Sequential Quasi Monte Carlo: extensions and new applications (up to infinite dimension)

N. Chopin (CREST-ENSAE)

joint work with Mathieu Gerber (Harvard)

nicolas.chopin@ensae.fr

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).

What's new?

- more efficient implementation of the Hilbert ordering
- extension to smoothing
- new examples with gain>1 although dimension=10
- Realisation that QMC gain is more important when good proposals are available
- application to diffusions (dim$=\infty$, end of talk)

Quick introduction to QMC

QMC basics

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 basics

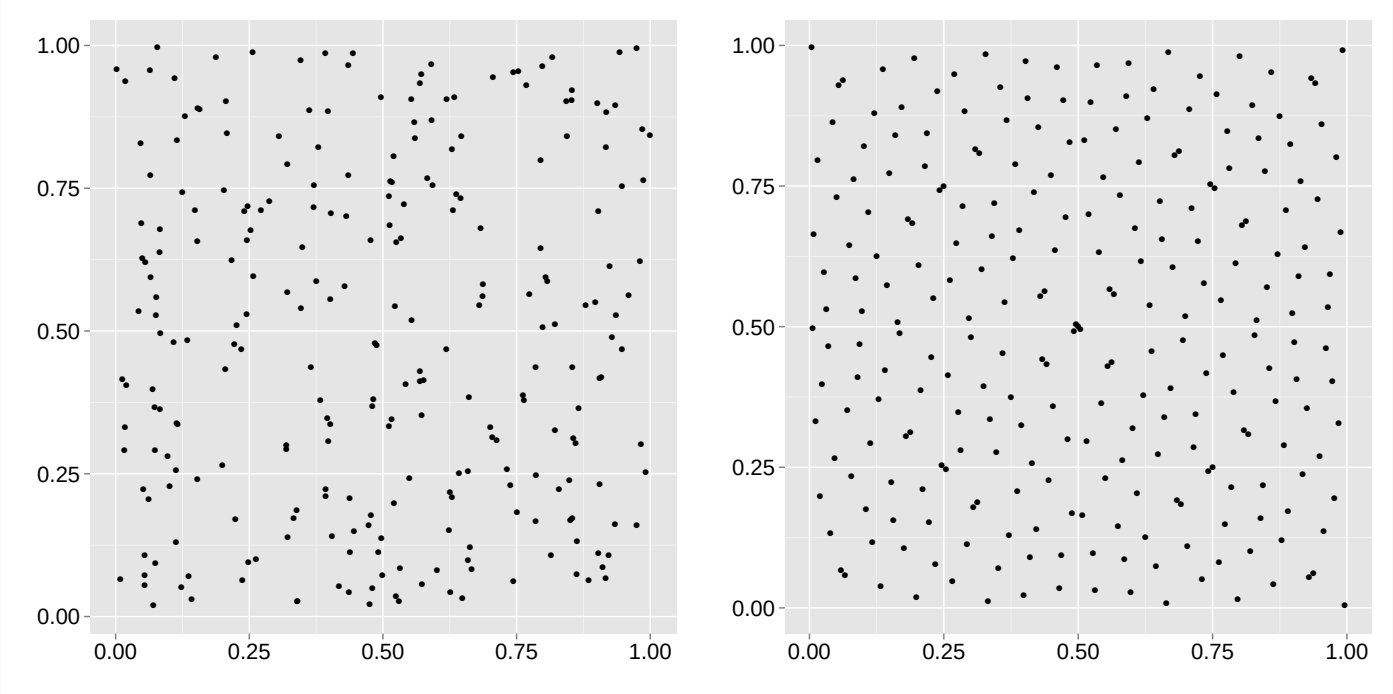
Consider the standard MC approximation

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

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 φ, 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 φ, 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 $D^\star(\mathbf{u}^{1:N}) = \mathcal{O}(N^{-1+\epsilon})$.

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 φ):

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

Motivation: Feynman-Kac models, particle filtering

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.

Application to rare events

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 $\mathbf{x}_s \in A_s$ for all $s \leq t$, $\mathbb{Q}_t$ is the dist' of $\mathbf{x}_t$ conditional on $\mathbf{x}_s \in A_s$ for $s \leq t$ and so on.

Application to HMMs (hidden Markov models)

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 (the law of $\mathbf{x}_t$ conditional on data $\mathbf{y}_{0:t}$), and Z_t into the likelihood of the data (the marginal density of $\mathbf{y}_{0:t}$).

Application to HMMs (hidden Markov models)

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 (the law of $\mathbf{x}_t$ conditional on data $\mathbf{y}_{0:t}$), and Z_t into the likelihood of the data (the marginal density of $\mathbf{y}_{0:t}$).

Applications in signal processing, Ecology, neurosciences...

Example: 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.

Simulated data

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

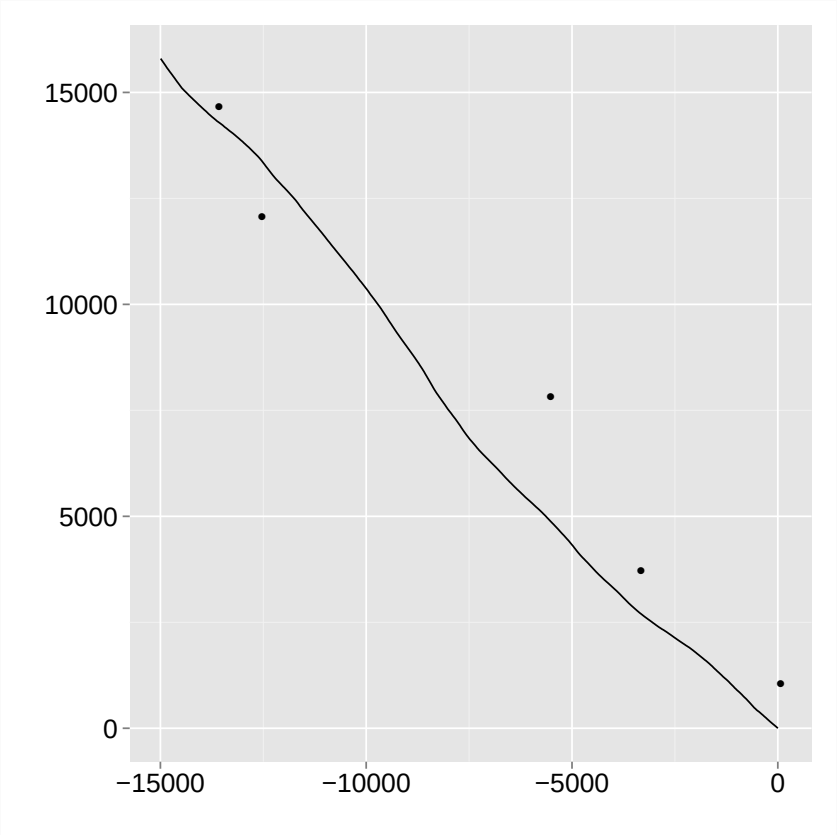


Figure 1: Simulated trajectory (15 min)

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

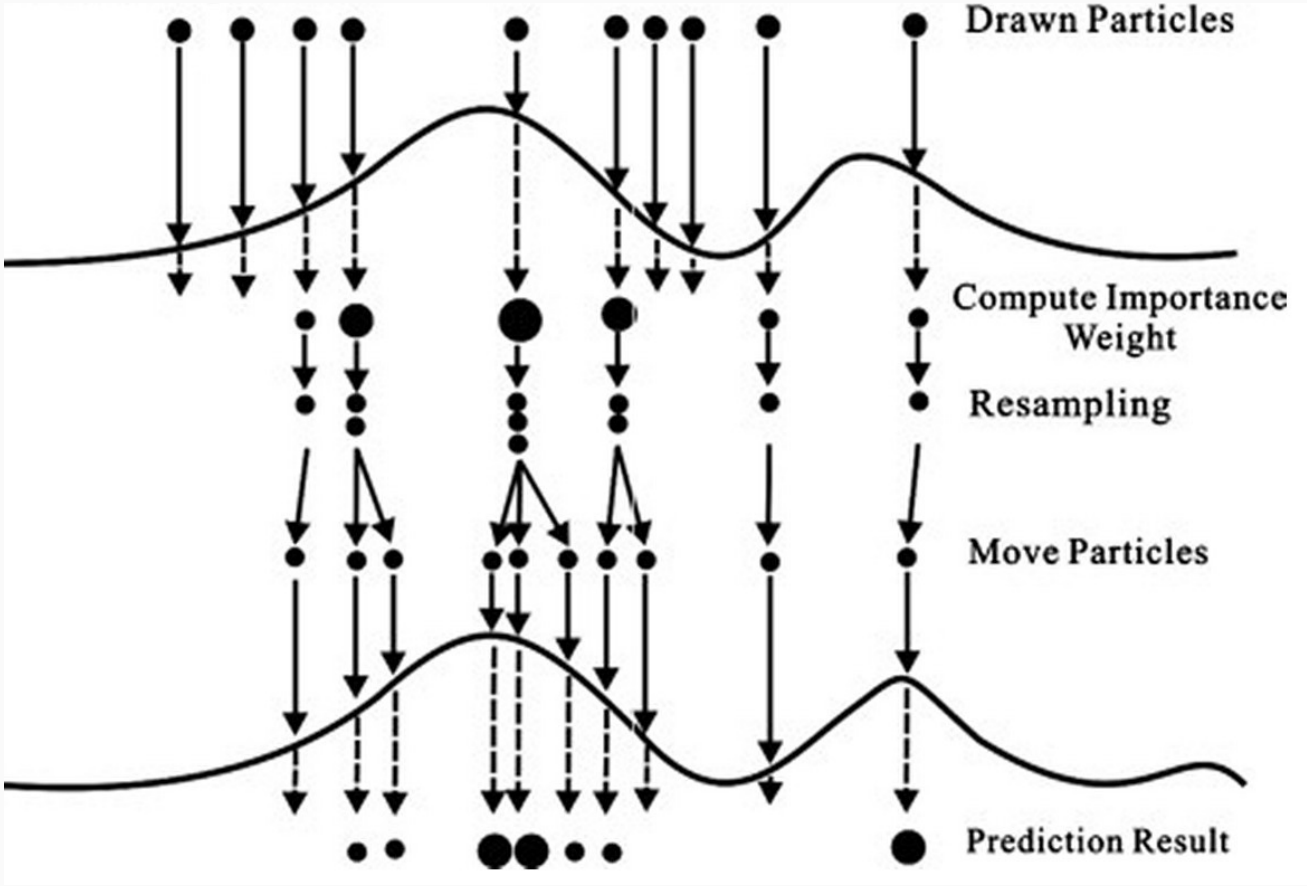
Recursively, for time $t = 1:T$,

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

Cartoon representation



Source: Chris Steinruecken

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 φ).

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 φ).

In addition, compute

$$Z_t^N = \prod_{s=0}^{t} \left\{ \frac{1}{N} \sum_{n=1}^{N} G_t(\mathbf{x}_{t-1}^{a_{t-1}^n}, \mathbf{x}_t^n) \right\}$$

as an approximation of Z_t (the likelihood of the data in a HMM).

Beyond boostrap filters: guided proposals

A PF such that $m_t(\mathbf{x}_{t-1}, \mathrm{d}\mathbf{x}_t)$ (the kernel used to simulate particles) matches $f_t(\mathbf{x}_{t-1}, \mathrm{d}\mathbf{x}_t)$ (the Markov kernel of $(\mathbf{x}_t)$ for the considered HMM) is called a bootstrap filter. However, it is possible, and often useful, to take $m_t \neq f_t$. Provided

$$G_t(x_{t-1}, x_t) = \frac{f_t(\mathbf{x}_{t-1}, \mathrm{d}\mathbf{x}_t) g(\mathbf{y}_t | \mathbf{x}_t)}{m_t(\mathbf{x}_{t-1}, \mathrm{d}\mathbf{x}_t)},$$

$\mathbb{Q}_t$ still matches the filtering distribution.

Beyond boostrap filters: guided proposals

A PF such that $m_t(\mathbf{x}_{t-1}, \mathrm{d}\mathbf{x}_t)$ (the kernel used to simulate particles) matches $f_t(\mathbf{x}_{t-1}, \mathrm{d}\mathbf{x}_t)$ (the Markov kernel of $(\mathbf{x}_t)$ for the considered HMM) is called a bootstrap filter. However, it is possible, and often useful, to take $m_t \neq f_t$. Provided

$$G_t(x_{t-1}, x_t) = \frac{f_t(\mathbf{x}_{t-1}, \mathrm{d}\mathbf{x}_t) g(\mathbf{y}_t | \mathbf{x}_t)}{m_t(\mathbf{x}_{t-1}, \mathrm{d}\mathbf{x}_t)},$$

$\mathbb{Q}_t$ still matches the filtering distribution.

The idea is to choose m_t so that the variance of G_t is as small as possible. The 'optimal' choice is the distribution of $\mathbf{x}_t$ conditional on $\mathbf{x}_{t-1}$ and $\mathbf{y}_t$ (usually intractable).

Beyond boostrap filters: guided proposals

A PF such that $m_t(\mathbf{x}_{t-1}, \mathrm{d}\mathbf{x}_t)$ (the kernel used to simulate particles) matches $f_t(\mathbf{x}_{t-1}, \mathrm{d}\mathbf{x}_t)$ (the Markov kernel of $(\mathbf{x}_t)$ for the considered HMM) is called a bootstrap filter. However, it is possible, and often useful, to take $m_t \neq f_t$. Provided

$$G_t(x_{t-1}, x_t) = \frac{f_t(\mathbf{x}_{t-1}, \mathrm{d}\mathbf{x}_t) g(\mathbf{y}_t | \mathbf{x}_t)}{m_t(\mathbf{x}_{t-1}, \mathrm{d}\mathbf{x}_t)},$$

$\mathbb{Q}_t$ still matches the filtering distribution.

The idea is to choose m_t so that the variance of G_t is as small as possible. The 'optimal' choice is the distribution of $\mathbf{x}_t$ conditional on $\mathbf{x}_{t-1}$ and $\mathbf{y}_t$ (usually intractable).

Notice how G_t depends on both $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$ in this case.

Simple example of a guided proposal

$$x_t | x_{t-1} \sim N(x_{t-1}, 1)$$
$$y_t = \mathbb{1}_{[0,\varepsilon]}(x_t)$$

and assume data such that $y_t = 1$ for all t.

Simple example of a guided proposal

$$x_t | x_{t-1} \sim N(x_{t-1}, 1)$$
$$y_t = \mathbb{1}_{[0,\varepsilon]}(x_t)$$

and assume data such that $y_t = 1$ for all t.

The bootstrap PF simulates particles from $N(x_{t-1}, 1)$, and kill particles that fall outside $[0, \varepsilon]$.

Instead, take $m_t(x_{t-1}, \mathrm{d}x_t)$ to be $N(x_{t-1}, 1)$ conditional on $x_t \in [0, \varepsilon]$. Then

$$G_t(x_{t-1}, x_t) = \Phi(\varepsilon - x_{t-1}) - \Phi(-x_{t-1})$$

and all particles fall in $[0, \varepsilon]$.

SQMC

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 Γ_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:

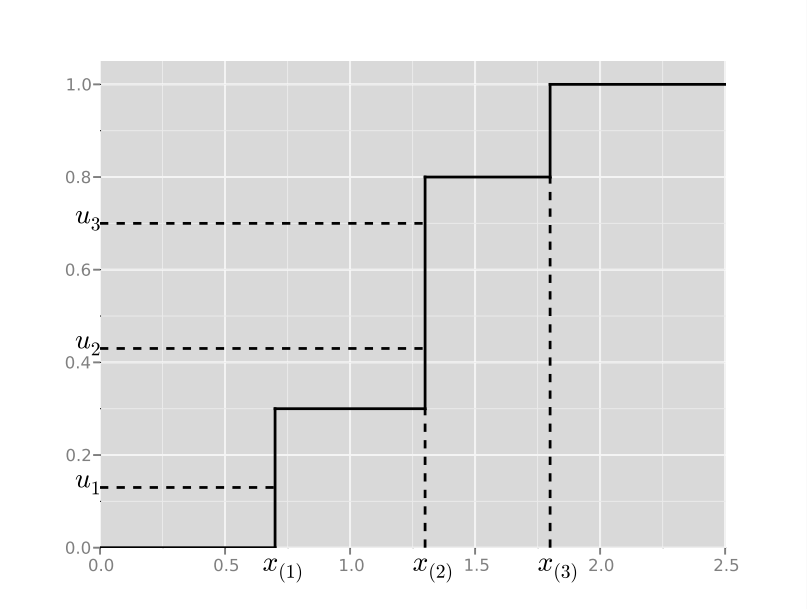
1. We will use the scalar u_t^n to choose the ancestor $\widetilde{\mathbf{x}}_{t-1}$.
2. We will use $\mathbf{v}_t^n$ to generate $\mathbf{x}_t^n$ as

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

where Γ_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 $d > 1$

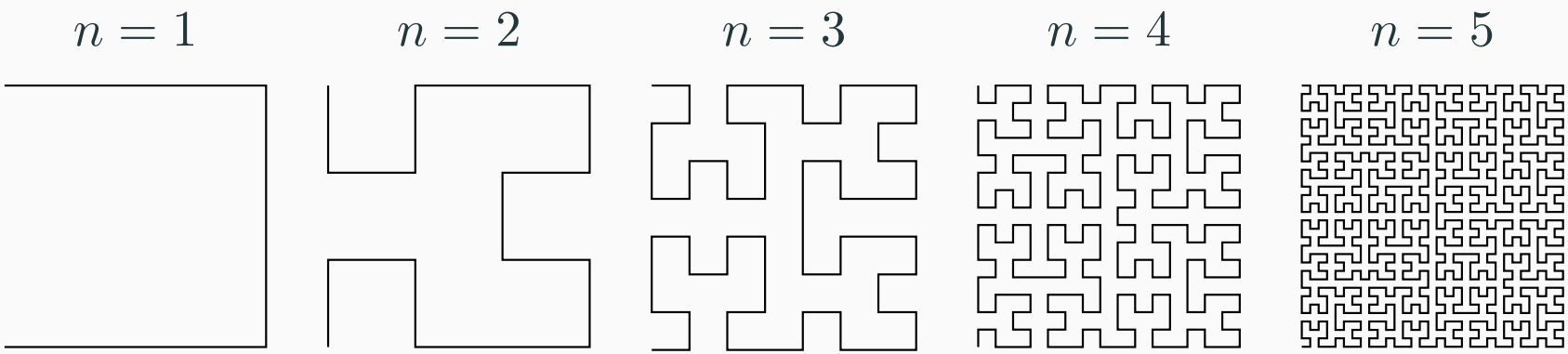
When $d > 1$, we cannot use the inverse CDF method to sample from the empirical distribution

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

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 ψ, 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: Marc van Dongen)

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 σ 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

- Related to array-RQMC (L'Ecuyer et al, 2006): same idea of using $(T+1)$ RQMC sequences of dimension $d+1$, rather than a single sequence of dimension $(T+1)d$.
- Because two sort operations are performed, the complexity of SQMC is $\mathcal{O}(N \log N)$ (while SMC is $\mathcal{O}(N)$).

Some remarks

- Related to array-RQMC (L'Ecuyer et al, 2006): same idea of using $(T+1)$ RQMC sequences of dimension $d+1$, rather than a single sequence of dimension $(T+1)d$.
- Because two sort operations are performed, the complexity of SQMC is $\mathcal{O}(N \log N)$ (while SMC is $\mathcal{O}(N)$).
- 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 Γ_t (e.g. multivariate inverse CDF).

Some remarks

- Related to array-RQMC (L'Ecuyer et al, 2006): same idea of using $(T+1)$ RQMC sequences of dimension $d+1$, rather than a single sequence of dimension $(T+1)d$.
- Because two sort operations are performed, the complexity of SQMC is $\mathcal{O}(N \log N)$ (while SMC is $\mathcal{O}(N)$).
- 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 Γ_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 develop QMC versions of 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 develop QMC versions of 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 φ, 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 φ,
$\mathbb{Q}_t^N(\varphi) - \mathbb{Q}_t(\varphi) \to 0$.

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 π_h is the image of π through π.

A few words on smoothing

Smoothing basics

Smoothing is approximating the dist' of the complete trajectory $\mathbf{x}_{0:T}$ at some final time T:

$$\bar{Q}_t(\varphi) = \frac{1}{Z_T}\mathbb{E}\left[\varphi(\mathbf{x}_{0:T})G_0(\mathbf{x}_0)\prod_{t=1}^{T} G_t(\mathbf{x}_{t-1}, \mathbf{x}_t)\right]$$

Usually it requires two passes:

1. A forward pass (performed by a standard PF)
2. A backward pass (often $\mathcal{O}(N^2)$)

FFBS (Forward filtering backward sampling)

In FFBS, we simulate smoothing trajectories from the output of the forward pass: component t of smoothing trajectory $\tilde{\mathbf{x}}_{0:t}^n$ is $\mathbf{x}_t^{B_t^n}$, where the indices B_t^n are simulated backwards in time:

- $B_T^n \sim \mathcal{M}(W_T^{1:N})$
- $B_{T-1}^n \sim \mathcal{M}(\tilde{W}_{T-1}^{1:N})$ with

$$\tilde{W}_{T-1}^n \propto W_{T-1}^n f_t(\mathbf{x}_t^{B_T^n} | \mathbf{x}_{T-1}^n)$$

- and so on.

QMC in FFBS

1. We can use either SMC or SQMC in the forward pass
2. for backward sampling, we can either use standard random numbers, or a QMC point set of length N and dimension $T + 1$.

QMC in FFBS

1. We can use either SMC or SQMC in the forward pass
2. for backward sampling, we can either use standard random numbers, or a QMC point set of length N and dimension $T + 1$.

That's 4 possibilities. They are all valid. In our experience it is mostly the forward pass that benefits from QMC.

Examples

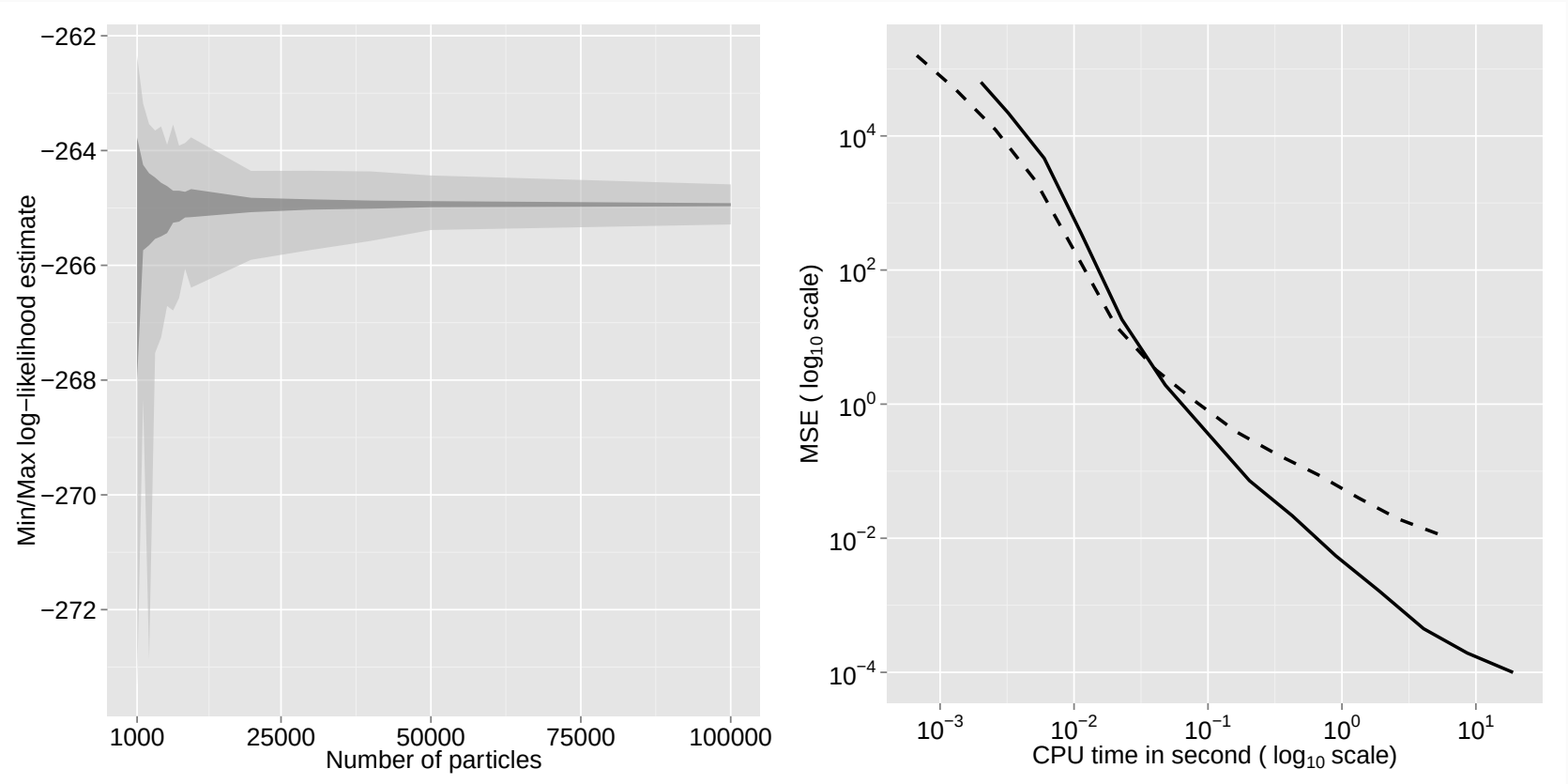
Examples: Kitagawa ($d = 1$)

Well known toy example (Kitagawa, 1998):

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

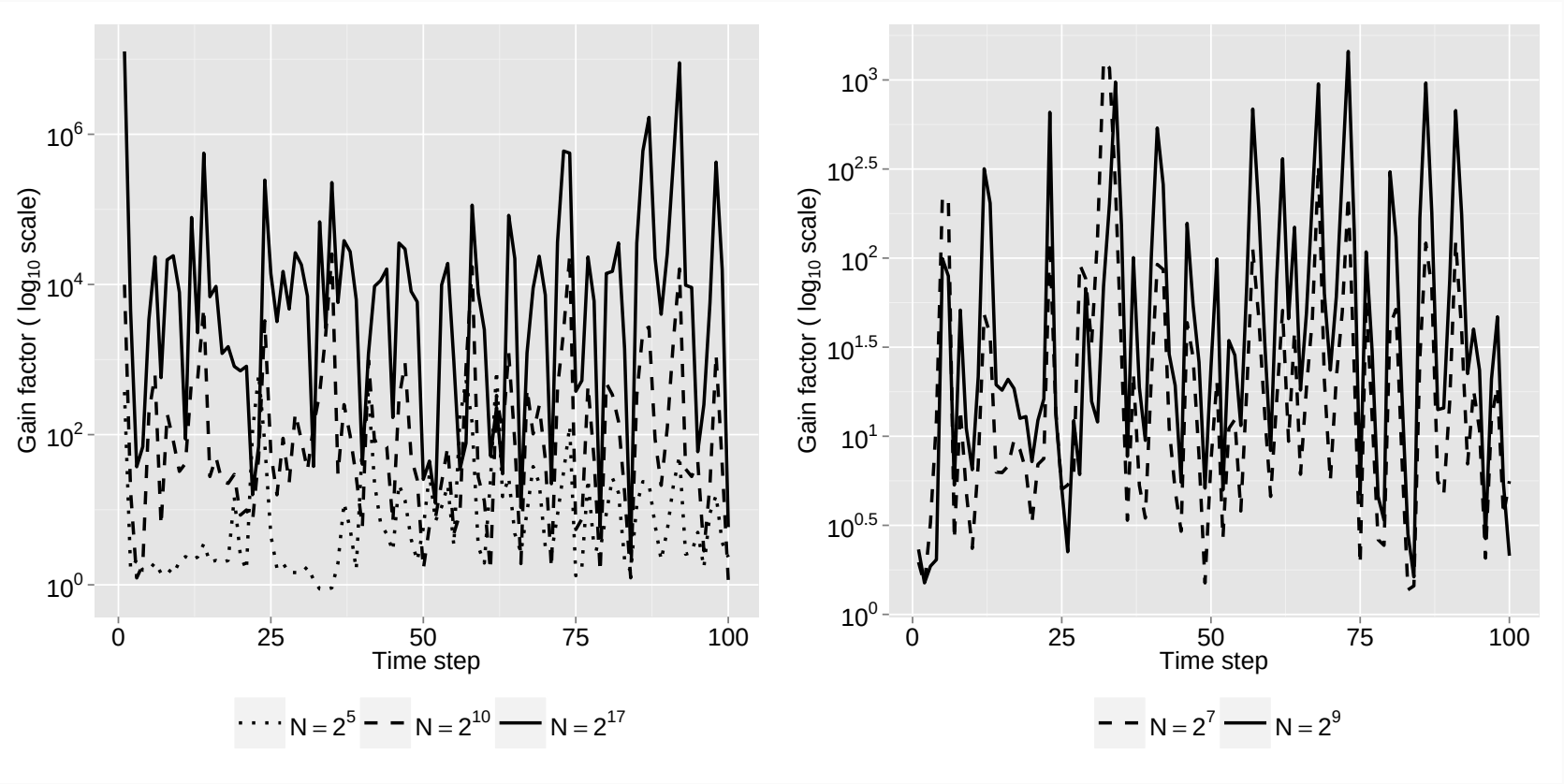
No paramater estimation (parameters are set to their true value).
We compare SQMC with SMC (based on systematic resampling) both in terms of N, and in terms of CPU time.

Kitagawa ($d = 1$)



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

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).

Autonomous positioning: results

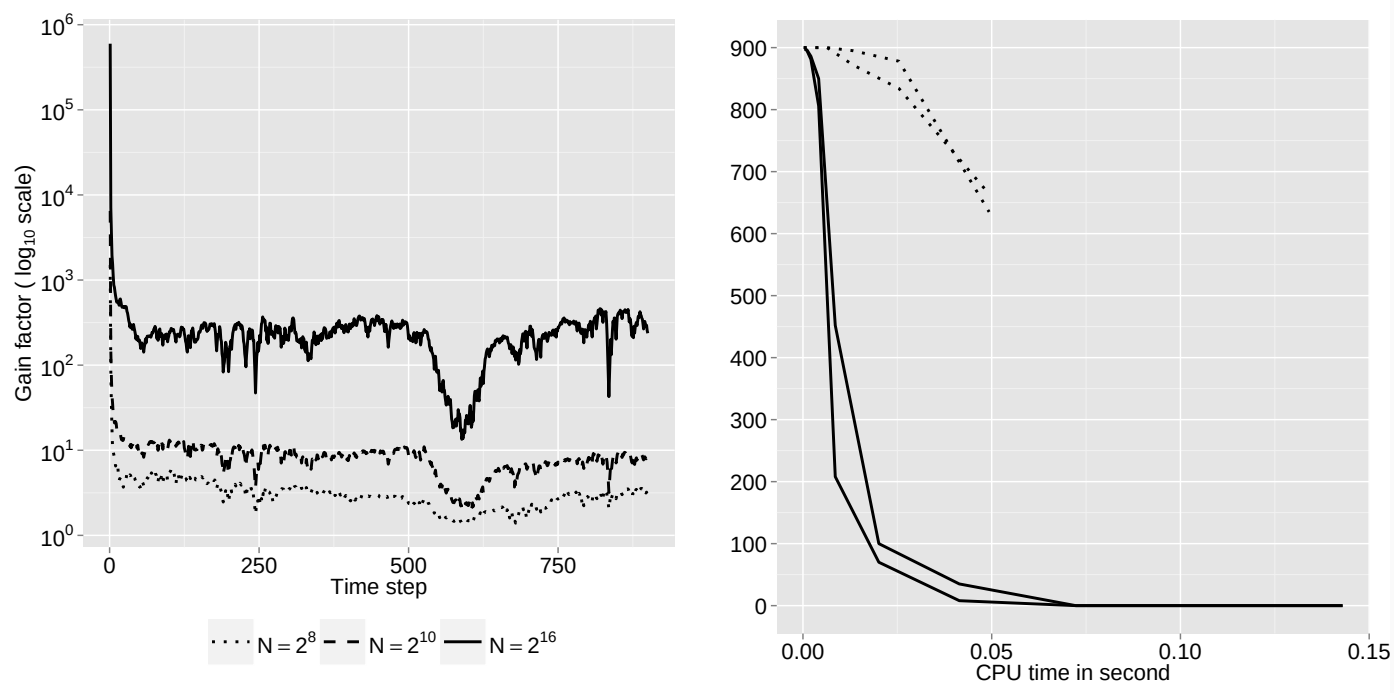


Figure 2: Left: Gain factor vs time (PF MSE/SQMC MSE); Right: number of time steps such that $\text{MSE}(\hat{x}_{t1}) > 0.01\text{Var}(x_{t1}|y_{0:t})$, as a function of CPU time

▸ Model

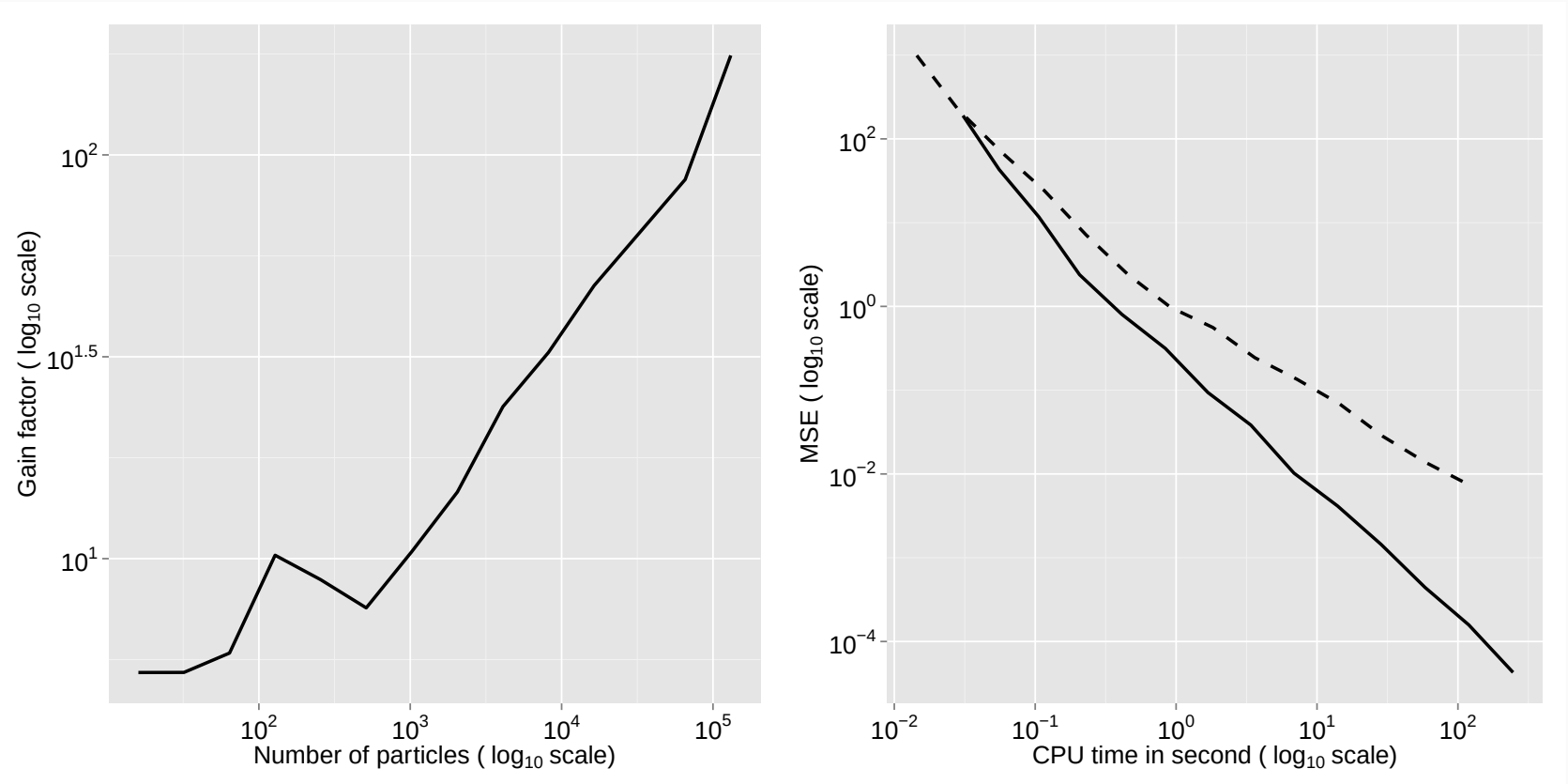
Multivariate Stochastic Volatility

Model is

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

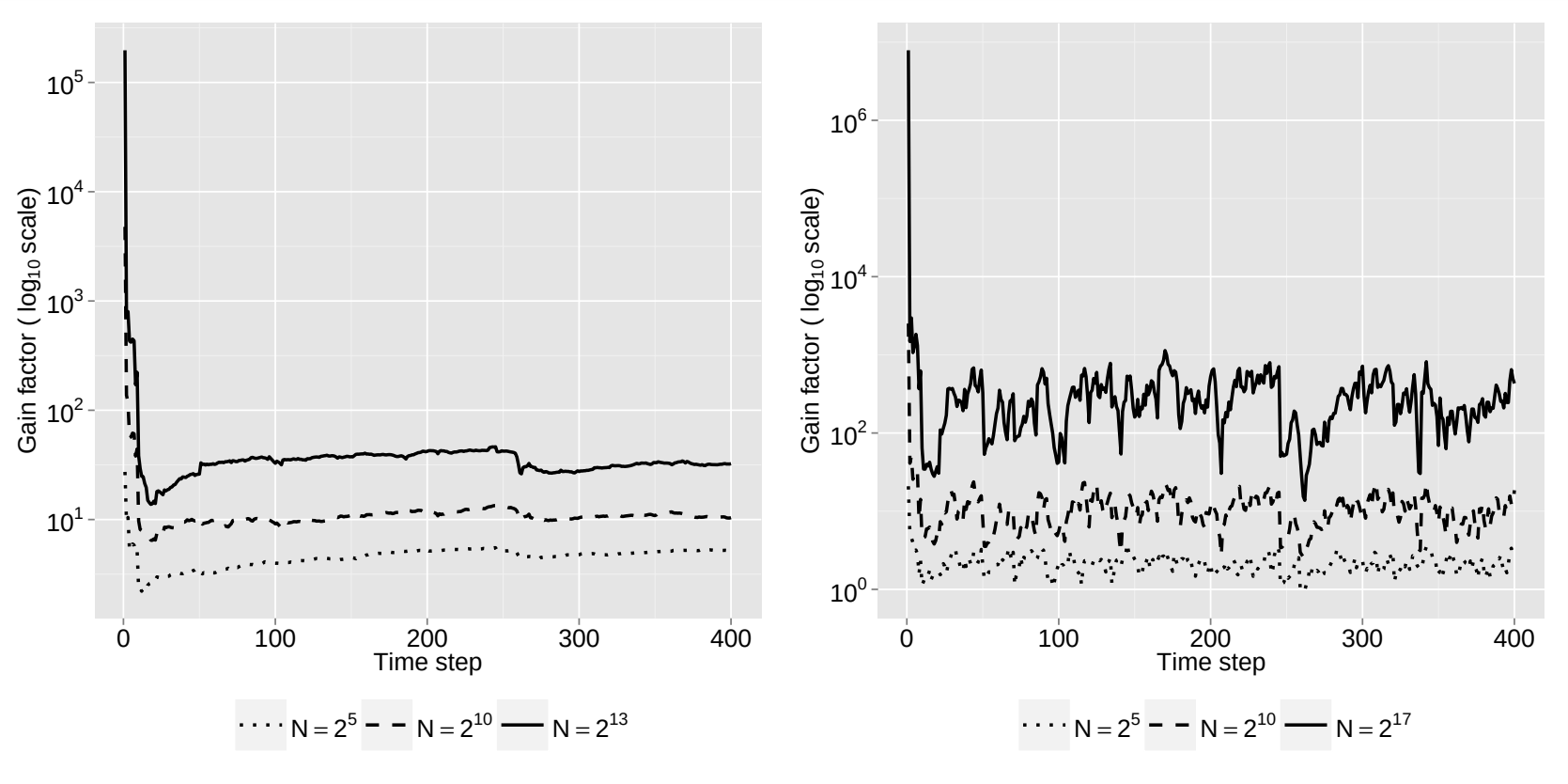
where $S_t = \mathrm{diag}(e^{x_{t1}}, \ldots, e^{x_{td}})$, with correlated noise terms: $(\boldsymbol{\epsilon}_t, \boldsymbol{\nu}_t) \sim N_{2d}(\mathbf{0}, \boldsymbol{C})$.

Multivariate Stochastic Volatility ($d = 2$)



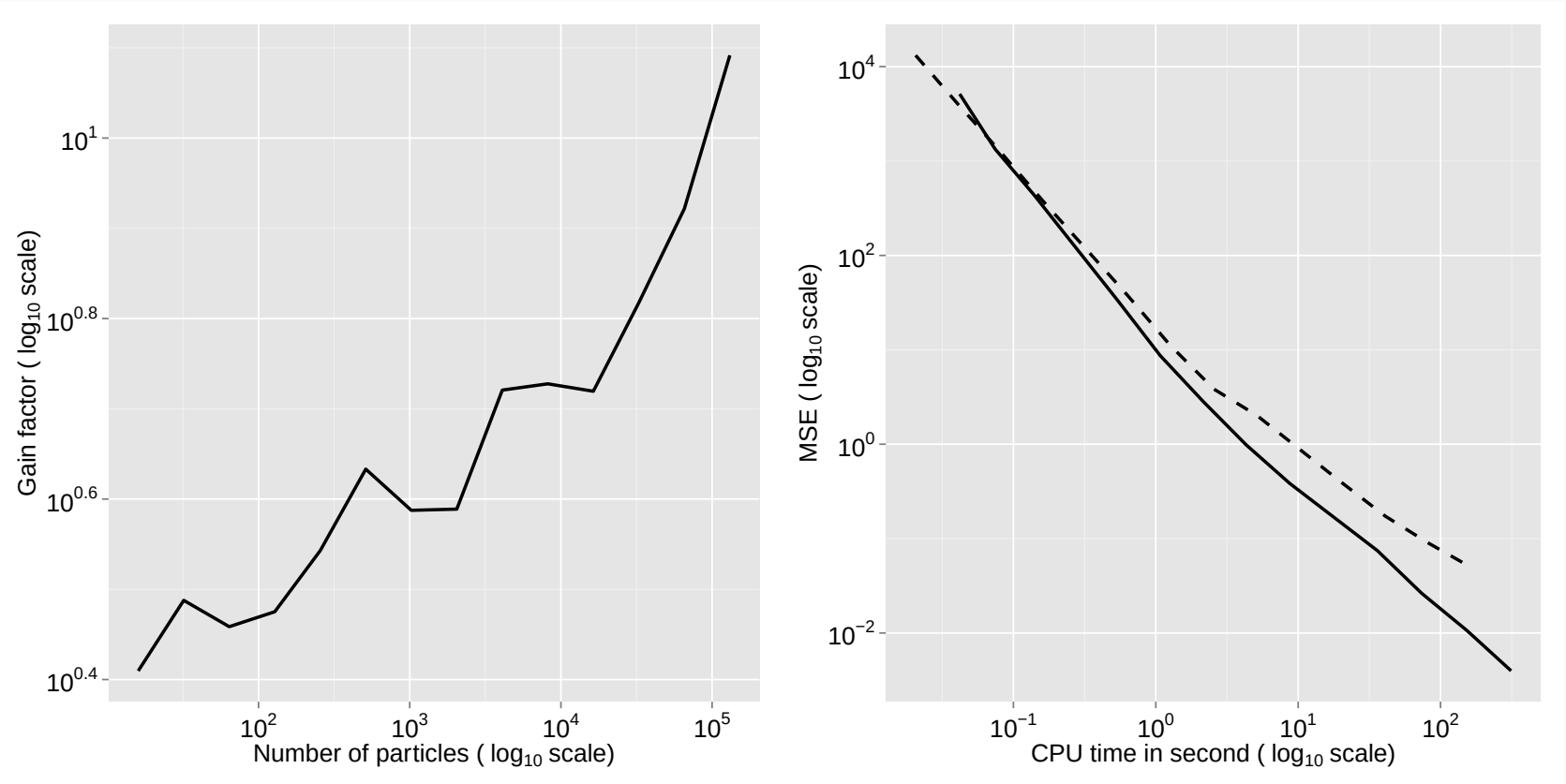
Log-likelihood evaluation (based on $T = 400$ data points and 200 independent runs).

Multivariate Stochastic Volatility ($d = 2$)



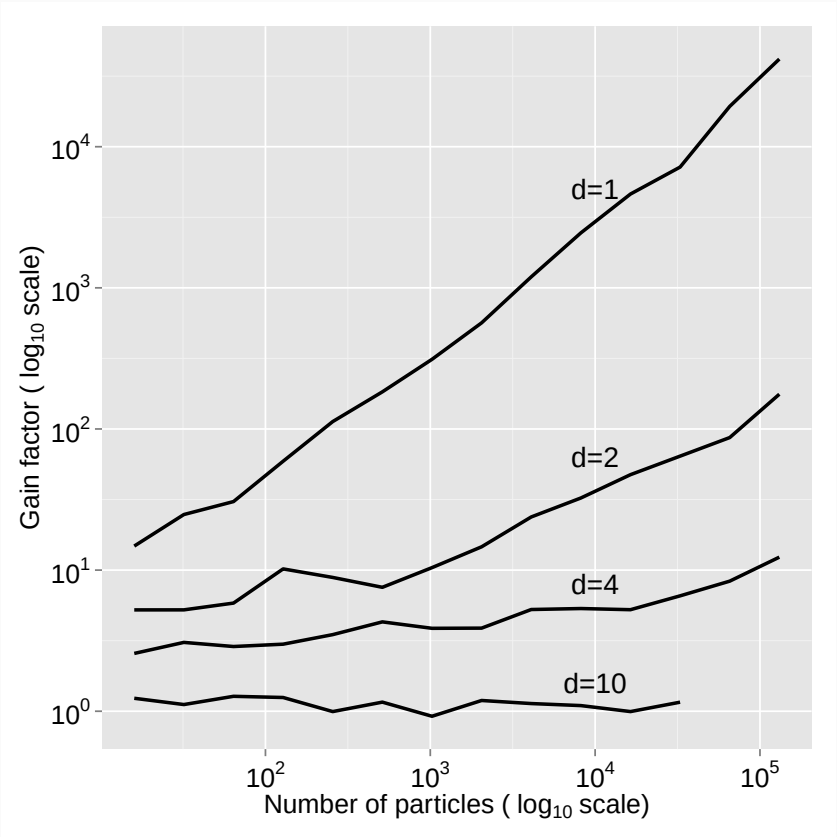
Log-likelihood evaluation (left) and filtering (right) as a function of t.

Multivariate Stochastic Volatility ($d = 4$)



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

Multivariate Stochastic volatility: dimension comparison



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

Remark on dimension

Increasing the dimension has two effects:

1. The Hilbert curve is less and less smooth (See also He and Owen, 2015)
2. The proportion of particles with non-negligible weights get small.

Remark on dimension

Increasing the dimension has two effects:

1. The Hilbert curve is less and less smooth (See also He and Owen, 2015)
2. The proportion of particles with non-negligible weights get small.

We managed recently to obtain reasonable gains even for $d = 10$ for a multivariate linear Gaussian model (using the optimal proposal).

Diffusion-driven SV model ($d = \infty$)

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

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 at times $t = 0, 1, \ldots, T$.

Euler Discretisation

For $M \geq 1$ (with $\delta = M^{-1}$), let $\boldsymbol{x}_t \in \mathbb{R}^M$ be such that:

$$\begin{cases} Y_t | Y_{t-1}, \mathbf{x}_t \sim \mathcal{N}_1\Big(Y_{t-1} + \mu_P + \beta\sigma_t^2 + \rho Z_t,\ (1-\rho^2)\sigma_t^2\Big) \\ \mathbf{x}_t(1) = \mathbf{x}_{t-1}(M) + \delta\mu(\mathbf{x}_{t-1}(M)) + \omega(\mathbf{x}_{t-1}(M))(W_{t-1+\delta} - W_{t-1}) \\ \vdots \\ \mathbf{x}_t(M) = \mathbf{x}_{t-1}(M-1) + \delta\mu(\mathbf{x}_{t-1}(M-1)) \\ \qquad\qquad + \omega(\mathbf{x}_{t-1}(M-1))(W_t - W_{t-\delta}) \end{cases}$$

and

$$\sigma_t^2 = \frac{1}{M}\sum_{m=1}^{M} e^{\mathbf{x}_{t-1}(m)}, \quad Z_t = \sum_{m=1}^{M} e^{\mathbf{x}_{t-1}(m)}\big(W_{t+m\delta} - W_{t+(m-1)\delta}\big).$$

Euler Discretisation

For $M \geq 1$ (with $\delta = M^{-1}$), let $\boldsymbol{x}_t \in \mathbb{R}^M$ be such that:

$$\begin{cases} Y_t | Y_{t-1}, \mathbf{x}_t \sim \mathcal{N}_1\Big(Y_{t-1} + \mu_P + \beta\sigma_t^2 + \rho Z_t,\ (1-\rho^2)\sigma_t^2\Big) \\ \mathbf{x}_t(1) = \mathbf{x}_{t-1}(M) + \delta\mu(\mathbf{x}_{t-1}(M)) + \omega(\mathbf{x}_{t-1}(M))(W_{t-1+\delta} - W_{t-1}) \\ \vdots \\ \mathbf{x}_t(M) = \mathbf{x}_{t-1}(M-1) + \delta\mu(\mathbf{x}_{t-1}(M-1)) \\ \qquad\qquad + \omega(\mathbf{x}_{t-1}(M-1))(W_t - W_{t-\delta}) \end{cases}$$

and

$$\sigma_t^2 = \frac{1}{M}\sum_{m=1}^{M} e^{\mathbf{x}_{t-1}(m)}, \quad Z_t = \sum_{m=1}^{M} e^{\mathbf{x}_{t-1}(m)}\big(W_{t+m\delta} - W_{t+(m-1)\delta}\big).$$

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

Resampling step collapses to $d = 1$

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

Resampling step collapses to $d = 1$

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

However, if we implement a bootstrap filter, we notice that (a) G_t depends only on $\mathbf{x}_t$; and (b) the simulation of $\tilde{\mathbf{x}}_t$ depends only on $\mathbf{x}_{t-1}(M)$ (the last component of $\mathbf{x}_{t-1}$), because process (X_t) is Markov. Thus we can collapse the choice of the ancestor to the choice of a scalar.

Resampling step collapses to $d = 1$

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

However, if we implement a bootstrap filter, we notice that (a) G_t depends only on $\mathbf{x}_t$; and (b) the simulation of $\tilde{\mathbf{x}}_t$ depends only on $\mathbf{x}_{t-1}(M)$ (the last component of $\mathbf{x}_{t-1}$), because process (X_t) is Markov. Thus we can collapse the choice of the ancestor to the choice of a scalar.

More generally, notion of effective dimension for the choice of the ancestors.

Mutation step of SQMC: Choice of Γ_t

To simulate $\mathbf{x}_t^n$ given $\mathbf{x}_{t-1}^n(M)$, we must simulate the innovation terms $W_{t+m\delta} - W_{t+(m-1)\delta}$.

- Forward approach: simulate the consecutive

$$W_{t+m\delta}^n - W_{t+(m-1)\delta}^n$$

  as IID $N(0, \delta)$ variates.

Mutation step of SQMC: Choice of Γ_t

To simulate $\mathbf{x}_t^n$ given $\mathbf{x}_{t-1}^n(M)$, we must simulate the innovation terms $W_{t+m\delta} - W_{t+(m-1)\delta}$.

- Forward approach: simulate the consecutive

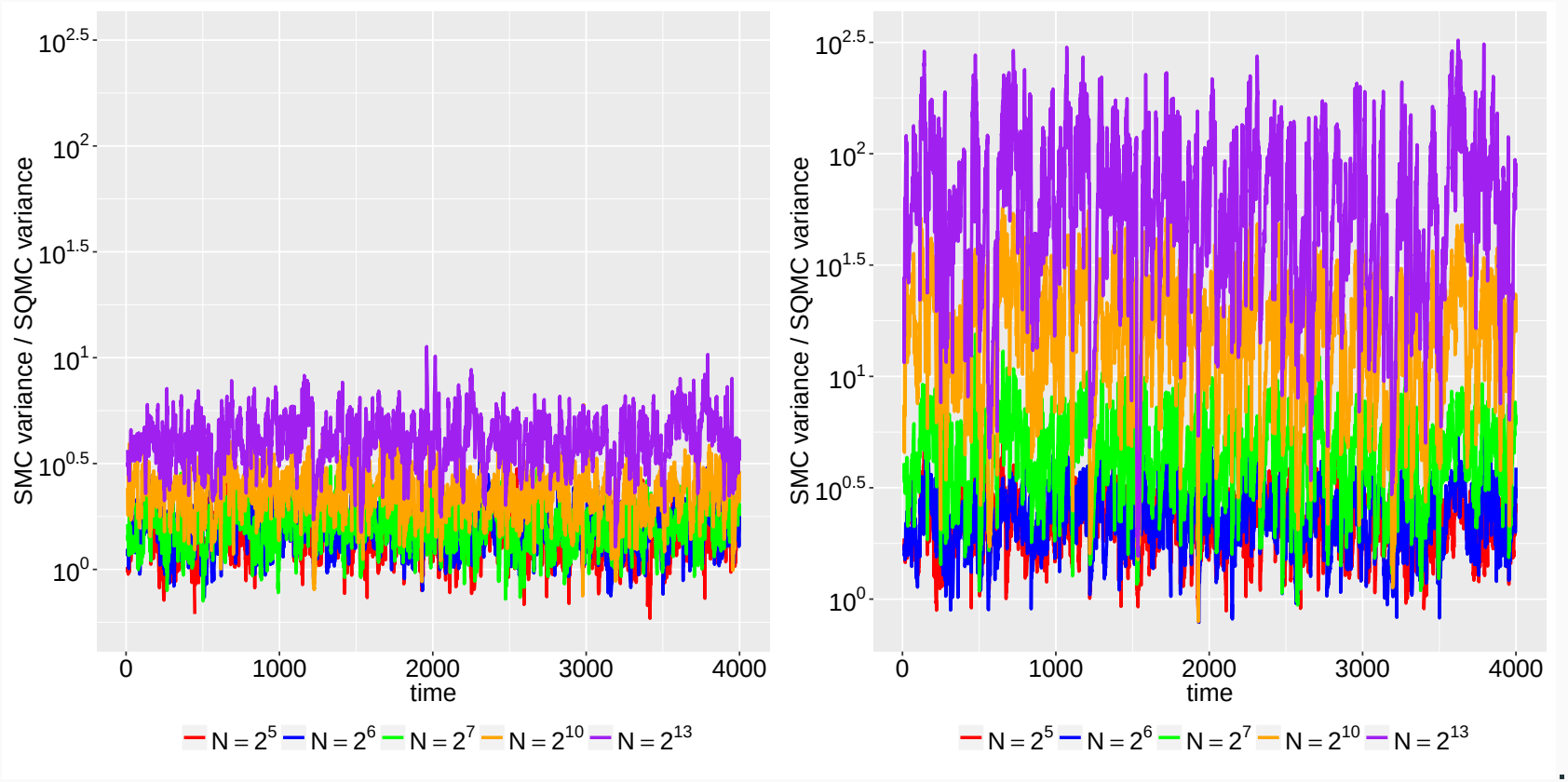
$$W_{t+m\delta}^n - W_{t+(m-1)\delta}^n$$

  as IID $N(0, \delta)$ variates.
- Brownian bridge: Simulate $W_{t+m\delta}$ (cond. on the previous ones) in the following order: $m = M$, $m = M/2$, $m = M/4$, $m = 3M/4$, ...

Diffusion driven SV model: Simulation set-up

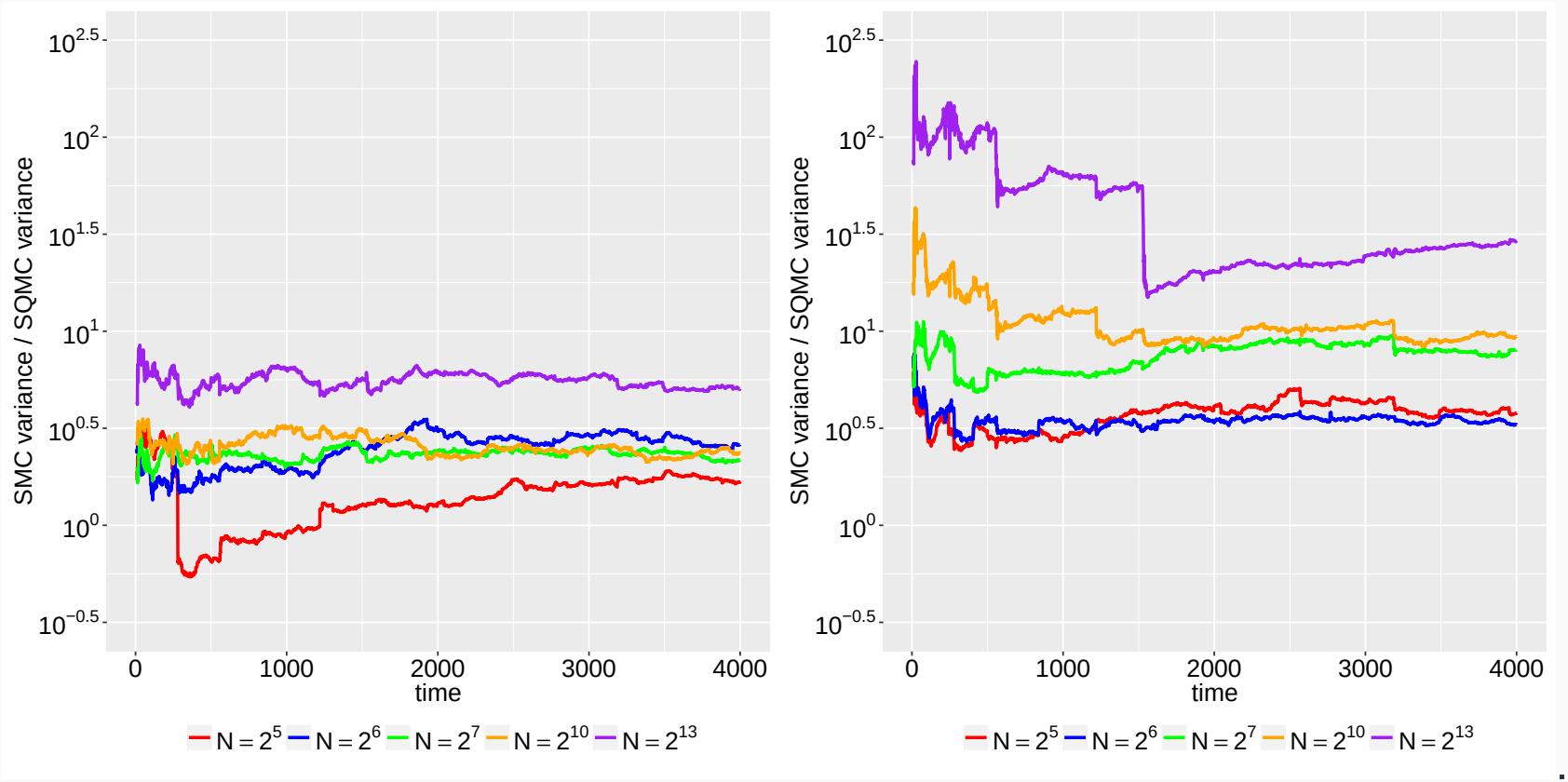
The parameters of the model are set to their estimated values for the daily return data on the closing price of the S&P 500 index from 5/5/1995 to 4/14/2003 (Chib et al., 2004).

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

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 $\mathbf{u}_t^n \sim U[0,1]^d$.
- impressive gains in performance (even for small N and moderate d).
- Supporting theory.
- C++ package by Mathieu (improved), Python library coming soon.

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**. Bernoulli (forthcoming), ArXiv:1506.06117.
- Chopin, N., and Gerber, M. **Application of sequential quasi-Monte Carlo to autonomous positioning** EUSIPCO 2015 proceedings, ArXiv:1503.01631.
- Forthcoming paper in MCQMC2016 proceedings.