# BiiPS: A software for inference in Bayesian graphical models with sequential Monte Carlo methods

Adrien Todeschini†, François Caron†, Pierrick Legrand† and Pierre Del Moral†

† Inria Bordeaux, Institut de Mathématiques de Bordeaux, Université de Bordeaux, France

## Context

The main factor in the success of MCMC methods is that they can be implemented with little efforts in a large variety of settings. Many softwares have been developed such as *WinBUGS* and *JAGS*, that helped to popularize Bayesian methods. These softwares allow the user to define his statistical model in a so-called BUGS language, then runs MCMC algorithms as a black box.

Although SMC methods have become a very popular class of numerical methods over the last 20 years, there is no such "black box software" for this class of methods. The *BiiPS* software aims at bridging this gap. From a graphical model defined in BUGS language, it automatically implements SMC algorithms and provides summaries of the posterior distributions.

## SMC/particle methods

- Based on interacting particles systems governed by two stochastic mechanisms:
  1. Mutation/Importance sampling: particles explore the space randomly and independently
  2. Selection/Resampling: the best suited particles are duplicated, others removed
- Designed to sample from a sequence of distributions $\pi_k(x_{1:k}) = p(x_{1:k}|y_{1:k})$ when we can only compute the unnormalized version $\gamma_k(x_{1:k})$

$$\pi_k(x_{1:k}) = \frac{p(x_{1:k}, y_{1:k})}{p(y_{1:k})} = \frac{\gamma_k(x_{1:k})}{Z_k}$$

### Generic SMC algorithm with N particles

- At time $1$: for $i = 1, \ldots, N$
  - Sample $x_1^{(i)} \sim q_1(x_1)$
  - Compute unnormalized weights $w_1^{(i)} = \frac{\gamma_1(x_1^{(i)})}{q_1(x_1^{(i)})}$ and estimate marginal likelihood $\hat{Z}_1 = \sum_{i=1}^N w_1^{(i)}$
- At time $k = 2, \ldots, T$: for $i = 1, \ldots, N$
  - Resample $\{x_{k-1}^{(i)}, w_{k-1}^{(i)}\}$ and set $w_{k-1}^{(i)} = 1/N$
  - Sample $x_k^{(i)} \sim q_k(x_k|x_{1:k-1})$
  - Compute unnormalized weights $w_k^{(i)} = w_{k-1}^{(i)} \frac{\gamma_k(x_{1:k}^{(i)})}{\gamma_{k-1}(x_{1:k-1}^{(i)}) q_k(x_k^{(i)}|x_{1:k-1}^{(i)})}$ and estimate marginal likelihood $\hat{Z}_k = \hat{Z}_{k-1} \sum_{i=1}^N w_k^{(i)}$

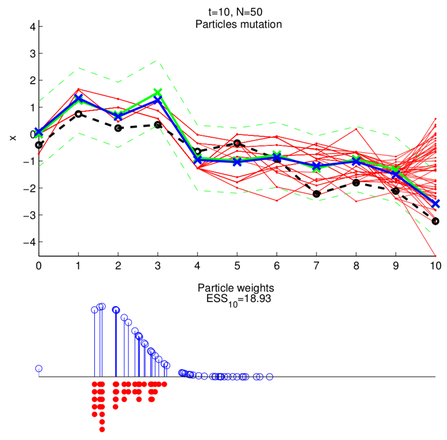


Figure: Particles genealogical tree

## Software features

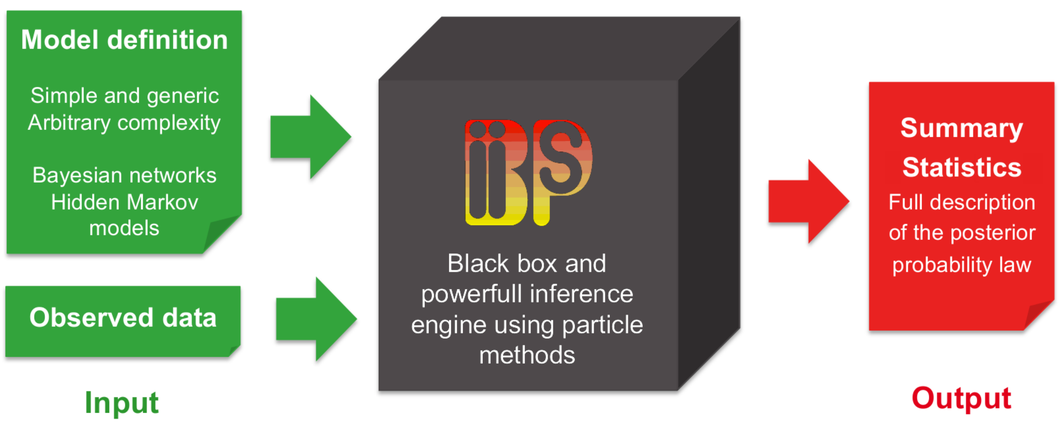


Figure: *BiiPS* input/output flowchart

### BUGS language compatible

- Includes most usual uni/multivariate continuous/discrete distributions
- Standard operators, usual functions, matrix operations, ...

### Development

- Free software adapted from *JAGS* © M. Plummer
- Core in C++ making use of Boost libraries
- R interface using *Rcpp* package
- Multi-platform: Linux, Windows, Mac

### SMC techniques

- Forward filtering
- Backward smoothing
- Usual resampling algorithms: multinomial, residual, stratified and systematic
- Conditional sampler for Gaussian conjugate prior

### Particle MCMC techniques

- Particle Independent Metropolis Hastings
- Particle Marginal Metropolis Hastings

## Future work

- Improve performance, parallelization, reduce memory footprint
- Interfaces: Matlab, Python, standalone
- More conjugate samplers, distributions and functions
- More advanced particle techniques
- Allow external user defined functions and samplers

## Example in financial econometrics

Consider infering the underlying volatility $x_{1:t}$ from observed price or rate data $y_{1:t}$

$$x_1 \sim \mathcal{N}(0, \tfrac{\sigma^2}{1-\alpha^2})$$
$$x_t \sim \mathcal{N}(\alpha x_{t-1}, \tfrac{\sigma^2}{1-\alpha^2}) \quad t > 1$$
$$y_t \sim \mathcal{N}(0, \beta^2 \exp(x_t)) \quad t > 1$$

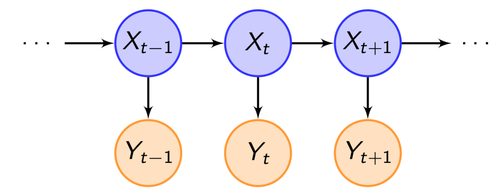


Figure: Hidden Markov Model

### BUGS language

```
alpha ~ dunif(0, 0.99)
prec.x <- (1-alpha^2) / sigma^2
x[1] ~ dnorm(0, prec.x)
for (t in 2:t.max) {
  f[t] <- alpha * x[t-1]
  x[t] ~ dnorm(f[t], prec.x)
  prec.y[t] <- 1 / (beta^2 * exp(x[t]))
  y[t] ~ dnorm(0, prec.y[t]) }
```

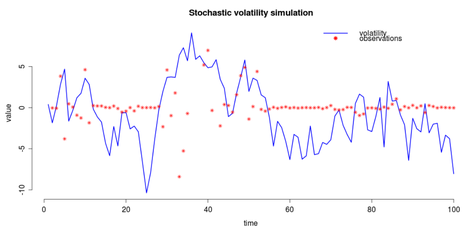


Figure: Volatility simulation

## RBiips package

### Inference of the volatility

```
data <- list(t.max=100, sigma=1.0,
             alpha=0.91, beta=0.5,
             y=y)
# Compile the model and load the data
model <- biips.model("volatility.bug",
                     data)
# Run SMC algorithm
out.smc <- smc.samples(model, "x",
                       n.part=1000)
# Summary statistics
x.summ <- summary(out.smc$x,
            fun=c("mean","quantiles"),
            probs=c(.05,.95))
plot(x.summ)
# Kernel density estimates
plot(density(out.smc$x, adjust=2))
```

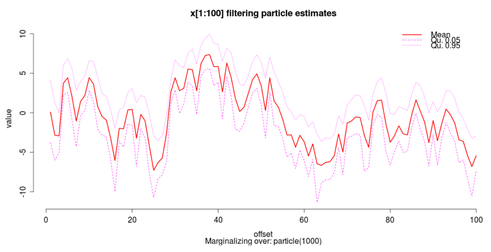


Figure: Summary statistics

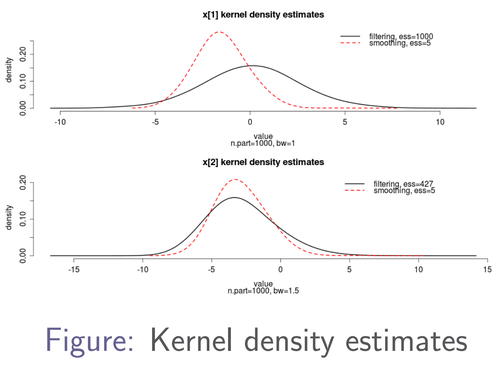


Figure: Kernel density estimates

### Estimation of the fixed parameter $\alpha$

```
data <- list(t.max=100, sigma=1.0,
             beta=0.5, y=y)
model <- biips.model("volatility.bug",
             data)
# Sensitivity analysis
out.sens <- smc.sensitivity(model,
             list(alpha=seq(0,.99,.01),
             n.part=100)
plot(param$alpha,
     out.sens$log.marg.like)
# Burn in PMMH algorithm
update.pmmh(model, "alpha",
             n.iter=1000, n.part=100)
# Generate PMMH samples
out.pmmh <- pmmh.samples(model,
             "alpha", n.iter=10000,
             n.part=100)
# PMMH mean value
print(mean(out.pmmh$alpha))
# PMMH trace plot and histogram
plot(out.pmmh$alpha)
hist(out.pmmh$alpha)
```

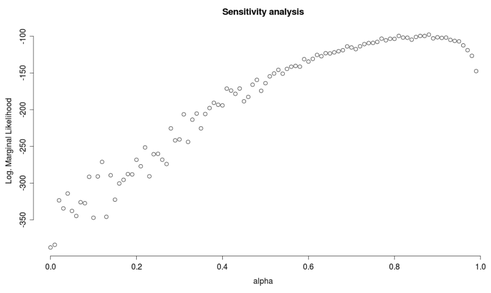


Figure: $\alpha$ sensitivity analysis

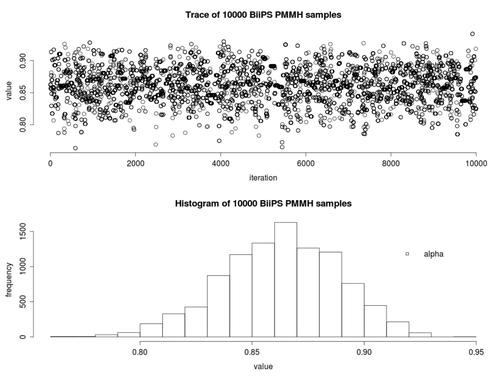


Figure: $\alpha$ PMMH samples: trace plot and histogram

## References

Andrieu, C., Doucet, A., and Holenstein, R. (2010).
Particle markov chain monte carlo methods.
*Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342.

Del Moral, P. (2004).
*Feynman-Kac formulae: genealogical and interacting particle systems with applications*.
Springer Verlag.

Doucet, A., De Freitas, N., and Gordon, N. (2001).
*Sequential Monte Carlo methods in practice*.
Springer Verlag.

Doucet, A. and Johansen, A. (2009).
A tutorial on particle filtering and smoothing: Fifteen years later.
*Handbook of Nonlinear Filtering*, pages 656–704.

Adrien.Todeschini@inria.fr
Francois.Caron@inria.fr

▷ http://alea.bordeaux.inria.fr/biips