Towards automatic calibration of the number of state particles within the SMC² algorithm

N. Chopin * J. Ridgway ** M. Gerber ***
O. Papaspiliopoulos ****

* CREST-ENSAE, Malakoff, and HEC Paris (e-mail: nicolas.chopin@ensae.fr)
** CREST-ENSAE, Malakoff, and Universite Dauphine France (e-mail: james.ridgway@ensae.fr)
*** Harvard University, USA (e-mail: mathieu.berger@fas.harvard.edu)
**** ICREA & Universitat Pompeu Fabra, Barcelona, Spain (e-mail: omiros.papaspiliopoulos@upf.edu)

Abstract: SMC² (Chopin et al., 2013) is an efficient algorithm for sequential estimation and state inference of state-space models. It generates $N_\theta$ parameter particles $\theta^n$, and, for each $\theta^n$, it runs a particle filter of size $N_x$ (i.e. at each time step, $N_x$ particles are generated in the state space $X$). We discuss how to automatically calibrate $N_x$ in the course of the algorithm. Our approach relies on conditional Sequential Monte Carlo updates, monitoring the state of the pseudo random number generator and on an estimator of the variance of the unbiased estimate of the likelihood that is produced by the particle filters, which is obtained using nonparametric regression techniques. We observe that our approach is both less CPU intensive and with smaller Monte Carlo errors than the initial version of SMC².

Keywords: Bayesian inference, Estimation algorithms, Hidden Markov models, Monte Carlo simulation, Particle filtering, State space models

1. INTRODUCTION

Consider a state-space model, with parameter $\theta \in \Theta$, latent Markov process $(x_t)_{t \geq 0}$, and observed process $(y_t)_{t \geq 0}$, taking values respectively in $X$ and $Y$. The model is defined through the following probability densities: $\theta$ has prior $p(\theta)$, $(x_t)_{t \geq 0}$ has initial law $\mu_0(x_0)$ and Markov transition $f_\theta^X(x_t|x_{t-1})$, and the $y_t$’s are conditionally independent, given the $x_t$’s, with density $f_\theta^Y(y_t|x_t)$. Sequential analysis of such a model amounts to computing recursively (in $t$) the posterior distributions

$$p(\theta, x_{0:t}|y_{0:t}) = \frac{p(\theta) \mu_0(x_0)}{p(y_{0:t})} \prod_{s=1}^t f_\theta^X(x_{s}|x_{s-1}) \prod_{s=0}^t f_\theta^Y(y_s|x_s),$$

or some of its marginals (e.g. $p(\theta|y_{0:t})$): the normalising constant $p(y_{0:t})$ of the above density is the marginal likelihood (evidence) of the data observed up to time $t$.

For a fixed $\theta$, the standard approach to sequential analysis of state-space models is particle filtering: one propagates $N_x$ particles in $X$ over time through mutation steps (based on proposal distribution $q_{t-1,\theta}(x_t|x_{t-1})$ at time $t$) and resampling steps; see Algorithm 1. Note the conventions: 1 : $N_x$ denotes the set of integers $\{1, \ldots, N_x\}$, $y_{0:t}$ is $(y_0, \ldots, y_t)$,

$$x_{1:N_x}^t = (x_{1}^t, \ldots, x_{N_x}^t), \quad x_{0:N_x}^t = (x_0^{1:N_x}, \ldots, x_t^{1:N_x}),$$

and so on.

The output of Algorithm 1 may be used in different ways: at time $t$, the quantity $\sum_{n=1}^{N_x} W^n_t \varphi(x^n_t)$ is a consistent (as $N_x \to +\infty$) estimator of the filtering expectation $\mathbb{E}[\varphi(x_t)|y_{0:t}, \theta]$; In addition, $\hat{\ell}_t(\theta)$ is an unbiased estimator of incremental likelihood $p(y_{0:t-1}|\tilde{\theta})$, and $\prod_{t=0}^T \hat{\ell}_t(\theta)$ is an unbiased estimator of the full likelihood $p(y_{0:T}|\theta)$ (Del Moral, 1996, Lemma 3).

In order to perform joint inference on parameter $\theta$ and state variables, Chopin et al. (2013) derived the SMC² sampler, that is, a SMC (Sequential Monte Carlo) algorithm in $\theta$–space, which generates and propagates $N_\theta$ values $\theta^n$ in $\Theta$, and which, for each $\theta^n$, runs a particle filter (i.e. Algorithm 1) for $\theta = \theta^n$, of size $N_x$. One issue however is how to choose $N_x$: if too big, then CPU time is wasted, while if taken too small, then the performance of the algorithm deteriorates. Chopin et al. (2013) give formal results (adapted from Andrieu et al. (2010)) that suggest that $N_x$ should grow at a linear rate during the course of the algorithm. They also propose a practical method for increasing $N_x$ adaptively, based on an importance sampling step where the $N_\theta$ particle systems, of size $N_x$, are replaced by new particle systems of size $N_{\text{new}}^{x}$. But this importance sampling step increases the degeneracy of the weights, which in return may lead to more frequent resampling steps, which are expensive. In this paper, we derive an...
random-walk Metropolis, with the random step calibrated
K
algorithm,
p
phase goes below some threshold ESS
min
an approximation of partial posterior parameter algorithm (Chopin, 2002) as Algorithm 2. For a model with
2.1 IBIS
based on importance sampling, but rather on a CSMC alternative way to increase
Algorithm 1.
Compute weights (a)
Sample
At time 0:
Sample
(b)
Sample
(c)
Compute weights (a)
Compute weights (b)
and incremental likelihood estimate
Recursively, from time t = 1 to time t = T:
(a) Sample \( \alpha^n \sim \mathcal{N}(W_{t-1}^{1:T}) \), the multinomial distribution which generates value i ∈ 1 : \( N_x \) with probability \( W_{t-1} \).
(b) Sample \( x^n_t \sim q_t(x^n_t|x^n_{t-1}) \).
(c) Compute weights

\[
\begin{align*}
    w_{0,\theta}(x^n_0) &= \frac{\mu_0(x^n_0)f(\alpha^n_0|x^n_0)}{q_0(x^n_0)} \\
    \text{normalised weights, } W^n_{0,\theta} &= w_{0,\theta}(x^n_0)/\sum_{i=1}^{N_x} w_{0,\theta}(x^n_0), \text{ and incremental likelihood estimate} \\
    \hat{\ell}_0(\theta) &= N_x^{-1} \sum_{n=1}^{N_x} w^n_{0,\theta}.
\end{align*}
\]

\[
\begin{align*}
    (b) \text{ Sample } \alpha^n_t \sim \mathcal{N}(W_{t-1}^{1:T}), \text{ the multinomial distribution which generates value i ∈ 1 : N}_x \text{ with probability}\ W_{t-1}.
    (c) \text{ Compute weights} \\
    w_{t,\theta}(x^n_{t-1}, x^n_t) &= \frac{f^X(x^n_t|x^n_{t-1})f(\alpha^n_t|x^n_{t-1})}{q_t(x^n_t|x^n_{t-1})} \\
    W^n_{t,\theta} &= \frac{w_{t,\theta}(x^n_{t-1}, x^n_t)}{\sum_{n=1}^{N_x} w_{t,\theta}(x^n_{t-1}, x^n_t)}
\end{align*}
\]

and incremental likelihood estimate
\[
\hat{\ell}_t(\theta) = N_x^{-1} \sum_{n=1}^{N_x} w_{t,\theta}(x^n_{t-1}, x^n_t).
\]

alternative way to increase \( N_x \) adaptively, which is not based on importance sampling, but rather on a CSMC (conditional Sequential Monte Carlo) update, which is less CPU intensive.

2. BACKGROUND ON SMC²

2.1 IBIS

To explain SMC², we first recall the structure of the IBIS algorithm (Chopin, 2002) as Algorithm 2. For a model with	parameter \( \theta \in \Theta \), prior \( p(\theta) \), data \( y_{0:T} \), and incremental likelihood \( p(y|y_{0:t-1}, \theta) \), IBIS provides at each iteration t
an approximation of partial posterior \( p(\theta|y_{0:t}) \). In practice, IBIS samples \( N_\theta \) particles \( \theta^m \) from the prior, then performs
sequential importance sampling steps, from \( p(\theta|y_{0:t-1}) \) to
(\( p(\theta|y_{0:t}) \)) using incremental weight
\[
\frac{p(\theta|y_{0:t})}{p(\theta|y_{0:t-1})} \propto p(y|y_{0:t-1}, \theta).
\]

To avoid weight degeneracy, one performs a resample-move step (described as Step (b) in Algorithm 2). When the ESS (effective sample size) of the weights, computed as:

\[
\text{ESS}(\omega^{1:N_x}) = \left( \frac{\sum_{m=1}^{N_x} \omega^m \right)^2}{\sum_{m=1}^{N_x} \omega^m^2} \in [1, N]\n\]
goes below some threshold ESSmin (e.g. \( N/2 \)), the \( \theta^m \)’s are resampled, then moved according to some Markov kernel \( K_t \) that leaves invariant the current target of the algorithm, \( p(\theta|y_{0:t}) \). This resample-move step re-introduces diversity among the \( \theta \)-particles.

A convenient default choice for \( K_t \) is several iterations of random-walk Metropolis, with the random step calibrated

Algorithm 1. Particle filter (PF, for fixed \( \theta \))

Operations involving superscript \( n \) must be performed for all \( n \in 1 : N_x \).

At time t:
(a) Sample \( x^n_0 \sim q_0(x_0) \).
(b) Compute weights
\[
\begin{align*}
    w_{0,\theta}(x^n_0) &= \frac{\mu_0(x^n_0)f(\alpha^n_0|x^n_0)}{q_0(x^n_0)} \\
    \text{normalised weights, } W^n_{0,\theta} &= w_{0,\theta}(x^n_0)/\sum_{i=1}^{N_x} w_{0,\theta}(x^n_0), \text{ and incremental likelihood estimate} \\
    \hat{\ell}_0(\theta) &= N_x^{-1} \sum_{n=1}^{N_x} w^n_{0,\theta}.
\end{align*}
\]

Algorithm 2. IBIS

Operations involving superscript \( m \) must be performed for all \( m \in 1 : N_\theta \).

(Init) Sample \( \theta^m \sim p(\theta) \), set \( \omega^m \leftarrow 1 \).

From time \( t = 0 \) to time \( t = T \), do
(a) Update importance weights
\[
\omega^m \leftarrow \omega^m \times p(y|y_{0:t-1}, \theta^m).
\]
(b) If ESS(\( \omega^{1:N_x} \)) ≤ ESSmin, sample (for all \( m \)) \( \tilde{\theta}^m \) from mixture
\[
\frac{1}{\sum_{m=1}^{N_\theta} \omega^m} \sum_{m=1}^{N_\theta} \omega^m K_t(\theta^m, d\theta),
\]
where \( K_t \) is a kernel with invariant distribution \( p(\theta|y_{0:t}) \); finally reset particle system to
\[
\theta^{1:N_\theta} \leftarrow \tilde{\theta}^{1:N_\theta}, \quad \omega^{1:N_\theta} \leftarrow (1, \ldots, 1).
\]

to the spread of the current particle population (i.e. variance of random step equals some fraction of the covariance matrix of the resampled particles).

The main limitation of IBIS is that it requires evaluating the likelihood increment \( p(y|y_{0:t-1}, \theta) \), which is typically
intractable for state-space models. On the other hand, we have seen that this quantity may be estimated unbiasedly
by particle filtering. This suggests combining IBIS (i.e. SMC in the \( \theta \)-dimension) with particle filtering (i.e. SMC in the \( x \)-dimension), as done in the SMC² algorithm.

2.2 SMC²

The general structure of SMC² is recalled as Algorithm 3. Essentially, one recognises the IBIS algorithm, where the
intractable incremental weight \( p(y|y_{0:t-1}, \theta^m) \) has been replaced by the unbiased estimate \( \hat{\ell}_t(\theta^m) \).
This estimate is obtained from a PF run for \( \theta = \theta^m \); thus \( N_\theta \) PFs are run in parallel. Denote \( \tilde{\alpha}^{1:N_\theta,m}, \tilde{a}^{1:N_\theta,m} \) the random variables generated by the PF associated to \( \theta^m \).

This ‘double-layer’ structure suggests that SMC² suffers from two levels of approximation, and as such that it
requires both \( N_x \to +\infty \) and \( N_\theta \to +\infty \) to converge. It turns out however that SMC² is valid for any fixed value of \( N_x \); that is, for any fixed \( N_x \geq 1 \), it converges as \( N_\theta \to +\infty \).

This property is intuitive in the simplified case when resampling-move steps are never triggered (i.e. take ESSmin = 0). Then SMC² collapses to importance sampling, with weights replaced by unbiased estimates, and it is easy to show convergence from first principles.

We now give a brief outline of the formal justification of SMC² for fixed \( N_x \), and refer to Chopin et al. (2013) for
more details. SMC² may be formalised as a SMC sampler for the sequence of extended distributions:

\[
\pi_t(\theta, x^{1:N_x}_{0:t}, a^{1:N_x}_{1:t}) = \frac{p(\theta)}{p(y_{0:t})} \psi_{t,\theta}(x^{1:N_x}_{0:t}, a^{1:N_x}_{1:t}) \prod_{s=0}^{t} \hat{\ell}_s(\theta),
\]

where \( \psi_{t,\theta} \) denotes the joint pdf of the random variables generated by a PF up to time t (for parameter \( \theta \), and \( \hat{\ell}_s(\theta) \) denotes the unbiased estimate of the likelihood incre-
Algorithm 3. SMC

Operations involving superscript \(m\) must be performed for all \(m \in 1:N_\theta\).

(\textbf{Init}) Sample \(\theta^m \sim p(\theta)\), set \(w^m \leftarrow 1\).

(a) For each \(\theta^m\), run iteration \(t\) of Algorithm 1, so as to obtain \((x_{0:t}^m, \pi_{1:N_r}^m, a_{1:t}^m)\), and \(\hat{L}_t(\theta^m)\).

(b) Update weights

\[
\omega^m \leftarrow \omega^m \times \hat{L}_t(\theta^m),
\]

(c) If \(\text{ESS}(\omega^m) \leq \text{ESS}_{\text{min}}\), sample \(\tilde{\theta}^m\) from mixture

\[
\frac{\omega^m}{\sum_{m=1}^{N_\theta} \omega^m} K_1(\left(\theta^m, x_{0:t}^m, a_{1:t}^m, \Omega_\theta\right), \left(\tilde{\theta}^m, \tilde{x}_{0:t}^m, \tilde{a}_{1:t}^m, \Omega_\theta\right), d\theta'),
\]

where \(K_1\) is a PMCMC kernel with invariant distribution \(\pi_t(\theta, x_{0:t}, a_{1:t})\) (see text); finally reset particle system to \(\left(\theta^m, x_{0:t}^m, a_{1:t}^m\right) \leftarrow \left(\tilde{\theta}^m, \tilde{x}_{0:t}^m, \tilde{a}_{1:t}^m\right)\) and \(w^m \leftarrow 1\), for all \(m\).

ment computed from that PF, \(\hat{L}_t(\theta) = N_x^{-1} \sum_{n=1}^{N_x} w_0(x_0^n)\), \(\hat{L}_t(\theta) = N_x^{-1} \sum_{n=1}^{N_x} w_0(x_0^n)\) for \(s > 0\); i.e. \(\hat{L}_t(\theta)\) is actually a function of \((\theta, x_{0:1}, a_{1:t})\).

One recognises in \(\pi_t\) the type of extended target distribution simulated by PMCMC (Particle MCMC, Andrieu et al. (2010)) algorithms. Note \(\pi_t\) is a proper probability density (it integrates to one), and that the marginal distribution of \(\theta\) is \(p(\theta|x_0)\). These two properties are easily deduced from the unbiasedness of \(\prod_{t=0}^{\infty} \hat{L}_t(\theta)\) (as an estimator of \(p(x_0|\theta)\)). In addition,

\[\pi_t(\theta, x_{0:t}, a_{1:t}) = \pi_{t-1}(\theta, x_{0:t-1}, a_{1:t-1}) \frac{p(\theta|x_{0:t}, a_{1:t})}{p(\theta|x_{0:t-1}, a_{1:t-1})} \hat{L}_t(\theta)\]

where one recognises in the second factor the distribution of the variables generated by a PF at time \(t\), conditional on those variables generated up to time \(t - 1\). Thus, the equation above justifies both Step (a) of Algorithm 3, where the particle filters are extended from time \(t - 1\) to \(t\), and Step (b), where the particles \((\theta^m, x_{0:t}^m, a_{1:t}^m)\) are reweighted by \(\hat{L}_t(\theta^m)\).

We describe in the following section PMCMC moves that may be used in Step (c). Before, we note that a naive implementation of SMC\(^2\) has a \(\mathcal{O}(tN_\theta N_\phi)\) memory cost at time \(t\), as one must stores in memory \((\theta^m, x_{0:t}^m, a_{1:t}^m)\) for each \(m \in 1:N_\theta\). This memory cost may be substantial even on a modern computer.

2.3 PMCMC moves

To make more explicit the dependence of the unbiased estimate of the likelihood on the variables generated during the course of PF, define

\[L_t(\theta, x_{0:t}, a_{1:t}) = \prod_{s=0}^{t} \hat{L}_s(\theta)\]

\[= \left\{ \frac{1}{N_x} \sum_{n=1}^{N_x} w_n(x_0^n) \right\} \prod_{s=1}^{t} \left\{ \frac{1}{N_x} \sum_{n=1}^{N_x} w_n(x_{s:n}^{m_s}, x_{s+1:n}^{m_s}) \right\}.
\]

The PMMH (Particle Markov Metropolis-Hastings) kernel, described as Algorithm 4, may be described informally as a Metropolis step in \(\theta\)-space, where the likelihood of both the current value and the proposed value have been replaced by unbiased estimators. Formally, as proven in Andrieu et al. (2010), it is in fact a standard Metropolis step with respect to the extended distribution \(\pi_t(\theta, x_{0:t}, a_{1:t})\); in particular it leaves invariant \(p(\theta|x_0)\). (For convenience, our description of PMMH assumes a random walk proposal, but PMMH is not restricted to this kind of proposal.)

Algorithm 4. Random walk PMMH update

Input: \((\theta, x_{0:t}, a_{1:t})\)

Output: \((\hat{\theta}, \hat{x}_{0:t}, \hat{a}_{1:t})\)

1. \(\theta^* = \theta + z, z \sim N(0, \Sigma_t)\).

2. Generate PF (Algorithm 1) for parameter \(\theta^*\); let \((\hat{\theta}^*, \hat{x}_{0:t}, \hat{a}_{1:t})\) be the output.

3. With probability \(1 \wedge r\),

\[r = \frac{p(\theta^*) L_t(\theta^*, x_{0:t}, a_{1:t})}{p(\theta) L_t(\theta, x_{0:t}, a_{1:t})}\]

let \((\hat{\theta}, \hat{x}_{0:t}, \hat{a}_{1:t}) \leftarrow (\theta^*, \hat{x}_{0:t}, \hat{a}_{1:t});\) otherwise \((\hat{\theta}, \hat{x}_{0:t}, \hat{a}_{1:t}) \leftarrow (\theta, \hat{x}_{0:t}, \hat{a}_{1:t})\).

In practice, we set \(\Sigma_t\), the covariance matrix of the proposal, to a fraction of the covariance matrix of the resampled \(\theta\)-particles.

One advantage of using PMMHM within SMC\(^2\) is that it does not require storing all the variables generated by the \(N_\phi\) PFs: operations at time \(t > 0\) require only having access to, for each \(m\), \((\theta^m, x_{0:t-1}^m, a_{1:t-1}^m)\) and \(L_{t-1}(\theta^m, x_{0:t-1}^m, a_{1:t-1}^m)\), which is computed recursively. Memory cost then reduces to \(\mathcal{O}(N_\phi N_\theta)\).

The Particle Gibbs approach is an alternative PMCMC step, based on the following property of target \(\pi_t\): if one extends \(\pi_t\) with random index \(k\), such that \(k \in 1:N_r\), and \(k \sim \mathcal{M}(W_{r:t}^{1:N_r})\), the normalised weights at the final iteration, then \(\pi_t\) the selected trajectory, together with \(\theta\), follow the posterior distribution \(p(\theta|x_{0:t}, [y_{0:t}])\); and (b) the remaining arguments of \(\pi_t\) follow a CSMC (conditional SMC) distribution, which corresponds to the distribution of the random variables generated by a PF, but conditional on one trajectory fixed to the selected trajectory; see Algorithm 5.

In contrast with PMMH, implementing particle Gibbs steps within SMC\(^2\) requires having access to all the variables \((\theta^m, x_{0:N_r}^m, a_{1:t}^m)\) at time \(t\), which as we have already discussed, might incur too big a memory cost.
Algorithm 5: Particle Gibbs update

Input: \((θ, x_{0:t}, a_{1:t}^{1:N_x})\)

Output: \((\tilde{θ}, \tilde{x}_{0:t}, \tilde{a}_{1:t}^{1:N_x})\)

1. Sample \(b_t \sim \mathcal{M}(W_{t}^{1:N_x})\), with \(W_{t} = w_{t, θ(x_{t-1}^{1:N_x}, x_t^s)})/ \sum_{i=1}^{N_x} w_{t, θ(x_{t-1}^{1:N_x}, x_i)}\). From \(s = t - 1\) to \(s = 0\), set \(b_s \leftarrow \tilde{a}_{s+1}^{s+1}\). Set \(\tilde{x}_s^s \leftarrow x_s^{s+1}, \tilde{a}_s^s = 1\) for all \(s \in 0:T\).

2. Sample \(\tilde{θ}\) from a MCMC step that leaves invariant distribution \(p(θ|x_{0:t}, y_{0:t})\), but with \(x_{0:t}\) set to \(x_{0:t}\).

3. Sample \((\tilde{x}_{0:t}, \tilde{a}_{1:t}^{1:N_x})\) as in Algorithm 1, but for parameter \(\tilde{θ}\) and conditionally on \(x_{0:t}\), that is: at time 0, generate \(\tilde{x}_0^0 \sim q_{0, \tilde{θ}}\) for \(n \in 2 : N\), at time 1, sample \(a_0^n \sim \mathcal{M}(W_{1}^{1:N_x})\), for \(n \in 2 : N\), and \(x_0^n \sim q_{1, \tilde{θ}}(\tilde{x}_0^{n-1})\), and so on.

2.4 Choosing \(N_x\)

Andrieu et al. (2010) show that, in order to obtain reasonable performance for PMMH, one should take \(N_x = O(t)\). Andrieu et al. (2013) show a similar result for Particle Gibbs.

In the context of SMC\(^2\), this suggests that \(N_x\) should be allowed to increase in the course of the algorithm. To that effect, Chopin et al. (2013) devised an exchange step, which consists in exchanging the current particle systems, of size \(N_x\), with new particle systems, of size \(N_x^{\text{new}}\), through importance sampling. In Chopin et al. (2013)’s implementation, the exchange step is triggered each time the acceptance rate of the PMMH step (as performed in Step 3. of Algorithm 4) is below a certain threshold, and \(N_x^{\text{new}} = 2N_x\) (i.e. \(N_x\) doubles every time).

The main drawback of this approach is that it introduces some weight degeneracy immediately after the resampling step. In particular, we will observe in our simulations that this prevents us from changing \(N_x\) too frequently, as the ESS of the weights then becomes too low.

In this paper, we discuss how to use a Particle Gibbs step in order to increase \(N_x\) without changing the weights.

3. PROPOSED APPROACH

3.1 Particle Gibbs and memory cost

We first remark that the Particle Gibbs step, Algorithm 5, offers a very simple way to change \(N_x\) during the course of the algorithm: In Step (2), simply re-generate a particle system (conditional on selected trajectory \(x_{0:t}\)) of size \(N_x^{\text{new}}\). But, as already discussed, such a strategy requires then to access past particle values \(x^n_s\) (and also \(a^n_s\)), rather than only current particle values \(x^n_t\).

This problem may be addressed in two ways. First, one may remark that, to implement Particle Gibbs, one needs to store only those \(x^n_s\) (and \(a^n_s\)) which have descendant among the \(N_x\) current particles \(x^n_t\). Jacob et al. (2013) developed such a path storage approach, and gave conditions on the mixing of Markov chain \((x_t)\) under which this approach has memory cost \(O(t + N_x \log N_x)\) for a single PF with \(N_x\) particles, run until time \(t\). Thus, an implementation of this approach within SMC\(^2\) would lead to a \(O(N_p (t + N_x \log N_x))\) memory cost.

A second approach, developed here, exploits the deterministic nature of PRNGs (pseudo-random number generators): a sequence \(z_0, z_1, \ldots, z_t\) of computer-generated random variates is actually a deterministic sequence determined by the initial state (seed) of the PRNG. It is sufficient to store that initial state and \(z_0\) in order to recover any \(z_i\) in the future. The trade-off is an increase in CPU cost, as each access to \(z_i\) require re-computing \(z_1, \ldots, z_i\).

We apply this idea to the variables \((x_{0:t}^{1:N_x}, a_{1:t}^{1:N_x})\). By close inspection of Algorithm 3, we note that variables in a ‘time slice’ \((x_{0:t}^{1:N_x}, a_{1:t}^{1:N_x})\), \(0 < s \leq t\) (or \(x_{0:t}^{1:N_x}\) at time \(0\)) are always generated jointly, either during Step (a), or during Step (c). In both cases, this time-slice is a deterministic function of the current PRNG state and the previous time slice. Thus, one may recover any time slice (when needed) by storing only (i) the PRNG state (immediately before the generation of the time slice); and (ii) in which Step (either (a) or (c)) the time slice was generated. This reduces the memory cost of SMC\(^2\) from \(O(tN_0N_x)\) to \(O(N_0(t + N_x))\).

Compared to the path storage approach mentioned above, our PRNG recycling approach has a larger CPU cost, a smaller memory cost, and does not require any conditions on the mixing properties of process \((x_t)\). Note that the CPU cost increase is within a factor of two, because each time a Particle Gibbs update is performed, the number of random variables that must be re-generated (i.e. \(x^n_s\) and \(a^n_s\) in Algorithm 5) roughly equals the number of random variables that are generated for the first time (i.e. the \(\tilde{x}_0^n\) and \(\tilde{a}_0^n\) in Algorithm 5).

3.2 Nonparametric estimation of \(N_x\)

As seen in Algorithm 3, a Particle Gibbs step will be performed each time the ESS goes below some threshold. That the ESS is low may indicate that \(N_x\) is also too low, and therefore that the variance of the likelihood estimates \(L_t(\theta^m, x_{0:t}^{1:N_x}, a_{1:t}^{1:N_x})\) is too high. Our strategy is to update (each time a Particle Gibbs step is performed) the current value of \(N_x\) to \(N_x^{\text{new}} = \tau/\sigma^2\), where \(\sigma^2\) is some (possibly rough) estimate of the variance of the \(\log\) likelihood estimates. This is motivated by results from Doucet et al. (2012), who also develop some theory that supports choosing \(\tau \approx 1\) is optimal (although their optimality results do not extend straightforwardly to our settings).

Assume \(Θ \subset \mathbb{R}^d\). To estimate \(σ^2\), we use backfitting to fit a GAM (generalized additive model) to the responses \(R^m = \log L_t(θ^m, x_{0:t}^{1:N_x}, a_{1:t}^{1:N_x})\):

\[
R^m = α + \sum_{j=1}^{d} f_j(C^m_j) + ε^m,
\]

using as covariates \(C^m_j\) the \(d\) principal components of the resampled \(θ\)-particles. The estimate \(σ^2\) is then the empirical variance of the residuals. See e.g. Chap. 9 of
We found this strategy to work well, with the caveat that choosing \( \tau \) required some trial and error.

### 3.3 Additional considerations

Using Particle Gibbs as our PMCMC move within SMC\(^2\) hast two advantages: (a) it makes it possible to change \( N_x \) without changing the weights, as explained above; and (b) it also makes it possible to update the \( \theta^m \) according to Gibbs or Metropolis step that leaves \( \theta|x_{0:t},y_{0:t} \) invariant; see Step (3) of Algorithm 5. For models where sampling from \( \theta|x_{0:t},y_{0:t} \) is not convenient, one may instead update \( \theta \) through several PMMH steps performed after the Particle Gibbs step.

### 4. NUMERICAL EXAMPLE

We consider the following stochastic volatility model: \( x_0 \sim N(\mu,\sigma^2/(1-\rho^2)) \), \( x_t - \mu = \rho(x_{t-1} - \mu) + \sigma \epsilon_t \), \( \epsilon_t \sim N(0,1) \) and \( y_t|^x_t \sim N(0,e^{x_t}) \); thus \( \theta = (\mu,\rho,\sigma) \), with \( \rho \in [-1,1] \), \( \sigma > 0 \). We assign independent priors to the components of \( \theta \); \( \mu \sim N(0,2^2) \), \( \rho \sim N(0,1) \) constrained to \([-1,1]\), and \( \sigma^2 \sim IG(3,0.5) \). The dataset consists in log-returns from the monthly SP500 index, observed from 29/05/2013 to 19/12/2014; \( T = 401 \).

Figure 1 plots the marginal posterior \( p(\rho,\sigma^2|y_{0:15}) \), as approximated by SMC\(^2\), run up to time 15. This figure illustrates the need for modelling nonparametrically the true likelihood as a function of \( \theta \), in order to estimate the variance of the estimated likelihood.

For this model, sampling jointly from \( \theta|x_{0:t},y_{0:t} \) is difficult, but it is easy to perform a Gibbs step that leaves invariant \( \theta|x_{0:t},y_{0:t} \), as the full conditionals of each component (e.g. \( \mu|\sigma,\rho,x_{0:t},y_{0:t} \) and so on) are standard distributions. Let’s call ‘full PG’ Algorithm 5, where Step 2 consists of this Gibbs step for \( \theta|x_{0:t},y_{0:t} \); and conversely let’s call ‘partial PG’ Algorithm 5 with \( \hat{\theta} = \theta \) in Step 2 (\( \theta \) is not updated).

We compare four versions of SMC\(^2\): (a) the standard version, as proposed in Chopin et al. (2013) (i.e. Step (c) of Algorithm 3 is a PMMH step, and that step is followed by an exchange step to double \( N_x \) when the acceptance rate of PMMH is below 20%); (b) the same algorithm, except that an exchange step is systematically performed after Step (c), and \( N_x \) is set to the value obtained with our non-parametric approach (see Section 3.2); (c) the version developed in this paper, with full PG steps (and \( N_x \) updated through the non-parametric procedure); (d) the same algorithm, but with partial PG steps, followed by 3 PMMH steps to update \( \theta \).

The point of Algorithm (b) is to show that adapting \( N_x \) too often during the course of the algorithm is not desirable when using the exchange step, as this leads to too much variance. The point of Algorithm (d) is to see how our approach performs when sampling from \( \theta|x_{0:t},y_{0:t} \) (either independently or through MCMC) is not feasible.

Figure 2 plots the evolution of \( N_x \) over time for the four SMC\(^2\) algorithms. One sees that, for these model and dataset, the CPU cost of the standard SMC\(^2\) algorithm is quite volatile, as \( N_x \) increases very quickly in certain runs. In fact certain runs are incomplete, as they were stopped when the CPU time exceeded 10 hours. On the other hand, the CPU cost of other versions is more stable across runs, and, more importantly, quite lower.

Figure 3 plots the empirical variance of the estimated marginal likelihood (evidence, \( p(y_{0:t}) \), normalised with...
the running time up to time step $t$. One observes that version (c) does quite better than (d), and far much better than (a). Results from Algorithm (b) were to variable to be included.

Figure 4 plots the acceptance rate of PMMH steps for Algorithms (a), (b) and (d). (Recall that Algorithm (c) does not perform PMMH steps). Note the poor performance of Algorithm (b). Figure 5 compares the box-plots of posterior estimates of $\sigma$ at final time $T$, obtained from several runs of Algorithms (c) and (d). Algorithm (c) shows slightly less variability, while being 30% faster on average. One sees that the improvement brought by ability to sample from $\theta|x_0:t, y_0:t$ is modest here for parameter estimation, but recall that in Figure 3, the improvement was more substantial.

ACKNOWLEDGEMENTS

We thank Pierre Jacob for useful comments.

REFERENCES

Andrieu, C., Doucet, A., and Holenstein, R. (2010). Particle Markov chain Monte Carlo methods. J. R. Statist. Soc. B, 72(3), 269–342. doi:10.1111/j.1467-9868.2009.00736.x.

Andrieu, C., Lee, A., and Vihola, M. (2013). Uniform Ergodicity of the Iterated Conditional SMC and Geometric Ergodicity of Particle Gibbs samplers. ArXiv e-prints.

Chopin, N. (2002). A sequential particle filter for static models. Biometrika, 89, 539–552.

Chopin, N., Jacob, P., and Papaspiliopoulos, O. (2013). SMC$^2$: A sequential Monte Carlo algorithm with particle Markov chain Monte Carlo updates. J. R. Statist. Soc. B, 75(3), 397–426.

Del Moral, P. (1996). Non-linear filtering: interacting particle resolution. Markov processes and related fields, 2(4), 555–581.

Doucet, A., Pitt, M., Deligiannidis, G., and Kohn, R. (2012). Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. ArXiv preprint.

Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J., and Tibshirani, R. (2009). The elements of statistical learning, volume 2. Springer.

Jacob, P., Murray, L., and Rubenthaler, S. (2013). Path storage in the particle filter. Statist. Comput., 1–10. doi:10.1007/s11222-013-9445-x. URL http://dx.doi.org/10.1007/s11222-013-9445-x.