Sequential Quasi Monte Carlo

N. Chopin (CREST-ENSAE)

nicolas.chopin@ensae.fr

joint work with Mathieu Gerber (Harvard)
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 $O(N^{-1+\epsilon})$. 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 $O(N^{-1+\epsilon})$. 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 \( \mathcal{U} \left([0,1]^d\right) \).

QMC replaces \( u^{1:N} \) 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 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:

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

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

There are various ways to construct point sets \( P_N = \{u^{1:N}\} \) so that \( D^*(u^{1:N}) = O(N^{-1+\epsilon}) \).
As a simple example of a low-discrepancy sequence in dimension one, $d = 1$, consider

\[
\begin{array}{ccccccc}
 1 & 1 & 3 & 1 & 3 & 5 & 7 \\
 2' & 4' & 4' & 8' & 8' & 8' & 8' \\
\end{array}
\]

or more generally,

\[
\frac{1}{p}, \ldots, \frac{p - 1}{p}, \frac{1}{p^2}, \ldots.
\]
As a simple example of a low-discrepancy sequence in dimension one, $d = 1$, consider

\[
\begin{align*}
1 & , 1 & 3 & 1 & 3 & 5 & 7 \\
2 & ' & 4 & ' & 4 & ' & 8 & ' & 8 & ' & 8 & ' & 8 & \cdots
\end{align*}
\]

or more generally,

\[
\begin{align*}
1 & , \ldots, \frac{p-1}{p} & , \frac{1}{p^2} & , \ldots
\end{align*}
\]

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 $u^n \sim \mathcal{U}([0, 1]^d)$ marginally. In this way

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

and one may evaluate the MSE through independent runs.
RQMC (randomised QMC)

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

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

and one may evaluate the MSE through independent runs.

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

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

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

and one may evaluate the MSE through independent runs.

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

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

$$\text{Var} \left\{ \frac{1}{N} \sum_{n=1}^{N} \varphi(u^n) \right\} = \mathcal{O}(N^{-3+\varepsilon})$$
To use QMC:
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);
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.
A Feynman-Kac model is made of:

- A Markov chain in $\mathcal{X}$: initial law is $m_0(dx)$, Markov kernel at iteration $t$ is $m_t(x_{t-1}, dx_t)$
- 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 \( m_0(dx) \), Markov kernel at iteration \( t \) is \( m_t(x_{t-1}, dx_t) \)
- 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

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

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

⇒ change of measure.
Take for instance

\[ G_t(x_{t-1}, x_t) = 1_{A_t}(x_t) \]

then \( Z_t \) is the probability that the \( x_t \in A_t \) for all \( t \), and so on.
Imagine a model for a Markov chain \((x_t)\) that is not observed directly, but through

\[
y_t = h(x_t) + \text{noise}
\]

and let \(g(y_t|x_t)\) be the density of \(y_t\) conditional on \(x_t\). Then, taking

\[
G_t(x_{t-1}, x_t) = g(y_t|x_t)
\]

turns \(Q_t\) into the filtering distribution of Markov chain \((x_t)\), conditional on data \(y_{0:t}\).
Imagine a model for a Markov chain \((x_t)\) that is not observed directly, but through

\[ y_t = h(x_t) + \text{noise} \]

and let \(g(y_t|x_t)\) be the density of \(y_t\) conditional on \(x_t\). Then, taking

\[ G_t(x_{t-1}, x_t) = g(y_t|x_t) \]

turns \(Q_t\) into the **filtering** distribution of Markov chain \((x_t)\), conditional on data \(y_{0:t}\).

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 $x_t^n$ of Markov chain $(x_t)$
2. reweight according to function $G_t$

Problem: variance of cumulative weights:

$$w(x_{0:t}^n) = \prod_{s=0}^{t} G_s(x_{s-1}^n, x_s^n)$$

increases over time (at exponential rate).
Particle Filtering: Basic idea

At time 0, use importance sampling, to go from $m_0(dx_0)$ to $Q_0(dx_0) \propto m_0(dx_0)G_0(x_0)$. We thus obtain the following approximation of $Q_0$:

$$Q_0^N(dx_0) = \frac{1}{\sum_{n=1}^{N} G_0(x_0^n)} \sum_{n=1}^{N} G_0(x_0^n) \delta_{x_0^n}(x_0)$$

To progress to time 1:

1. Choose one ‘ancestor’ $x_0^n$ with probability $\propto G_0(x_0^n)$; call $A_0^n$ the index of the selected ancestor.
2. Simulate $x_1^n \sim m_1(x_0^{A_0^n}, dx_1)$
3. Reweight, with weight $G_1(x_0^{A_0^n}, x_1^n)$
Operations must be be performed for all \( n \in 1 : N \).

At time 0,

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

(b) Compute

\[
W_n^0 = \frac{G_0(x_0^n)}{\sum_{m=1}^N G_0(x_0^m)}
\]

and

\[
Z_N^0 = N^{-1} \sum_{n=1}^N G_0(x_0^n).
\]

Recursively, for time \( t = 1 : T \),

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

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

(c) Compute

\[
W_t^n = \frac{G_t(x_{t-1}^{a_{t-1}^n}, x_t^n)}{\sum_{m=1}^N G_t(x_{t-1}^{a_{t-1}^m}, x_t^m)}
\]

and

\[
Z_N^t = Z_N^{t-1} \left\{ N^{-1} \sum_{n=1}^N G_t(x_{t-1}^{a_{t-1}^n}, x_t^n) \right\}.
\]
Cartoon representation

Source for image: some dark corner of the Internet.
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).
We can formalise the succession of Steps (a), (b) and (c) at iteration $t$ as an importance sampling step from random probability measure

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

(1)

to

$$\{\text{same thing}\} \times G_{t}(\tilde{x}_{t-1}, x_{t}).$$
We can formalise the succession of Steps (a), (b) and (c) at iteration $t$ as an importance sampling step from random probability measure

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

(1)

to

$$\{\text{same thing}\} \times G_{t}(\tilde{x}_{t-1}, x_{t}).$$

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.
More precisely, we are going to write the simulation from

\[ \sum_{n=1}^{N} W_{t-1}^n \delta x_{t-1}^n (d \tilde{x}_{t-1}) m_t (\tilde{x}_{t-1}, dx_t) \]

as a function of \( u_t^n = (u_t^n, v_t^n) \), \( u_t^n \in [0, 1] \), \( v_t^n \in [0, 1]^d \), such that:

1. We will use the scalar \( u_t^n \) to choose the ancestor \( \tilde{x}_{t-1} \).
2. We will use \( v_t^n \) to generate \( x_t^n \) as

\[ x_t^n = \Gamma_t (\tilde{x}_{t-1}, v_t^n) \]

where \( \Gamma_t \) is a deterministic function such that, for \( v_t^n \sim \mathcal{U} [0, 1]^d \), \( \Gamma_t (\tilde{x}_{t-1}, v_t^n) \sim m_t (\tilde{x}_{t-1}, dx_t) \).
Intermediate step

More precisely, we are going to write the simulation from

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

as a function of $u^n_t = (u^n_t, v^n_t)$, $u^n_t \in [0, 1]$, $v^n_t \in [0, 1]^d$, such that:

1. We will use the scalar $u^n_t$ to choose the ancestor $\tilde{x}_{t-1}$.
2. We will use $v^n_t$ to generate $x^n_t$ as

$$x^n_t = \Gamma_t(\tilde{x}_{t-1}, v^n_t)$$

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

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

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

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

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

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 $H : [0, 1] \to [0, 1]^d$. 
From $d = 1$ to $d > 1$

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

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

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 $[0, 1]$, then 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 \( u^{1:N}_0 \) in \([0, 1]^d\), and compute \( x^n_0 = \Gamma_0(u^n_0) \). (e.g. \( \Gamma_0 = F_{m_0}^{-1} \))

(b) Compute \( W^n_0 = G_0(x^n_0) / \sum_{m=1}^{N} G_0(x^m_0) \).

Recursively, for time \( t = 1 : T \),

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

(b) Hilbert sort: find permutation \( \sigma \) such that

\[
    h \circ \psi(x^{\sigma(1)}_{t-1}) \leq \ldots \leq h \circ \psi(x^{\sigma(N)}_{t-1}).
\]

(c) Generate \( a^{1:N}_{t-1} \) using inverse CDF Algorithm, with inputs \( \text{sort}(u^{1:N}_t) \) and \( W^{\sigma(1:N)}_{t-1} \), and compute

\[
    x^n_t = \Gamma_t(x^{\sigma(a^n_{t-1})}_{t-1}, v^{\sigma(n)}_t). \quad \text{(e.g. } \Gamma_t = F_{m_t}^{-1})
\]

(e) Compute

\[
    W^n_t = G_t(x^{\sigma(a^n_{t-1})}_{t-1}, x^n_t) / \sum_{m=1}^{N} G_t(x^{\sigma(a^m_{t-1})}_{t-1}, x^m_t).
\]
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 $O(N \log N)$. (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 $u_t \sim U[0, 1]^d$, 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 $x_{t-1}^{a_{t-1}^n}$. 
• If we use RQMC (randomised QMC) point sets $u_t^{1:N}$, then SQMC generates an unbiased estimate of the marginal likelihood $Z_t$. 
Extensions

- If we use RQMC (randomised QMC) point sets $u_1^N$, 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.)
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 $u^1_t:u^N_t$, 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.
Main results

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

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

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

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

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

Theory is non-standard and borrows heavily from QMC concepts.
Let $\mathcal{X} = [0, 1]^d$. Consistency results are expressed in terms of the star norm

$$\|Q^N_t - Q_t\|_* = \sup_{[0,b] \subset [0,1)^d} \left| \left( Q^N_t - Q_t \right)(B) \right| \to 0.$$ 

This implies consistency for bounded functions $\varphi$,

$$Q^N_t(\varphi) - Q_t(\varphi) \to 0.$$ 

The Hilbert curve conserves discrepancy:

$$\|\pi^N - \pi\|_* \to 0 \implies \|\pi^N_h - \pi_h\|_* \to 0$$

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):

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

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 $\text{MSE}(\text{SMC})/\text{MSE}(\text{SQMC})$. 
Application: autonomous positioning

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

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

$$x_t = x_{t-1} + T_s \nu_t + T_s \epsilon_t$$

and noise terms $\epsilon_t, \nu_t$ are Laplace-distributed.
Application: simulated data

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

**Figure**: Simulated trajectory (15 min)
**Application: results**

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

Model is

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

with possibly correlated noise terms: \((\epsilon_t, \nu_t) \sim N_{2d}(0, \mathbf{C})\).

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 ($d = 2$)

![Graphs showing log-likelihood evaluation and filtering as a function of $t$.]

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)

\[
\begin{align*}
\text{d}Y_t &= \{\mu P + \beta e^{X_t}\} \text{d}t + e^{X_t}/2 \text{d}B_t \\
\text{d}X_t &= \mu(X_t) \text{d}t + \omega(X_t) \text{d}W_t
\end{align*}
\]

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

\[
\begin{align*}
\mu(x) &= \kappa(\mu - e^x)e^{-x} - 0.5\omega^2 e^{-x} \\
\omega(x) &= \omega e^{-x/2}
\end{align*}
\]
Example: Diffusion driven SV model (e.g. Shephard, 2004)

\[
\begin{align*}
\frac{dY_t}{dt} &= \{\mu_P + \beta e^{X_t}\}dt + e^{X_t/2}dB_t \\
\frac{dX_t}{dt} &= \mu(X_t)dt + \omega(X_t)dW_t
\end{align*}
\]

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

\[
\begin{align*}
\mu(x) &= \kappa(\mu - e^x)e^{-x} - 0.5\omega^2e^{-x} \\
\omega(x) &= \omega e^{-x/2}
\end{align*}
\]

The \(Y_t\) are observed for \(t = 0, 1, \ldots, T\).
For $M \geq 1$ (with $\delta = M^{-1}$),

\[
\begin{align*}
Y_{t+1} \mid Y_t, \tilde{X}_{t+1} &\sim \mathcal{N}_1 \left( Y_t + \mu_P + \beta \tilde{\sigma}_{t+1}^2 + \rho \tilde{Z}_{t+1}, (1 - \rho^2)\tilde{\sigma}_{t+1}^2 \right) \\
\tilde{X}_{t+\delta} &= \tilde{X}_t + \delta \mu(\tilde{X}_t) + \omega(\tilde{X}_t)(W_{t+\delta} - W_t) \\
&\vdots \\
\tilde{X}_{t+1} &= \tilde{X}_{t+1-\delta} + \delta \mu(\tilde{X}_{t+1-\delta}) + \omega(\tilde{X}_{t+1-\delta})(W_{t+1} - W_{t+1-\delta})
\end{align*}
\]

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

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

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

\[
\begin{align*}
Y_{t+1}|Y_t, \tilde{X}_{t+1} &\sim \mathcal{N}_1 \left( Y_t + \mu_P + \beta \tilde{\sigma}^2_{t+1} + \rho \tilde{Z}_{t+1}, (1 - \rho^2) \tilde{\sigma}^2_{t+1} \right) \\
\tilde{X}_{t+\delta} &= \tilde{X}_t + \delta \mu(\tilde{X}_t) + \omega(\tilde{X}_t)(W_{t+\delta} - W_t) \\
\vdots \\
\tilde{X}_{t+1} &= \tilde{X}_{t+1-\delta} + \delta \mu(\tilde{X}_{t+1-\delta}) + \omega(\tilde{X}_{t+1-\delta})(W_{t+1} - W_{t+1-\delta})
\end{align*}
\]

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

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

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.
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 $\tilde{x}^n_{t+1}$ at iteration $t+1$ of SQMC:

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

  $$W^n_{t+m\delta} - W^n_{t+(m-1)\delta} = \sqrt{\delta} \Phi^{-1}(v^n_m), \quad m = 1, \ldots, M.$$
We consider the following two approaches to generate $\tilde{x}_{t+1}^n$ at iteration $t + 1$ of SQMC:

- First approach (forward approach): Set (with $v^n \in [0, 1]^M$)
  \[
  W^n_{t+m\delta} - W^n_{t+(m-1)\delta} = \sqrt{\delta} \Phi^{-1}(v^n_m), \quad m = 1, \ldots, M.
  \]

- Second approach: Use $v^n$ and a dimension reduction approach to simulate the values $\{\tilde{W}_n^{m\delta}\}_{m=1}^M$ of a standard Brownian motion $(\tilde{W}_s^n)_{s \in [0,1]}$, and set
  \[
  W^n_{t+m\delta} - W^n_{t+(m-1)\delta} = \tilde{W}^n_{t+m\delta} - \tilde{W}^n_{t+(m-1)\delta}, \quad m = 1, \ldots, M.
  \]
We consider the following two approaches to generate $\tilde{x}_{t+1}^{n}$ at iteration $t + 1$ of SQMC:

- **First approach (forward approach):** Set (with $v^n \in [0, 1]^M$)
  \[
  W_{t+m\delta}^n - W_{t+(m-1)\delta}^n = \sqrt{\delta}\Phi^{-1}(v_m^n), \quad m = 1, \ldots, M.
  \]

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

For the second approach, we will use the **Brownian Bridge** construction (Caflisch et al., 1997).
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).
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).
• Only requirement to replace SMC with SQMC is that the simulation of $x^n_t|x^n_{t-1}$ may be written as $x^n_t = \Gamma_t(x^n_{t-1}, u^n_t)$ where $u^n_t \sim U[0, 1]^d$.

• We observe very impressive gains in performance (even for small $N$ and not too small $d$).

• Supporting theory.
References

• Gerber, M., and Chopin, N. Convergence of sequential quasi-monte carlo smoothing algorithms. ArXiv preprint 1506.06117 (Bernoulli, forthcoming)
• Forthcoming paper in MCQMC2016 proceedings.