PARTICLE FILTERS: sequential MC

We will use the importance sampler as a framework since it lends itself naturally to Bayesian estimation.

Suppose we want to find

$$
\mathbb{E} [g(x) | y_{1:t}] = \int g(x) p(x | y_{1:t}) \, dx
$$

Bayesian estimate.

$Y_i$ are the observations with known errors.
MC:

\[ \hat{E}(g y) \approx \frac{1}{N} \sum_{i=1}^{N} g(X_i) \]

where \( X_i \sim p(x|y_{1:t}) \)

Convergence of estimator is

\[ \lim_{n \to \infty} \left| \hat{E}(g y_{1:t}) - E(g(y_{1:t})) \right| \leq O(N^{-\frac{1}{2}}) \]

In principle, this rate is not affected by the size of the dimension of \( x \).

**Importance Sampling**

We use an approximate distribution \( \pi(x) \).
We'd like \( \pi(x) \) to be close to \( p(x|y_{1:t}) \).
\( \pi(x) \) importance distribution

We would also like \( \pi(x) \) to generate samples easily. We don't require that \( \pi(x) \) be a proper distribution, but it makes things easier.

\[
\int g(x) \pi(x | y_{1:t}) \, dx \\
= \int \left[ g(x) \frac{p(x | y_{1:t})}{\pi(x)} \right] \pi(x) \, dx \tag{A} \\
\]

We also want the support of \( \pi(x) \) to be larger than \( p(x | y_{1:t}) \)

So (A) is interpreted as

\[
\left\langle g(x) \frac{p(x | y_{1:t})}{\pi(x)} \right\rangle_{\pi(x)} \\
\]

The expectation of \( \frac{gP}{\pi} \) wrt \( \pi(x) \)
Using sample averages:

$$\hat{E}(g) p(x|y; t) = \frac{1}{N} \sum_{i=1}^{N} g(x) \frac{p(x|y; t)}{\pi(x)}$$

where $X_i \sim \pi(x)$

$$= \sum_{i=1}^{N} w_i g(x_i)$$

$w_i$ are the "weights"

$$w_i = \frac{1}{N} \frac{p(x_i|y; t)}{\pi(x_i)}$$

Rule: the disadvantage of importance sampling is that we need $p(x_i|y; t)$ to be well known, but very commonly, we only know its shape, i.e. it's an
improper PDF (we don't know the normalization)

i.e. \[ p(x | y_{1:T}) = \frac{1}{Z} p(y_{1:T} | x) p(x) \]

\[ Z = \int p(x | y_{1:T}) \, dx \]

We can reformulate the importance sampler to handle not knowing \( Z \) (assuming \( p(x | y) \) is a posterior)

\[ \mathbb{E}(g(x) | y_{1:T}) = \int g(x) p(x | y_{1:T}) \, dx \]

\[ = \frac{\int g(x) p(y_{1:T} | x) p(x) \, dx}{\int p(y_{1:T} | x) p(x) \, dx} \]

Introduce importance distribution \( \psi(x) \)

\[ \int \left[ \frac{p(y_{1:T} | x) p(x) g(x)}{\psi(x)} \right] \psi(x) \, dx \]

\[ = \frac{\int \left[ \frac{p(y_{1:T} | x) p(x)}{\psi(x)} \right] \psi(x) \, dx}{\int \left[ \frac{p(y_{1:T} | x) p(x)}{\psi(x)} \right] \psi(x) \, dx} \]
In Serkka's book \( \mathbb{D}(x) = \mathcal{G}(x | y_{1:T}) \)

\[
\begin{align*}
\frac{1}{N} \sum_{i=1}^{N} \frac{p(y_{1:T} | x^i) p(x^i) g(x^i)}{\pi(x^i)} &\equiv \sum_{i=1}^{N} w_i g(x^i) \\
\end{align*}
\]

\( X^i \sim \mathcal{D}(x) \)

where

\[
\begin{align*}
w_i &= \frac{p(y_{1:T} | x^i) p(x^i)}{\pi(x^i)} \\
\end{align*}
\]

\[
\begin{align*}
\sum_{i=1}^{N} \frac{p(y_{1:T} | x^i) p(x^i)}{\pi(x^i)}
\end{align*}
\]

**Algorithm**

Given a likelihood \( p(y_{1:T} | x) \) and a prior \( p(x) \) we can form an importance sampling approximation of the posterior as follows:
1. Draw $N$ samples from the importance pdf
   \[ X^i \sim \tilde{\Pi}(X^i | y_{1:T}) \quad i = 1, 2, \ldots, N \]

2. Compute unnormalized weights
   \[ \tilde{w}_i = \frac{P(y_{1:T} | X^i) p(X^i)}{\tilde{\Pi}(X^i | y_{1:T})} \]

   Then
   \[ w_i = \frac{\tilde{w}_i}{\sum_{i=1}^{N} \tilde{w}_i} \]

3. \[ \mathbb{E}(g(x) | y_{1:T}) = \sum_{i=1}^{N} w_i g(X^i) \]

Remark: The approximate posterior pdf formed by the algorithm can be cast as
   \[ p(x | y_{1:T}) \approx \sum_{i=1}^{N} w_i \delta(x - X^i) \]
**SEQUENTIAL IMPORTANCE SAMPLING (SIS)**

The time dependent case:

(Prior) \( x_k \sim p(x_k | x_{k-1}) \) Markovian

(Model) \( x_k = f(x_{k-1}) dt + \sigma dW_t \)

(Likelihood) \( y_k \sim p(y_k | x_k) \)

(Observations) \( y_k = H x_k + V_k \)

We'll use weighted set of "particles" \( \{ W_i, x_i \} \) \( i=1, \ldots, N \) to represent samples from \( p(x_k | y_{1:k}) \) at every time step. The approximate expectation of an arbitrary function \( g(\cdot) \) can be calculated as the weighted sample average.

\[
E \left[ g(x_k) | y_{1:k} \right] \approx \sum_{i=1}^{N} W_i g(x_{i:k})
\]
Consider the calculation of the expectation of $g(.)$ with respect to a posterior

$$p(y_{1:k} | x_{0:k}) p(x_{0:k})$$

Assume that we have a Markovian process in $x_{k}$

$$p(x_{0:k} | y_{1:k}) = p(y_{k} | x_{0:k}, y_{1:k-1}) p(x_{0:k-1} | y_{1:k-1}) p(x_{0} | y_{1:1})$$

Now, construct the weights. Seek a recurrence relation that allows us to find $\{w_{k}\}$ in terms of
\[ \{w_{k-1}^i\} \]

We propose the importance pdf

\[ X_k^i \sim \tilde{\tau}(x_{0:k}^i | y_{1:k}) \]

The weights \( \text{(unnormalized)} \)

\[ \tilde{w}_k^i = \frac{p(y_{1:k} | x_k^i) p(x_k^i | x_{k-1}) p(x_{0:k-1} | y_{1:k-1})}{\tilde{\tau}(x_k^i)} \]

Suppose that we pick \( \tilde{\tau}(x) \) as a recursive rule:

\[ \tilde{\tau}(x_{0:k}^i | y_{1:k}) = \pi(x_k^i | x_{0:k-1}^i, y_{1:k}) \pi(x_{0:k-1}^i | y_{1:k-1}) \]

Then the weights

\[ \tilde{w}_k^i = \frac{p(y_{1:k} | x_k^i) p(x_k^i | x_{k-1}) p(x_{0:k-1} | y_{1:k-1})}{\tilde{\tau}(x_k^i | x_{0:k-1}^i, y_{1:k}) \pi(x_{0:k-1} | y_{1:k-1})} \]

So the latter fraction of the above expression will be \( w_{k-1}^i \).
\[ W_{k_i} = \frac{P(y_k | x_{k_i}) P(x_{k_i} | x_{k-1})}{\Pi(x_{k_i} | x_{0:k-1}, y_{1:k})} W_{k-1} \]

Here we're presuming that \( W_{k-1} \) has been normalized.

**Algorithm (SIS)**

1. **Draw** \( N \) samples \( x_0^i \) from
   \[ x_0^i \sim p(x_0) \quad i=1,2,..,N \]

   set \( w_0^i = \frac{1}{N} \) for all \( N \).

2. **For** each \( k=1,...,T \) **do**
   - **Draw** samples \( x_k^i \) from the importance pdf
     \[ x_k^i \sim \Pi(x_k | x_{0:k-1}, y_{1:k}) \quad i=1,..,N \]
   - \( \tilde{w}_k = W_{k-1} \sum_{i=1}^{N} P(y_k | x_{k_i}) P(x_{k_i} | x_{k-1}) \]
   - \( z = \frac{1}{N} \sum_{i=1}^{N} \tilde{w}_k \)
\[ w_i = \frac{\hat{w}_i}{Z} \]

Rule: Convenient to draw from a \( T \) \( (\theta) \) which has a CDF. Otherwise, you need to use an empirical strategy to obtain the sample.

Rule: Also typical (convenient) to use

\[
T_T(x_k|x_{0:k-1}, y_{1:k}) = T_T(x_k|x_{k-1}, y_{1:k})
\]

Markovian

THE DEGENERACY PROBLEM: particular to the SIS. We'll address this problem via a Resampling.

Pictorially, the problem is seen as follows:
 Rule: This is also known as a particle filter collapse.

The simplest "fix" is called **RESAMPLING**

SISR (Sequential Importance Sampling with Resampling)

Restart the process at each time step (observation step) or every once in a while.
**ALGORITHM**

1. Interpret each $w_i$ as the probability of obtaining a sample with index $i$ from set $\{x_i, i=1, \ldots, N\}$. 
2. Draw $N$ samples for that discrete distribution and replace old $w$ with new distribution. 
3. Set all weights of new $w_i = \frac{1}{N}$. 

11 particles ($N$)
Rule: Resampling can lead to changes in the variance estimates.
Resampling Algorithm:

Define $N_{eff} = \frac{1}{\sum_{i=1}^{N} (w_i)^2}$

The normalized weights

We see that if all weights

$w_i = \frac{1}{N}$ then $N_{eff} = N$ (ideal)

The other extreme is 1 weight is 1 others are 0:

Then $N_{eff} = 1$

.: Monitor $N_{eff}$ and resample when $N_{eff}$ is low, near 1.

The SISR would look like SIS, but have $N_{eff}$ to decide whether we need to resample:
SISR (SIR)

Draw N samples \( x_0^i \sim p(x_0) \) \( i = 1:N \)

set \( w_0^i = \frac{1}{N} \)

for \( k = 1:T \)

\[ \hat{x}_k^i \sim \pi(x_k | x_{k-1}^i, y_{1:k}) \]

\( i = 1:N \)

\[ \hat{w}_k^i = w_{k-1}^i \frac{p(y_k | \hat{x}_k^i) p(\hat{x}_k^i | x_{k-1}^i)}{\pi(x_k | x_{k-1}^i, y_{1:k})} \]

\[ w_k^i = \frac{1}{\sum_{i=1}^{N} \hat{w}_k^i} \hat{w}_k^i \]

Find Neff

if \( \text{Neff} > N_{\text{threshold}} \)

accept \( x_0^i : k = \hat{x}_0^i : k \)

else

Draw N samples from discrete
In the SISR, we might consider choosing
\( x_i = \frac{X - X_i}{w_i} \) and set weights \( w_i = \frac{1}{\sum_i w_i} \) for \( X_i \) and set weights \( w_i = \frac{1}{\sum_i w_i} \) for \( X_i \)

\[ X_i = \frac{X - X_i}{w_i} \]
**Bootstrap Filter**: a variant of SIR where the dynamics model \( p(x_t|x_{t-1}) \) is used as the TT, the importance distribution.

This makes for an easy code, but we now have simple sequential MC.

**Algorithm**

**Usual initialization**

\[
\begin{align*}
&\text{for } k = 1 : T \\
&\quad \text{Draw set } x_k^i \text{ for each point in the sample set } \{x_k^i, i = 1 : N\} \\
&\quad \text{Advance via model to get } x_k^i \sim p(x_k|x_{k-1}) \\
&\quad \text{Calculate weights } \hat{w}_k^i = p(y_k|x_k^i) \\
&\quad w_k^i = \frac{1}{N \hat{w}_k^i} \\
&\end{align*}
\]
When prior & likelihood are very mismatched, except to be extra careful on all particle filters.

Rule: When a model has low uncertainty you generate many particles that are near identical copies.

Rule: Often we inflate the variance of the model or the observations artificially in order to improve things, at the expense of poorer estimates. The hit is usually not in the mean, but in the posterior uncertainty.

**RAO-BLACKWELLIZED PF (RBPF)** (aka Mixture KF)
The basic idea here is to identify all aspects of the state variable that are linear & Gaussian or well approximated this way. You then split up the state

\[ x = (\tilde{x}, \tilde{r}) \]

We use something optimal for the linear Gaussian portion and particles for the other portion:

\[ P(x_{0:k} | y_{1:k}) = P(\tilde{x}_{0:k}, \tilde{r}_{0:k} | y_{1:k}) \]

\[ = P(\tilde{r}_{0:k} | y_{1:k}) P(\tilde{x}_{0:k} | \tilde{r}_{0:k}, y_{1:k}) \]

Use particles on \( \tilde{r}_{0:k} \) or 4D Var on \( \tilde{x}_{0:k} \)

Use Kalman

on \( \tilde{x}_{0:k} \)
This is set up as a predictor/corrector advancing first the Kalman part and then doing the particle part on the marginalized.