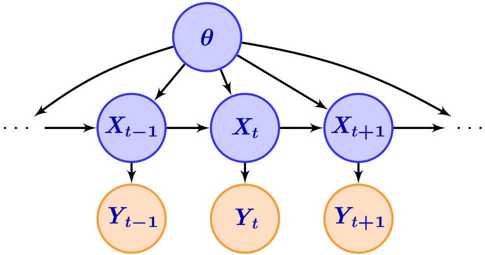
Particle MCMC

Recent algorithms that use SMC algorithms within a MCMC algorithm

- Particle Independant Metropolis-Hastings (PIMH)
- Particle Marginal Metropolis-Hastings (PMMH)

[Andrieu et al., 2010]

Static parameter estimation



Due to the successive resamplings, SMC estimations of $p(\theta|y_{1:T})$ might be poor.

The PMMH splits the variables in the graphical model into two sets:

- a set of variables X that will be sampled using a SMC algorithm
- a set $\theta = (\theta_1, \ldots, \theta_p)$ sampled with a MH proposal

PMMH

Standard PMMH algorithm

Set $\widehat{Z}(0) = 0$ and initialize $\theta(0)$

For $k = 1, \ldots, n_{\text{iter}}$,

- Sample $\theta^\star \sim q$
- Run a SMC to approximate $p(x_{1:T} | y_{1:T}, \theta^\star)$ with output $(X_{1:T}^{\star(i)}, W_T^{\star(i)})_{i=1,\ldots,N}$ and $\widehat{Z}^\star$
- With probability

$$\min\left(1, \frac{\widehat{Z}^\star}{\widehat{Z}(k-1)}\right)$$

  set $X_{1:T}(k) = X_{1:n}^{\star(\ell)}$, $\theta(k) = \theta^\star$ and $\widehat{Z}(k-1) = \widehat{Z}^\star$, where $\ell \sim \mathbf{Discrete}(W_T^{\star(1)}, \ldots, W_T^{\star(N)})$
- otherwise, keep previous iteration values

Outputs

- MCMC samples $(X_{1:n}(k), \theta(k))_{k=1,\ldots,n_{\text{iter}}}$

Static parameter estimation in the SSV model

We consider the following prior on the parameters $\boldsymbol{\alpha}$, $\boldsymbol{\pi}$, $\boldsymbol{\phi}$ and $\boldsymbol{\tau}$:

$$\begin{aligned} \alpha_1 &= \gamma_1 \\ \alpha_2 &= \gamma_1 + \gamma_2 \\ \gamma_1 &\sim \mathcal{N}(0, 100) \\ \gamma_2 &\sim \mathcal{TN}_{(0,+\infty)}(0, 100) \end{aligned}$$

$$\begin{aligned} \frac{1}{\sigma^2} &\sim \mathrm{Gamma}(2.001, 1) \\ \phi &\sim \mathcal{TN}_{(-1,1)}(0, 100) \\ \pi_{11} &\sim \mathrm{Beta}(.5, .5) \\ \pi_{22} &\sim \mathrm{Beta}(.5, .5) \end{aligned}$$

[Carvalho and Lopes, 2007]

SSV model with unknown parameters in BUGS language

switch_stoch_volatility_param.bug

```
model
{
  gamma[1,1] ~ dnorm(0, 1/100)
  gamma[2,1] ~ dnorm(0, 1/100)T(0,)
  alpha[1,1] <- gamma[1,1]
  alpha[2,1] <- gamma[1,1] + gamma[2,1]
  phi ~ dnorm(0, 1/100)T(-1,1)
  tau ~ dgamma(2.001, 1)
  sigma <- 1/sqrt(tau)
  pi[1,1] ~ dbeta(.5, .5)
  pi[1,2] <- 1.00 - pi[1,1]
  pi[2,2] ~ dbeta(.5, .5)
  pi[2,1] <- 1.00 - pi[2,2]
  ...
}
```

Matbiips

```
% *Compile BUGS model and sample data*
model_filename = 'switch_stoch_volatility_param.bug'; % BUGS model
    filename
model = biips_model(model_filename, data, 'sample_data',
    sample_data); % Create biips model and sample data
data = model.data;
```

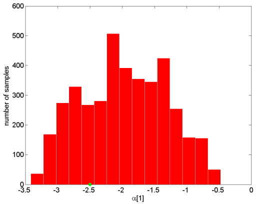
PMMH samples

Run a PMMH sampler to approximate $p(\alpha_1, \alpha_2, \sigma, \pi_{11}, \pi_{22}, \phi, X_{1,T}, C_{1:T} | Y_{1:T})$.

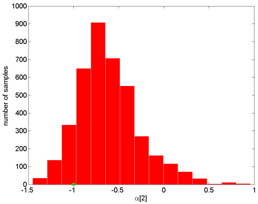
Matbiips

```
% *Parameters of the PMMH*
n_burn = 2000; % nb of burn-in/adaptation iterations
n_iter = 40000; % nb of iterations after burn-in
thin = 10; % thinning of MCMC outputs
n_part = 50; % nb of particles for the SMC
param_names = {'gamma[1,1]','gamma[2,1]', 'phi', 'tau', 'pi[1,1]',
    'pi[2,2]'}; % name of the variables updated with MCMC (others
    are updated with SMC)
latent_names = {'x','alpha[1,1]','alpha[2,1]', 'sigma'}; % name of
    the variables updated with SMC and that need to be monitored
% *Init PMMH*
inits = {-1, 1,.5,5,.8,.8};
obj_pmmh = biips_pmmh_init(model, param_names, 'inits', inits, '
    latent_names', latent_names); % creates a pmmh object
% *Run PMMH*
[obj_pmmh, stats_pmmh_update] = biips_pmmh_update(obj_pmmh, n_burn,
     n_part); % adaptation and burn-in iterations
[obj_pmmh, out_pmmh, log_post, log_marg_like, stats_pmmh] =...
    biips_pmmh_samples(obj_pmmh, n_iter, n_part,'thin', thin); %
    Samples
```

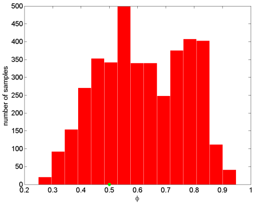
Posterior samples



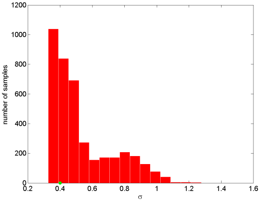
(e) α_1



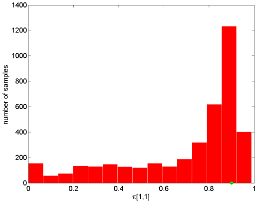
(f) α_2



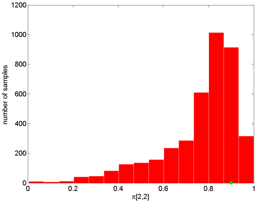
(g) ϕ



(h) σ



(i) π_{11}



(j) π_{22}

Other features of *Biips*

- Backward smoothing algorithm
- Particle Independent Metropolis-Hastings algorithm
- Automatic choice of the proposal distribution including **Optimal/Conditional samplers**: Gaussian-Gaussian, Beta-Bernouilli, Finite discrete
- Easy BUGS language extensions with user-defined Matlab/R functions

Bibliography I

Andrieu, C., Doucet, A., and Holenstein, R. (2010).
Particle Markov chain Monte Carlo methods.
Journal of the Royal Statistical Society B, 72:269–342.

Carvalho, C. M. and Lopes, H. F. (2007).
Simulation-based sequential analysis of Markov switching stochastic volatility models.
Computational Statistics & Data Analysis, 51(9):4526–4542.

Del Moral, P. (2004).
Feynman-Kac formulae. Genealogical and interacting particle systems with application.
Springer.

Doucet, A., de Freitas, N., and Gordon, N., editors (2001).
Sequential Monte Carlo Methods in practice.
Springer-Verlag.

Doucet, A. and Johansen, A. (2010).
A tutorial on particle filtering and smoothing: fifteen years later.
In Crisan, D. and Rozovsky, B., editors, *Oxford Handbook of Nonlinear Filtering*. Oxford University Press.

Gilks, W., Thomas, A., and Spiegelhalter, D. (1994).
A language and program for complex Bayesian modelling.
The Statistician, 43:169–177.

Bibliography II

Lunn, D., Jackson, C., Best, N., Thomas, A., and Spiegelhalter, D. (2012).
The BUGS Book: A Practical Introduction to Bayesian Analysis.
CRC Press/ Chapman and Hall.

Plummer, M. (2012).
JAGS Version 3.3.0 user manual.

Stan Development Team (2013).
Stan: A c++ library for probability and sampling, version 2.1.

THANK YOU

http://alea.bordeaux.inria.fr/biips