Sequential Monte Carlo

Nando de Freitas & Arnaud Doucet
University of British Columbia
December 2009
Tutorial overview

- **Introduction** Nando – 10min
- **Part I** Arnaud – 50min
  - Monte Carlo
  - Sequential Monte Carlo
  - Theoretical convergence
  - Improved particle filters
  - Online Bayesian parameter estimation
  - Particle MCMC
  - Smoothing
  - Gradient based online parameter estimation
- **Break** 15min
- **Part II** NdF – 45 min
  - Beyond state space models
  - Eigenvalue problems
  - Diffusion, protein folding & stochastic control
  - Time-varying Pitman-Yor Processes
  - SMC for static distributions
  - Boltzmann distributions & ABC
Tutorial overview

- **Introduction** Nando – 10min
- **Part I** Arnaud – 50min
  - Monte Carlo
  - Sequential Monte Carlo
  - Theoretical convergence
  - Improved particle filters
  - Online Bayesian parameter estimation
  - Particle MCMC
  - Smoothing
  - Gradient based online parameter estimation
- **Break** 15min
- **Part II** NdF – 45 min
  - Beyond state space models
  - Eigenvalue problems
  - Diffusion, protein folding & stochastic control
  - Time-varying Pitman-Yor Processes
  - SMC for static distributions
  - Boltzmann distributions & ABC

20th century
SMC in this community

Many researchers in the NIPS community have contributed to the field of Sequential Monte Carlo over the last decade.

- *Michael Isard* and *Andrew Blake* popularized the method with their Condensation algorithm for image tracking.
- Soon after, *Daphne Koller*, *Stuart Russell*, *Kevin Murphy*, *Sebastian Thrun*, *Dieter Fox* and *Frank Dellaert* and their colleagues demonstrated the method in AI and robotics.
- *Tom Griffiths* and colleagues have studied SMC methods in cognitive psychology.
The 20th century - Tracking

[Michael Isard & Andrew Blake (1996)]
The 20th century - Tracking

[Boosted particle filter of Kenji Okuma, Jim Little & David Lowe]
The 20\textsuperscript{th} century – State estimation

[Dieter Fox]
http://www.cs.washington.edu/ai/Mobile_Robotics/mcl/
The 20th century – State estimation

[Dieter Fox]
http://www.cs.washington.edu/ai/Mobile_Robotics/mcl/
The 20th century – State estimation
The 20th century – The birth

[Metropolis and Ulam, 1949]
Tutorial overview

- **Introduction**  Nando – 10min

- **Part I  Arnaud – 50min**
  - Monte Carlo
  - Sequential Monte Carlo
  - Theoretical convergence
  - Improved particle filters
  - Online Bayesian parameter estimation
  - Particle MCMC
  - Smoothing
  - Gradient based online parameter estimation

- **Break 15min**

- **Part II  NdF – 45 min**
  - Beyond state space models
  - Eigenvalue problems
  - Diffusion, protein folding & stochastic control
  - Time-varying Pitman-Yor Processes
  - SMC for static distributions
  - Boltzmann distributions & ABC
Arnaud’s slides will go here
Sequential Monte Carlo Methods

Nando De Freitas & Arnaud Doucet
UBC
\( \{X_k\}_{k \geq 1} \) hidden \( \mathcal{X} \)-valued Markov process with

\[
X_1 \sim \mu(x_1) \text{ and } X_k \mid (X_{k-1} = x_{k-1}) \sim f(x_k \mid x_{k-1}).
\]
State-Space Models

- $\{X_k\}_{k \geq 1}$ hidden $\mathcal{X}$-valued Markov process with
  \[ X_1 \sim \mu(x_1) \text{ and } X_k \mid (X_{k-1} = x_{k-1}) \sim f(x_k \mid x_{k-1}). \]

- $\{Y_k\}_{k \geq 1}$ observed $\mathcal{Y}$-valued process with observations conditionally independent given $\{X_k\}_{k \geq 1}$ with
  \[ Y_k \mid (X_k = x_k) \sim g(y_k \mid x_k). \]
State-Space Models

- \( \{X_k\}_{k \geq 1} \) hidden \( \mathcal{X} \)-valued Markov process with
  \[
  X_1 \sim \mu (x_1) \quad \text{and} \quad X_k \mid (X_{k-1} = x_{k-1}) \sim f (x_k \mid x_{k-1}).
  \]

- \( \{Y_k\}_{k \geq 1} \) observed \( \mathcal{Y} \)-valued process with observations conditionally independent given \( \{X_k\}_{k \geq 1} \) with
  \[
  Y_k \mid (X_k = x_k) \sim g (y_k \mid x_k).
  \]

- **Main Objective:** Estimate \( \{X_k\}_{k \geq 1} \) given \( \{Y_k\}_{k \geq 1} \) online/offline.
Inference in State-Space Models

- Given observations $y_{1:n} := (y_1, y_2, \ldots, y_n)$, inference about $X_{1:n} := (X_1, \ldots, X_n)$ relies on the posterior

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

where

$$p(x_{1:n}, y_{1:n}) = \mu(x_1) \prod_{k=2}^{n} f(x_k \mid x_{k-1}) \prod_{k=1}^{n} g(y_k \mid x_k)$$

$$p(y_{1:n}) = \int \cdots \int p(x_{1:n}, y_{1:n}) \, dx_{1:n}$$
Inference in State-Space Models

- Given observations $y_{1:n} := (y_1, y_2, \ldots, y_n)$, inference about $X_{1:n} := (X_1, \ldots, X_n)$ relies on the posterior

\[ p (x_{1:n} \mid y_{1:n}) = \frac{p (x_{1:n}, y_{1:n})}{p (y_{1:n})} \]

where

\[ p (x_{1:n}, y_{1:n}) = \mu (x_1) \prod_{k=2}^{n} f (x_k \mid x_{k-1}) \prod_{k=1}^{n} g (y_k \mid x_k), \]

\[ p (y_{1:n}) = \int \cdots \int p (x_{1:n}, y_{1:n}) \, dx_{1:n} \]

- We want to compute $p (x_{1:n} \mid y_{1:n})$ and $p (y_{1:n})$ sequentially in time $n$. 

For non-linear non-Gaussian models, numerical approximations are required.
Given observations $y_{1:n} := (y_1, y_2, \ldots, y_n)$, inference about $X_{1:n} := (X_1, \ldots, X_n)$ relies on the posterior

$$p(x_{1:n} | y_{1:n}) = \frac{p(x_{1:n}, y_{1:n})}{p(y_{1:n})}$$

where

$$p(x_{1:n}, y_{1:n}) = \mu(x_1) \prod_{k=2}^{n} f(x_k | x_{k-1}) \prod_{k=1}^{n} g(y_k | x_k),$$

$$p(y_{1:n}) = \int \cdots \int p(x_{1:n}, y_{1:n}) \, dx_{1:n}$$

We want to compute $p(x_{1:n} | y_{1:n})$ and $p(y_{1:n})$ sequentially in time $n$.

For non-linear non-Gaussian models, numerical approximations are required.
Assume you can generate $X_{1:n}^{(i)} \sim p(x_{1:n} | y_{1:n})$ where $i = 1, \ldots, N$ then MC approximation is

$$\hat{p}(x_{1:n} | y_{1:n}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X_{1:n}^{(i)}}(x_{1:n})$$
Monte Carlo Methods

- Assume you can generate $X_{1:n}^{(i)} \sim p(x_{1:n} \mid y_{1:n})$ where $i = 1, \ldots, N$ then MC approximation is

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

- **Integration is straightforward**

$$
\int \varphi_n(x_{1:n}) \hat{p}(x_{1:n} \mid y_{1:n}) \, dx_{1:n} = \frac{1}{N} \sum_{i=1}^{N} \varphi_n(X_{1:n}^{(i)}).
$$
Monte Carlo Methods

- Assume you can generate \(X_{1:n}^{(i)} \sim p(x_{1:n} | y_{1:n})\) where \(i = 1, \ldots, N\) then MC approximation is

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

- Integration is straightforward

\[
\int \varphi_n(x_{1:n}) \hat{p}(x_{1:n} | y_{1:n}) \, dx_{1:n} = \frac{1}{N} \sum_{i=1}^{N} \varphi_n(X_{1:n}^{(i)}).
\]

- Marginalisation is straightforward

\[
\hat{p}(x_k | y_{1:n}) = \int \hat{p}(x_k | y_{1:n}) \, dx_{1:k-1} \, dx_{k+1:n} = \frac{1}{N} \sum_{i=1}^{N} \delta_{X_{k}^{(i)}}(x_k)
\]
Monte Carlo Methods

- Assume you can generate $X_{1:n}^{(i)} \sim p(x_{1:n} | y_{1:n})$ where $i = 1, \ldots, N$ then MC approximation is

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

- Integration is straightforward

$$
\int \varphi_n(x_{1:n}) \hat{p}(x_{1:n} | y_{1:n}) \, dx_{1:n} = \frac{1}{N} \sum_{i=1}^{N} \varphi_n(X_{1:n}^{(i)})
$$

- Marginalisation is straightforward

$$
\hat{p}(x_k | y_{1:n}) = \int \hat{p}(x_k | y_{1:n}) \, dx_{1:k-1} \, dx_{k+1:n} = \frac{1}{N} \sum_{i=1}^{N} \delta_{X_{k}^{(i)}}(x_k)
$$

- **Problem:** Sampling from $p(x_{1:n} | y_{1:n})$ is impossible in general cases.
**Divide and conquer strategy**: Break the problem of sampling from $p(x_{1:n} \mid y_{1:n})$ into a collection of simpler subproblems. First approximate $p(x_1 \mid y_1)$ at time 1, then $p(x_{1:2} \mid y_{1:2})$ at time 2 and so on.
Divide and conquer strategy: Break the problem of sampling from $p(x_1:n \mid y_1:n)$ into a collection of simpler subproblems. First approximate $p(x_1 \mid y_1)$ at time 1, then $p(x_{1:2} \mid y_{1:2})$ at time 2 and so on.

Each target distribution is approximated by a cloud of random samples termed particles evolving according to importance sampling and resampling steps.
Importance Sampling

- Assume you have at time $n - 1$

$$\hat{p}(x_{1:n-1} | y_{1:n-1}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X_{1:n-1}}^{(i)}(x_{1:n-1}) .$$
Assume you have at time $n - 1$

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

By sampling $\tilde{X}^{(i)}_{n} \sim f(x_{n} \mid X^{(i)}_{n-1})$ and setting $\tilde{X}^{(i)}_{1:n} = (X^{(i)}_{1:n-1}, \tilde{X}^{(i)}_{n})$ then

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

- Assume you have at time $n - 1$
  \[
  \hat{p}(x_{1:n-1} \mid y_{1:n-1}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X_{1:n-1}^{(i)}}(x_{1:n-1}).
  \]

- By sampling $\tilde{X}_{n}^{(i)} \sim f(x_{n} \mid X_{n-1}^{(i)})$ and setting $\tilde{X}_{1:n}^{(i)} = (X_{1:n-1}^{(i)}, \tilde{X}_{n}^{(i)})$
  then
  \[
  \hat{p}(x_{1:n} \mid y_{1:n-1}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{\tilde{X}_{1:n}^{(i)}}(x_{1:n}).
  \]

- Our target at time $n$ is
  \[
  p(x_{1:n} \mid y_{1:n}) = \frac{g(y_{n} \mid x_{n})}{\int g(y_{n} \mid x_{n}) p(x_{1:n} \mid y_{1:n-1}) \, dx_{n}} p(x_{1:n} \mid y_{1:n-1})
  \]
  so by substituting $\hat{p}(x_{1:n} \mid y_{1:n-1})$ to $p(x_{1:n} \mid y_{1:n-1})$ we obtain
  \[
  \tilde{p}(x_{1:n} \mid y_{1:n}) = \sum_{i=1}^{N} W_{n}^{(i)} \delta_{\tilde{X}_{1:n}^{(i)}}(x_{1:n}), \quad W_{n}^{(i)} \propto g(y_{n} \mid \tilde{X}_{1:n}^{(i)}).
  \]
Resampling

We have a “weighted” approximation \( \bar{p}(x_{1:n} | y_{1:n}) \) of \( p(x_{1:n} | y_{1:n}) \)

\[
\bar{p}(x_{1:n} | y_{1:n}) = \sum_{i=1}^{N} W_n^{(i)} \delta_{\tilde{x}^{(i)}_{1:n}}(x_{1:n}).
\]
Resampling

- We have a “weighted” approximation $\tilde{p}(x_{1:n} \mid y_{1:n})$ of $p(x_{1:n} \mid y_{1:n})$:

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

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

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

to obtain

$$\hat{p}(x_{1:n} \mid y_{1:n}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X_{1:n}^{(i)}}(x_{1:n}).$$
Resampling

- We have a “weighted” approximation \( \tilde{p}(x_{1:n} \mid y_{1:n}) \) of \( p(x_{1:n} \mid y_{1:n}) \)

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

- To obtain \( N \) samples \( X_{1:n}^{(i)} \) approximately distributed according to \( p(x_{1:n} \mid y_{1:n}) \), we just resample

\[
X_{1:n}^{(i)} \sim \tilde{p}(x_{1:n} \mid y_{1:n})
\]

to obtain

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

- Particles with high weights are copied multiples times, particles with low weights die.
At time $n = 1$

- Sample $\tilde{X}_{1}^{(i)} \sim \mu(x_{1})$ then

$$
\tilde{p}(x_{1} | y_{1}) = \sum_{i=1}^{N} W_{1}^{(i)} \delta_{\tilde{X}_{1}^{(i)}}(x_{1}), \quad W_{1}^{(i)} \propto g(y_{1} | \tilde{X}_{1}^{(i)}).
$$
At time $n = 1$

- Sample $\tilde{X}_1^{(i)} \sim \mu(x_1)$ then

$$
\tilde{p}(x_1 | y_1) = \sum_{i=1}^{N} W_1^{(i)} \delta_{\tilde{X}_1^{(i)}}(x_1), \quad W_1^{(i)} \propto g(y_1 | \tilde{X}_1^{(i)}).
$$

- Resample $X_1^{(i)} \sim \tilde{p}(x_1 | y_1)$ to obtain $\hat{p}(x_1 | y_1) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X_1^{(i)}}(x_1)$. 
At time $n = 1$

- Sample $\tilde{X}_1^{(i)} \sim \mu(x_1)$ then

\[
\tilde{p}(x_1 | y_1) = \sum_{i=1}^{N} W_1^{(i)} \delta_{\tilde{X}_1^{(i)}}(x_1), \quad W_1^{(i)} \propto g(y_1 | \tilde{X}_1^{(i)}).
\]

- Resample $X_1^{(i)} \sim \tilde{p}(x_1 | y_1)$ to obtain $\hat{p}(x_1 | y_1) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X_1^{(i)}}(x_1)$.
At time $n = 1$

- Sample $\tilde{X}_1^{(i)} \sim \mu(x_1)$ then

$$\tilde{p}(x_1 | y_1) = \sum_{i=1}^{N} W_1^{(i)} \delta_{\tilde{X}_1^{(i)}}(x_1), \quad W_1^{(i)} \propto g(y_1 | \tilde{X}_1^{(i)}) .$$

- Resample $X_1^{(i)} \sim \tilde{p}(x_1 | y_1)$ to obtain $\tilde{p}(x_1 | y_1) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X_1^{(i)}}(x_1) .

At time $n \geq 2$

- Sample $\tilde{X}_n^{(i)} \sim f(x_n | X_{n-1}^{(i)})$, set $\tilde{X}_{1:n} = (X_{1:n-1}^{(i)}, \tilde{X}_n^{(i)})$ and

$$\tilde{p}(x_{1:n} | y_{1:n}) = \sum_{i=1}^{N} W_n^{(i)} \delta_{\tilde{X}_{1:n}^{(i)}}(x_{1:n}), \quad W_n^{(i)} \propto g(y_n | \tilde{X}_n^{(i)}) .$$
Bootstrap Filter (Gordon, Salmond & Smith, 1993)

At time $n = 1$

- Sample $\tilde{X}_1^{(i)} \sim \mu (x_1)$ then
  
  $$\tilde{p} (x_1 | y_1) = \sum_{i=1}^{N} W_1^{(i)} \delta_{\tilde{X}_1^{(i)}} (x_1), \quad W_1^{(i)} \propto g \left( y_1 | \tilde{X}_1^{(i)} \right).$$

- Resample $X_1^{(i)} \sim \tilde{p} (x_1 | y_1)$ to obtain $\tilde{p} (x_1 | y_1) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X_1^{(i)}} (x_1)$.

At time $n \geq 2$

- Sample $\tilde{X}_n^{(i)} \sim f \left( x_n | X_{n-1}^{(i)} \right)$, set $\tilde{X}_{1:n} = \left( X_{1:n-1}^{(i)}, \tilde{X}_n^{(i)} \right)$ and
  
  $$\tilde{p} (x_{1:n} | y_{1:n}) = \sum_{i=1}^{N} W_n^{(i)} \delta_{\tilde{X}_{1:n}^{(i)}} (x_{1:n}), \quad W_n^{(i)} \propto g \left( y_n | \tilde{X}_n^{(i)} \right).$$

- Resample $X_{1:n}^{(i)} \sim \tilde{p} (x_{1:n} | y_{1:n})$ to obtain $\tilde{p} (x_{1:n} | y_{1:n}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X_{1:n}^{(i)}} (x_{1:n})$. 

Nando De Freitas & Arnaud Doucet UBC ( ) Sequential Monte Carlo Methods 8 / 39
At time $n$, we get

$$
\hat{p}(x_{1:n} \mid y_{1:n}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x_{1:n}^{(i)}}(x_{1:n}).
$$
At time $n$, we get

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

The marginal likelihood estimate is given by

$$
\hat{p}(y_{1:n}) = \prod_{k=1}^{n} \hat{p}(y_{k} \mid y_{1:k-1}) = \prod_{k=1}^{n} \left( \frac{1}{N} \sum_{i=1}^{N} g \left(y_{k} \mid \tilde{X}_{k}^{(i)} \right) \right).
$$
At time $n$, we get

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

The marginal likelihood estimate is given by

$$\hat{p}(y_{1:n}) = \prod_{k=1}^{n} \hat{p}(y_k | y_{1:k-1}) = \prod_{k=1}^{n} \left( \frac{1}{N} \sum_{i=1}^{N} g(y_k | \tilde{X}_k^{(i)}) \right).$$

Computational complexity is $O(N)$ and memory requirements $O(nN)$. 
At time $n$, we get

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

The marginal likelihood estimate is given by

$$
\hat{p}(y_{1:n}) = \prod_{k=1}^{n} \hat{p}(y_k | y_{1:k-1}) = \prod_{k=1}^{n} \left( \frac{1}{N} \sum_{i=1}^{N} g(y_k | \tilde{X}_{k}^{(i)}) \right) .
$$

Computational complexity is $O(N)$ and memory requirements $O(nN)$.

If we are only interested in $p(x_n | y_{1:n})$ or $p(s_n(x_{1:n}) | y_{1:n})$ where $s_n(x_{1:n}) = \Psi_n(x_n, s_{n-1}(x_{1:n-1}))$ is fixed-dimensional then memory requirements $O(N)$.
Figure: \( p(x_1|y_1) \) and \( \hat{E}[X_1|y_1] \) (top) and particle approximation of \( p(x_1|y_1) \) (bottom).
Figure: $p(x_1|y_1)$, $p(x_2|y_{1:2})$ and $\widehat{\mathbb{E}}[X_1|y_1]$, $\widehat{\mathbb{E}}[X_2|y_{1:2}]$ (top) and particle approximation of $p(x_{1:2}|y_{1:2})$ (bottom)
Figure: $p(x_k|y_{1:k})$ and $\hat{E}[X_k|y_{1:k}]$ for $k = 1, 2, 3$ (top) and particle approximation of $p(x_{1:3}|y_{1:3})$ (bottom)
**Figure:** $p(x_k | y_{1:k})$ and $\widehat{E} [X_k | y_{1:k}]$ for $k = 1, \ldots, 10$ (top) and particle approximation of $p(x_{1:10} | y_{1:10})$ (bottom)
Figure: $p(x_k | y_{1:k})$ and $\widehat{E}[X_k | y_{1:k}]$ for $k = 1, \ldots, 24$ (top) and particle approximation of $p(x_{1:24} | y_{1:24})$ (bottom)
Degeneracy problem. For any $N$ and any $k$, there exists $n(k, N)$ such that for any $n \geq n(k, N)$

$$
\hat{p}(x_1:k \mid y_1:n) = \delta_{x_1:k}(x_1:k).
$$

$\hat{p}(x_1:n \mid y_1:n)$ is an unreliable approximation of $p(x_1:n \mid y_1:n)$ as $n \nearrow$.

Figure: Exact calculation of $\frac{1}{n} \mathbb{E} \left[ \sum_{k=1}^{n} X_k \mid y_1:n \right]$ via Kalman (blue) vs SMC estimate (red) for $N = 1000$. As $n$ increases, the SMC estimate deteriorates.
Numerous precise convergence results are available for SMC methods (Del Moral, 2004).
Convergence Results

- Numerous precise convergence results are available for SMC methods (Del Moral, 2004).
- Let $\varphi_n : \mathcal{X}^n \to \mathbb{R}$ and consider

\[
\overline{\varphi}_n = \int \varphi_n (x_{1:n}) \, p (x_{1:n} \, | \, y_{1:n}) \, dx_{1:n},
\]

\[
\hat{\varphi}_n = \int \varphi_n (x_{1:n}) \, \hat{p} (x_{1:n} \, | \, y_{1:n}) \, dx_{1:n} = \frac{1}{N} \sum_{i=1}^{N} \varphi_n \left( X_{1:n}^{(i)} \right).
\]
Numerous precise convergence results are available for SMC methods (Del Moral, 2004).

Let \( \phi_n : X^n \rightarrow \mathbb{R} \) and consider

\[
\overline{\phi}_n = \int \phi_n(x_{1:n}) \ p(x_{1:n} | y_{1:n}) \ dx_{1:n},
\]

\[
\hat{\phi}_n = \int \phi_n(x_{1:n}) \ \hat{p}(x_{1:n} | y_{1:n}) \ dx_{1:n} = \frac{1}{N} \sum_{i=1}^N \phi_n \left( X_{1:n}^{(i)} \right).
\]

Under very weak assumptions, we have for any \( p > 0 \)

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

and

\[
\lim_{N \rightarrow \infty} \sqrt{N} (\hat{\phi}_n - \overline{\phi}_n) \Rightarrow \mathcal{N} (0, \sigma_n^2).
\]
Convergence Results

- Numerous precise convergence results are available for SMC methods (Del Moral, 2004).
- Let \( \varphi_n : \mathcal{X}^n \to \mathbb{R} \) and consider

\[
\bar{\varphi}_n = \int \varphi_n(x_1:n) p(x_1:n \mid y_1:n) \, dx_1:n,
\]

\[
\hat{\varphi}_n = \int \varphi_n(x_1:n) \hat{p}(x_1:n \mid y_1:n) \, dx_1:n = \frac{1}{N} \sum_{i=1}^{N} \varphi_n \left( X_1^{(i)} \right).
\]

- Under very weak assumptions, we have for any \( p > 0 \)

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

and

\[
\lim_{N \to \infty} \sqrt{N} \left( \hat{\varphi}_n - \bar{\varphi}_n \right) \Rightarrow \mathcal{N} \left( 0, \sigma_n^2 \right).
\]

- **Very weak results**: \( C_n \) and \( \sigma_n^2 \) can increase with \( n \) and will for a path-dependent \( \varphi_n(x_1:n) \) as the degeneracy problem suggests!
Stronger Convergence Results

- **Exponentially stability assumption.** For any $x_1, x'_1$

\[
\frac{1}{2} \int \left| p(x_n \mid y_{2:n}, X_1 = x_1) - p(x_n \mid y_{2:n}, X_1 = x'_1) \right| \, dx_n \leq \alpha^n \text{ for } |\alpha| < 1.
\]
Stronger Convergence Results

- **Exponentially stability assumption.** For any $x_1, x'_1$
  \[
  \frac{1}{2} \int \left| p \left( x_n \| y_{2:n}, X_1 = x_1 \right) - p \left( x_n \| y_{2:n}, X_1 = x'_1 \right) \right| \, dx_n \leq \alpha^n \text{ for } |\alpha| < 1.
  \]

- **Marginal distribution.** For $\varphi_n (x_{1:n}) = \varphi (x_n)$,
  \[
  \mathbb{E} \left[ |\hat{\varphi}_n - \varphi_n|^p \right]^{1/p} \leq \frac{C}{\sqrt{N}},
  \]
  \[
  \lim_{N \rightarrow \infty} \sqrt{N} (\hat{\varphi}_n - \varphi_n) \Rightarrow \mathcal{N} (0, \sigma_n^2) \text{ where } \sigma_n^2 \leq D,
  \]
  where $C$ and $D$ typically exponential in $\dim(X_n)$.
Stronger Convergence Results

- **Exponentially stability assumption.** For any $x_1, x_1'$
  $$
  \frac{1}{2} \int \left| p \left( x_n \mid y_{2:n}, X_1 = x_1 \right) - p \left( x_n \mid y_{2:n}, X_1 = x_1' \right) \right| \, dx_n \leq \alpha^n \text{ for } |\alpha| < 1.
  $$

- **Marginal distribution.** For $\varphi_n (x_1:n) = \varphi (x_n)$,
  $$
  \mathbb{E} \left[ \left| \hat{\varphi}_n - \varphi_n \right|^p \right]^{1/p} \leq \frac{C}{\sqrt{N}},
  $$
  $$
  \lim_{N \to \infty} \sqrt{N} (\hat{\varphi}_n - \varphi_n) \Rightarrow \mathcal{N} \left( 0, \sigma^2_n \right) \text{ where } \sigma^2_n \leq D,
  $$
  where $C$ and $D$ typically exponential in $\text{dim}(X_n)$.

- **Marginal likelihood.**
  $$
  \lim_{N \to \infty} \sqrt{N} (\log \hat{p} (y_{1:n}) - \log p (y_{1:n})) \Rightarrow \mathcal{N} \left( 0, \overline{\sigma}^2_n \right) \text{ with } \overline{\sigma}^2_n \leq A \, n.
  $$
Stronger Convergence Results

- **Exponentially stability assumption.** For any $x_1, x'_1$

  \[
  \frac{1}{2} \int \left| p \left( x_n \mid y_{2:n}, X_1 = x_1 \right) - p \left( x_n \mid y_{2:n}, X_1 = x'_1 \right) \right| \, dx_n \leq \alpha^n \text{ for } |\alpha| < 1.
  \]

- **Marginal distribution.** For $\varphi_n (x_{1:n}) = \varphi (x_n)$,

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

  \[
  \lim_{N \to \infty} \sqrt{N} (\hat{\varphi}_n - \varphi_n) \Rightarrow \mathcal{N} (0, \sigma_n^2) \text{ where } \sigma_n^2 \leq D,
  \]

  where $C$ and $D$ typically exponential in $\text{dim}(X_n)$.

- **Marginal likelihood.**

  \[
  \lim_{N \to \infty} \sqrt{N} \left( \log \hat{p} (y_{1:n}) - \log p (y_{1:n}) \right) \Rightarrow \mathcal{N} (0, \bar{\sigma}_n^2) \text{ with } \bar{\sigma}_n^2 \leq A \, n.
  \]

- **Resampling is necessary.** Without resampling, we have

  \[
  \log \hat{p} (y_{1:n}) = \log \frac{1}{N} \sum_{i=1}^{N} \prod_{k=1}^{n} g \left( y_k \mid \tilde{X}_k^{(i)} \right)
  \]

  which has a variance increasing exponentially with $n$ even for trivial examples.
Improving the Sampling Step

- **Bootstrap filter.** Very inefficient for vague prior/peaky likelihood; e.g. 
  \[ p ( x_{n-1} | y_{1:n-1} ) = \mathcal{N} ( x_{n-1}; m, \sigma^2 ) , f ( x_n | x_{n-1} ) = \mathcal{N} ( x_n; x_{n-1}, \sigma_v^2 ) \]
  and \[ g ( y_n | x_n ) = \mathcal{N} ( y_n; x_n, \sigma_w^2 ). \]
Improving the Sampling Step

- **Bootstrap filter.** Very inefficient for vague prior/peaky likelihood; e.g.
  \[ p(x_{n-1}|y_{1:n-1}) = \mathcal{N}(x_{n-1}; m, \sigma^2), \]
  \[ f(x_n|x_{n-1}) = \mathcal{N}(x_n; x_{n-1}, \sigma^2_v) \]
  and
  \[ g(y_n|x_n) = \mathcal{N}(y_n; x_n, \sigma^2_w). \]

- **Optimal proposal/Perfect adaptation.** Resample
  \[ W_n \propto p(y_n|x_{n-1}), \]
  sample
  \[ p(x_n|y_n, x_{n-1}) \propto g(y_n|x_n) f(x_n|x_{n-1}). \]
Improving the Sampling Step

- **Bootstrap filter.** Very inefficient for vague prior/peaky likelihood; e.g.
  \[ p(x_{n-1} | y_1:n-1) = \mathcal{N}(x_{n-1}; m, \sigma^2), \quad f(x_n | x_{n-1}) = \mathcal{N}(x_n; x_{n-1}, \sigma^2_V) \]
  and
  \[ g(y_n | x_n) = \mathcal{N}(y_n; x_n, \sigma^2_w). \]

- **Optimal proposal/Perfect adaptation.** Resample
  \[ W_n \propto p(y_n | x_{n-1}), \quad \text{sample} \quad p(x_n | y_n, x_{n-1}) \propto g(y_n | x_n) f(x_n | x_{n-1}). \]
Improving the Sampling Step

- **Bootstrap filter.** Very inefficient for vague prior/peaky likelihood; e.g.
  \[ p \left( x_{n-1} | y_1:n-1 \right) = \mathcal{N} \left( x_{n-1}; m, \sigma^2 \right), \quad f \left( x_n | x_{n-1} \right) = \mathcal{N} \left( x_n; x_{n-1}, \sigma_v^2 \right) \]
  and
  \[ g \left( y_n | x_n \right) = \mathcal{N} \left( y_n; x_n, \sigma_w^2 \right). \]

- **Optimal proposal/Perfect adaptation.** Resample
  \[ W_n \propto p \left( y_n | x_{n-1} \right), \]
  sample
  \[ p \left( x_n | y_n, x_{n-1} \right) \propto g \left( y_n | x_n \right) f \left( x_n | x_{n-1} \right). \]

**Figure:**

\[ p \left( x_n | y_1:n-1 \right) = \int f \left( x_n | x_{n-1} \right) p \left( x_{n-1} | y_1:n-1 \right) dx_{n-1} \text{ (blue)}, \]
\[ \int p \left( x_n | y_n, x_{n-1} \right) p \left( x_{n-1} | y_1:n-1 \right) dx_{n-1} \text{ (green)}, \]
\[ g \left( y_n | x_n \right) \text{ (red)}. \]
Various standard improvements

- **Approximate optimal proposal.** Design analytical approximation via EKF, UKF \( \hat{p}(x_n|y_n, x_{n-1}) \) of \( p(x_n|y_n, x_{n-1}) \). Sample \( \hat{p}(x_n|y_n, x_{n-1}) \) and set

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

  see also Auxiliary Particle Filters (Pitt & Shephard, 1999)
Various standard improvements

- **Approximate optimal proposal.** Design analytical approximation via EKF, UKF \( \hat{p}(x_n|y_n, x_{n-1}) \) of \( p(x_n|y_n, x_{n-1}) \). Sample \( \hat{p}(x_n|y_n, x_{n-1}) \) and set

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

see also Auxiliary Particle Filters (Pitt & Shephard, 1999)

- **Resample Move** (Gilks & Berzuini, 1999). After the resampling step, you have \( X_{1:n}^{(i)} = X_{1:n}^{(j)} \) for \( i \neq j \). To add diversity among particles, use an MCMC kernel \( X'_{1:n}^{(i)} \sim K_n \left( x_{1:n} | X_{1:n}^{(i)} \right) \) where

\[
p(x'_{1:n}|y_{1:n}) = \int p(x_{1:n}|y_{1:n}) K_n \left( x'_{1:n} | x_{1:n} \right) dx_{1:n}
\]

Here \( K_n \) does not have to be ergodic.
Improving the Resampling Step

- Resample $N$ times $X_{1:n}^{(i)} \sim \tilde{p}(x_{1:n} | y_{1:n}) = \sum_{i=1}^{N} W_n^{(i)} \delta_{\tilde{X}_{1:n}^{(i)}}(x_{1:n})$ to obtain $\hat{p}(x_{1:n} | y_{1:n})$ is called \textit{multinomial resampling} as

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

where $\{N_n^{(i)}\}$ follow a multinomial with $\mathbb{E} \left[ N_n^{(i)} \right] = NW_n^{(i)}$, $\mathbb{V} \left[ N_n^{(1)} \right] = NW_n^{(i)} \left( 1 - W_n^{(i)} \right)$. 
Improving the Resampling Step

- Resample $N$ times $X_{1:n}^{(i)} \sim \tilde{p}(x_{1:n} \mid y_{1:n}) = \sum_{i=1}^{N} W_n^{(i)} \delta_{\tilde{X}_{1:n}^{(i)}}(x_{1:n})$ to obtain $\hat{p}(x_{1:n} \mid y_{1:n})$ is called *multinomial resampling* as

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

where $\{N_n^{(i)}\}$ follow a multinomial with $\mathbb{E} \left[ N_n^{(i)} \right] = NW_n^{(i)}$, $\mathbb{V} \left[ N_n^{(1)} \right] = NW_n^{(i)} \left( 1 - W_n^{(i)} \right)$.

- Better resampling steps can be designed with $\mathbb{E} \left[ N_n^{(i)} \right] = NW_n^{(i)}$ but smaller $\mathbb{V} \left[ N_n^{(i)} \right]$; e.g. stratified resampling (Kitagawa, 1996).
Online Bayesian Parameter Estimation

- Assume we have
  \[ X_n \mid (X_{n-1} = x_{n-1}) \sim f_\theta (x_n \mid x_{n-1}), \]
  \[ Y_n \mid (X_n = x_n) \sim g_\theta (y_n \mid x_n), \]

  where \( \theta \) is an *unknown* static parameter with prior \( p(\theta) \).
Online Bayesian Parameter Estimation

- Assume we have
  \[ X_n \mid (X_{n-1} = x_{n-1}) \sim f_\theta (x_n \mid x_{n-1}), \]
  \[ Y_n \mid (X_n = x_n) \sim g_\theta (y_n \mid x_n), \]
  where \( \theta \) is an unknown static parameter with prior \( p(\theta) \).

- Given data \( y_{1:n} \), inference relies on
  \[ p(\theta, x_{1:n} \mid y_{1:n}) = p(\theta \mid y_{1:n}) p_\theta (x_{1:n} \mid y_{1:n}) \]
  where
  \[ p(\theta \mid y_{1:n}) \propto p_\theta (y_{1:n}) p(\theta). \]
Assume we have

\[
X_n \mid (X_{n-1} = x_{n-1}) \sim f_{\theta}(x_n \mid x_{n-1}), \\
Y_n \mid (X_n = x_n) \sim g_{\theta}(y_n \mid x_n),
\]

where \(\theta\) is an \textit{unknown} static parameter with prior \(p(\theta)\).

Given data \(y_{1:n}\), inference relies on

\[
p(\theta, x_{1:n} \mid y_{1:n}) = p(\theta \mid y_{1:n}) p_{\theta}(x_{1:n} \mid y_{1:n})
\]

where

\[
p(\theta \mid y_{1:n}) \propto p_{\theta}(y_{1:n}) p(\theta).
\]

SMC methods apply as it is a standard model with extended state

\(Z_n = (X_n, \theta_n)\) where

\[
f(z_n \mid z_{n-1}) = \delta_{\theta_{n-1}}(\theta_n) f_{\theta_n}(x_n \mid x_{n-1}), \quad g(y_n \mid z_n) = g_{\theta}(y_n \mid x_n).
\]
For fixed $\theta$, $\mathbb{V} \left[ \log \hat{p}_\theta (y_{1:n}) \right]$ is in $Cn/N$. In a Bayesian context, the problem is even more severe as $p(\theta|y_{1:n}) \propto p_\theta (y_{1:n}) p(\theta)$. Exponential stability assumption cannot hold as $\theta_n = \theta_1$. 
Cautionary Warning

- For fixed $\theta$, $\mathbb{V} [\log \hat{p}_{\theta} (y_{1:n})]$ is in $Cn/N$. In a Bayesian context, the problem is even more severe as $p(\theta|y_{1:n}) \propto p_{\theta} (y_{1:n}) p(\theta)$. Exponential stability assumption cannot hold as $\theta_n = \theta_1$.

- To mitigate but NOT solve the problem, introduce MCMC steps on $\theta$; e.g. (Andrieu, D.&D.,1999; Fearnhead, 1998, 2002; Gilks & Berzuini 1999,2001,2003; Storvik, 2002; Polson & Johannes, 2007; Vercauteren et al., 2005).
For fixed $\theta$, $\mathbb{V} \left[ \log \hat{p}_\theta(y_{1:n}) \right]$ is in $Cn/N$. In a Bayesian context, the problem is even more severe as $p(\theta|y_{1:n}) \propto p_\theta(y_{1:n}) p(\theta)$. Exponential stability assumption cannot hold as $\theta_n = \theta_1$.

To mitigate but **NOT** solve the problem, introduce MCMC steps on $\theta$; e.g. (Andrieu, D.&D.,1999; Fearnhead, 1998, 2002; Gilks & Berzuini 1999,2001,2003; Storvik, 2002; Polson & Johannes, 2007; Vercauteren et al., 2005).

When $p(\theta|y_{1:n}, x_{1:n}) = p(\theta|s_n(x_{1:n}, y_{1:n}))$ where $s_n(x_{1:n}, y_{1:n})$ is fixed-dimensional, this is an elegant algorithm but still relies on $\hat{p}(x_{1:n}|y_{1:n})$ so degeneracy will creep in.
Cautionary Warning

- For fixed $\theta$, $\forall \log \hat{p}_\theta (y_{1:n})$ is in $Cn/N$. In a Bayesian context, the problem is even more severe as $p(\theta | y_{1:n}) \propto p_\theta (y_{1:n}) p(\theta)$. Exponential stability assumption cannot hold as $\theta_n = \theta_1$.

- To mitigate but **NOT** solve the problem, introduce MCMC steps on $\theta$; e.g. (Andrieu, D.&D., 1999; Fearnhead, 1998, 2002; Gilks & Berzuini 1999, 2001, 2003; Storvik, 2002; Polson & Johannes, 2007; Vercauteren et al., 2005).

- When $p(\theta | y_{1:n}, x_{1:n}) = p(\theta | s_n (x_{1:n}, y_{1:n}))$ where $s_n (x_{1:n}, y_{1:n})$ is fixed-dimensional, this is an elegant algorithm but still relies on $\hat{p}(x_{1:n} | y_{1:n})$ so degeneracy will creep in.

- As $\dim(Z_n) = \dim(X_n) + \dim(\theta)$, such methods are not recommended for high-dimensional $\theta$, especially with vague priors.
Given at time $n - 1$, the approximation at time $n$

$$\hat{p}(\theta, x_{1:n-1} \mid y_{1:n-1}) = \frac{1}{N} \sum_{i=1}^{N} \delta(\theta^{(i)}_{n-1}, x^{(i)}_{1:n-1}) (\theta, x_{1:n-1}).$$
Example of SMC with MCMC for Parameter Estimation

- Given at time $n - 1$, the approximation at time $n$
  \[
  \hat{p}(\theta, x_{1:n-1} \mid y_{1:n-1}) = \frac{1}{N} \sum_{i=1}^{N} \delta(\theta^{(i)}_{n-1}, x^{(i)}_{1:n-1}) (\theta, x_{1:n-1}) .
  \]

- Sample $\tilde{X}^{(i)}_n \sim f_{\theta^{(i)}_{n-1}} (x_n \mid X^{(i)}_{n-1})$, set $\tilde{X}^{(i)}_{1:n} = (X^{(i)}_{1:n-1}, \tilde{X}^{(i)}_n)$ and
  \[
  \tilde{p}(\theta, x_{1:n} \mid y_{1:n}) = \sum_{i=1}^{N} W^{(i)}_n \delta(\theta^{(i)}_{n-1}, \tilde{X}^{(i)}_{1:n}) (x_{1:n}) ,
  W^{(i)}_n \propto g_{\theta^{(i)}_{n-1}} (y_n \mid \tilde{X}^{(i)}_n) .
  \]
Example of SMC with MCMC for Parameter Estimation

- Given at time $n - 1$, the approximation at time $n$

$$\hat{p}(\theta, x_{1:n-1} | y_{1:n-1}) = \frac{1}{N} \sum_{i=1}^{N} \delta(\theta_{n-1}^{(i)}, x_{1:n-1}^{(i)}) (\theta, x_{1:n-1}) .$$

- Sample $\tilde{X}_{n}^{(i)} \sim f_{\theta_{n-1}^{(i)}} ( x_{n} | X_{n-1}^{(i)} )$, set $\tilde{X}_{1:n}^{(i)} = (X_{1:n-1}^{(i)}, \tilde{X}_{n}^{(i)})$ and

$$\tilde{p}(\theta, x_{1:n} | y_{1:n}) = \sum_{i=1}^{N} W_{n}^{(i)} \delta(\theta_{n-1}^{(i)}, \tilde{X}_{1:n}^{(i)}) (x_{1:n}) , \ W_{n}^{(i)} \propto g_{\theta_{n-1}^{(i)}} (y_{n} | \tilde{X}_{n}^{(i)}) .$$

- Resample $X_{1:n}^{(i)} \sim \tilde{p}(x_{1:n} | y_{1:n})$ then sample $\theta_{n}^{(i)} \sim p(\theta | y_{1:n}, X_{1:n}^{(i)})$ to obtain

$$\hat{p}(\theta, x_{1:n} | y_{1:n}) = \frac{1}{N} \sum_{i=1}^{N} \delta(\theta_{n}^{(i)}, x_{1:n}^{(i)}) (\theta, x_{1:n}) .$$
SMC estimate of $\mathbb{E} [\theta | y_{1:n}]$, as $n$ increases the degeneracy creeps in.
Given data $y_{1:n}$, inference relies on

$$p(\theta, x_{1:n} | y_{1:n}) = p(\theta | y_{1:n}) p_{\theta}(x_{1:n} | y_{1:n})$$

where

$$p(\theta | y_{1:n}) \propto p_{\theta}(y_{1:n}) p(\theta).$$
Offline Bayesian Parameter Estimation

- Given data $y_{1:n}$, inference relies on
  \[ p(\theta, x_{1:n} \mid y_{1:n}) = p(\theta \mid y_{1:n}) p_\theta(x_{1:n} \mid y_{1:n}) \]
  where
  \[ p(\theta \mid y_{1:n}) \propto p_\theta(y_{1:n}) p(\theta). \]
- For a given $\theta$, SMC can estimate both $p_\theta(x_{1:n} \mid y_{1:n})$ and $p_\theta(y_{1:n})$. 
Offline Bayesian Parameter Estimation

- Given data $y_{1:n}$, inference relies on

$$p(\theta, x_{1:n} | y_{1:n}) = p(\theta | y_{1:n}) p_\theta(x_{1:n} | y_{1:n})$$

where

$$p(\theta | y_{1:n}) \propto p_\theta(y_{1:n}) p(\theta).$$

- For a given $\theta$, SMC can estimate both $p_\theta(x_{1:n} | y_{1:n})$ and $p_\theta(y_{1:n})$.

- Is it possible to use SMC within MCMC to sample from $p(\theta, x_{1:n} | y_{1:n})$?
To sample from a target $\pi(z)$, the MH sampler generates a Markov chain $\{Z^{(i)}\}$ according to the following mechanism. Given $Z^{(i-1)}$, propose a candidate $Z^* \sim q\left(z^* | Z^{(i-1)}\right)$ and with probability

$$\alpha\left(Z^{(i-1)}, Z^*\right) = \min\left(1, \frac{\pi(Z^*) q\left(Z^{(i-1)} | Z^*\right)}{\pi(Z^{(i-1)}) q\left(Z^* | Z^{(i-1)}\right)}\right)$$

set $Z^{(i)} = Z^*$, otherwise $Z^{(i)} = Z^{(i-1)}$. 
To sample from a target $\pi(z)$, the MH sampler generates a Markov chain $\{Z^{(i)}\}$ according to the following mechanism. Given $Z^{(i-1)}$, propose a candidate $Z^* \sim q\left(z^*|Z^{(i-1)}\right)$ and with probability

$$
\alpha \left( Z^{(i-1)}, Z^* \right) = \min \left( 1, \frac{\pi(Z^*) q\left(Z^{(i-1)}|Z^*\right)}{\pi(Z^{(i-1)}) q\left(Z^*|Z^{(i-1)}\right)} \right)
$$

set $Z^{(i)} = Z^*$, otherwise $Z^{(i)} = Z^{(i-1)}$.

It can be easily shown that

$$
\pi(z') = \int \pi(z) K(z'|z) \, dz
$$

where $K(z'|z)$ is the transition kernel of the MH and under weak assumptions $Z^{(i)} \sim \pi(z)$ as $i \to \infty$. 

Consider the following so-called marginal MH algorithm which target
\[
p(\theta, x_{1:n} \mid y_{1:n}) = p(\theta \mid y_{1:n}) p_\theta(x_{1:n} \mid y_{1:n})
\]
using the proposal
\[
q((x_{1:n}^*, \theta^*) \mid (x_{1:n}, \theta)) = q(\theta^* \mid \theta) p_{\theta^*}(x_{1:n}^* \mid y_{1:n}).
\]
Marginal Metropolis-Hastings Sampler

- Consider the following so-called marginal MH algorithm which targets
  \[ p(\theta, x_{1:n} \mid y_{1:n}) = p(\theta \mid y_{1:n}) p_\theta(x_{1:n} \mid y_{1:n}) \]
  using the proposal
  \[ q((x_{1:n}^*, \theta^*) \mid (x_{1:n}, \theta)) = q(\theta^* \mid \theta) p_{\theta^*}(x_{1:n}^* \mid y_{1:n}). \]
- The MH acceptance probability is
  \[
  \min \left( 1, \frac{p(\theta^*, x_{1:n}^* \mid y_{1:n}) q((x_{1:n}, \theta) \mid (x_{1:n}^*, \theta^*)))}{p(\theta, x_{1:n} \mid y_{1:n}) q((x_{1:n}^*, \theta^*) \mid (x_{1:n}, \theta))} \right) \\
  = \min \left( 1, \frac{p_{\theta^*}(y_{1:n}) p(\theta^*) q(\theta \mid \theta^*)}{p_{\theta}(y_{1:n}) p(\theta) q(\theta^* \mid \theta)} \right)
  \]
Marginal Metropolis-Hastings Sampler

- Consider the following so-called marginal MH algorithm which target

\[ p(\theta, x_{1:n} | y_{1:n}) = p(\theta | y_{1:n}) p(\theta, x_{1:n} | y_{1:n}) \]

using the proposal

\[ q((x_{1:n}^*, \theta^*) | (x_{1:n}, \theta)) = q(\theta^* | \theta) p_{\theta^*}(x_{1:n}^* | y_{1:n}). \]

- The MH acceptance probability is

\[
\frac{\min\left(1, \frac{p(\theta^*, x_{1:n}^* | y_{1:n})}{p(\theta, x_{1:n} | y_{1:n})} \frac{q((x_{1:n}^*, \theta) | (x_{1:n}, \theta^*))}{q((x_{1:n}^*, \theta^*) | (x_{1:n}, \theta))}\right)}{p_{\theta^*}(y_{1:n}) p(\theta^*) q(\theta^* | \theta)} = \min\left(1, \frac{p_{\theta^*}(y_{1:n})}{p_{\theta}(y_{1:n})} \frac{p(\theta^*) q(\theta | \theta^*)}{q(\theta^* | \theta)}\right)
\]

- **Problem:** We do not know \( p_{\theta}(y_{1:n}) \) analytically and cannot sample from \( p_{\theta}(x_{1:n} | y_{1:n}) \) so this algorithm cannot be implemented.
Consider the following so-called marginal MH algorithm which target
\[ p(\theta, x_{1:n} | y_{1:n}) = p(\theta | y_{1:n}) p_\theta(x_{1:n} | y_{1:n}) \]
using the proposal
\[ q((x_{1:n}^*, \theta^*) | (x_{1:n}, \theta)) = q(\theta^* | \theta) p_{\theta^*}(x_{1:n}^* | y_{1:n}). \]

The MH acceptance probability is
\[
\min \left( 1, \frac{p(\theta^*, x_{1:n}^* | y_{1:n}) q((x_{1:n}, \theta) | (x_{1:n}^*, \theta^*))}{p(\theta, x_{1:n} | y_{1:n}) q((x_{1:n}^*, \theta^*) | (x_{1:n}, \theta))} \right)
\]
\[= \min \left( 1, \frac{p_{\theta^*}(y_{1:n}) p(\theta^*) q(\theta | \theta^*)}{p_\theta(y_{1:n}) p(\theta) q(\theta^* | \theta)} \right)\]

**Problem**: We do not know \( p_\theta(y_{1:n}) \) analytically and cannot sample from \( p_\theta(x_{1:n} | y_{1:n}) \) so this algorithm cannot be implemented.

**“Idea”**: Use SMC approximations of \( p_\theta(x_{1:n} | y_{1:n}) \) and \( p_\theta(y_{1:n}) \).
At iteration $i$, given $\{\theta (i-1), X_{1:n}(i-1), \hat{p}_{\theta(i-1)}(y_{1:n})\}$ then sample $\theta^* \sim q(\theta | \theta(i-1))$, run an SMC algorithm to obtain $\hat{p}_{\theta^*}(x_{1:n} | y_{1:n})$ and $\hat{p}_{\theta^*}(y_{1:n})$. 
At iteration $i$, given $\{\theta (i - 1), X_{1:n} (i - 1), \hat{p}_{\theta(i-1)} (y_{1:n})\}$ then sample $\theta^* \sim q (\theta | \theta (i - 1))$, run an SMC algorithm to obtain $\hat{p}_{\theta^*} (x_{1:n} | y_{1:n})$ and $\hat{p}_{\theta^*} (y_{1:n})$.

Sample $X_{1:n}^* \sim \hat{p}_{\theta^*} (x_{1:n} | y_{1:n})$. 
Particle Marginal MH Sampler

- At iteration $i$, given $\{\theta(i-1), X_{1:n}(i-1), \hat{p}_{\theta(i-1)}(y_{1:n})\}$ then sample $\theta^* \sim q(\theta|\theta(i-1))$, run an SMC algorithm to obtain $\hat{p}_{\theta^*}(x_{1:n}|y_{1:n})$ and $\hat{p}_{\theta^*}(y_{1:n})$.
- Sample $X_{1:n}^* \sim \hat{p}_{\theta^*}(x_{1:n}|y_{1:n})$.
- With probability

$$\min \left(1, \frac{\hat{p}_{\theta^*}(y_{1:n}) p(\theta^*)}{\hat{p}_{\theta(i-1)}(y_{1:n}) p(\theta(i-1))} \frac{q(\theta(i-1)|\theta^*)}{q(\theta^*|\theta(i-1))} \right)$$

set $\{\theta(i), X_{1:n}(i), \hat{p}_{\theta(i)}(y_{1:n})\} = \{\theta^*, X_{1:n}^*, \hat{p}_{\theta^*}(y_{1:n})\}$ otherwise set $\{\theta(i), X_{1:n}(i), \hat{p}_{\theta(i)}(y_{1:n})\} = \{\theta(i-1), X_{1:n}(i-1), \hat{p}_{\theta(i-1)}(y_{1:n})\}$. 
This algorithm (without sampling $X_{1:n}$) was proposed as an approximate MCMC algorithm to sample from $p(\theta | y_{1:n})$ in (Fernandez-Villaverde & Rubio-Ramirez, 2007).
This algorithm (without sampling $X_{1:n}$) was proposed as an approximate MCMC algorithm to sample from $p(\theta|y_{1:n})$ in (Fernandez-Villaverde & Rubio-Ramirez, 2007).

Whatever being $N \geq 1$, this algorithm admits exactly $p(\theta, x_{1:n}|y_{1:n})$ as invariant distribution (Andrieu, D. & Holenstein, 2010). A particle version of the Gibbs sampler also exists.
This algorithm (without sampling $X_{1:n}$) was proposed as an approximate MCMC algorithm to sample from $p(\theta|y_{1:n})$ in (Fernandez-Villaverde & Rubio-Ramirez, 2007).

Whatever being $N \geq 1$, this algorithm admits exactly $p(\theta, x_{1:n}|y_{1:n})$ as invariant distribution (Andrieu, D. & Holenstein, 2010). A particle version of the Gibbs sampler also exists.

The higher $N$, the better the performance of the algorithm: $N$ scales roughly linearly with $n$. 
Validity of the Particle Marginal MH Sampler

- This algorithm (without sampling $X_{1:n}$) was proposed as an approximate MCMC algorithm to sample from $p(\theta | y_{1:n})$ in (Fernandez-Villaverde & Rubio-Ramirez, 2007).
- Whatever being $N \geq 1$, this algorithm admits exactly $p(\theta, x_{1:n} | y_{1:n})$ as invariant distribution (Andrieu, D. & Holenstein, 2010). A particle version of the Gibbs sampler also exists.
- The higher $N$, the better the performance of the algorithm: $N$ scales roughly linearly with $n$.
- Particularly useful in scenarios where $X_n$ moderate dimensional & $\theta$ high dimensional. Admits the plug and play property (Ionides et al., 2006).
Two species $X_t^1$ (prey) and $X_t^2$ (predator)

Pr \( \{ X_{t+dt}^1 = x_t^1 + 1, X_{t+dt}^2 = x_t^2 \mid x_t^1, x_t^2 \} \) = $\alpha x_t^1 dt + o(dt)$,

Pr \( \{ X_{t+dt}^1 = x_t^1 - 1, X_{t+dt}^2 = x_t^2 + 1 \mid x_t^1, x_t^2 \} \) = $\beta x_t^1 x_t^2 dt + o(dt)$,

Pr \( \{ X_{t+dt}^1 = x_t^1, X_{t+dt}^2 = x_t^2 - 1 \mid x_t^1, x_t^2 \} \) = $\gamma x_t^2 dt + o(dt)$,

observed at discrete times

\[ Y_n = X_{n\Delta}^1 + W_n \text{ with } W_n \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2). \]
Inference for Stochastic Kinetic Models

- Two species $X^1_t$ (prey) and $X^2_t$ (predator)

\[
\begin{align*}
\Pr \left( X^1_{t+dt} = x^1_t + 1, X^2_{t+dt} = x^2_t \mid x^1_t, x^2_t \right) &= \alpha x^1_t dt + o(dt), \\
\Pr \left( X^1_{t+dt} = x^1_t - 1, X^2_{t+dt} = x^2_t + 1 \mid x^1_t, x^2_t \right) &= \beta x^1_t x^2_t dt + o(dt), \\
\Pr \left( X^1_{t+dt} = x^1_t, X^2_{t+dt} = x^2_t - 1 \mid x^1_t, x^2_t \right) &= \gamma x^2_t dt + o(dt),
\end{align*}
\]

observed at discrete times

\[Y_n = X^1_{n\Delta} + W_n \text{ with } W_n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2) .\]

- We are interested in the kinetic rate constants $\theta = (\alpha, \beta, \gamma)$ a priori distributed as (Boys et al., 2008)

\[\alpha \sim \mathcal{G}(1, 10), \quad \beta \sim \mathcal{G}(1, 0.25), \quad \gamma \sim \mathcal{G}(1, 7.5).\]
Inference for Stochastic Kinetic Models

- Two species $X^1_t$ (prey) and $X^2_t$ (predator)

$$\Pr\left( X^1_{t+dt} = x^1_t + 1, X^2_{t+dt} = x^2_t \mid x^1_t, x^2_t \right) = \alpha x^1_t dt + o(dt),$$
$$\Pr\left( X^1_{t+dt} = x^1_t - 1, X^2_{t+dt} = x^2_t + 1 \mid x^1_t, x^2_t \right) = \beta x^1_t x^2_t dt + o(dt),$$
$$\Pr\left( X^1_{t+dt} = x^1_t, X^2_{t+dt} = x^2_t - 1 \mid x^1_t, x^2_t \right) = \gamma x^2_t dt + o(dt),$$

observed at discrete times

$$Y_n = X^1_{n\Delta} + W_n \text{ with } W_n \sim \mathcal{N}(0, \sigma^2).$$

- We are interested in the kinetic rate constants $\theta = (\alpha, \beta, \gamma)$ a priori distributed as (Boys et al., 2008)

$$\alpha \sim \mathcal{G}(1, 10), \quad \beta \sim \mathcal{G}(1, 0.25), \quad \gamma \sim \mathcal{G}(1, 7.5).$$

- MCMC methods require reversible jumps, PMMH requires only forward simulation.
Experimental Results

Simulated data

Posterior distributions

α

β

γ
SMC Fixed-Lag Smoothing Approximation

- Direct SMC approximations of $p(x_{1:n} \mid y_{1:n})$ and its marginals $p(x_k \mid y_{1:n})$ gets poorer as $n \nearrow$. 
SMC Fixed-Lag Smoothing Approximation

- Direct SMC approximations of \( p(x_{1:n} \mid y_{1:n}) \) and its marginals \( p(x_k \mid y_{1:n}) \) gets poorer as \( n \uparrow \).
- The fixed-lag smoothing approximation (Kitagawa & Sato, 2001) relies on

\[
p(x_{1:k} \mid y_{1:n}) \approx p(x_{1:k} \mid y_{1:k+\Delta}) \text{ for } \Delta \text{ large enough.}
\]
Direct SMC approximations of $p(\mathbf{x}_{1:n}|\mathbf{y}_{1:n})$ and its marginals $p(\mathbf{x}_k|\mathbf{y}_{1:n})$ gets poorer as $n \uparrow$.

The fixed-lag smoothing approximation (Kitagawa & Sato, 2001) relies on

$$p(\mathbf{x}_{1:k}|\mathbf{y}_{1:n}) \approx p(\mathbf{x}_{1:k}|\mathbf{y}_{1:k+\Delta})$$

for $\Delta$ large enough.

**Algorithmically**: stop resampling $\left\{ \mathbf{X}_k^{(i)} \right\}$ beyond time $k + \Delta$. 
SMC Fixed-Lag Smoothing Approximation

- Direct SMC approximations of $p(x_{1:n} | y_{1:n})$ and its marginals $p(x_k | y_{1:n})$ gets poorer as $n \uparrow$.
- The fixed-lag smoothing approximation (Kitagawa & Sato, 2001) relies on

$$p(x_{1:k} | y_{1:n}) \approx p(x_{1:k} | y_{1:k+\Delta})$$

for $\Delta$ large enough.

- **Algorithmically**: stop resampling $\left\{ X_k^{(i)} \right\}$ beyond time $k + \Delta$.
- Computational cost is $O(Nn)$ but non-vanishing bias as $N \rightarrow \infty$ (Olsson & al., 2006).
SMC Fixed-Lag Smoothing Approximation

- Direct SMC approximations of \( p(\mathbf{x}_{1:n} \mid \mathbf{y}_{1:n}) \) and its marginals \( p(\mathbf{x}_k \mid \mathbf{y}_{1:n}) \) gets poorer as \( n \to \infty \).
- The fixed-lag smoothing approximation (Kitagawa & Sato, 2001) relies on

  \[
p(\mathbf{x}_{1:k} \mid \mathbf{y}_{1:n}) \approx p(\mathbf{x}_{1:k} \mid \mathbf{y}_{1:k+\Delta}) \text{ for } \Delta \text{ large enough.}
  \]

- **Algorithmically**: stop resampling \( \{X_k^{(i)}\} \) beyond time \( k + \Delta \).
- Computational cost is \( \mathcal{O}(Nn) \) but non-vanishing bias as \( N \to \infty \) (Olsson & al., 2006).
- Picking \( \Delta \) is difficult. \( \Delta \) too small results in \( p(\mathbf{x}_{1:k} \mid \mathbf{y}_{1:k+\Delta}) \) being a poor approximation of \( p(\mathbf{x}_{1:k} \mid \mathbf{y}_{1:n}) \). \( \Delta \) too large improves the approximation but particle degeneracy creeps in.
**Forward filtering Backward smoothing** (FFBS).

\[
p(x_k \mid y_1:n) = \int p(x_{k+1} \mid y_1:n) \frac{f(x_{k+1} \mid x_k) p(x_k \mid y_{1:k})}{p(x_{k+1} \mid y_{1:n})} \, dx_{k+1}
\]

- smoother at \(k\)
- smoother at \(k+1\)
- filter at \(k\)
- backward transition \(p(x_k \mid y_{1:n}, x_{k+1})\)
SMC Forward Filtering Backward Smoothing

- **Forward filtering Backward smoothing (FFBS).**

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

- **SMC Implementation:** For \( k = 1, \ldots, n \), compute \( \hat{p}(x_k | y_{1:k}) \). For \( k = n - 1, \ldots, 1 \), compute \( \hat{p}(x_k | y_{1:n}) = \sum_{i=1}^{N} W_{k|n}^{(i)} \delta X_{k}^{(i)}(x_k) \) with cost \( O(N^2 n) \) using

\[
W_{k|n}^{(i)} = \sum_{j=1}^{N} W_{k+1|n}^{(j)} \frac{f(X_{k+1}^{(j)} | X_{k}^{(i)})}{\sum_{l=1}^{N} f(X_{k+1}^{(j)} | X_{k}^{(l)})}.
\]
SMC Forward Filtering Backward Smoothing

- **Forward filtering Backward smoothing (FFBS).**

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

- **SMC Implementation:** For \( k = 1, \ldots, n \), compute \( \hat{p}(x_k | y_{1:k}) \). For \( k = n-1, \ldots, 1 \), compute \( \hat{p}(x_k | y_{1:n}) = \sum_{i=1}^N W_{k|n}^{(i)} \delta_{X_k^{(i)}}(x_k) \) with cost \( \mathcal{O}(N^2n) \) using

\[
W_{k|n}^{(i)} = \sum_{j=1}^N W_{k+1|n}^{(j)} \frac{f(X_{k+1}^{(j)} | X_k^{(i)})}{\sum_{l=1}^N f(X_{k+1}^{(j)} | X_k^{(l)})}.
\]

- For \( \varphi_n(x_{1:n}) = \sum_{k=1}^{n-1} s_k(x_k, x_{k+1}) \), the SMC FFBS estimates \( \{\hat{\varphi}_n\} \) can be computed online exactly (Del Moral, D. & Singh, 2009).
SMC Forward Filtering Backward Smoothing

- **Forward filtering Backward smoothing (FFBS).**

\[
p(x_k | y_{1:n}) = \int p(x_{k+1} | y_{1:n}) \left[ f(x_{k+1} | x_k) p(x_k | y_{1:k}) / p(x_{k+1} | y_{1:n}) \right] \ dx_{k+1}
\]

\[
\text{smoother at } k + 1
\]

\[
\text{filter at } k
\]

\[
\text{backward transition } p(x_k | y_{1:n}, x_{k+1})
\]

- **SMC Implementation:** For \( k = 1, \ldots, n \), compute \( \hat{p}(x_k | y_{1:k}) \). For \( k = n - 1, \ldots, 1 \), compute \( \hat{p}(x_k | y_{1:n}) = \sum_{i=1}^{N} W^{(i)}_{k|n} \delta_{x^{(i)}_k} (x_k) \) with cost \( O(N^2 n) \) using

\[
W^{(i)}_{k|n} = \sum_{j=1}^{N} W^{(j)}_{k+1|n} \frac{f(X^{(j)}_{k+1} | X^{(i)}_k)}{\sum_{l=1}^{N} f(X^{(j)}_{k+1} | X^{(l)}_k)}.
\]

For \( \varphi_n (x_{1:n}) = \sum_{k=1}^{n-1} s_k (x_k, x_{k+1}) \), the SMC FFBS estimates \( \{\hat{\varphi}_n\} \) can be computed online exactly (Del Moral, D. & Singh, 2009).

- Sampling from \( \hat{p}(x_{1:n} | y_{1:n}) \) costs \( O(Nn) \) (Godsill, D. & West, 2004) but \( O(n) \) through rejection sampling (Douc et al., 2009).
SMC Generalized Two-Filter Smoothing

- **Generalized Two-Filter smoothing (TFS)**

\[ p(x_k, x_{k+1} | y_1:n) \propto \frac{p(x_k | y_{1:k}) f(x_{k+1} | x_k)}{\bar{p}(x_{k+1})} \cdot \frac{\bar{p}(x_{k+1} | y_{k+1:n})}{p(x_{k+1} | y_{k+1:n})}, \]

\[ \bar{p}(x_{k+1} | y_{k+1:n}) \propto p(y_{k+1:n} | x_{k+1}) \bar{p}(x_{k+1}). \]
SMC Generalized Two-Filter Smoothing

- **Generalized Two-Filter smoothing (TFS)**

\[ p(x_k, x_{k+1} \mid y_1:n) \propto p(x_k \mid y_1:k) f(x_{k+1} \mid x_k) \frac{\bar{p}(x_{k+1} \mid y_{k+1:n})}{\bar{p}(x_{k+1})}, \]

\[ \bar{p}(x_{k+1} \mid y_{k+1:n}) \propto p(y_{k+1:n} \mid x_{k+1}) \bar{p}(x_{k+1}). \]

- **SMC Implementation**: For \( k = 1, \ldots, n \), compute \( \hat{p}(x_k \mid y_1:k) \). For \( k = n, \ldots, 1 \), compute \( \hat{p}(x_{k+1} \mid y_{k+1:n}) \). Combine the forward and backward filters to obtain

\[ \hat{p}(x_k, x_{k+1} \mid y_1:n) \propto \hat{p}(x_k \mid y_1:k) \frac{f(x_{k+1} \mid x_k)}{\bar{p}(x_{k+1})} \hat{p}(x_{k+1} \mid y_{k+1:n}) \]
SMC Generalized Two-Filter Smoothing

- **Generalized Two-Filter smoothing (TFS)**

\[
p(x_k, x_{k+1} \mid y_1:n) \propto \underbrace{p(x_k \mid y_{1:k})}_{\text{forward filter}} \ f(x_{k+1} \mid x_k) \underbrace{\bar{p}(x_{k+1} \mid y_{k+1:n})}_{\text{generalized backward filter}} \underbrace{\bar{p}(x_k)}_{\text{artificial prior}},
\]

\[
\bar{p}(x_{k+1} \mid y_{k+1:n}) \propto p(y_{k+1:n} \mid x_k) \bar{p}(x_k) .
\]

- **SMC Implementation**: For \( k = 1, \ldots, n \), compute \( \hat{p}(x_k \mid y_{1:k}) \). For \( k = n, \ldots, 1 \), compute \( \hat{p}(x_{k+1} \mid y_{k+1:n}) \). Combine the forward and backward filters to obtain

\[
\hat{p}(x_k, x_{k+1} \mid y_1:n) \propto \hat{p}(x_k \mid y_{1:k}) \ \frac{f(x_{k+1} \mid x_k)}{\bar{p}(x_{k+1})} \hat{p}(x_{k+1} \mid y_{k+1:n})
\]

- **Cost**: \( O(N^2n) \) but \( O(Nn) \) through rejection sampling (Briers, D. & Maskell, 2008) and importance sampling (Fearnhead, Wynell & Tawn, 2008; Briers, D. & Singh, 2005).
Convergence Results

- **Exponentially stability assumption.** For any $x_1, x'_1$

\[
\frac{1}{2} \int \left| p \left( x_n \mid y_{2:n}, X_1 = x_1 \right) - p \left( x_n \mid y_{2:n}, X_1 = x'_1 \right) \right| \, dx_n \leq \alpha^n \text{ for } |\alpha| < 1.
\]
Convergence Results

- **Exponentially stability assumption.** For any $x_1, x'_1$

$$\frac{1}{2} \int | p(x_n | y_{2:n}, X_1 = x_1) - p(x_n | y_{2:n}, X_1 = x'_1) | \, dx_n \leq \alpha^n \text{ for } |\alpha| < 1.$$ 

- **Additive functionals.** If $\varphi_n(x_{1:n}) = \sum_{k=1}^{n} \varphi(x_k)$, we have for the standard path-based SMC estimate (Poyiadjis, D. & Singh, 2009)

$$\lim_{N \to \infty} \sqrt{N} (\hat{\varphi}_n - \bar{\varphi}_n) \Rightarrow \mathcal{N} (0, \sigma^2_n) \text{ where } \underline{\sigma}_n^2 \leq \sigma^2_n \leq \overline{\sigma}_n^2.$$ 

For the FFBS and TFS estimates (Douc et al., 2009; Del Moral, D. & Singh, 2009), we have

$$\lim_{N \to \infty} \sqrt{N} (\hat{\varphi}_n - \bar{\varphi}_n) \Rightarrow \mathcal{N} (0, \sigma^2_n) \text{ where } \sigma^2_n \leq C_n$$
Convergence Results

- **Exponentially stability assumption.** For any $x_1, x'_1$

\[
\frac{1}{2} \int \left| p \left( x_n \mid y_{2:n}, X_1 = x_1 \right) - p \left( x_n \mid y_{2:n}, X_1 = x'_1 \right) \right| \, dx_n \leq \alpha^n \text{ for } |\alpha| < 1.
\]

- **Additive functionals.** If $\varphi_n (x_{1:n}) = \sum_{k=1}^{n} \varphi (x_k)$, we have for the standard path-based SMC estimate (Poyiadjis, D. & Singh, 2009)

\[
\lim_{N \to \infty} \sqrt{N} \left( \hat{\varphi}_n - \bar{\varphi}_n \right) \Rightarrow \mathcal{N} \left( 0, \sigma_n^2 \right) \text{ where } \underline{A}n^2 \leq \sigma_n^2 \leq \overline{A}n^2.
\]

For the FFBS and TFS estimates (Douc et al., 2009; Del Moral, D. & Singh, 2009), we have

\[
\lim_{N \to \infty} \sqrt{N} \left( \hat{\varphi}_n - \bar{\varphi}_n \right) \Rightarrow \mathcal{N} \left( 0, \sigma_n^2 \right) \text{ where } \sigma_n^2 \leq \sigma_n^2 \leq Cn\]

- **Tradeoff between computational and statistical efficiency.**
Consider a linear Gaussian model

\[
X_1 \sim \mathcal{N} \left( 0, \frac{\sigma^2}{1 - \phi^2} \right) \quad \text{and} \quad X_k = \phi X_{k-1} + \sigma_V V_k, \quad V_k \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)
\]

\[
Y_k = cX_k + \sigma_W W_k, \quad W_k \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1).
\]
Consider a linear Gaussian model

\[ X_1 \sim \mathcal{N}\left(0, \frac{\sigma^2}{1 - \phi^2}\right) \quad \text{and} \quad X_k = \phi X_{k-1} + \sigma_V V_k, \quad V_k \text{ i.i.d. } \mathcal{N}(0, 1) \]

\[ Y_k = cX_k + \sigma_W W_k, \quad W_k \text{ i.i.d. } \mathcal{N}(0, 1). \]

We simulate 10,000 observations for
\[ \theta = (\phi, \sigma_V, c, \sigma_W) = (0.8, 0.5, 1.0, 1.0). \]
Experimental Results

Consider a linear Gaussian model

\[ X_1 \sim \mathcal{N} \left( 0, \frac{\sigma^2}{1 - \phi^2} \right) \quad \text{and} \quad X_k = \phi X_{k-1} + \sigma_V V_k, \quad V_k \overset{\text{i.i.d.}}{\sim} \mathcal{N} (0, 1) \]

\[ Y_k = cX_k + \sigma_W W_k, \quad W_k \overset{\text{i.i.d.}}{\sim} \mathcal{N} (0, 1). \]

We simulate 10,000 observations for \( \theta = (\phi, \sigma_V, c, \sigma_W) = (0.8, 0.5, 1.0, 1.0) \).

We compute the score vector using Fisher’s identity

\[ \nabla \log p_\theta (y_{1:n}) = \int \nabla \log p_\theta (x_{1:n}, y_{1:n}) \ p_\theta (x_{1:n} | y_{1:n}) \ dx_{1:n} \]

at the true value of \( \theta \) and compare to its true value.
Empirical Variance for Standard vs FFBS Approximations

Standard path-based (left) vs FFBS (right); the vertical scale is different
**Gradient ascent**: To maximise \( p_\theta(y_{1:n}) \) w.r.t \( \theta \), use at iteration \( k + 1 \)

\[
\theta_{k+1} = \theta_k + \nabla \log p_\theta(y_{1:n})|_{\theta=\theta_k}
\]

where \( \nabla \log p_\theta(y_{1:n})|_{\theta=\theta_k} \) is computed using Fisher’s identity or IPA (Coquelin, Deguest & Munos, 2009) and any SMC smoothing algorithm.
Parameter Estimation using Gradient Ascent/EM

- **Gradient ascent**: To maximise $p_\theta(y_{1:n})$ w.r.t $\theta$, use at iteration $k + 1$

$$ \theta_{k+1} = \theta_k + \nabla \log p_\theta(y_{1:n})|_{\theta=\theta_k} $$

where $\nabla \log p_\theta(y_{1:n})|_{\theta=\theta_k}$ is computed using Fisher’s identity or IPA (Coquelin, Deguest & Munos, 2009) and any SMC smoothing algorithm.

- **EM algorithm**: To maximise $p_\theta(y_{1:n})$ w.r.t $\theta$, the EM uses at iteration $k + 1$

$$ \theta_{k+1} = \arg \max Q(\theta_k, \theta). $$

where

$$ Q(\theta_k, \theta) = \int \log p_\theta(x_{1:n}, y_{1:n}) \ p_{\theta_k}(x_{1:n}|y_{1:n}) \ dx_{1:n} $$

can be computed using any SMC smoothing algorithm.
In the online implementation (Le Gland & Mevel, 1997), update the parameter at time $n + 1$ using

$$\theta_{n+1} = \theta_n + \gamma_{n+1} \nabla \log p_{\theta_1:n}(y_n|y_{1:n-1})$$

where $\sum_n \gamma_n = \infty$, $\sum_n \gamma_n^2 < \infty$ and

$$\nabla \log p_{\theta_1:n}(y_n|y_{1:n-1}) = \nabla \log p_{\theta_1:n}(y_{1:n}) - \nabla \log p_{\theta_1:n-1}(y_{1:n-1}).$$
In the online implementation (Le Gland & Mevel, 1997), update the parameter at time $n + 1$ using

$$
\theta_{n+1} = \theta_n + \gamma_{n+1} \nabla \log p_{\theta_1:n}(y_n|y_{1:n-1})
$$

where $\sum_n \gamma_n = \infty$, $\sum_n \gamma_n^2 < \infty$ and

$$
\nabla \log p_{\theta_1:n}(y_n|y_{1:n-1}) = \nabla \log p_{\theta_1:n}(y_{1:n}) - \nabla \log p_{\theta_1:n-1}(y_{1:n-1}).
$$

An estimate of $\nabla \log p_{\theta_1:n}(y_n|y_{1:n-1})$ with a time-uniform bounded variance can be computed using online SMC FFBS estimate (Del Moral, D. & Singh, 2009).
In the online implementation (Le Gland & Mevel, 1997), update the parameter at time $n + 1$ using

$$\theta_{n+1} = \theta_n + \gamma_{n+1} \nabla \log p_{\theta_{1:n}}(y_n|y_{1:n-1})$$

where $\sum_n \gamma_n = \infty$, $\sum_n \gamma_n^2 < \infty$ and

$$\nabla \log p_{\theta_{1:n}}(y_n|y_{1:n-1}) = \nabla \log p_{\theta_{1:n}}(y_{1:n}) - \nabla \log p_{\theta_{1:n-1}}(y_{1:n-1}).$$

An estimate of $\nabla \log p_{\theta_{1:n}}(y_n|y_{1:n-1})$ with a time-uniform bounded variance can be computed using online SMC FFBS estimate (Del Moral, D. & Singh, 2009).

A numerically stable SMC implementation of online EM (e.g. Cappé, 2009; Elliott, Ford & Moore, 2002) can also be implemented using online SMC FFBS estimate.
Online Parameter Estimation using Gradient Ascent/EM

- In the online implementation (Le Gland & Mevel, 1997), update the parameter at time $n + 1$ using

$$\theta_{n+1} = \theta_n + \gamma_{n+1} \nabla \log p_{\theta_1:n} (y_n | y_{1:n-1})$$

where $\sum_n \gamma_n = \infty$, $\sum_n \gamma_n^2 < \infty$ and

$$\nabla \log p_{\theta_1:n} (y_n | y_{1:n-1}) = \nabla \log p_{\theta_1:n} (y_{1:n}) - \nabla \log p_{\theta_1:n-1} (y_{1:n-1}).$$

- An estimate of $\nabla \log p_{\theta_1:n} (y_n | y_{1:n-1})$ with a time-uniform bounded variance can be computed using online SMC FFBS estimate (Del Moral, D. & Singh, 2009).

- A numerically stable SMC implementation of online EM (e.g. Cappé, 2009; Elliott, Ford & Moore, 2002) can also be implemented using online SMC FFBS estimate.

- These non-Bayesian procedures do not suffer from the degeneracy problem but require long data sets for convergence.
Tutorial overview

- **Introduction** Nando – 10min
- **Part I** Arnaud – 50min
  - Monte Carlo
  - Sequential Monte Carlo
  - Theoretical convergence
  - Improved particle filters
  - Online Bayesian parameter estimation
  - Particle MCMC
  - Smoothing
  - Gradient based online parameter estimation
- **Break** 15min
- **Part II** NdF – 45 min
  - Beyond state space models
  - Eigenvalue problems
  - Diffusion, protein folding & stochastic control
  - Time-varying Pitman-Yor Processes
  - SMC for static distributions
  - Boltzmann distributions & ABC
Sequential Monte Carlo (recap)

\[ P(X_0)P(X_1|X_0)P(Y_1|X_1)P(X_2|X_1)P(Y_2|X_2)P(X_3|X_2)P(Y_3|X_3) / P(X_0:3|Y_1:3) \]
SMC methods can be used to sample approximately from any sequence of growing distributions \( f_{\frac{1}{n}} g_{n-1} \)

\[
\frac{1}{n} (x_{1:n}) = \frac{f_n(x_{1:n})}{Z_n}
\]

where

\[
\begin{align*}
\{ f_n : X^n &\to R^+ \text{ is known point-wise.} \\
\{ Z_n &= \int f_n(x_{1:n}) dx_{1:n}
\end{align*}
\]

We introduce a proposal distribution \( q_n (x_{1:n}) \) to approximate \( Z_n \):

\[
Z_n = \frac{f_n(x_{1:n})}{q_n(x_{1:n})} q_n(x_{1:n}) dx_{1:n} = W_n(x_{1:n}) q_n(x_{1:n}) dx_{1:n}
\]
Let us construct the proposal sequentially: Introduce $q_n(x_{n|1:n-1})$ to sample component $X_n$ given $X_{1:n-1} = x_{1:n-1}$.

Then the importance weight becomes:

$$W_n = W_{n-1} \frac{f_n(x_{1:n})}{f_{n-1}(x_{1:n-1}) q_n(x_{n|1:n-1})}$$
SMC algorithm

1. Initialize at time \( n = 1 \)

2. At time \( n \geq 2 \)

   2.1 Sample \( \bar{X}_{n}^{(i)} \to q_{n} \) and augment \( \bar{X}_{1:n}^{(i)} = X_{1:n_{i}}^{(i)} \)

   2.2 Compute the sequential weight

\[
\frac{W_{n}^{(i)}}{f_{n_{i}} \bar{X}_{1:n_{i}}^{(i)} q_{n} \bar{X}_{n}^{(i)} - \bar{X}_{1:n_{i}}^{(i)}}
\]

Then the target approximation is:

\[
\mathcal{E}_{n}(x_{1:n}) = \frac{1}{N} \sum_{i=1}^{N} X_{1:n}^{(i)} \}
\]

2. Resample \( X_{1:n}^{(i)} \to \mathcal{E}_{n}(x_{1:n}) \) to obtain \( \mathcal{E}_{n}(x_{1:n}) = \frac{1}{N} \sum_{i=1}^{N} X_{1:n}^{(i)} \).
Example 1: Bayesian filtering

\[ f_n(x_{1:n}) = p(x_{1:n} ; y_{1:n}) ; \quad 1_/n(x_{1:n}) = p(x_{1:n} j y_{1:n}) ; \quad Z_n = p(y_{1:n}) ; \]

\[ q_n(x_n j x_{1:n} i 1) = f(x_n j x_{1:n} i 1); \]
Example 2: Eigen-particles

Computing eigen-pairs of exponentially large matrices and operators is an important problem in science. I will give two motivating examples:

i. Diffusion equation & Schrodinger’s equation in quantum physics

ii. Transfer matrices for estimating the partition function of Boltzmann machines

Both problems are of enormous importance in physics and learning.
Quantum Monte Carlo

\[
\left( -\frac{1}{2} \sum_{i=1}^{N} \nabla_{i}^{2} + \sum_{i=1}^{N} v(r_{i}) + \frac{1}{2} \sum_{i=1}^{N} \sum_{\substack{j=1 \atop j \neq i}}^{N} \frac{1}{|r_{i} - r_{j}|} \right) \Psi(r_{1}, r_{2}, \ldots, r_{N}) = E \Psi(r_{1}, r_{2}, \ldots, r_{N})
\]

We can map this multivariable differential equation to an eigenvalue problem:

\[
\tilde{A}(r) K(s_{j}) dr = \lambda \tilde{A}(s)
\]

In the discrete case, this is the largest eigenpair of the $M \times M$ matrix $A$:

\[
A x = \lambda x \quad \forall x \quad x(r_{i}) a(r_{i}; s) = \lambda x(s) \quad s = 1; 2; \ldots; M
\]

where $a(r_{i}; s)$ is the entry of $A$ at row $r_{i}$ and column $s$.

Transfer matrices of Boltzmann Machines

\[
Z = \sum_{j=1}^{m} \exp \left( \sum_{i=1}^{n} A \left( \frac{j}{m} \right) \right)
\]

[see e.g. Onsager, Nimalan Mahendran]
Power method

Let $A$ have $M$ linearly independent eigenvectors, then any vector $v$ may be represented as a linear combination of the eigenvectors of $A$: $v = \sum_i c_i x_i$, where $c$ is a constant. Consequently, for sufficiently large $n$,

$$A^n v = c_1 \lambda_1^n x_1$$
Particle power method

Succesive matrix-vector multiplication maps to Kernel-function multiplication (a path integral) in the continuous case:

\[
\begin{array}{c}
Z \quad Z \\ \phi \phi \phi v(x_1) K(x_k|x_{k-1}) dx_{1:n} \quad \frac{1}{4} c_{1,s}^{n} \tilde{A}(x_n) \\
\end{array}
\]

The particle method is obtained by defining

\[
Y^n f(x_{1:n}) = v(x_1) K(x_k|x_{k-1})
\]

Consequently \( c_{s,1} \), \( Z_n \) and \( \tilde{A}(x_n) \). The largest eigenvalue \( \lambda_1 \) of \( K \) is given by the ratio of successive partition functions:

\[
\lambda_1 = \frac{Z_n}{Z_{n-1}}
\]

The importance weights are

\[
W_n = W_{n-1} \frac{v(x_1)}{Q(x_{1:n}) v(x_1)} \frac{1}{4} c_{1,s}^{n} K(x_k|x_{k-1}) \]

\[
= W_{n-1} \frac{K(x_n|x_{n-1})}{Q(x_{n-1})}
\]
Example 3: Particle diffusion

A particle $X_n g_{n+1}$ evolves in a random medium

$$X_1 \mapsto 1 (\emptyset), \quad X_{n+1} | X_n = x \mapsto p(\emptyset x):$$

At time $n$, the probability of it being killed is $1 - g(X_n)$ with $0 \leq g(x) \leq 1$.

One wants to approximate $\Pr (T > n)$. 
Example 3: Particle diffusion

Again, we obtain our familiar path integral:

\[
Pr(T > n) = \mathbb{E}_1 \left[ \text{Probability of not being killed at } n \text{ given } X_{1:n} \right] \\
= \mathcal{Z} \mathcal{Z} \mathcal{Y} \mathcal{Y} \mathcal{Y} \sum_{k=2}^{1} (x_1) p(x_k | x_{k-1}) g(x_k) \int_{k=1}^{\infty} \{Z\} \text{ Probability to survive at } n
\]

Consider

\[
f_n(x_{1:n}) = \mathcal{Y} \mathcal{Y} \sum_{k=2}^{1} (x_1) p(x_k | x_{k-1}) g(x_k)
\]

\[
\frac{1}{n} (x_{1:n}) = \frac{f_n(x_{1:n})}{Z_n} \text{ where } Z_n = Pr(T > n)
\]

SMC is then used to compute \(Z_n\), the probability of not being killed at time \(n\), and to approximate the distribution of the paths having survived at time \(n\).

[Del Moral & AD, 2004]
Example 4: SAWs

Goal: Compute the volume $Z_n$ of a self-avoiding random walk, with uniform distribution on a lattice:

$$\frac{1}{n} (x_{1:n}) = Z_n I_1 D_n (x_{1:n})$$

where

$$D_n = \{ x_{1:n} \in E_n \text{ such that } x_k \succ x_{k+1} \text{ and } x_k \notin x_i \text{ for } k \neq i \}$$

and

$$Z_n = \text{cardinality of } D_n$$

SAWs on lattices are often used to study polymers and protein folding.

[See e.g. Peter Grassberger (PERM) & Alena Shmygelska; Rosenbluth Method]
Example 5: Stochastic control

Consider a Fredholm equation of the 2nd kind (e.g. Bellman backup):

\[ v(x_0) = r(x_0) + \int K(x_0; x_1)v(x_1)dx_1 \]

This expression can be easily transformed into a path integral (Von Neumann series representation):

\[ v(x_0) = r(x_0) + \sum_{n=1}^{\infty} \int r(x_n)K(x_{k_{n-1}}; x_k)dx_{1:n} \]

The SMC sampler again follows by choosing

\[ f_0(x_0) = r(x_0) \]

\[ f_n(x_{0:n}) = r(x_n) \prod_{k=1}^{n} K(x_{k_{n-1}}; x_k) \]

In this case we have a trans-dimensional distribution, so we do a little bit more work when implementing the method. 

[AD & Vladislav Tadic, 2005]
Particle smoothing can be used in the E step of the EM algorithm for MDPs.

\[
\tilde{p}_\theta (x_{0:k}, a_{0:k} | k, r_k) = r(x_k, a_k) p_\theta (x_{0:k}, a_{0:k} | k) \tilde{p}_\theta (r_k | k)
\]

[See e.g. Matt Hoffman et al, 2007]
Example 6: Dynamic Dirichlet processes

[François Caron, Manuel Davy & AD, 2007]
SMC for static models

Let \( f_{\frac{1}{n}} g_{n-1} \) be a sequence of probability distributions defined on \( X \) such that each \( f_{\frac{1}{n}} (x) \) is known up to a normalizing constant, i.e.

\[
\frac{1}{n} f_{\frac{1}{n}} (x) = \frac{Z_{n-1}^{\frac{1}{n}}}{\prod_{i=1}^{n} \hat{Z}} f_{n}(x)
\]

Unknown Known

We want to sample approximately from \( \frac{1}{n} f_{\frac{1}{n}} (x) \) and compute \( Z_n \) sequentially.

This differs from the standard SMC, where \( \frac{1}{n} f_{\frac{1}{n}} (x_{1:n}) \) is defined on \( X^n \).

\[
\frac{1}{n} f_{\frac{1}{n}} (x) = Z^{-\frac{1}{n}} \prod_{i=1}^{n} e^{i \sum_{j} w_{ij} x_{i}}
\]
Static SMC applications

2 Sequential Bayesian Inference: $\frac{1}{n}(x) = p(x|y_{1:n})$:

2 Global optimization: $\frac{1}{n}(x) / \left[\frac{1}{4}(x)\right]^n$ with $f_n$ increasing sequence such that $f_n \rightarrow 1$:

2 Sampling from a fixed target $\frac{1}{n}(x) / \left[\frac{1}{4}(x)\right]^n \left[\frac{1}{4}(x)\right]^{1_i} \left[\frac{1}{4}(x)\right]^{1_f}$ where $1_1$ is easy to sample from. Use sequence $1_1 = 1 > 1_{n_i} > 1_n > 1_{final} = 0$. Then $\frac{1}{4}(x) / 1_1(x)$ and $\frac{1}{4}_{final}(x) / \frac{1}{4}(x)$

2 Rare event simulation $\frac{1}{4}(A) \triangleq 1$: $\frac{1}{n}(x) / \frac{1}{4}(x)1_{E_n}(x)$ with $Z_1$ known. Use sequence $E_1 = X \frac{3}{4} E_{n_1} \frac{3}{4} E_n \frac{3}{4} E_{final} = A$. Then $Z_{final} = \frac{1}{4}(A)$:

2 Classical CS problems: SAT, constraint satisfaction, computing volumes in high dimensions, matrix permanents and so on.
Static SMC derivation

Construct an artificial distribution that is the product of the target distribution that we want to sample from and a backward kernel $L$:

$$\varphi_n(x_{1:n}) = Z_n^{-1} f_n(x_{1:n}), \text{ where } f_n(x_{1:n}) = \frac{\prod_{k=1}^{n} f_n(x_k)}{\text{target}} L_k(x_k|x_{k+1})$$

such that $\frac{1}{\varphi_n}(x_n) = \int \varphi_n(x_{1:n}) \, dx_{1:n}$.

The importance weights become:

$$W_n = \frac{f_n(x_{1:n})}{K_n(x_{1:n})} = W_{n+1} \frac{K_{n+1}(x_{1:n+1})}{f_{n+1}(x_{1:n+1})} \frac{f_n(x_{1:n})}{K_n(x_{1:n})} = W_{n+1} \frac{f_n(x_n) L_{n+1}(x_{n+1}|x_n)}{f_{n+1}(x_{n+1}) K_{n+1}(x_{n+1}|x_n)}$$

For the proposal $K(\cdot)$, we can use any MCMC kernel.

We only care about $\frac{1}{\varphi_n}(x_n) = Z_n^{-1} f_n(x_n)$ so no degeneracy problem.

---

[Pierre Del Moral, AD, Ajay Jasra, 2006]
Static SMC algorithm

1. Initialize at time $n = 1$

2. At time $n \geq 2$
   
   (a) Sample $X_{n}^{(i)} \sim K_{n} \times_{n_{j=1}} X_{n_{j=1}}^{(i)}$ and augment $X_{n_{i=1:n}} = X_{n_{i=1}}^{(i)} ; X_{n}$

   (b) Compute the importance weights
   
   $$W_{n}^{(i)} = W_{n_{i=1}}^{(i)} \frac{f_{n} X_{n}^{(i)} L_{n_{i=1}} X_{n_{i=1}}^{(i)} - X_{n}^{(i)} \cdot X_{n}^{(i)} - X_{n_{i=1}}^{(i)}}{f_{n_{i=1}} X_{n_{i=1}}^{(i)} K_{n} X_{n}^{(i)} - X_{n_{i=1}}^{(i)}}$$

   Then the weighted approximation is

   $$\mathbf{b}_{n}^{n} (x_{n}) = \frac{1}{N} \sum_{i=1}^{N} W_{n}^{(i)} \pm X_{n}^{(i)} (x_{n})$$

   (c) Resample $X_{n}^{(i)} \sim \mathbf{b}_{n}^{n} (x_{n})$ to obtain $\mathbf{b}_{n}^{n} (x_{n}) = \frac{1}{N} \sum_{i=1}^{N} \pm X_{n}^{(i)} (x_{n})$. 
Static SMC: Choice of $L$

² A default (easiest) choice consists of using a $\frac{1}{n}$-invariant MCMC kernel $K_n$ and the corresponding reversed kernel $L_{n+1}$:

$$L_{n+1}(x_{n+1}|x_n) = \frac{\frac{1}{n} (x_{n+1}) K_n (x_{n} | x_{n+1})}{\frac{1}{n} (x_n)}$$

² In this case, the weights simplify to:

$$W^{(i)}_n = W^{(i)}_{n+1} \frac{f_n X^{(i)}_{n+1}}{f_{n+1} X^{(i)}_{n+1}}$$

² This particular choice appeared independently in physics and statistics (Jarzynski, 1997; Crooks, 1998; Gilks & Berzuini, 2001; Neal, 2001): In machine learning, it’s often referred to as annealed importance sampling.

² Smarter choices of $L$ can be sometimes implemented in practice.
Example 1: Deep Boltzmann machines

\[
\frac{1}{4}(x) = 1
\]

\[
\frac{1}{2}(x) \quad \frac{1}{3}(x) \quad \frac{1}{4}(x) = Z \prod_{i \neq j} \mathcal{A}(x_i; x_j)
\]

\[
\text{W}^{(i)}_2 / \mathcal{A}(X^{(i)}_{2;3}; X^{(i)}_{2;2})
\]

\[
\text{W}^{(i)}_3 / \mathcal{A}(X^{(i)}_{3;1}; X^{(i)}_{3;5})
\]

[Firas Hamze, Hot coupling, 2005] [Peter Carbonetto, 2007, 2009]
Some results for undirected graphs

- Fully-Connected Random: HC, SMCG, Gibbs, Loopy
- Fully-Connected Homogeneous: HC, SMCG, Gibbs, Loopy
- Grid Model Random: NULL
- Grid Model Homogeneous: HC, SMCG, Gibbs, Loopy
Example 2: ABC

Consider a Bayesian model with prior $p(\mu)$ and likelihood $L(yj\mu)$ for data $y$. The likelihood is assumed to be intractable but we can sample from it.

**ABC algorithm:**

1. Sample $\mu^{(i)} \sim p(\mu)$
2. Hallucinate data $Z^{(i)} \sim L \left( zj\mu^{(i)} \right)$
3. Accept samples if hallucinations look like the data $|d^i y; Z^{(i)}|$, where $d : Y \subseteq Y! R^+$ is a metric.

The samples are approximately distributed according to:

$$\frac{1}{4} \left( \mu; xj y \right) / p(\mu) L \left( xj\mu \right) 1_{d(y;}z) \cdot$$

The hope is that $\frac{1}{4} \left( \mu j y \right) \frac{1}{4} \frac{1}{4}(\mu j y)$ for very small $\epsilon$.

Inefficient for $\epsilon$ small!

[Beaumont, 2002]
SMC samplers for ABC

1. Define a sequence of artificial targets $f_{\frac{1}{4}} (\mu j y) g_{n=1;\ldots;P}$ where

   \[ "_1 = 1 , \ldots , "_P = \ldots \]:

2. We can use SMC to sample from $f_{\frac{1}{4}} (\mu j y) g_{n=1;\ldots;P}$ by adopting a Metropolis-Hastings proposal kernel $K_n ( (\mu_n ; z_n) j (\mu_{n-1} ; z_{n-1} ))$, with importance weights

   \[ W_n^{(i)} = W_1^{(i)} \frac{1}{\int \frac{d y; Z_{n\mid i}^{(i)}}{Z_{n\mid i}^{(i)}} \cdot "_n \cdot \ldots \cdot "_n , \]

3. Smarter algorithms have been proposed, which for example, compute the parameters $"_n$ and of $K_n$ adaptively.

[Pierre Del Moral, AD, Ajay Jasra, 2009]
Final remarks

• SMC is a general, easy and flexible strategy for sampling from any arbitrary sequence of targets and for computing their normalizing constants.

• SMC is benefiting from the advent of GPUs.

• SMC remains limited to moderately high-dimensional problems.

Thank you!

Nando de Freitas & Arnaud Doucet
Naïve SMC for static models

At time $n-1$, you have particles $X_{n-1}^{(i)} \sim \frac{1}{\pi_{n-1}}(x_{n-1})$.

Move the particles according to a transition kernel

$$X_n^{(i)} \sim K_n(x_{n-1}) X_{n-1}^{(i)}$$

hence marginally

$$Z X_n^{(i)} \sim \frac{1}{\pi_n}(x_n) \text{ where } \frac{1}{\pi_n}(x_n) = \frac{1}{\pi_{n-1}}(x_{n-1}) K_n(x_{n-1}) X_{n-1}^{(i)}$$

Our target is $\frac{1}{\pi_n}(x_n)$ so the importance weight is

$$W_n^{(i)} = \frac{1}{\pi_n} X_n^{(i)}$$

Problem: $\frac{1}{\pi_n}(x_n)$ does not admit an analytical expression in general cases.