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 $\mathcal{U}\left([0,1]^d\right)$.

QMC basics

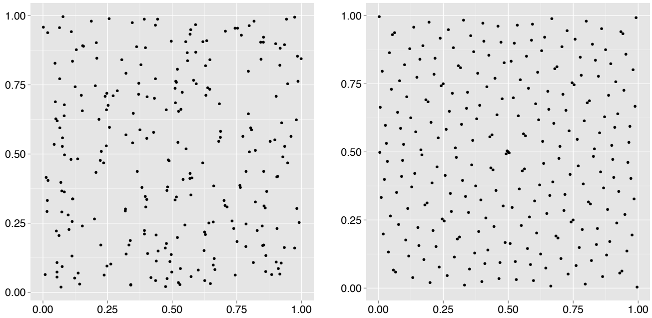
Consider the standard MC approximation

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

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:

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

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

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 $\mathbf{u}^n = \boldsymbol{w} + \mathbf{v}^n \equiv 1$, 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):

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

Particle Filtering: Hidden Markov models

Consider an unobserved Markov chain $(\mathbf{x}_t)$, $\mathbf{x}_0 \sim m_0(\mathrm{d}\mathbf{x}_0)$ and

$$\mathbf{x}_t | \mathbf{x}_{t-1} = \mathbf{x}_{t-1} \sim m_t(\mathbf{x}_{t-1}, \mathrm{d}\mathbf{x}_t)$$

taking values in $\mathcal{X} \subset \mathbb{R}^d$, and an observed process $(\mathbf{y}_t)$,

$$\mathbf{y}_t | \mathbf{x}_t \sim g(\mathbf{y}_t | \mathbf{x}_t).$$

Particle Filtering: Hidden Markov models

ENSAE CREST

Consider an unobserved Markov chain $(\mathbf{x}_t)$, $\mathbf{x}_0 \sim m_0(\mathrm{d}\mathbf{x}_0)$ and

$$\mathbf{x}_t | \mathbf{x}_{t-1} = \mathbf{x}_{t-1} \sim m_t(\mathbf{x}_{t-1}, \mathrm{d}\mathbf{x}_t)$$

taking values in $\mathcal{X} \subset \mathbb{R}^d$, and an observed process $(\mathbf{y}_t)$,

$$\mathbf{y}_t | \mathbf{x}_t \sim g(\mathbf{y}_t | \mathbf{x}_t).$$

Sequential analysis of HMMs amounts to recover quantities such as $p(x_t|y_{0:t})$ (filtering), $p(x_{t+1}|y_{0:t})$ (prediction), $p(y_{0:t})$ (marginal likelihood), etc., recursively in time. Many applications in engineering (tracking), finance (stochastic volatility), epidemiology, ecology, neurosciences, etc.

Taking $G_t(\mathbf{x}_{t-1}, \mathbf{x}_t) := g_t(\mathbf{y}_t|\mathbf{x}_t)$, we see that sequential analysis of a HMM may be cast into a Feynman-Kac model. In particular, filtering amounts to computing

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

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

and expectations are wrt the law of the Markov chain $(\mathbf{x}_t)$.

ENSAE CREST

Taking $G_t(\mathbf{x}_{t-1}, \mathbf{x}_t) := g_t(\mathbf{y}_t|\mathbf{x}_t)$, we see that sequential analysis of a HMM may be cast into a Feynman-Kac model. In particular, filtering amounts to computing

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

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

and expectations are wrt the law of the Markov chain $(\mathbf{x}_t)$.

Note: FK formalism has other applications that sequential analysis of HMM. In addition, for a given HMM, there is a more than one way to define a Feynmann-Kac formulation of that model.

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 $Z_0^N = N^{-1}\sum_{n=1}^N G_0(\mathbf{x}_0^n)$.

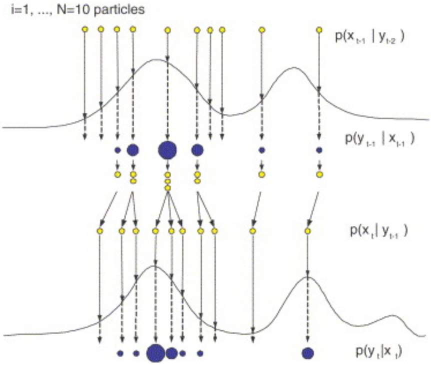
Recursively, for time $t = 1:T$,

(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 φ). 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

ENSAE CREST

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:

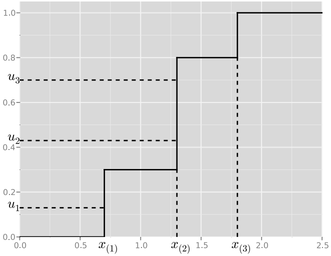
1. We will use the scalar u_t^n to choose the ancestor $\widetilde{\mathbf{x}}_{t-1}$.
2. We will use $\mathbf{v}_t^n$ to generate $\mathbf{x}_t^n$ as

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

   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 $d > 1$

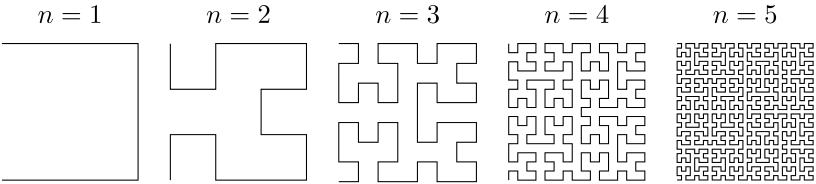
When $d > 1$, we cannot use the inverse CDF method to sample from the empirical distribution

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

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: 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 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

- 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 Γ_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 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

ENSAE CREST

- 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.

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.

Application: autonomous positioning

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

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

and noise terms ϵ_t, ν_t are Laplace-distributed.

Application: simulated data

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

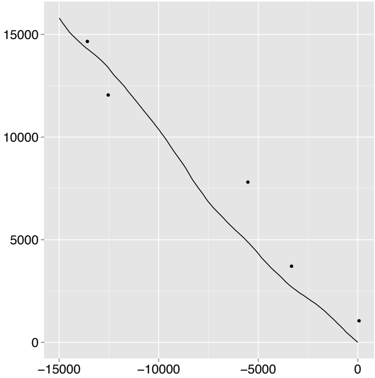


Figure : Simulated trajectory (15 min)

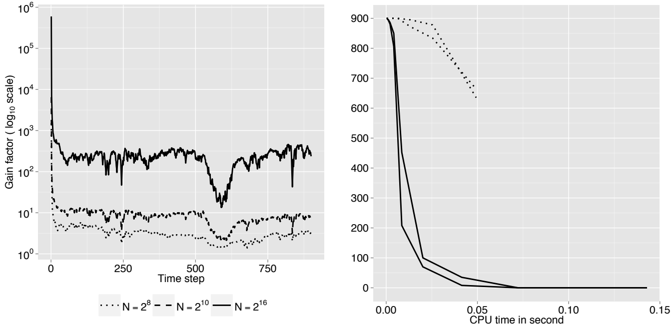


Figure : Left: Gain factor vs time (PF MSE/SQMC MSE); Right: number of time steps such that $\mathrm{MSE}(\hat{x}_{t1}) > 0.01\mathrm{Var}(x_{t1}|y_{0:t})$, as a function of CPU time

Conclusion

ENSAE CREST

- 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 or $d = 6$).
- Supporting theory.

Further work

- Adaptive resampling (triggers resampling steps when weight degeneracy is too high).
- Adapt SQMC to situations where sampling from $m_t(\mathbf{x}_{t-1}^n, \mathrm{d}\mathbf{x}_t)$ involves some accept/reject mechanism.
- Adapt SQMC to situations where sampling from $m_t(\mathbf{x}_{t-1}^n, \mathrm{d}\mathbf{x}_t)$ is a Metropolis step. In this way, we could develop SQMC counterparts of SMC samplers (Del Moral et al, 2006).
- SQMC^2 (QMC version of SMC^2, C. et al, 2013)?

Further work

- Adaptive resampling (triggers resampling steps when weight degeneracy is too high).
- Adapt SQMC to situations where sampling from $m_t(\mathbf{x}_{t-1}^n, \mathrm{d}\mathbf{x}_t)$ involves some accept/reject mechanism.
- Adapt SQMC to situations where sampling from $m_t(\mathbf{x}_{t-1}^n, \mathrm{d}\mathbf{x}_t)$ is a Metropolis step. In this way, we could develop SQMC counterparts of SMC samplers (Del Moral et al, 2006).
- SQMC^2 (QMC version of SMC^2, C. et al, 2013)?

Paper on Arxiv, was published last year as a read paper in JRSSB.

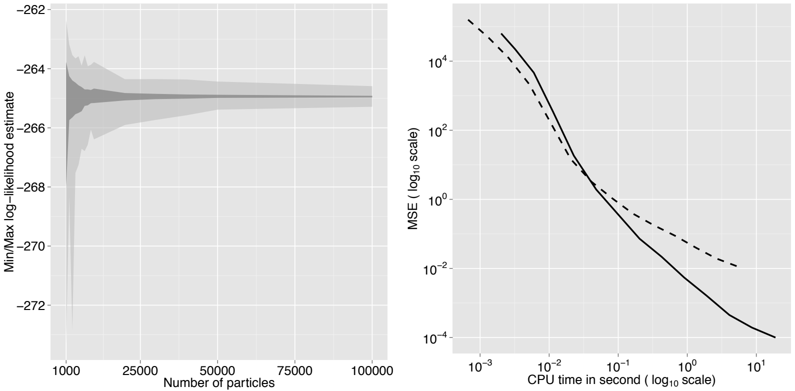
ENSAE CREST

Well known toy example (Kitagawa, 1998):

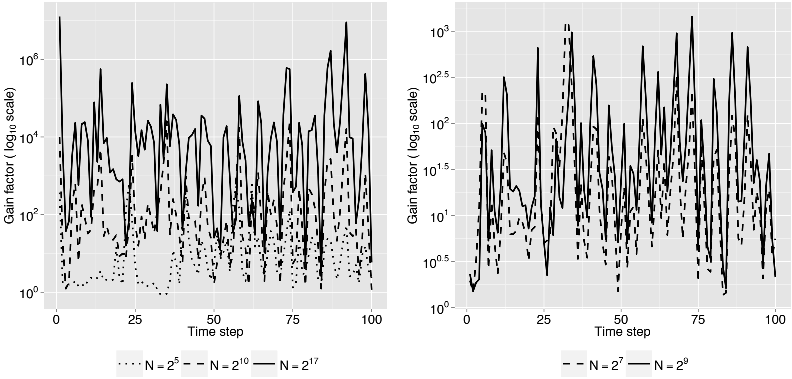
$$\begin{cases} y_t = \frac{x_t^2}{a} + \epsilon_t \\ x_t = b_1 x_{t-1} + b_2 \frac{x_{t-1}}{1+x_{t-1}^2} + b_3 \cos(b_4 t) + \sigma \nu_t \end{cases}$$

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).



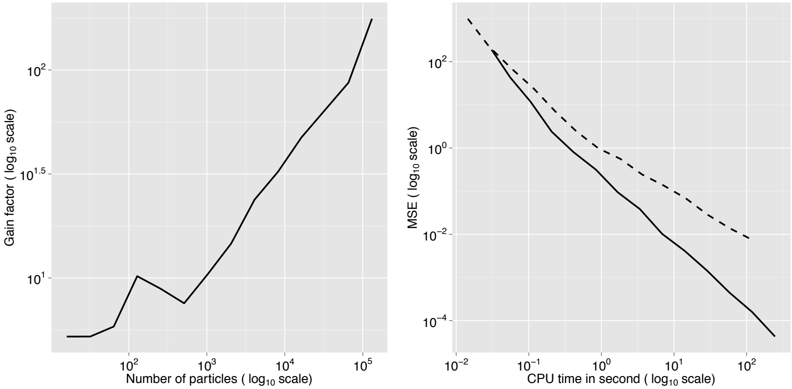
Filtering: computing $\mathbb{E}(\mathbf{x}_t|\mathbf{y}_{0:t})$ at each iteration t. Gain factor is MSE(SMC)/MSE(SQMC).

Examples: Multivariate Stochastic Volatility

Model is

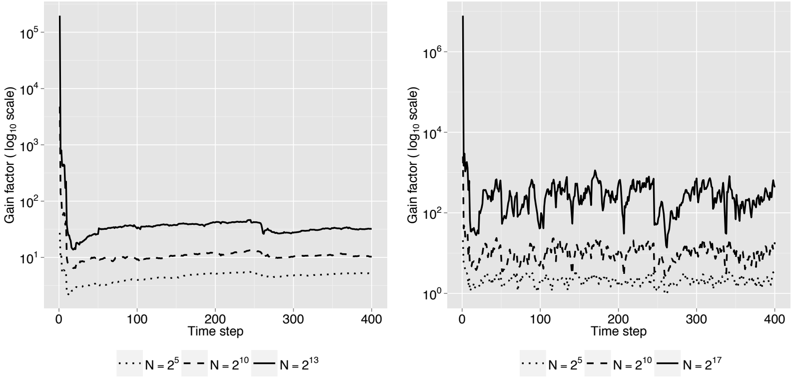
$$\begin{cases} \mathbf{y}_t = S_t^{\frac{1}{2}} \boldsymbol{\epsilon}_t \\ \mathbf{x}_t = \boldsymbol{\mu} + \Phi(\mathbf{x}_{t-1} - \boldsymbol{\mu}) + \Psi^{\frac{1}{2}} \boldsymbol{\nu}_t \end{cases}$$

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 $d = 4$.



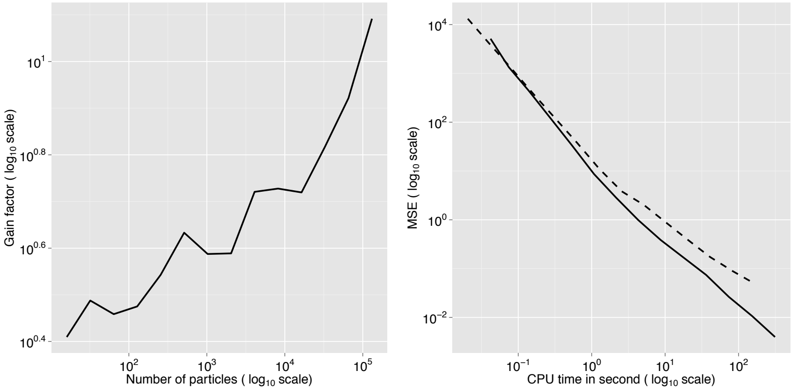
Log-likelihood evaluation (based on $T = 400$ data points and 200 independent runs).

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)