
# Sequential Quasi Monte Carlo

N. Chopin (CREST-ENSAE)

nicolas.chopin@ensae.fr

joint work with Mathieu Gerber (Harvard)

# Outline


Particle filtering (a.k.a. Sequential Monte Carlo) is a set of Monte Carlo techniques for sequential inference in state-space models. The error rate of PF is therefore $\mathcal{O}_P(N^{-1/2})$.

# Outline


Particle filtering (a.k.a. Sequential Monte Carlo) is a set of Monte Carlo techniques for sequential inference in state-space models. The error rate of PF is therefore $\mathcal{O}_P(N^{-1/2})$.

Quasi Monte Carlo (QMC) is a substitute for standard Monte Carlo (MC), which typically converges at the faster rate $\mathcal{O}(N^{-1+\epsilon})$. However, standard QMC is usually defined for IID problems.

# Outline


Particle filtering (a.k.a. Sequential Monte Carlo) is a set of Monte Carlo techniques for sequential inference in state-space models. The error rate of PF is therefore $\mathcal{O}_P(N^{-1/2})$.

Quasi Monte Carlo (QMC) is a substitute for standard Monte Carlo (MC), which typically converges at the faster rate $\mathcal{O}(N^{-1+\epsilon})$. However, standard QMC is usually defined for IID problems.

The purpose of this work is to derive a QMC version of PF, which we call SQMC (Sequential Quasi Monte Carlo).

Consider the standard MC approximation

$$\frac{1}{N}\sum_{n=1}^{N}\varphi(\mathbf{u}^n) \approx \int_{[0,1]^d}\varphi(\mathbf{u})\mathrm{d}\mathbf{u}$$

where the $N$ vectors $\mathbf{u}^n$ are IID variables simulated from $\mathcal{U}\left([0,1]^d\right)$.

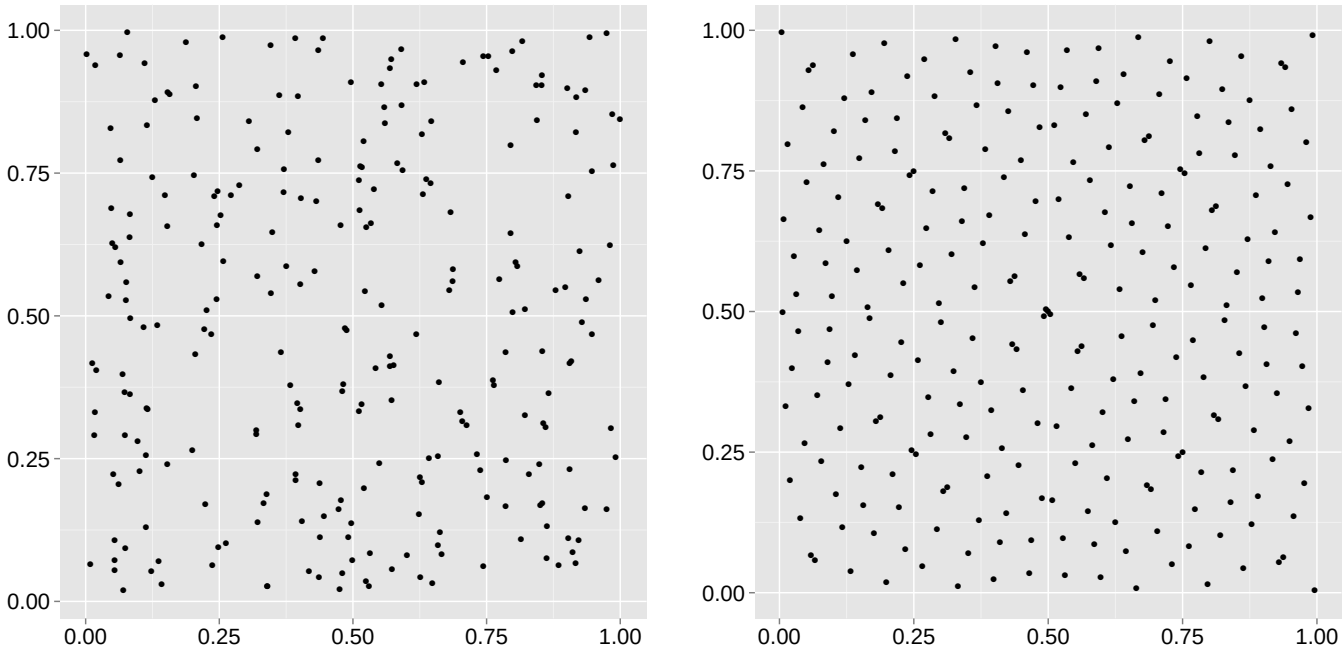
Consider the standard MC approximation

$$\frac{1}{N}\sum_{n=1}^{N}\varphi(\mathbf{u}^n) \approx \int_{[0,1]^d}\varphi(\mathbf{u})\mathrm{d}\mathbf{u}$$

where the $N$ vectors $\mathbf{u}^n$ are IID variables simulated from $\mathcal{U}\left([0,1]^d\right)$.

QMC replaces $\mathbf{u}^{1:N}$ by a set of $N$ points that are more evenly distributed on the hyper-cube $[0,1]^d$. This idea is formalised through the notion of discrepancy.

# QMC vs MC in one plot




QMC versus MC: $N = 256$ points sampled independently and uniformly in $[0,1]^2$ (left); QMC sequence (Sobol) in $[0,1]^2$ of the same length (right)

# Discrepancy

Koksma–Hlawka inequality:

$$\left| \frac{1}{N} \sum_{n=1}^{N} \varphi(\mathbf{u}^n) - \int_{[0,1]^d} \varphi(\mathbf{u}) \, \mathrm{d}\mathbf{u} \right| \leq V(\varphi) D^\star(\mathbf{u}^{1:N})$$

where $V(\varphi)$ depends only on $\varphi$, and the star discrepancy is defined as:

$$D^\star(\mathbf{u}^{1:N}) = \sup_{[\mathbf{0},\boldsymbol{b}]} \left| \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}\left(\mathbf{u}^n \in [\mathbf{0}, \boldsymbol{b}]\right) - \prod_{i=1}^{d} b_i \right|.$$

# Discrepancy

Koksma–Hlawka inequality:

$$\left| \frac{1}{N} \sum_{n=1}^{N} \varphi(\mathbf{u}^n) - \int_{[0,1]^d} \varphi(\mathbf{u}) \, \mathrm{d}\mathbf{u} \right| \leq V(\varphi) D^\star(\mathbf{u}^{1:N})$$

where $V(\varphi)$ depends only on $\varphi$, and the star discrepancy is defined as:

$$D^\star(\mathbf{u}^{1:N}) = \sup_{[\mathbf{0},\boldsymbol{b}]} \left| \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}\left(\mathbf{u}^n \in [\mathbf{0}, \boldsymbol{b}]\right) - \prod_{i=1}^{d} b_i \right|.$$

There are various ways to construct point sets $P_N = \left\{\mathbf{u}^{1:N}\right\}$ so that $D^\star(\mathbf{u}^{1:N}) = \mathcal{O}(N^{-1+\epsilon})$.

# Examples: Van der Corput, Halton


As a simple example of a low-discrepancy sequence in dimension one, $d = 1$, consider

$$\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{3}{8}, \frac{5}{8}, \frac{7}{8} \dots$$

or more generally,

$$\frac{1}{p}, \dots, \frac{p-1}{p}, \frac{1}{p^2}, \dots .$$

# Examples: Van der Corput, Halton


As a simple example of a low-discrepancy sequence in dimension one, $d = 1$, consider

$$\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{3}{8}, \frac{5}{8}, \frac{7}{8} \dots$$

or more generally,

$$\frac{1}{p}, \dots, \frac{p-1}{p}, \frac{1}{p^2}, \dots .$$

In dimension $d > 1$, a Halton sequence consists of a Van der Corput sequence for each component, with a different $p$ for each component (the first $d$ prime numbers).

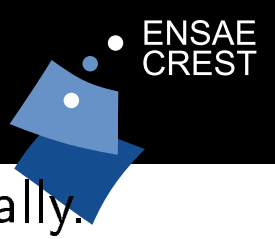
# RQMC (randomised QMC)


RQMC randomises QMC so that each $\mathbf{u}^n \sim \mathcal{U}\left([0,1]^d\right)$ marginally. In this way

$$\mathbb{E}\left\{\frac{1}{N}\sum_{n=1}^{N}\varphi(\mathbf{u}^n)\right\} = \int_{[0,1]^d}\varphi(\mathbf{u})\,\mathrm{d}\mathbf{u}$$

and one may evaluate the MSE through independent runs.

# RQMC (randomised QMC)




RQMC randomises QMC so that each $\mathbf{u}^n \sim \mathcal{U}\left([0,1]^d\right)$ marginally. In this way

$$\mathbb{E}\left\{\frac{1}{N}\sum_{n=1}^{N}\varphi(\mathbf{u}^n)\right\} = \int_{[0,1]^d}\varphi(\mathbf{u})\,\mathrm{d}\mathbf{u}$$

and one may evaluate the MSE through independent runs.

A simple way to generate a RQMC sequence is to take $\mathbf{u}^n = \boldsymbol{w} + \boldsymbol{v}^n \equiv 1$, where $\boldsymbol{w} \sim U([0,1]^d)$ and $\mathbf{v}^{1:N}$ is a QMC point set.

# RQMC (randomised QMC)

RQMC randomises QMC so that each $\mathbf{u}^n \sim \mathcal{U}\left([0,1]^d\right)$ marginally. In this way

$$\mathbb{E}\left\{\frac{1}{N}\sum_{n=1}^{N}\varphi(\mathbf{u}^n)\right\} = \int_{[0,1]^d}\varphi(\mathbf{u})\,\mathrm{d}\mathbf{u}$$

and one may evaluate the MSE through independent runs.

A simple way to generate a RQMC sequence is to take $\mathbf{u}^n = \boldsymbol{w} + \boldsymbol{v}^n \equiv 1$, where $\boldsymbol{w} \sim U([0,1]^d)$ and $\mathbf{v}^{1:N}$ is a QMC point set.

Owen (1995, 1997a, 1997b, 1998) developed RQMC strategies such that (for a certain class of smooth functions $\varphi$):

$$\mathrm{Var}\left\{\frac{1}{N}\sum_{n=1}^{N}\varphi(\mathbf{u}^n)\right\} = \mathcal{O}(N^{-3+\varepsilon})$$

# QMC take-home message


To use QMC:

# QMC take-home message


To use QMC:

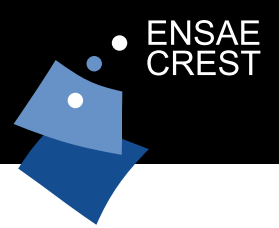
1. rewrite your MC algorithm as a deterministic function of uniform variables;

# QMC take-home message


To use QMC:

1. rewrite your MC algorithm as a deterministic function of uniform variables;
2. replace these uniform variables by a QMC or RQMC sequence (RQMC is better);

# QMC take-home message




To use QMC:

1. rewrite your MC algorithm as a deterministic function of uniform variables;
2. replace these uniform variables by a QMC or RQMC sequence (RQMC is better);
3. pray for increased performance.

# Feynmann-Kac models: definition


A Feynman-Kac model is made of:

- A Markov chain in $\mathcal{X}$: initial law is $m_0(\mathrm{d}\mathbf{x})$, Markov kernel at iteration $t$ is $m_t(\mathbf{x}_{t-1}, \mathrm{d}\mathbf{x}_t)$
- A sequence of potential functions $G_0 : \mathcal{X} \to \mathbb{R}^+$, $G_t : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$

# Feynmann-Kac models: definition

A Feynman-Kac model is made of:

- A Markov chain in $\mathcal{X}$: initial law is $m_0(\mathrm{d}\mathbf{x})$, Markov kernel at iteration $t$ is $m_t(\mathbf{x}_{t-1}, \mathrm{d}\mathbf{x}_t)$
- A sequence of potential functions $G_0 : \mathcal{X} \to \mathbb{R}^+$, $G_t : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$

Aim is to compute sequentially quantities such as

$$\mathbb{Q}_t(\varphi) = \frac{1}{Z_t}\mathbb{E}\left[\varphi(\mathbf{x}_t) G_0(\mathbf{x}_0) \prod_{s=1}^{t} G_s(\mathbf{x}_{s-1}, \mathbf{x}_s)\right],$$

$$\text{with } Z_t = \mathbb{E}\left[G_0(\mathbf{x}_0) \prod_{s=1}^{t} G_s(\mathbf{x}_{s-1}, \mathbf{x}_s)\right].$$

$\Rightarrow$ change of measure.


Take for instance

$$G_t(\mathbf{x}_{t-1}, \mathbf{x}_t) = \mathbb{1}_{A_t}(\mathbf{x}_t)$$

then $Z_t$ is the probability that the $\mathbf{x}_t \in A_t$ for all $t$, and so on.

# Feynmann-Kac models: application to filtering


Imagine a model for a Markov chain $(\mathbf{x}_t)$ that is not observed directly, but through

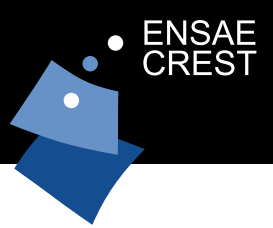
$$\mathbf{y}_t = h(\mathbf{x}_t) + \text{noise}$$

and let $g(\mathbf{y}_t|\mathbf{x}_t)$ be the density of $\mathbf{y}_t$ conditional on $\mathbf{x}_t$. Then, taking

$$G_t(\mathbf{x}_{t-1}, \mathbf{x}_t) = g(\mathbf{y}_t|\mathbf{x}_t)$$

turns $\mathbb{Q}_t$ into the filtering distribution of Markov chain $(\mathbf{x}_t)$, conditional on data $\mathbf{y}_{0:t}$.

# Feynmann-Kac models: application to filtering




Imagine a model for a Markov chain $(\mathbf{x}_t)$ that is not observed directly, but through

$$\mathbf{y}_t = h(\mathbf{x}_t) + \text{noise}$$

and let $g(\mathbf{y}_t|\mathbf{x}_t)$ be the density of $\mathbf{y}_t$ conditional on $\mathbf{x}_t$. Then, taking

$$G_t(\mathbf{x}_{t-1}, \mathbf{x}_t) = g(\mathbf{y}_t|\mathbf{x}_t)$$

turns $\mathbb{Q}_t$ into the filtering distribution of Markov chain $(\mathbf{x}_t)$, conditional on data $\mathbf{y}_{0:t}$.

Applications: target tracking, Ecology, neurosciences...

# Particle Filtering: why?

For a given Feynman-Kac model, a possible approach to approximate $\mathbb{Q}_t$ sequentially would be (sequential) importance sampling:

1. At time $t$, simulate $N$ copies $\mathbf{x}_t^n$ of Markov chain $(\mathbf{x}_t)$
2. reweight according to function $G_t$

Problem: variance of cumulative weigts:

$$w(\mathbf{x}_{0:t}^n) = \prod_{s=0}^{t} G_s(\mathbf{x}_{s-1}^n, \mathbf{x}_s^n)$$

increases over time (at exponential rate).

# Particle Filtering: Basic idea


At time 0, use importance sampling, to go from $m_0(\mathrm{d}\mathbf{x}_0)$ to $\mathbb{Q}_0(\mathrm{d}\mathbf{x}_0) \propto m_0(\mathrm{d}\mathbf{x}_0) G_0(x_0)$. We thus obtain the following approximation of $\mathbb{Q}_0$:

$$\mathbb{Q}_0^N(\mathrm{d}\mathbf{x}_0) = \frac{1}{\sum_{n=1}^N G_0(x_0^n)} \sum_{n=1}^N G_0(x_0^n) \delta_{\mathbf{x}_0^n}(\mathbf{x}_0)$$

To progress to time 1:

1. Choose one 'ancestor' $\mathbf{x}_0^n$ with probability $\propto G_0(\mathbf{x}_0^n)$; call $A_0^n$ the index of the selected ancestor.
2. Simulate $\mathbf{x}_1^n \sim m_1(\mathbf{x}_0^{A_0^n}, \mathrm{d}\mathbf{x}_1)$
3. Reweight, with weight $G_1(\mathbf{x}_0^{A_0^n}, \mathbf{x}_1^n)$

# Particle filtering: the algorithm

Operations must be be performed for all $n \in 1:N$.
At time 0,

(a) Generate $\mathbf{x}_0^n \sim m_0(\mathrm{d}\mathbf{x}_0)$.

(b) Compute $W_0^n = G_0(\mathbf{x}_0^n) / \sum_{m=1}^N G_0(\mathbf{x}_0^m)$ and $Z_0^N = N^{-1} \sum_{n=1}^N G_0(\mathbf{x}_0^n)$.

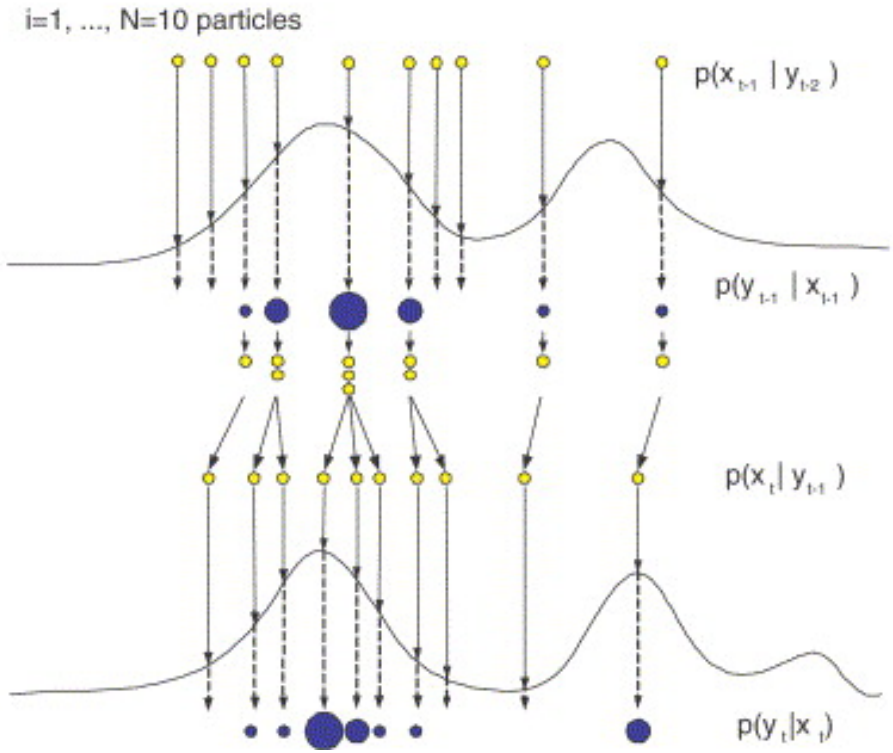
Recursively, for time $t = 1:T$,

(a) Generate $a_{t-1}^n \sim \mathcal{M}(W_{t-1}^{1:N})$.

(b) Generate $\mathbf{x}_t^n \sim m_t(\mathbf{x}_{t-1}^{a_{t-1}^n}, \mathrm{d}\mathbf{x}_t)$.

(c) Compute $W_t^n = G_t(\mathbf{x}_{t-1}^{a_{t-1}^n}, \mathbf{x}_t^n) / \sum_{m=1}^N G_t(\mathbf{x}_{t-1}^{a_{t-1}^m}, \mathbf{x}_t^m)$ and $Z_t^N = Z_{t-1}^N \left\{ N^{-1} \sum_{n=1}^N G_t(\mathbf{x}_{t-1}^{a_{t-1}^n}, \mathbf{x}_t^n) \right\}$.

# Cartoon representation




Source for image: some dark corner of the Internet.

# PF output


At iteration $t$, compute

$$\mathbb{Q}_t^N(\varphi) = \sum_{n=1}^{N} W_t^n \varphi(\mathbf{x}_t^n)$$

to approximate $\mathbb{Q}_t(\varphi)$ (the filtering expectation of $\varphi$). In addition, compute

$$Z_t^N$$

as an approximation of $Z_t$ (the likelihood of the data).

# Formalisation


We can formalise the succession of Steps (a), (b) and (c) at iteration $t$ as an importance sampling step from random probability measure

$$\sum_{n=1}^{N} W_{t-1}^{n} \delta_{\mathbf{x}_{t-1}^{n}}(\mathrm{d}\widetilde{\mathbf{x}}_{t-1}) m_t(\widetilde{\mathbf{x}}_{t-1}, \mathrm{d}\mathbf{x}_t) \tag{1}$$

to

$$\{\text{same thing}\} \times G_t(\widetilde{\mathbf{x}}_{t-1}, \mathbf{x}_t).$$

# Formalisation

We can formalise the succession of Steps (a), (b) and (c) at iteration $t$ as an importance sampling step from random probability measure

$$\sum_{n=1}^{N} W_{t-1}^{n} \delta_{\mathbf{x}_{t-1}^{n}}(\mathrm{d}\widetilde{\mathbf{x}}_{t-1}) m_t(\widetilde{\mathbf{x}}_{t-1}, \mathrm{d}\mathbf{x}_t) \tag{1}$$

to

$$\{\text{same thing}\} \times G_t(\widetilde{\mathbf{x}}_{t-1}, \mathbf{x}_t).$$

Idea: use QMC instead of MC to sample $N$ points from (1); i.e. rewrite sampling from (1) this as a function of uniform variables, and use low-discrepancy sequences instead.

# Intermediate step

More precisely, we are going to write the simulation from

$$\sum_{n=1}^{N} W_{t-1}^{n} \delta_{\mathbf{x}_{t-1}^{n}}(\mathrm{d}\widetilde{\mathbf{x}}_{t-1}) m_t(\widetilde{\mathbf{x}}_{t-1}, \mathrm{d}\mathbf{x}_t)$$

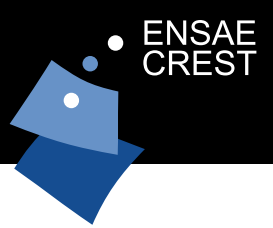
as a function of $\mathbf{u}_t^n = (u_t^n, \mathbf{v}_t^n)$, $u_t^n \in [0,1]$, $\mathbf{v}_t^n \in [0,1]^d$, such that:

1. We will use the scalar $u_t^n$ to choose the ancestor $\widetilde{\mathbf{x}}_{t-1}$.
2. We will use $\mathbf{v}_t^n$ to generate $\mathbf{x}_t^n$ as

$$\mathbf{x}_t^n = \Gamma_t(\widetilde{\mathbf{x}}_{t-1}, \mathbf{v}_t^n)$$

where $\Gamma_t$ is a deterministic function such that, for $\mathbf{v}_t^n \sim \mathcal{U}[0,1]^d$, $\Gamma_t(\widetilde{\mathbf{x}}_{t-1}, \mathbf{v}_t^n) \sim m_t(\widetilde{\mathbf{x}}_{t-1}, \mathrm{d}\mathbf{x}_t)$.

# Intermediate step




More precisely, we are going to write the simulation from

$$\sum_{n=1}^{N} W_{t-1}^{n} \delta_{\mathbf{x}_{t-1}^{n}}(\mathrm{d}\widetilde{\mathbf{x}}_{t-1}) m_t(\widetilde{\mathbf{x}}_{t-1}, \mathrm{d}\mathbf{x}_t)$$

as a function of $\mathbf{u}_t^n = (u_t^n, \mathbf{v}_t^n)$, $u_t^n \in [0,1]$, $\mathbf{v}_t^n \in [0,1]^d$, such that:

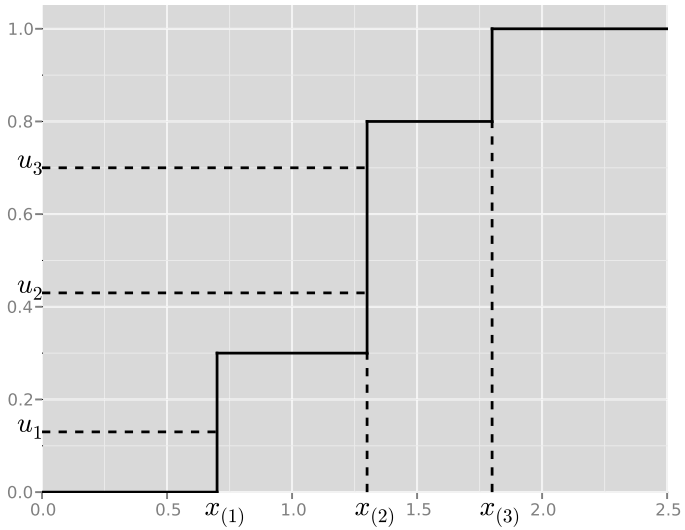
1. We will use the scalar $u_t^n$ to choose the ancestor $\widetilde{\mathbf{x}}_{t-1}$.
2. We will use $\mathbf{v}_t^n$ to generate $\mathbf{x}_t^n$ as

$$\mathbf{x}_t^n = \Gamma_t(\widetilde{\mathbf{x}}_{t-1}, \mathbf{v}_t^n)$$

where $\Gamma_t$ is a deterministic function such that, for $\mathbf{v}_t^n \sim \mathcal{U}[0,1]^d$, $\Gamma_t(\widetilde{\mathbf{x}}_{t-1}, \mathbf{v}_t^n) \sim m_t(\widetilde{\mathbf{x}}_{t-1}, \mathrm{d}\mathbf{x}_t)$.

The main problem is point 1.

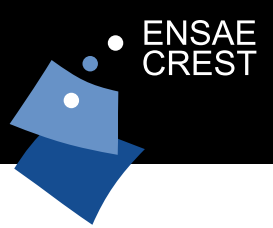
# Case $d = 1$




Simply use the inverse transform method: $\tilde{\mathbf{x}}_{t-1}^n = \hat{F}^{-1}(u_t^n)$, where $\hat{F}$ is the empirical cdf of

$$\sum_{n=1}^{N} W_{t-1}^n \delta_{\mathbf{x}_{t-1}^n}(\mathrm{d}\widetilde{\mathbf{x}}_{t-1}).$$

# From $d = 1$ to $d > 1$




When $d > 1$, we cannot use the inverse CDF method to sample from the empirical distribution

$$\sum_{n=1}^{N} W_{t-1}^{n} \delta_{\mathbf{x}_{t-1}^{n}}(\mathrm{d}\widetilde{\mathbf{x}}_{t-1}).$$

Idea: we "project" the $\mathbf{x}_{t-1}^{n}$'s into $[0,1]$ through the (generalised) inverse of the Hilbert curve, which is a fractal, space-filling curve $H : [0,1] \to [0,1]^{d}$.

# From $d = 1$ to $d > 1$

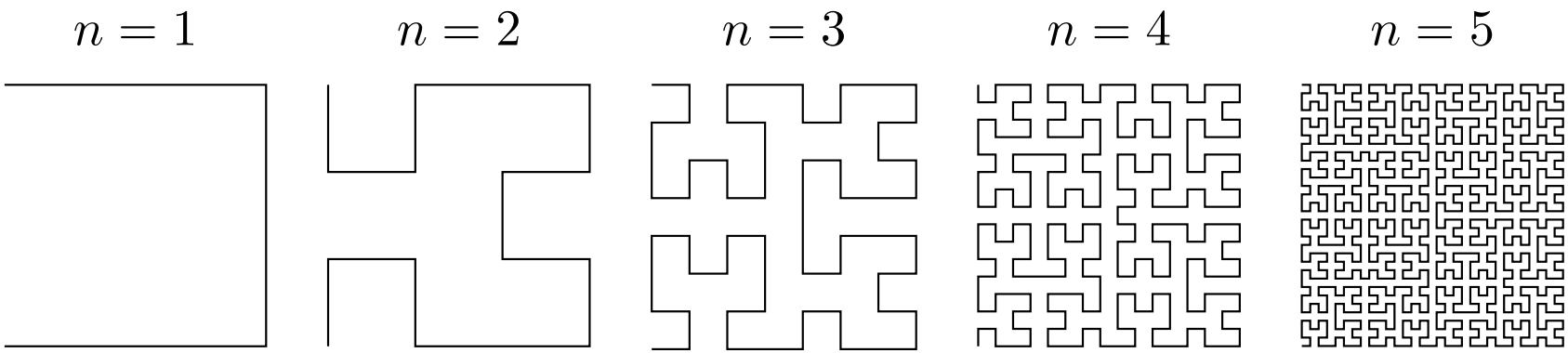
When $d > 1$, we cannot use the inverse CDF method to sample from the empirical distribution

$$\sum_{n=1}^{N} W_{t-1}^{n} \delta_{\mathbf{x}_{t-1}^{n}}(\mathrm{d}\widetilde{\mathbf{x}}_{t-1}).$$

Idea: we "project" the $\mathbf{x}_{t-1}^{n}$'s into $[0,1]$ through the (generalised) inverse of the Hilbert curve, which is a fractal, space-filling curve $H : [0,1] \to [0,1]^d$.

More precisely, we transform $\mathcal{X}$ into $[0,1]^d$ through some function $\psi$, then we transform $[0,1]^d$ into $[0,1]$ through $h = H^{-1}$.

# Hilbert curve




The Hilbert curve is the limit of this sequence. Note the locality property of the Hilbert curve: if two points are close in $[0, 1]$, then the the corresponding transformed points remains close in $[0, 1]^d$. (Source for the plot: Wikipedia)

# SQMC Algorithm


At time 0,

(a) Generate a QMC point set $\mathbf{u}_0^{1:N}$ in $[0,1]^d$, and compute $\mathbf{x}_0^n = \Gamma_0(\mathbf{u}_0^n)$. (e.g. $\Gamma_0 = F_{m_0}^{-1}$)

(b) Compute $W_0^n = G_0(\mathbf{x}_0^n) / \sum_{m=1}^N G_0(\mathbf{x}_0^m)$.

Recursively, for time $t = 1 : T$,

(a) Generate a QMC point set $\mathbf{u}_t^{1:N}$ in $[0,1]^{d+1}$; let $\mathbf{u}_t^n = (u_t^n, \mathbf{v}_t^n)$.

(b) Hilbert sort: find permutation $\sigma$ such that $h \circ \psi(\mathbf{x}_{t-1}^{\sigma(1)}) \leq \ldots \leq h \circ \psi(\mathbf{x}_{t-1}^{\sigma(N)})$.

(c) Generate $a_{t-1}^{1:N}$ using inverse CDF Algorithm, with inputs $\mathrm{sort}(u_t^{1:N})$ and $W_{t-1}^{\sigma(1:N)}$, and compute $\mathbf{x}_t^n = \Gamma_t(\mathbf{x}_{t-1}^{\sigma(a_{t-1}^n)}, \mathbf{v}_t^{\sigma(n)})$. (e.g. $\Gamma_t = F_{m_t}^{-1}$)

(e) Compute $W_t^n = G_t(\mathbf{x}_{t-1}^{\sigma(a_{t-1}^n)}, \mathbf{x}_t^n) / \sum_{m=1}^N G_t(\mathbf{x}_{t-1}^{\sigma(a_{t-1}^m)}, \mathbf{x}_t^m)$.

# Some remarks


- Because two sort operations are performed, the complexity of SQMC is $\mathcal{O}(N \log N)$. (Compare with $\mathcal{O}(N)$ for SMC.)

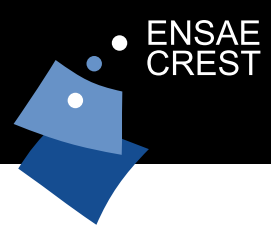
# Some remarks


- Because two sort operations are performed, the complexity of SQMC is $\mathcal{O}(N \log N)$. (Compare with $\mathcal{O}(N)$ for SMC.)
- The main requirement to implement SQMC is that one may simulate from Markov kernel $m_t(x_{t-1}, \mathrm{d}\mathbf{x}_t)$ by computing $\mathbf{x}_t = \Gamma_t(\mathbf{x}_{t-1}, \mathbf{u}_t)$, where $\mathbf{u}_t \sim \mathcal{U}[0,1]^d$, for some deterministic function $\Gamma_t$ (e.g. multivariate inverse CDF).

# Some remarks


- Because two sort operations are performed, the complexity of SQMC is $\mathcal{O}(N \log N)$. (Compare with $\mathcal{O}(N)$ for SMC.)
- The main requirement to implement SQMC is that one may simulate from Markov kernel $m_t(x_{t-1}, \mathrm{d}\mathbf{x}_t)$ by computing $\mathbf{x}_t = \Gamma_t(\mathbf{x}_{t-1}, \mathbf{u}_t)$, where $\mathbf{u}_t \sim \mathcal{U}[0,1]^d$, for some deterministic function $\Gamma_t$ (e.g. multivariate inverse CDF).
- The dimension of the point sets $\mathbf{u}_t^{1:N}$ is $1 + d$: first component is for selecting the parent particle, the $d$ remaining components is for sampling $\mathbf{x}_t^n$ given $\mathbf{x}_{t-1}^{a_{t-1}^n}$ .

# Extensions

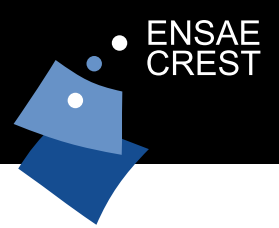



- If we use RQMC (randomised QMC) point sets $\mathbf{u}_t^{1:N}$, then SQMC generates an unbiased estimate of the marginal likelihood $Z_t$.

# Extensions

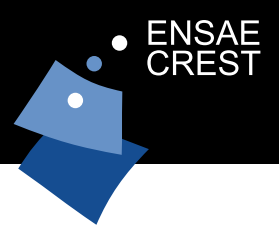

- If we use RQMC (randomised QMC) point sets $\mathbf{u}_t^{1:N}$, then SQMC generates an unbiased estimate of the marginal likelihood $Z_t$.
- This means we can use SQMC within the PMCMC framework. (More precisely, we can run e.g. a PMMH algorithm, where the likelihood of the data is computed via SQMC instead of SMC.)

# Extensions




- If we use RQMC (randomised QMC) point sets $\mathbf{u}_t^{1:N}$, then SQMC generates an unbiased estimate of the marginal likelihood $Z_t$.
- This means we can use SQMC within the PMCMC framework. (More precisely, we can run e.g. a PMMH algorithm, where the likelihood of the data is computed via SQMC instead of SMC.)
- We can also adapt quite easily the different particle smoothing algorithms: forward smoothing, backward smoothing, two-filter smoothing.

# Extensions




- If we use RQMC (randomised QMC) point sets $\mathbf{u}_t^{1:N}$, then SQMC generates an unbiased estimate of the marginal likelihood $Z_t$.
- This means we can use SQMC within the PMCMC framework. (More precisely, we can run e.g. a PMMH algorithm, where the likelihood of the data is computed via SQMC instead of SMC.)
- We can also adapt quite easily the different particle smoothing algorithms: forward smoothing, backward smoothing, two-filter smoothing.

# Main results

We were able to establish the following types of results: consistency

$$\mathbb{Q}_t^N(\varphi) - \mathbb{Q}_t(\varphi) \to 0, \quad \text{as } N \to +\infty$$

for certain functions $\varphi$, and rate of convergence

$$\mathrm{MSE}\left[\mathbb{Q}_t^N(\varphi)\right] = o(N^{-1})$$

(under technical conditions, and for certain types of RQMC point sets).
Theory is non-standard and borrows heavily from QMC concepts.

# Some concepts used in the proofs

Let $\mathcal{X} = [0,1]^d$. Consistency results are expressed in terms of the star norm

$$\|\mathbb{Q}_t^N - \mathbb{Q}_t\|_\star = \sup_{[\mathbf{0},\boldsymbol{b}]\subset[0,1)^d} \left| \left(\mathbb{Q}_t^N - \mathbb{Q}_t\right)(B) \right| \to 0.$$

This implies consistency for bounded functions $\varphi$,
$\mathbb{Q}_t^N(\varphi) - \mathbb{Q}_t(\varphi) \to 0$.
The Hilbert curve conserves discrepancy:

$$\|\pi^N - \pi\|_\star \to 0 \quad \Rightarrow \quad \|\pi_h^N - \pi_h\|_\star \to 0$$

where $\pi \in \mathcal{P}([0,1]^d)$, $h : [0,1]^d \to [0,1]$ is the (pseudo-)inverse of the Hilbert curve, and $\pi_h$ is the image of $\pi$ through $\pi$.

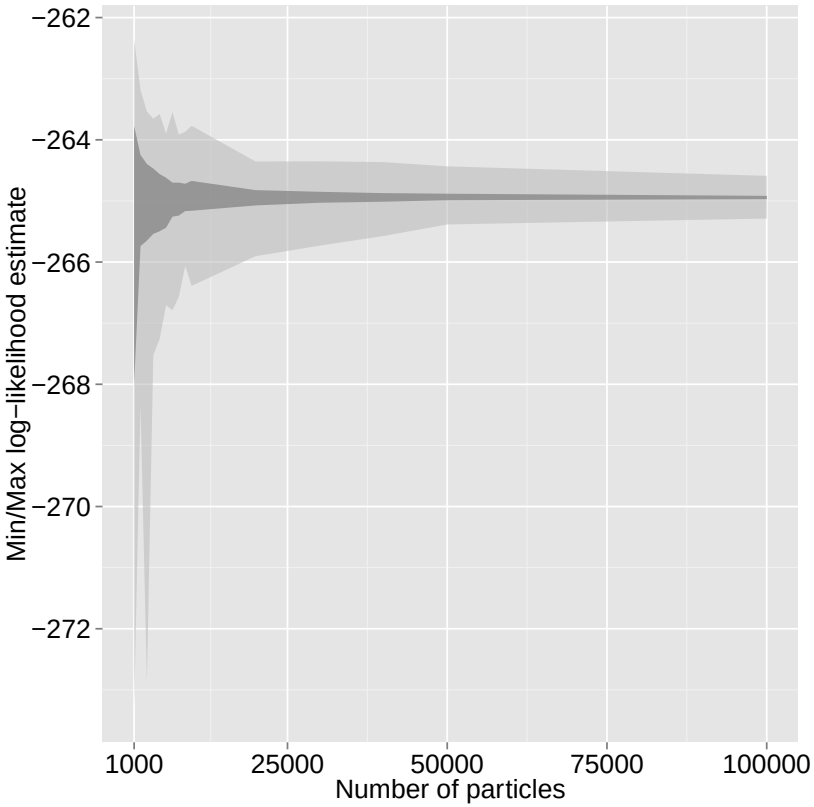
# Examples: Kitagawa ($d = 1$)

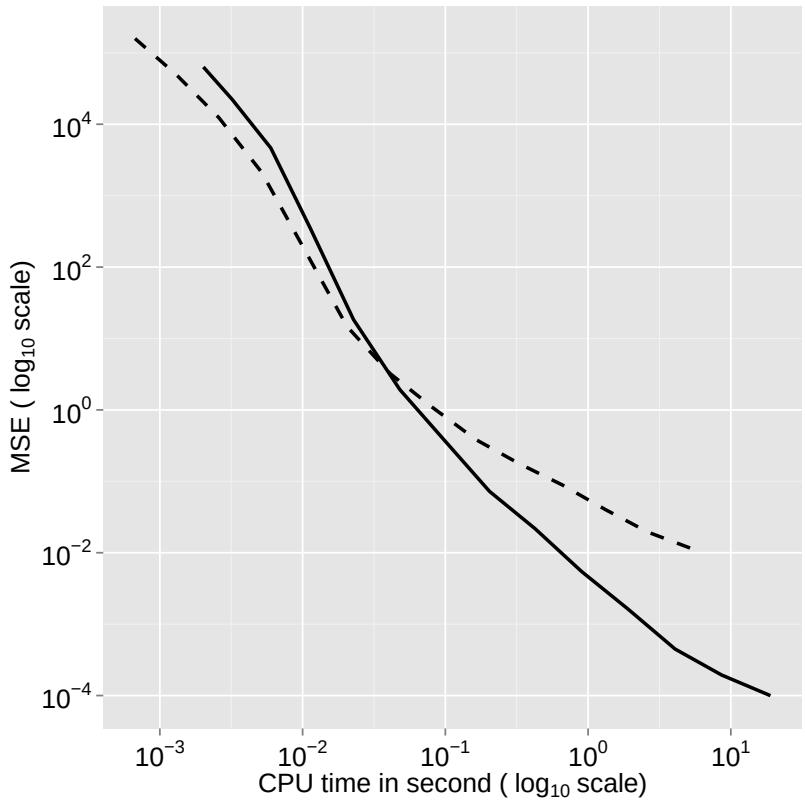

Well known toy example (Kitagawa, 1998):

$$\begin{cases} y_t = \frac{x_t^2}{a} + \epsilon_t \\ x_t = b_1 x_{t-1} + b_2 \frac{x_{t-1}}{1+x_{t-1}^2} + b_3 \cos(b_4 t) + \sigma \nu_t \end{cases}$$

No paramater estimation (parameters are set to their true value).
We compare SQMC with SMC (based on systematic resampling) both in terms of $N$, and in terms of CPU time.

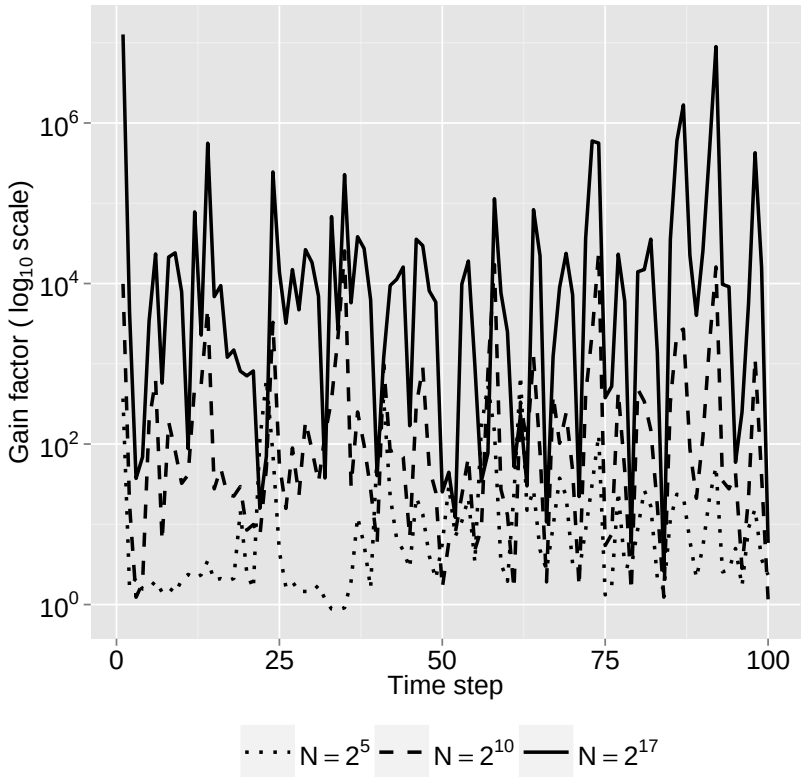
# Examples: Kitagawa ($d = 1$)

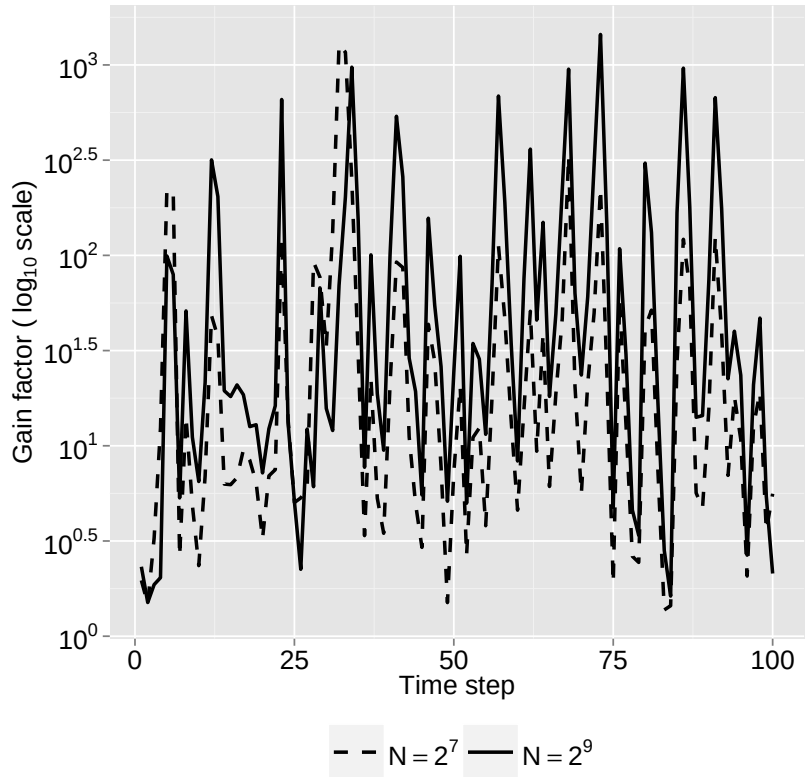



Log-likelihood evaluation (based on $T = 100$ data point and 500 independent SMC and SQMC runs).

# Examples: Kitagawa ($d = 1$)




Filtering: computing $\mathbb{E}(\mathbf{x}_t|\mathbf{y}_{0:t})$ at each iteration $t$. Gain factor is MSE(SMC)/MSE(SQMC).

# Application: autonomous positioning


Vehicle moves in 2D space, acquires its speeds every $T_s$ seconds, and receives $d_y$ radio signals. Model is:

$$y_{ti} = 10 \log_{10} \left( \frac{P_{i0}}{\|r_i - \mathbf{x}_t\|^{\alpha_i}} \right) + \nu_{it}, \quad i = 1, \ldots, d_y$$

$$\mathbf{x}_t = \mathbf{x}_{t-1} + T_s \boldsymbol{v}_t + T_s \boldsymbol{\epsilon}_t$$

and noise terms $\boldsymbol{\epsilon}_t$, $\boldsymbol{\nu}_t$ are Laplace-distributed.

# Application: simulated data

$T_s = 1\text{s}$, $d_y = 5$ (5 emiters), $\alpha_i = 0.95$.

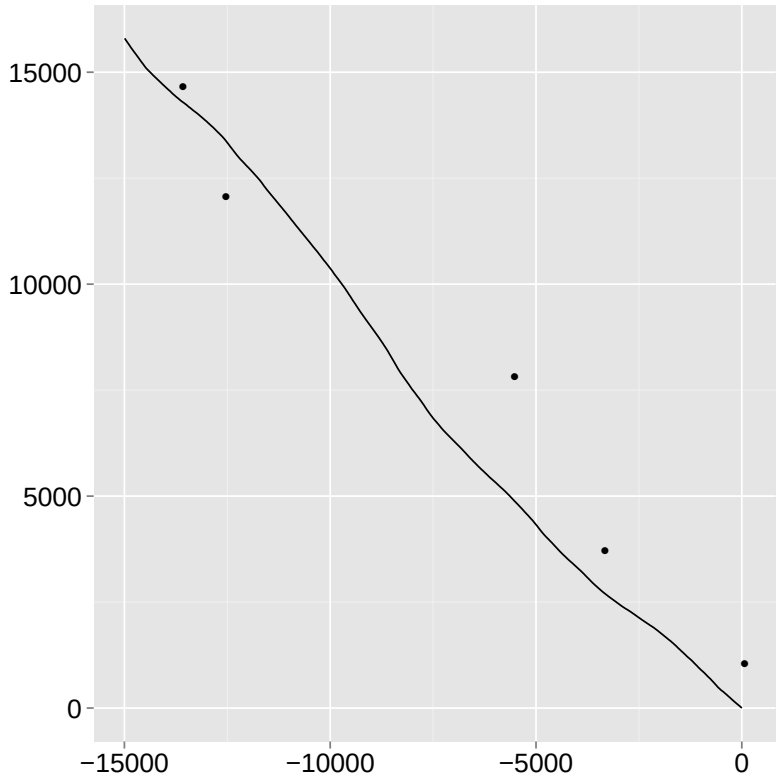



Figure : Simulated trajectory (15 min)

# Application: results

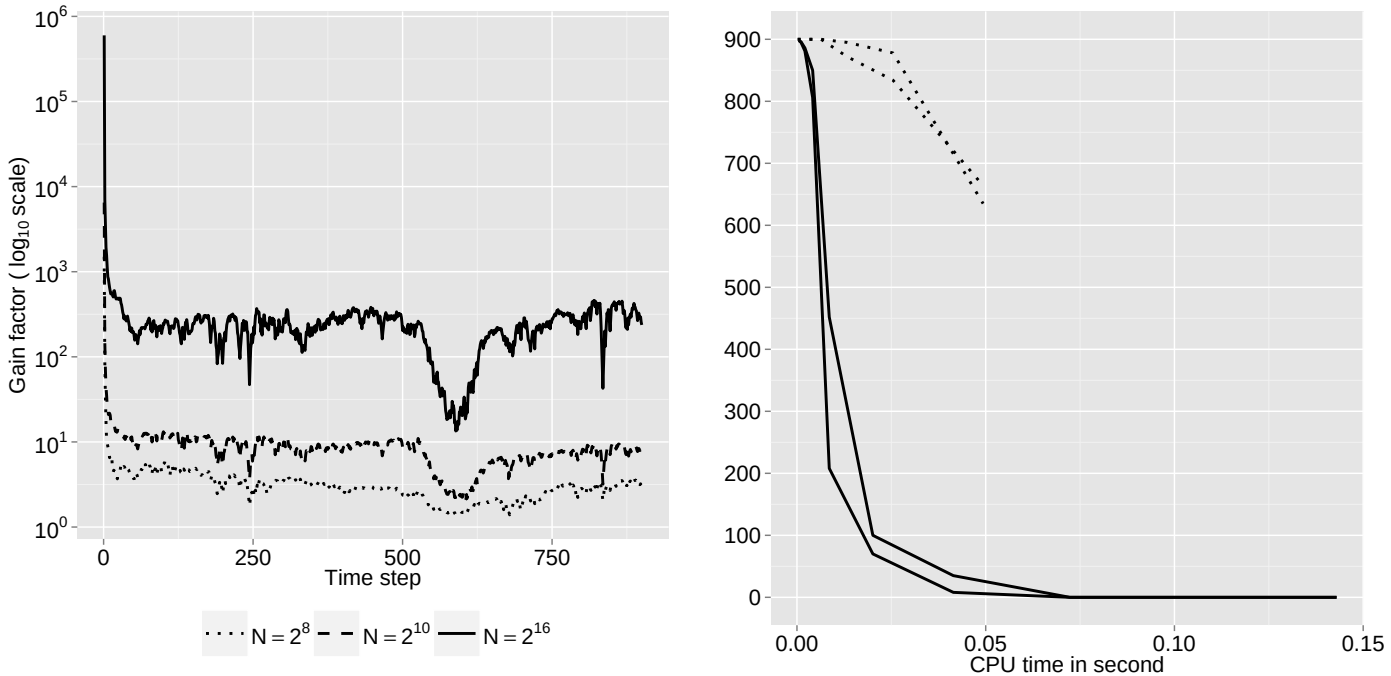



Figure : Left: Gain factor vs time (PF MSE/SQMC MSE); Right: number of time steps such that $\mathrm{MSE}(\hat{x}_{t1}) > 0.01\mathrm{Var}(x_{t1}|y_{0:t})$, as a function of CPU time

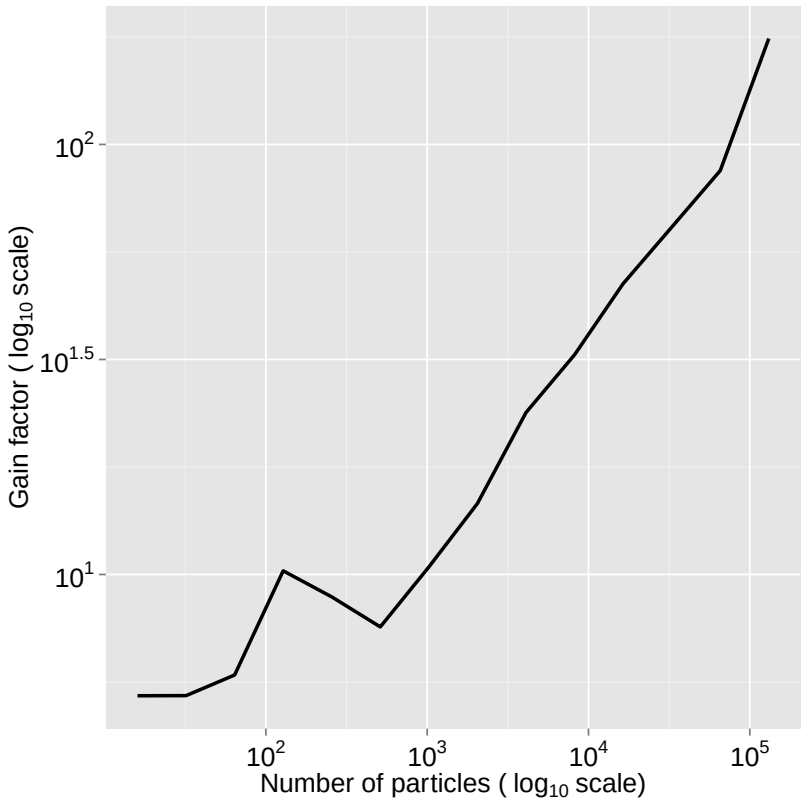
# Examples: Multivariate Stochastic Volatility

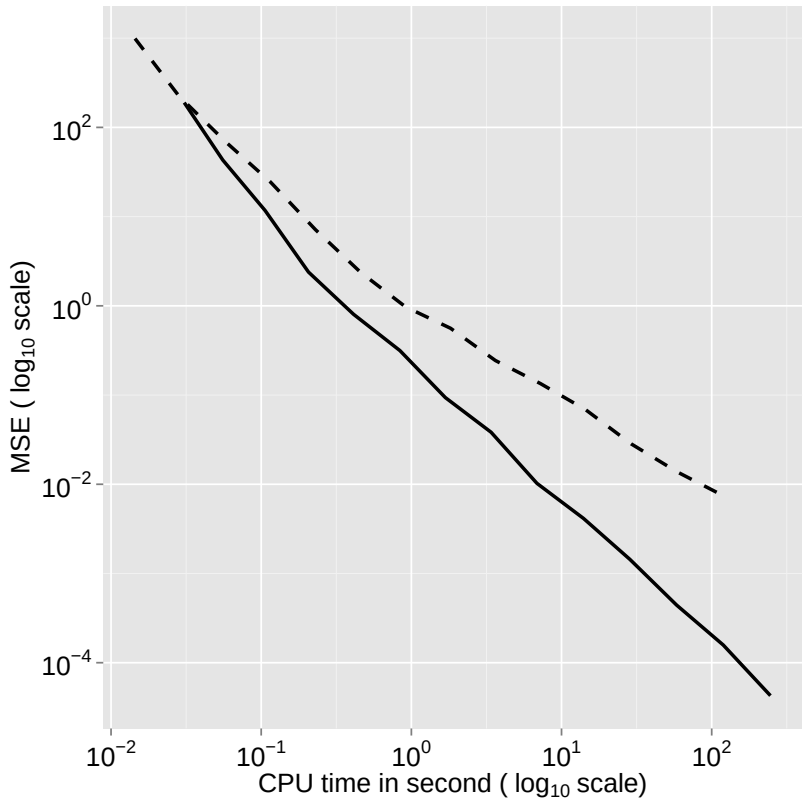

Model is

$$\begin{cases} \mathbf{y}_t = S_t^{\frac{1}{2}} \boldsymbol{\epsilon}_t \\ \mathbf{x}_t = \boldsymbol{\mu} + \Phi(\mathbf{x}_{t-1} - \boldsymbol{\mu}) + \Psi^{\frac{1}{2}} \boldsymbol{\nu}_t \end{cases}$$

with possibly correlated noise terms: $(\boldsymbol{\epsilon}_t, \boldsymbol{\nu}_t) \sim N_{2d}(\mathbf{0}, \boldsymbol{C})$.
We shall focus on $d = 2$ and $d = 4$.

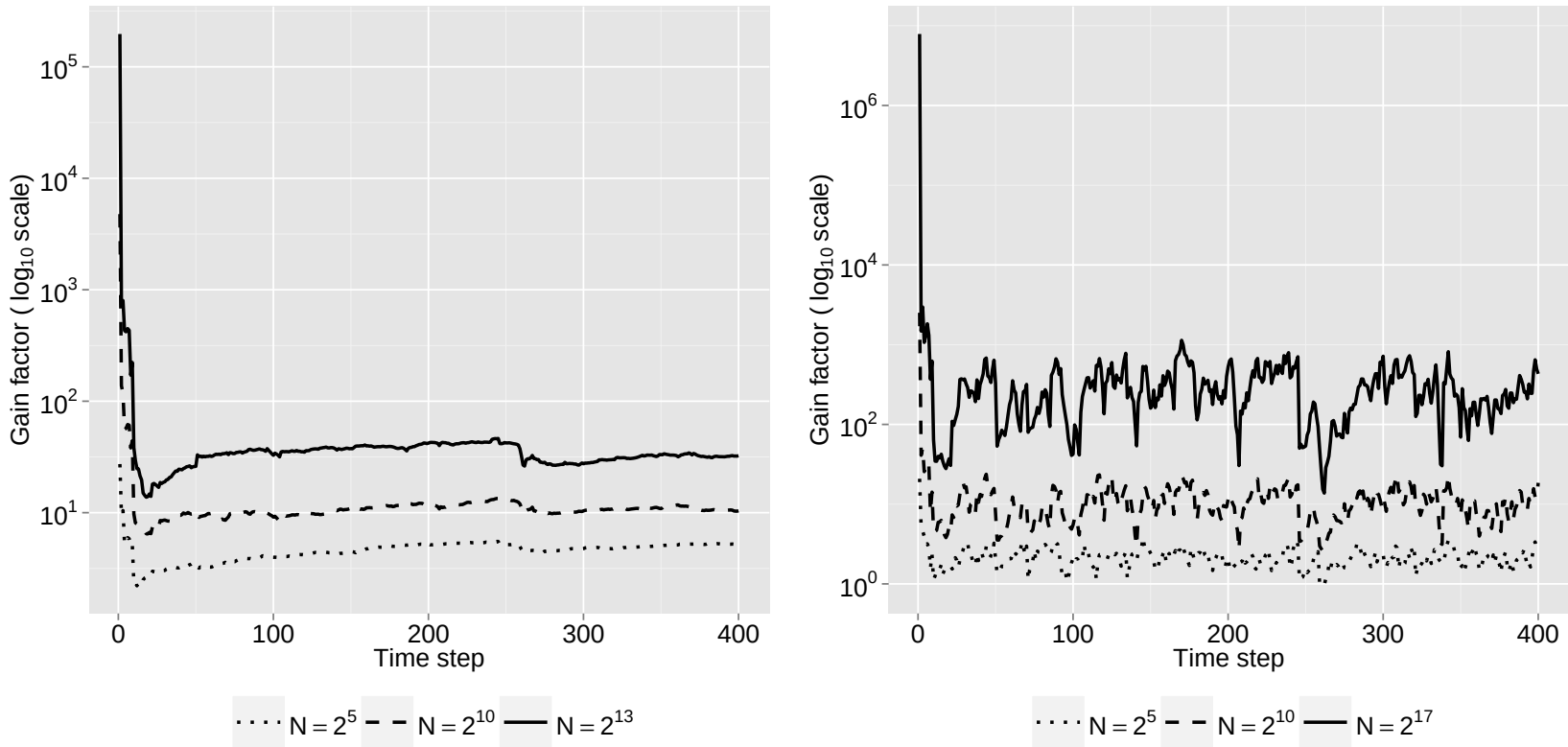
# Examples: Multivariate Stochastic Volatility ($d = 2$)



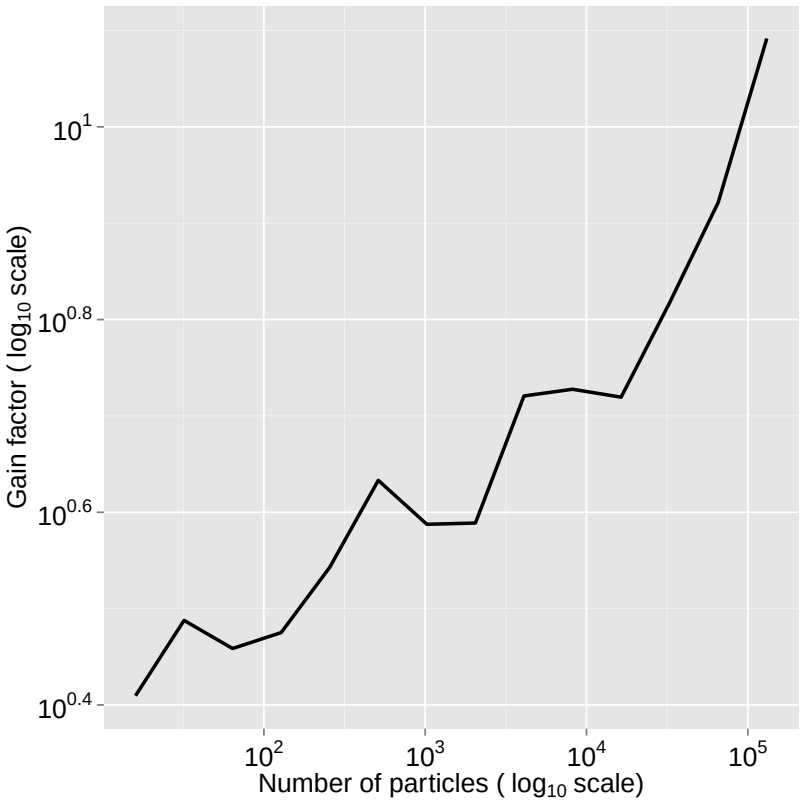

Log-likelihood evaluation (based on $T = 400$ data points and 200 independent runs).

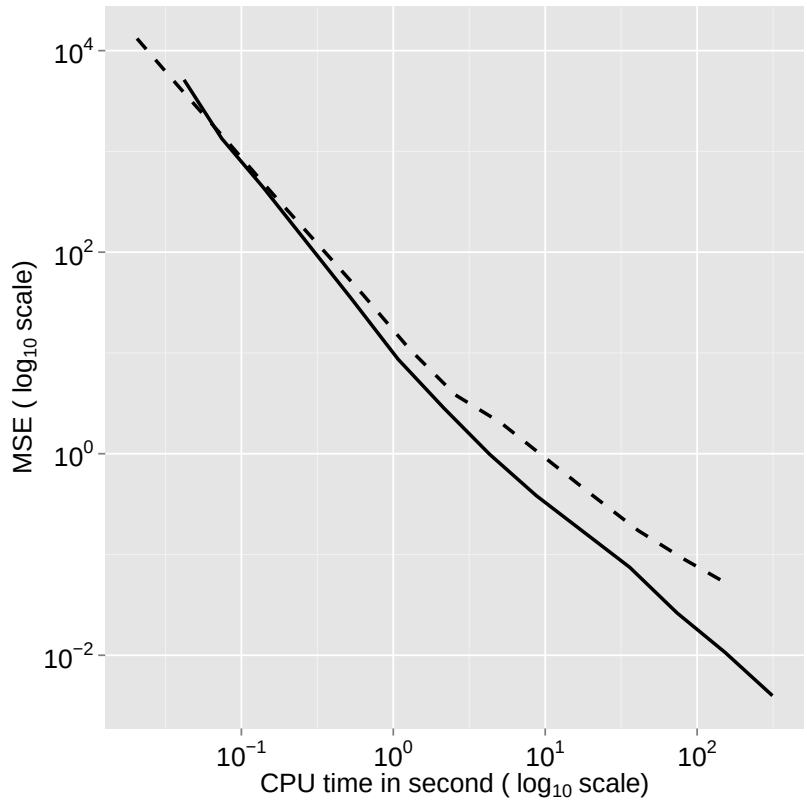
# Examples: Multivariate Stochastic Volatility ($d = 2$)




Log-likelihood evaluation (left) and filtering (right) as a function of $t$.

# Examples: Multivariate Stochastic Volatility ($d = 4$)




Log-likelihood estimation (based on $T = 400$ data points and 200 independent runs)

# Example: Diffusion driven SV model (e.g. Shephard, 2004)

$$\begin{cases} \mathrm{d}Y_t = \{\mu_P + \beta e^{X_t}\}\mathrm{d}t + e^{X_t/2}\mathrm{d}B_t \\ \mathrm{d}X_t = \mu(X_t)\mathrm{d}t + \omega(X_t)\mathrm{d}W_t \end{cases}$$

where $(B_t)_{t\geq 0}$ and $(W_t)_{t\geq 0}$ are Brownian motions with correlation coefficient $\rho \in (-1, 1)$ and

$$\mu(x) = \kappa(\mu - e^x)e^{-x} - 0.5\omega^2 e^{-x}$$
$$\omega(x) = \omega e^{-x/2}$$

# Example: Diffusion driven SV model (e.g. Shephard, 2004)

$$\begin{cases} \mathrm{d}Y_t = \{\mu_P + \beta e^{X_t}\}\mathrm{d}t + e^{X_t/2}\mathrm{d}B_t \\ \mathrm{d}X_t = \mu(X_t)\mathrm{d}t + \omega(X_t)\mathrm{d}W_t \end{cases}$$

where $(B_t)_{t\geq 0}$ and $(W_t)_{t\geq 0}$ are Brownian motions with correlation coefficient $\rho \in (-1, 1)$ and

$$\mu(x) = \kappa(\mu - e^x)e^{-x} - 0.5\omega^2 e^{-x}$$
$$\omega(x) = \omega e^{-x/2}$$

The $Y_t$ are observed for $t = 0, 1, \ldots, T$.

# Discretized version

For $M \geq 1$ (with $\delta = M^{-1}$),

$$\begin{cases} Y_{t+1}|Y_t, \tilde{\boldsymbol{X}}_{t+1} \sim \mathcal{N}_1\Big(Y_t + \mu_P + \beta\tilde{\sigma}^2_{t+1} + \rho\tilde{Z}_{t+1},\ (1-\rho^2)\tilde{\sigma}^2_{t+1}\Big) \\ \tilde{X}_{t+\delta} = \tilde{X}_t + \delta\mu(\tilde{X}_t) + \omega(\tilde{X}_t)(W_{t+\delta} - W_t) \\ \vdots \\ \tilde{X}_{t+1} = \tilde{X}_{t+1-\delta} + \delta\mu(\tilde{X}_{t+1-\delta}) + \omega(\tilde{X}_{t+1-\delta})(W_{t+1} - W_{t+1-\delta}) \end{cases}$$

where $\tilde{\boldsymbol{X}}_{t+1} = (\tilde{X}_{t+\delta}, \ldots, \tilde{X}_{t+1}) \in \mathbb{R}^M$ and

$$\tilde{\sigma}^2_{t+1} = \frac{1}{M}\sum_{m=1}^{M} e^{\tilde{X}_{t+m\delta}}, \quad \tilde{Z}_{t+1} = \sum_{m=1}^{M} e^{\tilde{X}_{t+m\delta}/2}\big(W_{t+m\delta} - W_{t+(m-1)\delta}\big).$$

# Discretized version

For $M \geq 1$ (with $\delta = M^{-1}$),

$$\begin{cases} Y_{t+1}|Y_t, \tilde{\boldsymbol{X}}_{t+1} \sim \mathcal{N}_1\Big(Y_t + \mu_P + \beta\tilde{\sigma}^2_{t+1} + \rho\tilde{Z}_{t+1},\ (1-\rho^2)\tilde{\sigma}^2_{t+1}\Big) \\ \tilde{X}_{t+\delta} = \tilde{X}_t + \delta\mu(\tilde{X}_t) + \omega(\tilde{X}_t)(W_{t+\delta} - W_t) \\ \vdots \\ \tilde{X}_{t+1} = \tilde{X}_{t+1-\delta} + \delta\mu(\tilde{X}_{t+1-\delta}) + \omega(\tilde{X}_{t+1-\delta})(W_{t+1} - W_{t+1-\delta}) \end{cases}$$

where $\tilde{\boldsymbol{X}}_{t+1} = (\tilde{X}_{t+\delta}, \ldots, \tilde{X}_{t+1}) \in \mathbb{R}^M$ and

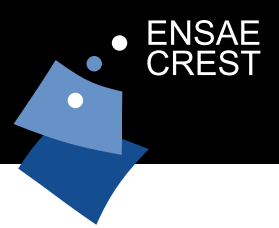
$$\tilde{\sigma}^2_{t+1} = \frac{1}{M}\sum_{m=1}^{M} e^{\tilde{X}_{t+m\delta}}, \quad \tilde{Z}_{t+1} = \sum_{m=1}^{M} e^{\tilde{X}_{t+m\delta}/2}\big(W_{t+m\delta} - W_{t+(m-1)\delta}\big).$$

For this model, $M = 10$ is a reasonable choice (Chib et al., 2004).

# Resampling step of SQMC


A naive application of SQMC would imply working in dimension $M = 10$, in particular for Hilbert ordering.

# Resampling step of SQMC




A naive application of SQMC would imply working in dimension $M = 10$, in particular for Hilbert ordering.

However, since $X_t$ is Markov, we can reduce this particular step to dimension 1.

# Mutation step of SQMC: Choice of $\Gamma_t$

We consider the following two approaches to generate $\tilde{\mathbf{x}}_{t+1}^n$ at iteration $t+1$ of SQMC:

- First approach (forward approach): Set (with $\mathbf{v}^n \in [0,1)^M$)

$$W_{t+m\delta}^n - W_{t+(m-1)\delta}^n = \sqrt{\delta}\Phi^{-1}(v_m^n), \quad m = 1, \ldots, M.$$

# Mutation step of SQMC: Choice of $\Gamma_t$

We consider the following two approaches to generate $\tilde{\mathbf{x}}_{t+1}^n$ at iteration $t+1$ of SQMC:

- First approach (forward approach): Set (with $\mathbf{v}^n \in [0,1)^M$)

$$W_{t+m\delta}^n - W_{t+(m-1)\delta}^n = \sqrt{\delta}\Phi^{-1}(v_m^n), \quad m = 1, \ldots, M.$$

- Second approach: Use $\mathbf{v}^n$ and a dimension reduction approach to simulate the values $\{\widetilde{W}_{m\delta}^n\}_{m=1}^M$ of a standard Brownian motion $(\widetilde{W}_s^n)_{s\in[0,1]}$, and set

$$W_{t+m\delta}^n - W_{t+(m-1)\delta}^n = \widetilde{W}_{t+m\delta}^n - \widetilde{W}_{t+(m-1)\delta}^n, \quad m = 1, \ldots, M.$$

# Mutation step of SQMC: Choice of $\Gamma_t$

We consider the following two approaches to generate $\tilde{\mathbf{x}}_{t+1}^n$ at iteration $t+1$ of SQMC:

- First approach (forward approach): Set (with $\mathbf{v}^n \in [0,1)^M$)

$$W_{t+m\delta}^n - W_{t+(m-1)\delta}^n = \sqrt{\delta}\Phi^{-1}(v_m^n), \quad m = 1, \ldots, M.$$

- Second approach: Use $\mathbf{v}^n$ and a dimension reduction approach to simulate the values $\{\widetilde{W}_{m\delta}^n\}_{m=1}^M$ of a standard Brownian motion $(\widetilde{W}_s^n)_{s\in[0,1]}$, and set

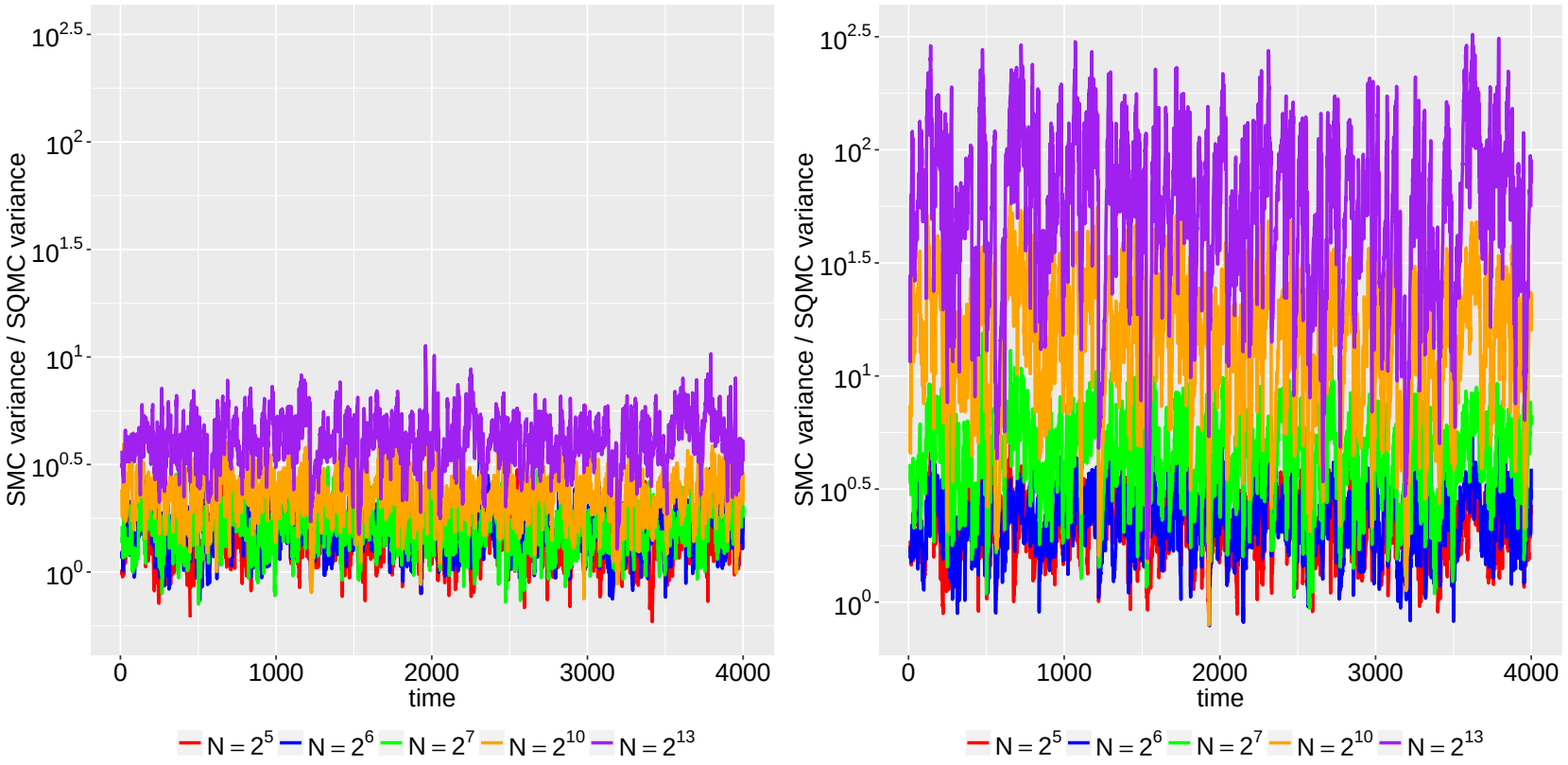
$$W_{t+m\delta}^n - W_{t+(m-1)\delta}^n = \widetilde{W}_{t+m\delta}^n - \widetilde{W}_{t+(m-1)\delta}^n, \quad m = 1, \ldots, M.$$

For the second approach, we will use the Brownian Bridge construction (Caflisch et al., 1997).

# Diffusion driven SV model: Simulation set-up

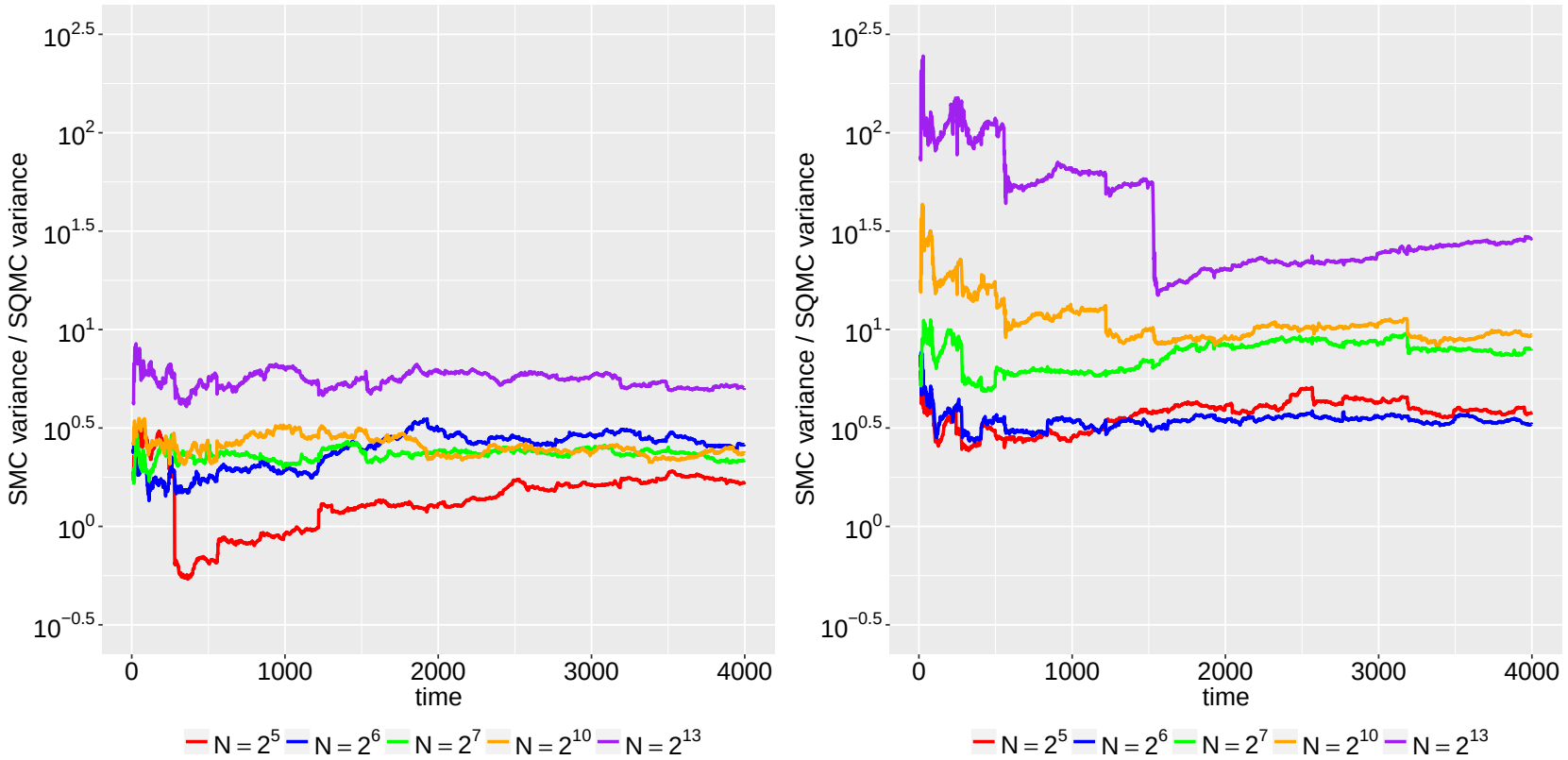

The parameters of the model are set to their estimated values for the daily return data on the closing price of the S&P 500 index from 5/5/1995 to 4/14/2003 (Chib et al., 2004).

# Diffusion driven SV model: Simulation Results




Estimation of $\mathbb{E}[X_t|Y_{0:T}]$ for $t \in \{1, \ldots, T\}$ and for different values of $N$ (and based 100 independent SMC and SQMC runs). SQMC is implemented with the forward method (left) and with the Brownian Bridge method (right).

# Diffusion driven SV model: Simulation Results




Estimation of the log-likelihood for different values of $N$ (and based 100 independent SMC and SQMC runs). SQMC is implemented with the forward method (left) and with the Brownian Bridge method (right).

# Conclusion

- Only requirement to replace SMC with SQMC is that the simulation of $\mathbf{x}_t^n | \mathbf{x}_{t-1}^n$ may be written as a $\mathbf{x}_t^n = \Gamma_t(\mathbf{x}_{t-1}^n, \mathbf{u}_t^n)$ where $\mathbf{u}_t^n \sim U[0,1]^d$.
- We observe very impressive gains in performance (even for small $N$ and not to small $d$).
- Supporting theory.

# References


- Gerber, M., and Chopin, N. Sequential quasi Monte Carlo. J. R. Stat. Soc. Ser. B. Stat. Methodol. 77, 3 (2015), 509–579. (read paper)
- Gerber, M., and Chopin, N. Convergence of sequential quasi-monte carlo smoothing algorithms. ArXiv preprint 1506.06117 (Bernoulli, forthcoming)
- Chopin, N., and Gerber, M. Application of sequential quasi-Monte Carlo to autonomous positioning. ArXiv preprint 1503.01631 (EUSIPCO 2015 proceedings).
- Forthcoming paper in MCQMC2016 proceedings.