Sequential Quasi Monte Carlo

N. Chopin (CREST-ENSAE)

nicolas.chopin@ensae.fr

joint work with Mathieu Gerber (CREST, Université de Lausanne)
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 $O_P(N^{-1/2})$. 
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 $O_P(N^{-1/2})$.

Quasi Monte Carlo (QMC) is a substitute for standard Monte Carlo (MC), which typically converges at the faster rate $O_P(N^{-1+\epsilon})$. However, standard QMC is usually defined for IID problems.
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 $O_P(N^{-1/2})$.

Quasi Monte Carlo (QMC) is a substitute for standard Monte Carlo (MC), which typically converges at the faster rate $O_P(N^{-1+\epsilon})$. 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

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

where the $N$ vectors $u^n$ are IID variables simulated from $\mathcal{U}[0,1]^d$. 
Consider the standard MC approximation

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

where the \( N \) vectors \( u^n \) are IID variables simulated from \( U[0,1]^d \).

In SQMC, we replace \( u^{1:N} \) 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)
Koksma–Hlawka inequality:

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

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

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

There are various ways to construct point sets \( P_N = \{u^{1:N}\} \) so that \( D^*(u^{1:N}) = O(N^{-1+\epsilon}) \). (Describing these different constructions is beyond the scope of this talk.)
Consider a Markov chain \((x_t), \ x_0 \sim m_0(dx_0)\) and 
\(x_t | x_{t-1} = x_{t-1} \sim m_t(x_{t-1}, dx_t)\) taking values in \(\mathcal{X} \subset \mathbb{R}^d\), and a sequence of potential functions \(G_t : \mathcal{X} \times \mathcal{X} \to \mathbb{R}\). One would like to compute quantities such as

\[
Q_t(\varphi) = \frac{1}{Z_t} \mathbb{E} \left[ \varphi(x_t) G_0(x_0) \prod_{s=1}^{t} G_s(x_{s-1}, x_s) \right],
\]

with \(Z_t = \mathbb{E} \left[ G_0(x_0) \prod_{s=1}^{t} G_s(x_{s-1}, x_s) \right].\)
The Feynman-Kac formalism is rather abstract, but consider a Hidden Markov model, with latent Markov process \((x_t)\), observed process \((y_t)\), such that \(y_t \mid x_t \sim f^Y (y_t \mid x_t)\). If we take

\[
G_t(x_{t-1}, x_t) = f^Y (y_t \mid x_t)
\]

we observe that \(Q_t(\varphi)\) is the filtering expectation of function \(\varphi\), and that \(Z_t\) is the marginal likelihood of the data \(y_{0:t}\).

\[
Z_t = \int_{\mathcal{X}^{T+1}} m_0(dx_0) f^Y (y_0 \mid x_0) \prod_{s=1}^t \left\{ m_s(x_{s-1}, dx_s) f^Y (y_s \mid x_s) \right\}.
\]
Particle filtering: the algorithm

Operations must be performed for all $n \in 1 : N$.

At time 0,

(a) Generate $x_0^n \sim m_0(dx_0)$.

(b) Compute $W_0^n = G_0(x_0^n) / \sum_{m=1}^{N} G_0(x_0^m)$ and $Z_0^N = N^{-1} \sum_{n=1}^{N} G_0(x_0^n)$.

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

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

(b) Generate $x_t^n \sim m_t(x_{t-1}^{a_{t-1}^{n-1}}, dx_t)$.

(c) Compute $W_t^n = G_t(x_{t-1}^{a_{t-1}^{n-1}}, x_t^n) / \sum_{m=1}^{N} G_t(x_{t-1}^{a_{t-1}^{m-1}}, x_t^m)$ and $Z_t^N = Z_{t-1}^N \left\{ N^{-1} \sum_{n=1}^{N} G_t(x_{t-1}^{a_{t-1}^{n-1}}, x_t^n) \right\}$. 
At iteration $t$, compute

$$Q_t^N(\varphi) = \sum_{n=1}^{N} W_t^n \varphi(x_t^n)$$

to approximate $Q_t(\varphi)$ (the filtering expectation of $\varphi$). In addition, compute

$$Z_t^N$$

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

Source for image: some dark corner of the Internet.
A closer look at resampling

The resampling step (a) ensures that particle $x_{t-1}^n$ has $O_{t-1}^n$ off-springs, where is $O_{t-1}^n$ random and such that
$$E(O_{t-1}^n) = NW_{t-1}^n.$$
A closer look at resampling

The resampling step (a) ensures that particle $x_{t-1}^n$ has $O_{t-1}^n$ off-springs, where is $O_{t-1}^n$ random and such that

$$\mathbb{E}(O_{t-1}^n) = NW_{t-1}^n.$$

In this way, particles with large (normalised) weight get many off-springs, while particles with small weights are likely to generate zero off-springs.
A closer look at resampling

The resampling step (a) ensures that particle $x_{t-1}^n$ has $O_{t-1}^n$ off-springs, where is $O_{t-1}^n$ random and such that
\[ \mathbb{E}(O_{t-1}^n) = NW_{t-1}^n . \]

In this way, particles with large (normalised) weight get many off-springs, while particles with small weights are likely to generate zero off-springs.

One way to formalise resampling is that of drawing $N$ times independently from
\[
\sum_{n=1}^{N} \mathcal{W}_{t-1}^n \delta_{x_{t-1}^n} (d\tilde{x}_{t-1})
\]
which is our current approx. of $Q_{t-1}(dx_{t-1})$. 
In fact, we can formalise the succession of the resampling step (a) and the mutation step (b) at iteration $t$ as an importance sampling step from random probability measure

$$
\bar{Q}_t^N \left( d(\tilde{x}_{t-1}, x_t) \right) = \sum_{n=1}^{N} W_{t-1}^n \delta_{x_{t-1}^n} \left( d\tilde{x}_{t-1} \right) m_t \left( \tilde{x}_{t-1}, d x_t \right)
$$

to

$$
Q_t^N \left( d(\tilde{x}_{t-1}, x_t) \right) \propto \bar{Q}_t^N \left( d(\tilde{x}_{t-1}, x_t) \right) G_t \left( \tilde{x}_{t-1}, x_t \right).
$$
Formalisation

In fact, we can formalise the succession of the resampling step (a) and the mutation step (b) at iteration $t$ as an importance sampling step from random probability measure $Q_N^t(d(\tilde{x}_{t-1}, x_t)) = \sum_{n=1}^{N} W_t d_{\tilde{x}_{t-1}} \delta_{\tilde{x}_{t-1}} m_t(\tilde{x}_{t-1}, dx_t)$

to

$$Q_N^t(d(\tilde{x}_{t-1}, x_t)) \propto Q_N^t(d(\tilde{x}_{t-1}, x_t)) G_t(\tilde{x}_{t-1}, x_t).$$

Idea: use QMC instead of MC to sample $N$ points from $Q_N^t(d(\tilde{x}_{t-1}, x_t))$. The main difficulty is that this distribution is partly discrete, partly continuous.
Case $d = 1$

Let $u^n_t = (v^n_t, w^n_t)$ be uniform variates in $[0, 1]^2$. Then

1. Use the inverse transform to obtain $\tilde{x}^{n-1}_t = F^{-1}(v^n_t)$, where $\hat{F}$ is the empirical cdf of $\sum_{n=1}^{N} W^n_{t-1} \delta_{x^n_{t-1}}(d\tilde{x}_{t-1})$.

2. Sample $x^n_t \sim m_t(\tilde{x}^{n-1}_t, dx_t)$ as: $x^n_t = \Gamma_t(x^n_{t-1}, w^n_t)$, where $\Gamma_t$ is e.g. the inverse CDF of $m_t(\tilde{x}^{n-1}_t, dx_t)$ (or some other appropriate deterministic function)
From $d = 1$ to $d > 1$

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

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

Idea: we “project” the $x_{t-1}^{n}$’s into $[0, 1]$ through the (generalised) inverse of the Hilbert curve, which is a fractal, space-filling curve $H : [0, 1] \rightarrow [0, 1]^d$. 
From $d = 1$ to $d > 1$

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

$$\sum_{n=1}^{N} \mathcal{W}_{t-1}^n \delta_{x_{t-1}^n}(d\tilde{x}_{t-1}).$$

Idea: we “project” the $x_{t-1}^n$’s into $[0, 1]$ through the (generalised) inverse of the Hilbert curve, which is a fractal, space-filling curve $H : [0, 1] \rightarrow [0, 1]^d$.

More precisely, we transform $\mathcal{X}$ into $[0, 1]^d$ through some function $\psi$, then we transform $[0, 1]^d$ into $[0, 1]$ through $h = H^{-1}$. 
The Hilbert curve is the limit of this sequence. Note the locality property of the Hilbert curve: if two points are close in $[0, 1]^d$, then the corresponding transformed points remains close in $[0, 1]$. (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 $\mathcal{W}_0^n = G_0(\mathbf{x}_0^n) / \sum_{m=1}^{N} G_0(\mathbf{x}_0^m)$.

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

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

(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 $\text{sort}(\mathbf{u}_t^{1:N})$ and $\mathcal{W}_t^{\sigma(1:N)}$, and compute $\mathbf{x}_t^n = \Gamma_t(\mathbf{x}_{t-1}^{\sigma(a_{t-1}^n)}, v_t^n)$. (e.g. $\Gamma_t = F_{m_t}^{-1}$)

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

Some remarks

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

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

- The main requirement to implement SQMC is that one may simulate from Markov kernel $m_t(x_{t-1}, dx_t)$ by computing $x_t = \Gamma_t(x_{t-1}, u_t)$, where $u_t \sim \mathcal{U}[0, 1]^d$, 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 $m_t(x_{t-1}, dx_t)$ by computing $x_t = \Gamma_t(x_{t-1}, u_t)$, where $u_t \sim \mathcal{U}[0, 1]^d$, for some deterministic function $\Gamma_t$ (e.g. multivariate inverse CDF).
- The dimension of the point sets $u_t^{1:N}$ is $1 + d$: first component is for selecting the parent particle, the $d$ remaining components is for sampling $x_t^n$ given $x_{t-1}^{a_t^n}$. 
Extensions

- If we use RQMC (randomised QMC) point sets $u_t^{1:N}$, 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 $u^1_N$, 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 $u_{1:t}^{1:N}$, 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 $u_{1:N}^t$, 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

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

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

$$\text{MSE} \left[ Q_t^N(\varphi) \right] = o_P(N^{-1})$$

(under technical conditions, and for certain types of RQMC point sets).

Theory is non-standard and borrows heavily from QMC concepts.
Examples: Kitagawa ($d = 1$)

Well known toy example (Kitagawa, 1998):

\[
\begin{align*}
y_t &= \frac{x_t^2}{a} + \epsilon_t \\
x_t &= b_1 x_{t-1} + b_2 \frac{x_{t-1}}{1+x_{t-1}^2} + b_3 \cos(b_4 t) + \sigma \nu_t
\end{align*}
\]

No parameter 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 \(\mathbb{E}(x_t | y_{0:t})\) at each iteration \(t\). Gain factor is \(\text{MSE(SMC)}/\text{MSE(SQMC)}\).
Examples: Multivariate Stochastic Volatility

Model is

\[
\begin{aligned}
  y_t &= S_t^{\frac{1}{2}} \epsilon_t \\
  x_t &= \mu + \Phi(x_{t-1} - \mu) + \Psi^{\frac{1}{2}} \nu_t
\end{aligned}
\]

with possibly correlated noise terms: \((\epsilon_t, \nu_t) \sim N_{2d}(0, C)\).
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 SMC and SQMC runs).
Examples: Multivariate Stochastic Volatility ($d = 2$)

Filtering.
Examples: Multivariate Stochastic Volatility \( (d = 4) \)

Log-likelihood estimation.
Only requirement to replace SMC with SQMC is that the simulation of $x_t^n|x_{t-1}^n$ may be written as $x_t^n = \Gamma_t(x_{t-1}^n, u_t^n)$ where $u_t^n \sim U[0, 1]^d$.

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 $m_t(x^n_{t-1}, dx_t)$ involves some accept/reject mechanism (e.g. Metropolis). In this way, we could develop SQMC counterparts of SMC samplers (Del Moral et al, 2006).
- SQMC² (C. et al, 2013)?
Further work

- Adaptive resampling (triggers resampling steps when weight degeneracy is too high).
- Adapt SQMC to situations where sampling from $m_t(x_{t-1}^n, dx_t)$ involves some accept/reject mechanism (e.g. Metropolis). In this way, we could develop SQMC counterparts of SMC samplers (Del Moral et al, 2006).
- SQMC$^2$ (C. et al, 2013)?

Soon on arxiv!