Introduction to Sequential Monte Carlo Methods

Smoothing using SMC

Auxiliary Particle Filter

Prof. Nicholas Zabaras
Materials Process Design and Control Laboratory
Sibley School of Mechanical and Aerospace Engineering
101 Frank H. T. Rhodes Hall
Cornell University
Ithaca, NY 14853-3801

Email: nzabaras@gmail.com
URL: http://mpdc.mae.cornell.edu/

May 9, 2013
Contents

- References and SMC Resources
- Bayesian Recursion for the State-Space Model, Convergence and Error Estimates
- Smoothing with SMC
- Forward-Backward Pass
- Auxiliary Particle Filter
References and SMC Resources

References and SMC Resources


- A Doucet et al., *Efficient Block Sampling Strategies for SMC*, JCGS, 2006

- C. Andrieu and A. Doucet, *Particle Filtering for Partially Observed Gaussian State-Space Models*, JRSS B, 2002

- R Chen and J Liu, *Mixture Kalman Filters*, JRSSB, 2000


References and SMC Resources


- C. Andrieu, A.D. & R. Holenstein, *Particle Markov chain Monte Carlo methods (with discussion)*, JRSS B, 2010

- A Doucet, *Sequential Monte Carlo Methods and Particle Filters*, List of Papers, Codes, and Viedo lectures on SMC and particle filters

- Pierre Del Moral, *Feynman-Kac models and interacting particle systems*
References and SMC Resources

- J. Carpenter, P. Clifford and P. Fearnhead, *An Improved Particle Filter for Non-linear Problems*


- R. Chen and J.S. Liu, *Predictive Updating Methods with Application to Bayesian Classification*, JRSS B, 1996


- A. Doucet, Short Courses Lecture Notes (*A*, *B*, *C*)

- P. Del Moral, *Feynman-Kac Formulae*, Springer-Verlag, 2004
Video Lectures

- Sequential MC Methods, N. de Freitas and A. Doucet, 2010
- Sequential Monte Carlo Methods, Arnuad Doucet, 2007
- Sequential MC Methods, M. Davy, 2007
- Particle Filters, S. Godsill, 2009
Let us summarize our state space model where the objective is to compute \( p(x_{1:n} \mid y_{1:n}) \)

We can write the following recursion equation:

\[
p(x_{1:n} \mid y_{1:n}) = \frac{p(x_{1:n}, y_{1:n})}{p(y_{1:n})} = \frac{g(y_n \mid x_n) f(x_n \mid x_{n-1}) p(x_{1:n-1}, y_{1:n-1})}{p(y_n \mid y_{1:n-1}) p(y_{1:n-1})} = \frac{g(y_n \mid x_n) f(x_n \mid x_{n-1}) p(x_{1:n-1} \mid y_{1:n-1})}{p(y_n \mid y_{1:n-1})}
\]

where

\[
p(y_n \mid y_{1:n-1}) = \int p(y_n, x_n \mid y_{1:n-1}) dx_n = \int g(y_n \mid x_n) p(x_n \mid y_{1:n-1}) dx_n
\]

\[
= \int g(y_n \mid x_n) f(x_n \mid x_{n-1}) p(x_{n-1} \mid y_{1:n-1}) dx_{n-1:n}
\]

We can write our update equation above in two steps:

*Step I - Prediction:* \( p(x_{1:n} \mid y_{1:n-1}) = f(x_n \mid x_{n-1}) p(x_{1:n-1} \mid y_{1:n-1}) \)

*Step II - Update:* \( p(x_{1:n} \mid y_{1:n}) = \frac{g(y_n \mid x_n) p(x_{1:n} \mid y_{1:n-1})}{p(y_n \mid y_{1:n-1})} \propto g(y_n \mid x_n) p(x_{1:n} \mid y_{1:n-1}) \)
Monte Carlo Implementation of the Prediction

- Assume that at step n-1 we have the following Monte Carlo approximation:

$$\hat{p}(x_{1:n-1} \mid y_{1:n-1}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{\tilde{X}_{1:n-1}^{(i)}}(x_{1:n-1})$$

- Considering

$$p(x_{1:n} \mid y_{1:n-1}) = f(x_{n} \mid x_{n-1}) p(x_{1:n-1} \mid y_{1:n-1}),$$

by sampling

$$\tilde{X}_{n}^{(i)} \sim f(x_{n}, \tilde{X}_{n-1}^{(i)})$$

and setting:

$$\tilde{X}_{1:n}^{(i)} = \left(\tilde{X}_{1:n-1}^{(i)}, \tilde{X}_{n}^{(i)}\right)$$

we predict:

$$\hat{p}(x_{1:n} \mid y_{1:n-1}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{\tilde{X}_{1:n}^{(i)}}(x_{1:n})$$
Importance Sampling Implementation of Update Step

- Our update equation for our target distribution is:

\[
p(x_{1:n} | y_{1:n}) = \frac{g(y_n | x_n) p(x_{1:n} | y_{1:n-1})}{p(y_n | y_{1:n-1})}
\]

- Substituting our Monte Carlo approximation

\[
\hat{p}(x_{1:n} | y_{1:n-1}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{\tilde{x}_{1n}^{(i)}}(x_{1:n})
\]

we obtain:

\[
\hat{p}(y_n | y_{1:n-1}) = \int g(y_n | x_n) \hat{p}(x_{1:n} | y_{1:n-1}) dx_{1:n} = \int g(y_n | x_n) \frac{1}{N} \sum_{i=1}^{N} \delta_{\tilde{x}_{1n}^{(i)}}(x_{1:n}) dx_{1:n} \Rightarrow
\]

\[
\hat{p}(y_n | y_{1:n-1}) = \frac{1}{N} \sum_{i=1}^{N} g(y_n | \tilde{X}_{1n}^{(i)})
\]

- Finally, we can approximate our target distribution as:

\[
\hat{p}(x_{1:n} | y_{1:n}) = \frac{g(y_n | x_n) \hat{p}(x_{1:n} | y_{1:n-1})}{\hat{p}(y_n | y_{1:n-1})}
\]

\[
= \frac{g(y_n | x_n) \frac{1}{N} \sum_{i=1}^{N} \delta_{\tilde{x}_{1n}^{(i)}}(x_{1:n})}{\hat{p}(y_n | y_{1:n-1})}
\]

\[
= \sum_{i=1}^{N} \frac{g(y_n | \tilde{X}_{1n}^{(i)})}{\sum_{j=1}^{N} g(y_n | \tilde{X}_{1n}^{(j)})} \delta_{\tilde{x}_{1n}^{(i)}}(x_{1:n})
\]
Thus our update equation is:

$$\hat{p}(x_{1:n} \mid y_{1:n}) = \sum_{i=1}^{N} W_n^{(i)} \delta_{\tilde{X}_{1:n}^{(i)}}(x_{1:n})$$

$$W_n^{(i)} \propto g(y_n \mid \tilde{X}_n^{(i)}), \text{ with } \sum_{i=1}^{N} W_n^{(i)} = 1$$

To obtain $N$ samples $X_{1:n}^{(i)}$ approximately distributed according to $p(x_{1:n} \mid y_{1:n})$, resample $N$ times with replacement:

$$X_{1:n}^{(i)} \sim \hat{p}(x_{1:n} \mid y_{1:n})$$

to obtain (at a cost $O(N))$:

$$\hat{p}(x_{1:n} \mid y_{1:n}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X_{1:n}^{(i)}}(x_{1:n}) = \sum_{i=1}^{N} \frac{N_n^{(i)}}{N} \delta_{\tilde{X}_{1:n}^{(i)}}(x_{1:n})$$

$$\{N_n^{(i)}\} \sim \text{Multinomial} \{W_n^{(i)}\},$$

$$E\left[ N_n^{(i)} \right] = NW_n^{(i)}, \text{ Var}\left[ N_n^{(i)} \right] = NW_n^{(i)} \left(1 - W_n^{(i)}\right)$$
### The SMC Algorithm

**At step n=1:**
- Sample $\widetilde{X}^{(i)}_1 \sim \mu(x_1)$ and then approximate:

$$\hat{p}(x_1 | y_1) = \sum_{i=1}^{N} W^{(i)}_1 \delta_{\widetilde{X}^{(i)}_1}(x_1), \quad W^{(i)}_1 \propto g(y_1, \widetilde{X}^{(i)}_1)$$

- Resample $X^{(i)}_1 \sim \tilde{p}(x_1 | y_1)$ to obtain

$$\hat{p}(x_1 | y_1) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X^{(i)}_1}(x_1)$$

**At step n≥2:**
- Sample $\widetilde{X}^{(i)}_1 \sim f(x_n | X^{(i)}_{n-1})$, set $\widetilde{X}^{(i)}_{n-1} \sim (X^{(i)}_{n-1}, \widetilde{X}^{(i)}_n)$ and then approximate:

$$\hat{p}(x_{1:n} | y_{1:n}) = \sum_{i=1}^{N} W^{(i)}_n \delta_{\widetilde{X}^{(i)}_{1:n}}(x_{1:n}), \quad W^{(i)}_n \propto g(y_n | \widetilde{X}^{(i)}_n), \text{ with } \sum_{i=1}^{N} W^{(i)}_n = 1$$

- Resample $X^{(i)}_{1:n} \sim \hat{p}(x_{1:n} | y_{1:n})$ to finally obtain:

$$\hat{p}(x_{1:n} | y_{1:n}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X^{(i)}_{1:n}}(x_{1:n})$$

- N Gordon, D J Salmond, AFM Smith, [Novel Approach to nonlinear non Gaussian Bayesian state estimation](https://dx.doi.org/10.1049/IEE-2193), IEE, 1993

---

*Bayesian Scientific Computing, Spring 2013 (N. Zabaras)*
At step \( n \) we thus have:

\[
\hat{p}(x_{1:n} \mid y_{1:n}) = \sum_{i=1}^{N} W_n^{(i)} \delta_{\tilde{X}_{1:n}^{(i)}}(x_{1:n}) \, , \quad W_n^{(i)} \propto g(y_n \mid \tilde{X}_n^{(i)}) \, , \quad \text{with} \quad \sum_{i=1}^{N} W_n^{(i)} = 1
\]

Using these approximations, we can now compute the following marginal likelihood:

\[
\hat{p}(x_{1:n} \mid y_{1:n}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{\tilde{X}_{1:n}^{(i)}}(x_{1:n})
\]

The computational cost of SMC is \( O(N) \) at each step and memory requirements \( O(nN) \). The cost remains \( O(N) \) if our interest is on computing \( p(x_n \mid y_{1:n}) \) or \( p(s_n(x_{1:n}) \mid y_{1:n}) \) where \( s_n(x_{1:n}) = \Phi_n(x_n, s_{n-1}(x_{1:n-1})) \) where \( \Phi \) is a function of the state.
Improving the Sampling Step

- The SMC algorithm discussed earlier is very inefficient. This is particularly true for vague prior \( f(x_n | x_{n-1}) \) or picky likelihood \( g(y_n | x_n) \)

\[
f(x_n | x_{n-1}) = \mathcal{N}(x_n; x_{n-1}, \sigma_v^2)
\]

- Consider the following:

\[
g(y_n | x_n) = \mathcal{N}(y_n; x_n, \sigma_w^2)
\]

\[
p(x_{n-1} | y_{1:n-1}) = \mathcal{N}(x_{n-1}; m, \sigma^2)
\]

- Optimal Proposal/Perfect adaptation:

\[
\text{Sample}: \ p(x_n | y_n, x_{n-1}) \propto g(y_n | x_n) f(x_n | x_{n-1}), \ Resample: W_n \propto p(y_n | x_{n-1})
\]

\[
p(x_n | y_{1:n-1}) = \int p(x_n | y_n, x_{n-1}) p(x_{n-1} | y_{1:n-1}) dx_{n-1}
\]

\[
\int f(x_n | x_{n-1}) p(x_{n-1} | y_{1:n-1}) dx_{n-1}
\]
Approximate Optimal Proposal

- Design analytical approximation using the EKF, UKF

\[ \hat{p}(x_n \mid y_n, x_{n-1}) \text{ of } p(x_n \mid y_n, x_{n-1}) \]

- We then sample \( \hat{p}(x_n \mid y_n, x_{n-1}) \)

and set the weights as:

\[ W_n \propto \frac{g(y_n \mid x_n) f(x_n \mid x_{n-1})}{\hat{p}(x_n \mid y_n, x_{n-1})} \]

M.K. Pitt and N. Shephard, Filtering via Simulation: Auxiliary Particle Filter, JASA, 1999
Resample Move: Adding Diversity

- After the resampling step, you have

\[ X_{1:n}^{(i)} = X_{1:n}^{(j)} \text{ for } i \neq j \]

- To add diversity among particles, use an MCMC kernel

\[ X_{1:n}^{(i)} \sim q_n \left( x_{1:n} \mid X_{1:n}^{(i)} \right) \]

where:

\[ p \left( x_{1:n}^{'} \mid y_{1:n} \right) = \int p \left( x_{1:n} \mid y_{1:n} \right) q_n \left( x_{1:n}^{'} \mid x_{1:n} \right) dx_{1:n} \]

- Note that the MCMC kernel does not need to be ergodic.

W Gilks and C. Berzuini, Following a moving target: MC inference for dynamic Bayesian Models, JRSS B, 2001
Degeneracy Problem

- For any $n$ and $k$, there exist $n(k,N)$ s.t. for any $n \geq n(k,N)$

$$\hat{p}(x_{1:k} \mid y_{1:k}) = \delta_{x^*_{1:k}}(x_{1:k})$$

i.e. we obtain one single delta Dirac function.

- Thus $\hat{p}(x_{1:n} \mid y_{1:n})$ is an unreliable representation of

$$p(x_{1:n} \mid y_{1:n}), \ n \to \infty, \ \text{thus SMC does not approximate well joint distributions as } n \to \infty.$$
Degeneracy Problem

- The Figure below shows:

\[
\frac{1}{n} \mathbb{E} \left[ \sum_{k=1}^{N} X_k \mid y_{1:n} \right]
\]

Computed with Kalman filter (blue) versus SMC (red) for N=1000. As n increases, the SMC estimate obviously deteriorates.

From SMC lecture, A. Doucet.
Consider any function $\phi_n : \mathcal{X}^n \rightarrow \mathbb{R}$ and consider computing its expectation with SMC:

\[
\phi_n = \int \phi_n (x_{1:n}) p (x_{1:n} \mid y_{1:n}) \, dx_{1:n} \quad \text{(exact)}
\]

\[
\hat{\phi}_n = \int \phi_n (x_{1:n}) p (x_{1:n} \mid y_{1:n}) \, dx_{1:n} = \frac{1}{N} \sum_{n=1}^{N} \phi_n (X_{1:n}^{(i)}) \quad \text{(SMC)}
\]

Assuming (e.g. bounded likelihood), one can show that for $p > 0$

\[
\mathbb{E} \left[ \left( \phi_n - \hat{\phi}_n \right)^p \right]^{1/p} \leq \frac{C_n}{\sqrt{N}}
\]

\[
\lim_{N \to \infty} \sqrt{N} \left( \phi_n - \hat{\phi}_n \right) \rightarrow \mathcal{N} \left( 0, \sigma_n^2 \right)
\]

Unfortunately, $C_n$ and $\sigma_n^2$ can increase with $n$ and will for a path-dependent $\phi_n (x_{1:n})$ as the degeneracy problem suggests.
Convergence Results

Assume an exponentially stability condition: For any \( x_1, x_1' \)

\[
\frac{1}{2} \int \left| p(x_{1:n} | y_{1:n}, X_1 = x_1) - p(x_{1:n} | y_{1:n}, X_1 = x_1') \right| dx_n \leq a^n, \forall |a| < 1
\]

Then one can show for any function \( \varphi_n(x_{1:n}) = \varphi(x_n) \)

\[
E \left[ \left| \varphi_n - \hat{\varphi}_n \right|^p \right]^{1/p} \leq \frac{C}{\sqrt{N}}
\]

\[
\lim_{N \to \infty} \sqrt{N} \left( \varphi_n - \hat{\varphi}_n \right) \rightarrow \mathcal{N} \left( 0, \sigma^2_n \right) \text{ with } \sigma^2_n \leq D
\]
Convergence Results

- The following convergence result can be shown for the marginal likelihood:

\[
\lim_{N \to \infty} \sqrt{N} \left( \log \hat{p}(y_{1:n}) - \log p(y_{1:n}) \right) \to \mathcal{N} \left( 0, \sigma_n^2 \right), \quad \sigma_n^2 \leq An
\]

Now the variance increases linearly with \( n \) rather than exponentially.

- Note that without resampling, the variance increases exponentially:

\[
\log \hat{p}(y_{1:n}) = \log \frac{1}{N} \sum_{n=1}^{N} g \left( y_k | \tilde{X}_k^{(i)} \right)
\]
We have seen that due to the degeneracy problem, the SMC approximation of $p(y_{1:n}, y_{1:n})$ is not good and only the SMC approximation of the marginals $p(y_{k-L+1:k}, y_{1:k})$ can be reasonable.

We can still use SMC for smoothing (e.g. compute $p(x_k | y_{1:n})$) using the forward/backward and the two-filter formulas.

These approximations rely on the marginals $p(x_k | y_{1:k})$. 
Forward/Backward Pass

- **Forward Pass:** Compute and store $p(x_k \mid y_{1:k})$ and $p(x_{k+1} \mid y_{1:k})$ for $k=1,2,\ldots,n$ using the updating recursion.

- **Particle Forward Pass:** Run a SMC forward and store $\hat{p}_N(x_k \mid y_{1:k})$, $k = 1, 2, \ldots, n$

- **Backward Pass:** Use for $k=n-1,n-2,\ldots,1$ the following recursion

$$p(x_k \mid y_{1:n}) = p(x_k \mid y_{1:k}) \int \frac{f(x_{k+1} \mid x_k)}{p(x_{k+1} \mid y_{1:k})} p(x_{k+1} \mid y_{1:n}) \, dx_{k+1}$$

- **Particle Backward Pass:**

$$W_{k-1|n}^{(i)} = W_{k-1}^{(i)} \sum_{l=1}^{N} W_{k|n}^{(l)} \frac{f(X_k^{(l)} \mid X_{k-1}^{(l)})}{\sum_{j=1}^{N} W_{k-1}^{(j)} f(X_k^{(l)} \mid X_{k-1}^{(l)})}$$

$$\hat{p}_N(x_k \mid y_{1:n}) = \sum_{i=1}^{N} W_{k|n}^{(i)} \delta_{X_k^{(i)}}(x_k)$$
The memory requirements are $O(Nn)$ and the complexity $O(N^2)$.

\[
\hat{p}_N(x_k | y_{1:n}) = \hat{p}_N(x_k | y_{1:k}) \int \frac{f(x_{k+1} | x_k) \hat{p}_N(x_{k+1} | y_{1:n})}{p(x_{k+1} | y_{1:k}) \hat{p}_N(x_k | y_{1:k})} dx_{k+1}
\]

\[
= \sum_{i=1}^{N} W^{(i)}_k \delta_{X^{(i)}_k} (x_k) \sum_{l=1}^{N} W^{(l)}_{k+1|n} \delta_{X^{(l)}_k} (x_{k+1}) f(X^{(l)}_{k+1} | x_k)
\]

\[
= \sum_{i=1}^{N} W^{(i)}_{k|n} \delta_{X^{(i)}_k} (x_k)
\]
This straightforward approach for smoothing is an improvement over using the marginals of $p(x_{1:n} \mid y_{1:n})$.

Note that in computing $p(x_k \mid y_{1:k})$, we reweight the particles $\{X_k^{(i)}\}$ but we keep their locations. Thus we assume that $p(x_k \mid y_{1:k})$ and $p(x_k \mid y_{1:n})$ have high probability in the same regions of the space. This may not be the case.
The two-filter formulas is based on

$$p(x_k | y_{1:n}) = \frac{p(x_k | y_{1:k}) p(y_{k+1:n} | x_k)}{p(y_{k+1:n} | y_{1:k})}$$

The backward filter satisfies the following

$$p(y_{k+1:n} | x_k) = \int p(y_{k+2:n} | x_{k+1}) g(y_{k+1} | x_{k+1}) f(x_{k+1} | x_k) dx_{k+1}$$

To implement a SMC approximation of the two filter formula, we need to approximate $$p(y_{k+1:n} | x_k)$$ and make the product of two SMC approximations.
Two Filter Smoothing

- Note that we can have
  \[ \int p(y_{k+1:n} | x_k) \, dx_k = \infty \]

  so this is not a proper probability measure:
  \[
  \frac{p(y_{k+1:n} | x_k)}{\int p(y_{k+1:n} | x_k) \, dx_k}
  \]

- It is thus impossible to have an SMC approximation of it.

- We simply cannot multiply two SMC approximations.
We use the following generalized two-filter formula:

\[
p(x_k | y_{1:n}) = \int p(x_{k-1} | y_{1:k-1}) \frac{f(x_k | x_{k-1})}{\eta_k(x_k)} \tilde{p}(x_k | y_{k:n}) dx_{k-1}
\]

where:

\[
\tilde{p}(x_k | y_{k:n}) = \frac{p(y_{k:n} | x_k) \eta_k(x_k)}{\int p(y_{k:n} | x_k) \eta_k(x_k) dx_k}
\]

for an artificial prior distribution \( \eta_k(x_k) \)

Note that now \( \tilde{p}(x_k | y_{k:n}) \) is a standard PDF in argument \( x_k \) and can be approximated using sequential Monte Carlo.
To estimate \( \tilde{p}(x_k \mid y_{k:n}) \) using SMC, we introduce a sequence of distributions:

\[
\tilde{p}(x_n \mid y_{k:n}) = \eta_n(x_n)g(y_n \mid x_n)
\]

\[
\tilde{p}(x_{k:n} \mid y_{k:n}) \propto \eta_k(x_k)\prod_{i=k+1}^{n} f(x_i \mid x_{i-1})\prod_{i=k}^{n} g(y_i \mid x_i), \quad k < n
\]

Note that with this construction we now have:

\[
\int \tilde{p}(x_{k:n} \mid y_{k:n}) \, dx_{k+1:n} = \tilde{p}(x_k \mid y_{k:n})
\]

and standard SMC can be applied.
**SISR Algorithm**

- At time n:
  Sample N particles $\tilde{X}_n^{(i)} \sim \tilde{q}\left(x_n \mid y_n\right)$, and compute
  \[
  \tilde{w}\left(\tilde{X}_n^{(i)}, y_n\right) = \frac{\eta_n\left(\tilde{X}_n^{(i)}\right) g\left(y_n \mid \tilde{X}_n^{(i)}\right)}{\tilde{q}\left(\tilde{X}_n^{(i)} \mid y_n\right)}, \tilde{W}_n^{(i)} \sim \tilde{w}\left(\tilde{X}_n^{(i)}, y_n\right)
  \]
  Resample $\left\{\tilde{X}_n^{(i)}, \tilde{W}_n^{(i)}\right\}$ to obtain new particles also denoted as $\left\{\tilde{X}_n^{(i)}\right\}$

- At time k, k<n
  Sample N particles $\tilde{X}_k^{(i)} \sim \tilde{q}\left(x_k \mid y_k, \tilde{X}_{k+1}^{(i)}\right)$, and compute
  \[
  w\left(\tilde{X}_{k:k+1}^{(i)}, y_k\right) = \frac{\eta_k\left(\tilde{X}_k^{(i)}\right) f\left(\tilde{X}_{k+1}^{(i)} \mid \tilde{X}_k^{(i)}\right) g\left(y_k \mid \tilde{X}_k^{(i)}\right)}{\eta_{k+1}\left(\tilde{X}_{k+1}^{(i)}\right) \tilde{q}\left(\tilde{X}_k^{(i)} \mid y_k, \tilde{X}_{k+1}^{(i)}\right)}, \tilde{W}_k^{(i)} \sim w\left(\tilde{X}_{k:k+1}^{(i)}, y_k\right)
  \]
  Resample $\left\{\tilde{X}_k^{(i)}, \tilde{W}_k^{(i)}\right\}$ to obtain new particles also denoted as $\left\{\tilde{X}_k^{(i)}\right\}$
An obvious choice is the following:

\[ \eta_k(x_k) = p(y_k) = \int \mu(x_1) \prod_{i=2}^{k} f(x_i | x_{i-1}) \, dx_{1:k-1} \]

from which we see that

\[ \frac{\eta_k(x_k) f(x_{k+1} | x_k)}{\eta_{k+1}(x_{k+1})} = p(x_k | x_{k+1}) \]

If \( \int \mu(x) f(x' | x) \, dx = \mu(x') \), then \( \eta_k(x) = \mu(x) \forall k \)

If \( f \) is \( \mu \)-reversible, i.e. if \( \mu(x) f(x' | x) = \mu(x') f(x | x') \), then:

\[ p(x_k | x_{k+1}) = f(x_{k+1} | x_k) \]
Additional degree of freedom is necessary because no analytical expression for $p(x_k)$ is available in general models.

For example:

$$X_{n+1} = \frac{X_n}{2} + 25\frac{X_n}{1 + X_n^2} + 8\cos(1.2n) + \sigma_v V_{n+1}$$

As a guideline, you can take any PDF as long as:

$$\eta_k(x_k) f(x_{k+1} | x_k) g(y_k | x_k) < \infty$$

$$\frac{\eta_{k+1}(x_{k+1}) q(x_k | x_{k+1}, y_k)}{\eta_k(x_k)} < \infty$$
Run standard SMC to approximate \( p(x_k \mid y_{1:k}) \) for \( k=1,2,\ldots,n \).

Run backward SMC to approximate \( \tilde{p}(x_k \mid y_{k:n}) \), \( k = n, n-1, \ldots, 1 \).

Use generalized two-filter formula identity approximation:

\[
p(x_k \mid y_{1:n}) \propto \int p(x_{k-1} \mid y_{1:k-1}) \frac{f(x_k \mid x_{k-1})}{\eta_k(x_k)} \tilde{p}(x_k \mid y_{k:n}) \, dx_{k-1} \Rightarrow
\]

\[
\tilde{p}(x_k \mid y_{1:n}) \propto \int \tilde{p}(x_{k-1} \mid y_{1:k-1}) \frac{f(x_k \mid x_{k-1})}{\eta_k(x_k)} \tilde{p}(x_k \mid y_{k:n}) \, dx_{k-1}
\]

\[
\propto \sum_{l=1}^{N} W_{k-1}^{(i)} \tilde{W}_k^{(j)} \frac{f(\tilde{X}_k^{(j)} \mid X_{k-1}^{(i)})}{\eta_k(\tilde{X}_k^{(j)})} \delta_{\tilde{X}_k^{(j)}}(x_k)
\]

The complexity of this algorithm is the same as that for the forward-backward algorithm \( O(N^2) \) but does not rely on filtering support and is more efficient.
Example

- Consider the following time series model:

\[ X_k = \frac{X_{k-1}}{2} + 25 \frac{X_{k-1}}{1 + X_{k-1}^2} + 8 \cos(1.2k) + V_k, \quad V_k \sim \mathcal{N}(0, 15), \quad X_1 \sim \mathcal{N}(0, 5) \]

\[ Y_k = \frac{X_k^2}{20} + W_k, \quad W_k \sim \mathcal{N}(0, 0.01) \]

- We chose an unscented approximation to the optimal importance function.

- We compare the two filter smoothing algorithm approximating both the prior t-f(1) and invariant distribution t-f(2) using the transition density together with the forward backward smoothing algorithm f-b.
Example

- Average RMS values for 100 MC runs of 50 time epochs

<table>
<thead>
<tr>
<th>N</th>
<th>f-b</th>
<th>t-f(1)</th>
<th>t-f(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>90.14</td>
<td>41.34</td>
<td>41.03</td>
</tr>
<tr>
<td>100</td>
<td>86.40</td>
<td>41.66</td>
<td>40.38</td>
</tr>
<tr>
<td>500</td>
<td>81.12</td>
<td>43.56</td>
<td>45.14</td>
</tr>
<tr>
<td>1000</td>
<td>83.36</td>
<td>39.73</td>
<td>39.99</td>
</tr>
</tbody>
</table>

- Average ESS values for 100 MC runs of 50 time epochs

<table>
<thead>
<tr>
<th>N</th>
<th>f-b</th>
<th>t-f(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>34.8</td>
<td>47.2</td>
</tr>
<tr>
<td>100</td>
<td>67.7</td>
<td>94.3</td>
</tr>
<tr>
<td>500</td>
<td>327.9</td>
<td>472.2</td>
</tr>
<tr>
<td>1000</td>
<td>645.2</td>
<td>940.2</td>
</tr>
</tbody>
</table>

From A. Doucet Lecture Notes
Summary

- The generalized two-filter formula outperforms generally the forward-backward algorithms. It is also more widely applicable; e.g. exact smoothing for partially observed diffusions.

- These smoothing algorithms are in $O(N^2)$ but fast versions can be implemented for dedicated applications.

- These smoothing algorithms will only be interesting for long time series compared to the standard SMC method.
Auxiliary Particle Filters
The most popular alternative to **Sequential Importance Sampling and Resampling** (SISR) is the class of **Auxiliary Particle Filters** (APF) (Pitt & Shephard, JASA, 1999).

The APF owes its name as it has been introduced using auxiliary variables.

It is widely believed in the literature that the APF is significantly different from SISR.

Recently, a novel interpretation of the APF is given by A. Doucet as an SISR algorithm.

A Generic SISR Algorithm

Consider an arbitrary sequence of probability densities \( \{\pi_t(x_{1:t})\}_{t=1} \). To sample sequentially from these distributions, the SISR algorithm proceeds as follows:

At time 1

- Sampling step:
  For \( i = 1:N \), sample \( X_1^{(i)} \sim q_1(\cdot) \)

- Resampling step:
  For \( i = 1:N \), compute weights \( w_i(X_1^{(i)}) = \frac{\pi_1(X_1^{(i)})}{q_1(X_1^{(i)})} \) and \( W_1^{(i)} = \frac{w_i(X_1^{(i)})}{\sum_{j=1}^N w_j(X_1^{(j)})} \)

  For \( i = 1:N \), sample \( \hat{X}_1^{(i)} \sim \sum_{j=1}^N W_1^{(j)} \delta_{X_1^{(j)}}(dx_1) \) where \( \delta_x(\cdot) \) denotes the singular distribution located at \( x \).

At time \( t \) (\( t>1 \))

- Sampling step
  For \( i = 1:N \), sample \( X_t^{(i)} \sim q_t(\cdot | \hat{X}_{t-1}^{(i)}) \)
A Generic SISR Algorithm

- Resampling step
  For $i = 1:N$, compute weights
  \[
  w_t\left(\hat{X}_{t-1}^{(i)}, X_t^{(i)}\right) = \frac{\pi_t\left(\hat{X}_{t-1}^{(i)}, X_t^{(i)}\right)}{\pi_{t-1}\left(\hat{X}_{t-1}^{(i)}\right)q_t\left(X_t^{(i)} | \hat{X}_{t-1}^{(i)}\right)}
  \]
  \[
  W_t^{(i)} = \frac{w_t\left(\hat{X}_{t-1}^{(i)}, X_t^{(i)}\right)}{\sum_{j=1}^{N}w_t\left(\hat{X}_{t-1}^{(j)}, X_t^{(j)}\right)}
  \]
  For $i = 1:N$, sample
  \[
  \hat{X}_{t}^{(i)} \sim \sum_{j=1}^{N}W_t^{(j)}\delta_{\left(\hat{X}_{t-1}^{(j)}, X_t^{(i)}\right)}(dx_{1:t})
  \]
Consider an unobserved Markov process \( \{X_t\} \) such that
\[
X_1 \sim \mu(\bullet), \quad X_t \mid (X_{t-1} = x_{t-1}) \sim f(\cdot | x_{t-1})
\]
The observation \( \{Y_t\} \) are conditionally independent given \( \{X_t\} \) and distributed according to
\[
Y_t \mid (X_t = x_t) \sim g(\cdot | x_t)
\]

The standard SISR algorithm for filtering corresponds to the case in which we set \( \pi_t(x_{1:t}) = p(x_{1:t} | y_{1:t}) \)

The APF attempts to bias the particles in promising regions of the space.
To achieve this, it targets not the distributions $p(x_{1:t} \mid y_{1:t})$ but

$$
\pi_t(x_{1:t}) = \hat{p}(x_{1:t} \mid y_{1:t+1}) \propto \hat{p}(y_{t+1} \mid x_t) p(x_{1:t} \mid y_{1:t})
$$

with $\hat{p}(y_{t+1} \mid x_t)$ an approximation of

$$
p(y_{t+1} \mid x_t) = \int g(y_{t+1} \mid x_{t+1}) f(x_{t+1} \mid x_t) dx_{t+1}
$$

if $p(y_{t+1} \mid x_t)$ is not know analytically.

The SISR algorithm is not fundamentally changed by modifying the target; we simply substitute with $\hat{p}(x_{1:t} \mid y_{1:t+1})$ all places of the SMC algorithm where we had $p(x_{1:t} \mid y_{1:t+1})$. 
In APF, suppose the importance distribution can be expressed as

\[ q_t(x_{1:t}) = q_{t-1}(x_{1:t-1})q_t(x_t | x_{t-1}) \]

The recursive importance weights are formulated as

\[
w_t(x_{1:t}) = \frac{\pi_t(x_{1:t})}{q_t(x_{1:t})} = \frac{\hat{p}(x_t | y_{1:t+1})}{q_t(x_{1:t})} \propto \frac{\hat{p}(y_{t+1} | x_t)p(x_{1:t} | y_{1:t})}{q_{t-1}(x_{1:t-1})q_t(x_t | x_{t-1})} \]

\[
\propto \frac{p(x_{1:t-1} | y_{1:t-1})\hat{p}(y_{t} | x_{t-1})\hat{p}(y_{t+1} | x_t)\hat{p}(y_{t} | x_t)g(y_t | x_t)}{q_{t-1}(x_{1:t-1})q_t(x_t | x_{t-1})\hat{p}(y_t | x_{t-1})} \]

\[
\propto w_{t-1}(x_{1:t-1})\frac{\hat{p}(y_{t+1} | x_t)\hat{p}(y_{t} | x_t)g(y_t | x_t)}{q_t(x_t | x_{t-1})\hat{p}(y_t | x_{t-1})} \]

where

\[
w_{t-1}(x_{1:t-1}) = \frac{\pi_{t-1}(x_{1:t-1})}{q_{t-1}(x_{1:t-1})} = \frac{\hat{p}(x_{1:t-1} | y_{1:t})}{q_{t-1}(x_{1:t-1})} \propto \frac{\hat{p}(y_{t} | x_{t-1})p(x_{1:t-1} | y_{1:t-1})}{q_{t-1}(x_{1:t-1})q_t(x_{1:t-1})} \]

Auxiliary Particle Filters
As we are typically interested in \( p(x_{1:t} \mid y_{1:t}) \) and not \( \hat{p}(x_{1:t} \mid y_{1:t+1}) \), we need to perform an importance sampling correction.

Suppose at step \( t-1 \), we have a particle \( \hat{X}_{1:t-1}^{(i)} \). Then we get the sample at step \( t \) by

\[
X_t^{(i)} \sim q_t \left( \cdot \mid \hat{X}_{t-1}^{(i)} \right)
\]

Assume the objective is to estimate a test function \( \varphi_t \) such that

\[
\bar{\varphi}_t = \int \varphi_t(x_{1:t}) p(x_{1:t} \mid y_{1:t}) \, dx_{1:t}
\]

The resulting APF estimate is given by

\[
\tilde{\varphi}_{t, \text{APF}}^N = \sum_{i=1}^{N} \tilde{W}_t^{(i)} \, \varphi_t \left( X_{1:t-1}^{(i)}, X_t^{(i)} \right)
\]

where

\[
\tilde{W}_t^{(i)} = \frac{\tilde{w}_t \left( \hat{X}_{1:t-1}^{(i)}, X_t^{(i)} \right)}{\sum_{j=1}^{N} \tilde{w}_t \left( \hat{X}_{1:t-1}^{(i)}, X_t^{(i)} \right)}
\]
The unnormalized weight can be obtained by
\[
\tilde{w}_t(x_{1:t}) = \frac{p(x_{1:t} | y_{1:t})}{\pi_t(x_{1:t-1}) q_t(x_t | x_{t-1})} = \frac{p(x_{1:t} | y_{1:t})}{p(x_{1:t-1} | y_{1:t-1}) \hat{p}(y_t | x_{t-1}) q_t(x_t | x_{t-1})}
\]
\[
\propto \frac{f(x_t | x_{t-1}) g(y_t | x_t)}{\hat{p}(y_t | x_{t-1}) q_t(x_t | x_{t-1})}
\]

We usually select the importance distribution \( q_t(x_t | x_{t-1}) \) as an approximation to
\[
p(x_t | y_t, x_{t-1}) = \frac{g(y_t | x_t) f(x_t | x_{t-1})}{p(y_t | x_{t-1})}
\]
which is often referred to as the optimal importance distribution.

When it is possible to select \( q_t(x_t | x_{t-1}) = p(x_t | y_t, x_{t-1}) \) and \( \hat{p}(y_{t+1} | x_t) = p(y_{t+1} | x_t) \), we obtain the so-called “perfect adaption” case in which the importance weights \( \tilde{w}_t(x_{1:t}) \) are all equal. Then there is no need to reweight.
APF Algorithm

- At time 1
  - For $i = 1:N$
    - Sample $X_1^{(i)} \sim q_1(\cdot)$, and set
      $$\tilde{W}_1^{(i)} \propto \frac{g(y_1 | X_1^{(i)}) f(X_1^{(i)})}{q_1(X_1^{(i)})}$$

- At time $n > 1$
  - For $i = 1:N$, set
    $$W_{n-1}^{(i)} \propto \tilde{W}_{n-1}^{(i)} \times \hat{p}(y_n | X_{n-1}^{(i)})$$
  - Resample \( \{X_{n-1}^{(i)}, W_{n-1}^{(i)}\} \) to obtain \( \{X_{n-1}^{(i)}', \frac{1}{N}\} \)
  - For $i = 1:N$
    - Set $X_{n-1}^{(i)} = X_{n-1}^{(i)}$
  - Sample $X_n^{(i)} \sim q_n(\cdot | X_{n-1}^{(i)})$
  - Set
    $$\tilde{W}_n^{(i)} \propto \frac{f(X_n^{(i)} | X_{n-1}^{(i)}) g(y_n | X_n^{(i)})}{\hat{p}(y_n | X_{n-1}^{(i)}) q_n(X_n^{(i)} | X_{n-1}^{(i)})}$$
This interpretation of the APF shows that one should select
\[ \hat{p}(y_t | x_{t-1})q_t(x_t | x_{t-1}) \]
with heavier tails than
\[ f(x_t | x_{t-1})g(y_t | x_t) \]

It is suggested to use:
\[ \hat{p}(y_t | x_{t-1}) = g(y_t | \mu(x_{t-1})) \]
where \( \mu(x_{t-1}) \) is the mode, mean or median of \( f(x_t | x_{t-1}) \). In most cases this will lead to unbounded importance weights.

The APF will not be beneficial compared to the standard SISR unless you have a good approximation of both \( p(y_t | x_{t-1}) \) and \( p(x_t | y_t, x_{t-1}) \).

In practice, you should build directly an approximation of \( p(x_t, y_t | x_{t-1}) \).

Contrary to popular belief, the APF will not always outperform the SISR even when we can use \( p(y_t | x_{t-1}) \) and \( p(x_t | y_t, x_{t-1}) \).
Consider the following binary state-space model with common state and observation spaces:

\[ E = \{0,1\} \quad p(x_1 = 0) = 0.5, \quad p(x_t = x_{t-1}) = 1 - \delta, \]
\[ y \in E \quad p(y_t = x_t) = 1 - \epsilon \]

Suppose we are interested in

\[ \varphi = \int \varphi(x_{1:2}) p(x_{1:2} \mid y_{1:2}) \, dx_{1:2} \]

where \( \varphi(x_{1:2}) = x_2 \)

The observation data at the first two steps are \( y_{1:2} = \{0,1\} \)

Example

- In this example, the optimal proposal distribution can be obtained:
  \[ q_1 = p(x_1 | y_1), \quad q_2 = p(x_2 | y_2, x_1) \]

- Analytic expression of \( p(y_n | x_{n-1}) \) is also available such that:
  \[
p(y_2 | x_1) = \int g(y_2 | x_2) f(x_2 | x_1) dx_2
  \]
  when \( x_1 = 0 \), \( p(y_2 = 1 | x_1 = 0) = (1 - \delta) \varepsilon + (1 - \varepsilon) \delta \)
  when \( x_1 = 1 \), \( p(y_2 = 1 | x_1 = 0) = \delta \varepsilon + (1 - \varepsilon)(1 - \delta) \)

- We choose different values for parameters \( \delta \) and \( \varepsilon \). \( N = 3000 \) particles are used in sampling procedures. 1000 runs are performed to compute the variance of estimated integral \( \varphi \).
Example

\[ \varepsilon = 0.25 \]

\[ \delta \]

\[ \text{variance} \times \text{No. particles} \]

- Direct sampling
- APF

A Matlab implementation