Intro to particle filters (aka Sequential Monte Carlo)

MVE187-MSA101 “Computational methods for Bayesian statistics”, 2021-2022

Umberto Picchini
@uPicchini

Chalmers University of Technology and University of Gothenburg
Sweden

This lecture will explore possibilities offered by particle filters, also known as Sequential Monte Carlo (SMC) methods, when applied to parameter estimation problems.

We will not give a thorough introduction to SMC methods.

Instead, we will introduce the most basic (popular) SMC method with the main goal of performing inference for model parameters for a state space / Hidden Markov model.

Results anticipation (next lecture)
Thanks to recent and astonishing results, we show how it is possible to construct a practical method for **exact Bayesian inference** on the parameters of a state-space model.

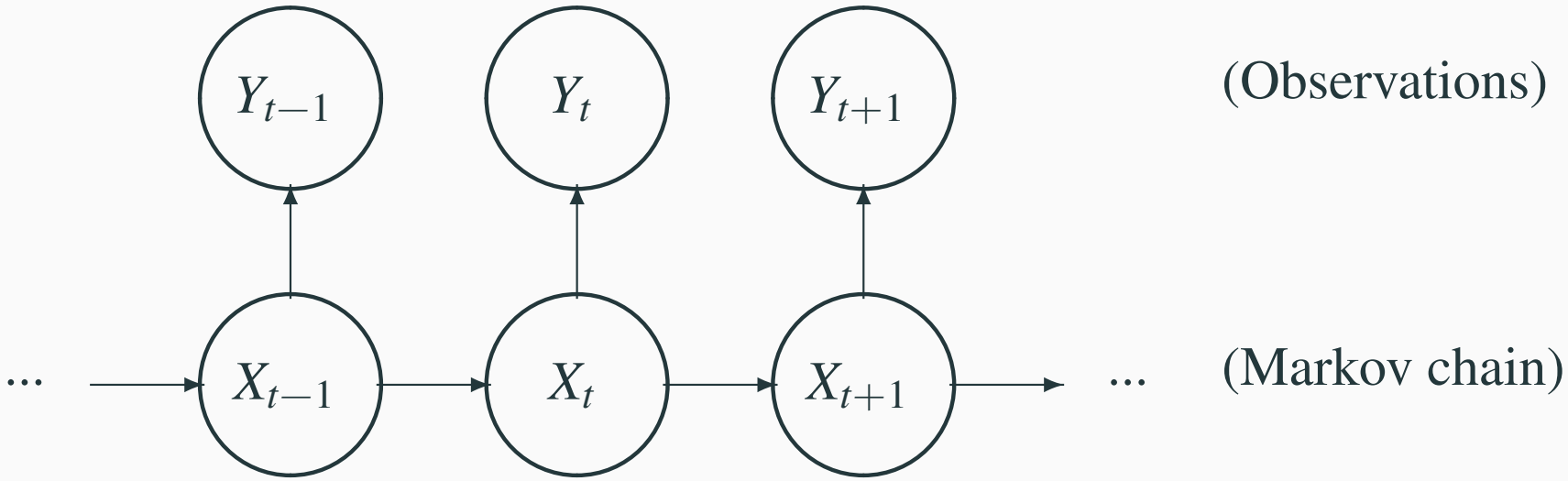
Working framework for my 2 lectures

For this and next lecture we consider the (important) case where data are modelled according to an Hidden Markov Model (HMM) or a state-space model (SSM).

A state-space model refers to a class of probabilistic graphical models that describes the probabilistic dependence between latent state variables and the observed measurements of a **dynamical system**.

SSMs as "graphical models"

Graphically:



$$(Y_t|X_t = x_t;\theta) \sim p(y_t|x_t,\theta) \qquad \text{(Observation density)}$$

$$(X_{t+1}|X_t = x_t,\theta) \sim p(x_{t+1}|x_t,\theta) \qquad \text{(Transition density)}$$

$$X_0 \sim \pi(x_0|\theta) \qquad \text{(Initial distribution)}$$

From the arrows, obviously X_{t+1} depends on X_t only.

Notation: I use the colon-notation $a : b$ to mean $a, a+1, ..., b$.

- Our system of interest is $\{X_t\}_{t \geqslant 0}$. This is a *latent/unobservable* stochastic process.
- (Unavailable) values of the system on a discrete time grid $\{0, 1, ..., T\}$ are $X_{0:T} = \{X_0, ..., X_T\}$

  (we consider integer times t for ease of notation).
- Actual observations (data) are noisy $y_{1:T} = \{y_1, ..., y_T\}$
- That is at time t each y_t is a noisy/corrupted version of the true state of the system X_t.
- $\{X_t\}_{t \geqslant 0}$ is assumed a Markovian stochastic process.
- Observations y_t are assumed conditionally independent *given* the corresponding latent state X_t.
- I denote the model parameters with θ.

In previous lectures you have seen the Kalman filter.

Not sure how much detail has been covered in previous lectures but: when the latent system X is linear and random noises are Gaussian distributed then

- (predictive distribution) $p(x_t|y_1, ..., y_{1:t-1})$ is Gaussian;
- (filtering distribution) $p(x_t|y_1, ..., y_{1:t})$ is Gaussian;
- (likelihood term) $p(y_t|y_1, ..., y_{1:t-1})$ is Gaussian;

and all three distributions have mean and covariance that can be computed iteratively (not reported for brevity, see [1]).

[1] Simo Särkkä (2013). Bayesian Filtering and Smoothing. Free pdf at http://users.aalto.fi/~ssarkka/pub/cup_book_online_20131111.pdf

Example: Gaussian random walk

A most trivial example (linear, Gaussian SSM).

$$x_t = x_{t-1} + \xi_t, \quad \xi_t \sim_{iid} N(0, q^2)$$
$$y_t = x_t + e_t, \quad e_t \sim_{iid} N(0, r^2)$$

Therefore

$$p(x_t|x_{t-1}) = N(x_t; x_{t-1}, q^2)$$
$$p(y_t|x_t) = N(y_t; x_t, r^2)$$

Here the parameter to infer would be $\theta = (q, r)$.

So we want to infer model parameters θ.

We need the likelihood function both for frequentist estimation (maximum likelihood) and Bayesian estimation (eg for MCMC).

The likelihood function of θ is

$$p(y_1, ..., y_T|\theta) = p(y_1|\theta) \prod_{t=2}^{T} p(y_t|y_{1:t-1}, \theta)$$

but we'll see computing this is often not trivial.

Example: SSM with latent SDE

A way less trivial example: here we have a continuous-time model

$$dx_t = f(x_t;\theta)dt + g(x_t;\theta)dB_t, \quad dB_t \sim_{iid} N(0,dt)$$
$$y_t = x_t + e_t, \quad e_t \sim_{iid} N(\cdot,\cdot)$$

Under conditions, the stochastic differential equation (SDE) has a solution $\{x_t\}$ that is a Markov process.

Huge problem: $p(x_t|x_s;\cdot)$ generally unknown! $(s < t)$.

Notation: I will keep writing $\mathbf{p}(\mathbf{x_t}|\mathbf{x_{t-1}};\cdot)$ instead of $\mathbf{p}(\mathbf{x_t}|\mathbf{x_s};\cdot)$, as if the continuous-time process was evolving between integer times.

Notation reminder

Always remember X_0 is NOT the state for the first observational time, that one is X_1. Instead X_0 is the (typically unknown) system's initial state for process $\{X_t\}$.

X_0 can be set to be a deterministic constant (as in the example we are going to discuss soon).

In general $\{X_t\}$ and $\{Y_t\}$ can be either continuous– or discrete–valued stochastic processes.

However in the following we assume $\{X_t\}$ and $\{Y_t\}$ to be defined on continuous spaces.

Our goal

Main goal

We introduce general methods for SSM producing inference for the vector of parameters θ. We will be particularly interested in Bayesian inference.

A quick look into the final goal

- I denote with $p(y_{1:T}|\theta)$ is the *likelihood of the measurements* conditionally on θ.
- $\pi(\theta)$ is the *prior density* of θ. Encloses knowledge about θ before we "see" our current data $y_{1:T}$.
- Bayes theorem gives the *posterior distribution*:

$$\pi(\theta|y_{1:T}) = \frac{p(y_{1:T}|\theta)\pi(\theta)}{p(y_{1:T})} \propto p(y_{1:T}|\theta)\pi(\theta)$$

- inference based on $\pi(\theta|y_{1:T})$ is called *Bayesian inference*.
- $p(y_{1:T})$ is the *marginal likelihood* (evidence), independent of θ.
- Goal: obtain samples from $\pi(\theta|y_{1:T})$.
- Remark: θ is a random quantity in the Bayesian framework.

Am I obsessed with the Bayesian approach? NO! However it sometimes offers the easiest way to deal with complex, non-trivial models. ▸ Some good reads

The Bayesian approach actually opens the possibility for a surprising result (next lecture...).

The likelihood function for SSMs

- In a SSM data are not independent, they are only *conditionally independent*$\rightarrow$ *complication!*:

$$p(y_{1:T}|\theta) = p(y_1|\theta)\prod_{t=2}^{T} p(y_t|y_{1:t-1},\theta) = ?$$

Except for the simplest cases, we generally don't have a closed-form expression for the product above because we do not know how to calculate $p(y_t|y_{1:t-1},\theta)$.

Exceptions are for example linear/Gaussian models where Kalman filtering can be applied.

In a SSM the observed process is assumed to depend on the latent Markov process $\{X_t\}$: we can write

$$p(y_{1:T}|\theta) = \int p(y_{1:T}, x_{0:T}|\theta) dx_{0:T} = \int \underbrace{p(y_{1:T}|x_{0:T}, \theta)}_{\text{use cond. indep.}} \times \underbrace{p(x_{0:T}|\theta)}_{\text{use Markovianity}} \, dx_{0:T}$$

$$= \int \prod_{t=1}^{T} p(y_t|x_t, \theta) \times \left\{ p(x_0|\theta) \prod_{t=1}^{T} p(x_t|x_{t-1}, \theta) \right\} dx_{0:T}$$

Problems

- The expression above is a $(T+1)$-dimensional integral ☹
- For most (nontrivial) models, transition densities $p(x_t|x_{t-1})$ are **unknown** ☹

Special cases

- In some simple cases, closed form solutions do exist: for example when the SSM is linear and Gaussian (see the Gaussian random walk example) then the classic Kalman filter gives the exact likelihood.
- In the SSM literature important (Gaussian) approximations are given by the extended and unscented Kalman filters. However, approximations offered by particle filters (a.k.a. sequential Monte Carlo) are presently the state-of-art for general non-linear non-Gaussian SSM.

What if we had $p(y_{1:T}|\theta)$...

Ideally if we had $p(y_{1:T}|\theta)$ we could use Metropolis-Hastings (MH) to sample from the posterior $\pi(\theta|y_{1:T})$. At iteration $r+1$ we have:

1. current value is θ_r, propose a new $\theta^* \sim q(\theta^*|\theta_r)$, e.g. $\theta^* \sim N(\theta_r, \Sigma_\theta)$ for some covariance matrix Σ_θ.
2. compute
$$A = \frac{p(y_{1:T}|\theta^*)}{p(y_{1:T}|\theta_r)} \times \frac{\pi(\theta^*)}{\pi(\theta_r)} \times \frac{q(\theta_r|\theta^*)}{q(\theta^*|\theta_r)}$$
Draw a uniform $u \sim U(0,1)$ and if $u < A$ **accept** θ^* and set $\theta_{r+1} := \theta^*$. Otherwise, **reject** θ^*, set $\theta_{r+1} := \theta_r$.
3. Set $r := r+1$, go to 1 and repeat.

By repeating steps 1-3 as much as wanted we are guaranteed that, by discarding a "long enough" number of iterations (*burnin*), the remaining draws form

- a Markov chain (hence dependent values) having $\pi(\theta|y_{1:T})$ as their stationary distribution.
- for a vector valued $\theta \in \mathbb{R}^p$ create p separate histograms of the draws pertaining each component of θ. Such histograms represent the posterior **marginals** $\pi(\theta_j|y_{1:T}), j = 1, ..., p$.

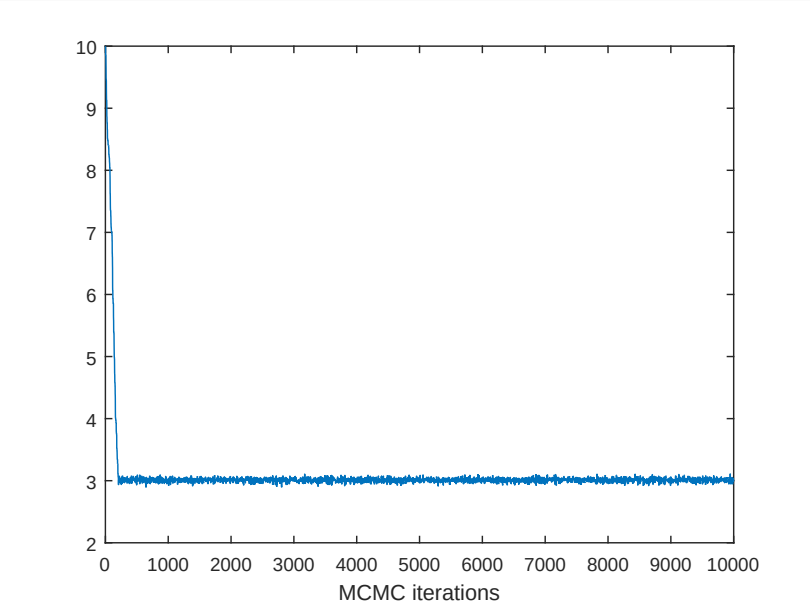
Brush up MH with a simple example

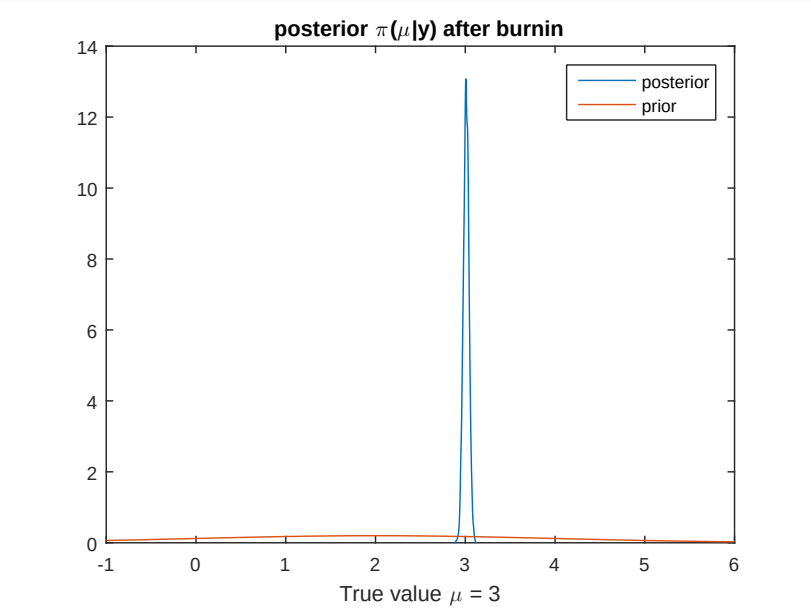
A simple toy problem not related to state-space modelling.

- Data: $n = 1,000$ observations i.i.d $y_j \sim N(3,1)$.
- Now assume $\mu = E(y_j)$ unknown. Estimate μ.
- Assume for example a "wide" prior: $\mu \sim N(2, 2^2)$ (do not look at data!)
- Gaussian random walk to propose $\mu^* = \mu_r + 0.1 \cdot \xi$, with $\xi \sim N(0,1)$. Notice Gaussian proposals are symmetric, hence $q(\mu^*|\mu_r) = q(\mu_r|\mu^*)$ therefore

$$A = \frac{p(y_{1:n}|\mu^*)}{p(y_{1:n}|\mu_r)} \times \frac{\pi(\mu^*)}{\pi(\mu_r)}$$

Start *far* at $\mu = 10$. Produce 10,000 iterations.





The posterior on the right plot discards the initial 500 draws (burnin).

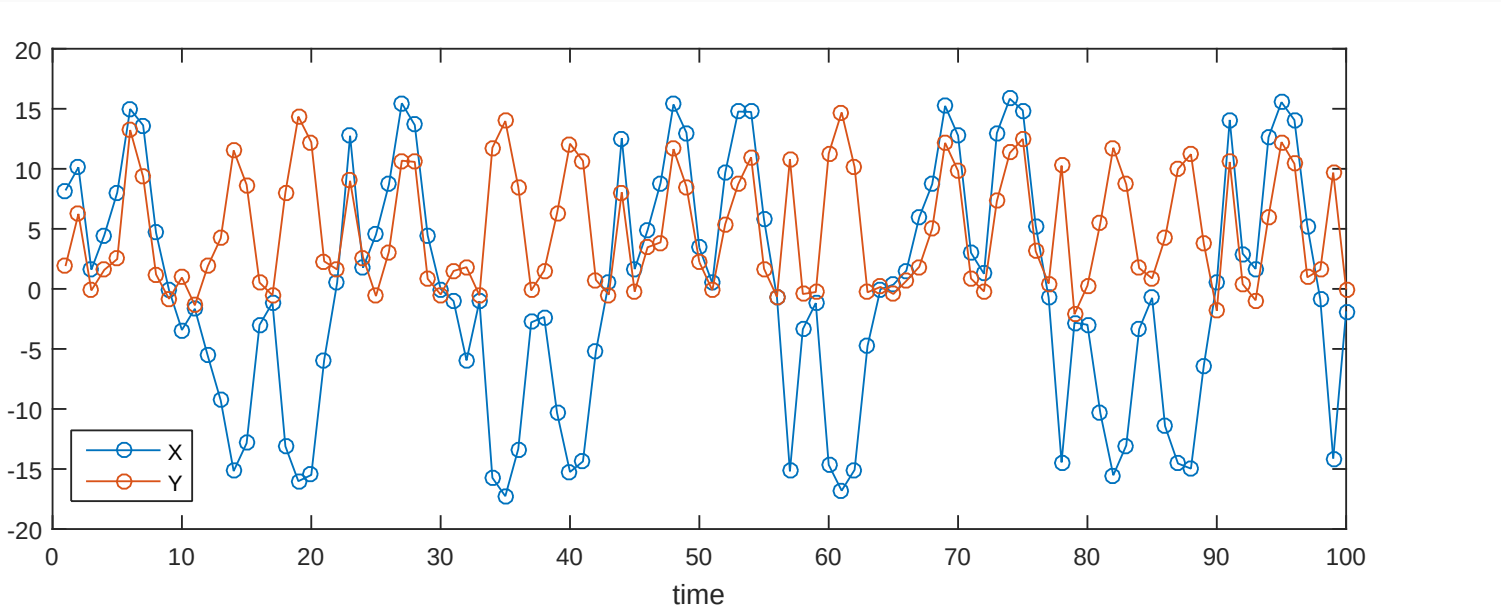
Example: a nonlinear SSM

$$\begin{cases} x_t = 0.5x_{t-1} + 25\frac{x_{t-1}}{(1+x_{t-1}^2)} + 8\cos(1.2(t-1)) + v_t, \\ y_t = 0.05x_t^2 + e_t, \end{cases}$$

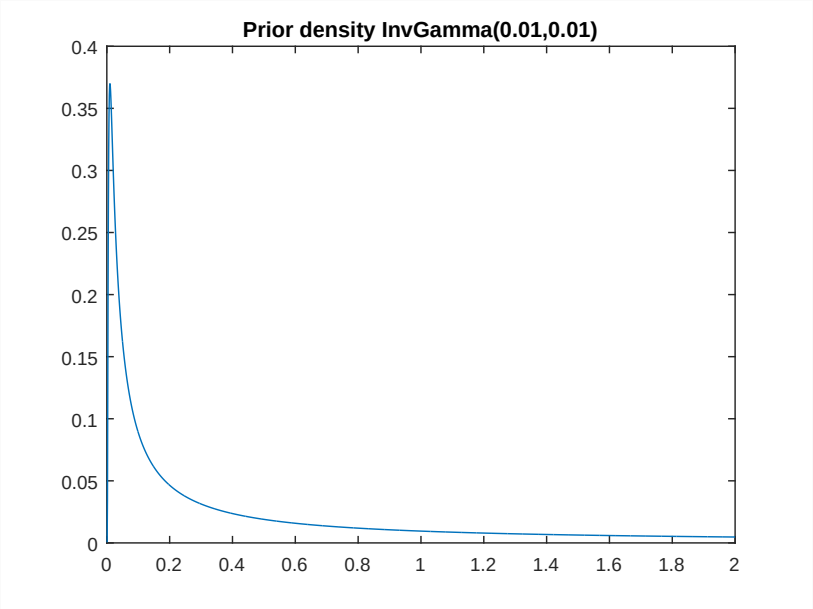
with a deterministic $x_0 = 0$, measurements at times $t = 1, 2, ..., 100$.

$v_t \sim N(0, q)$, $e_t \sim N(0, r)$ all independent.

Data generated with $q = 0.1$ and $r = 1$.



Priors: $q \sim InvGamma(0.01, 0.01)$, $r \sim InvGamma(0.01, 0.01)$



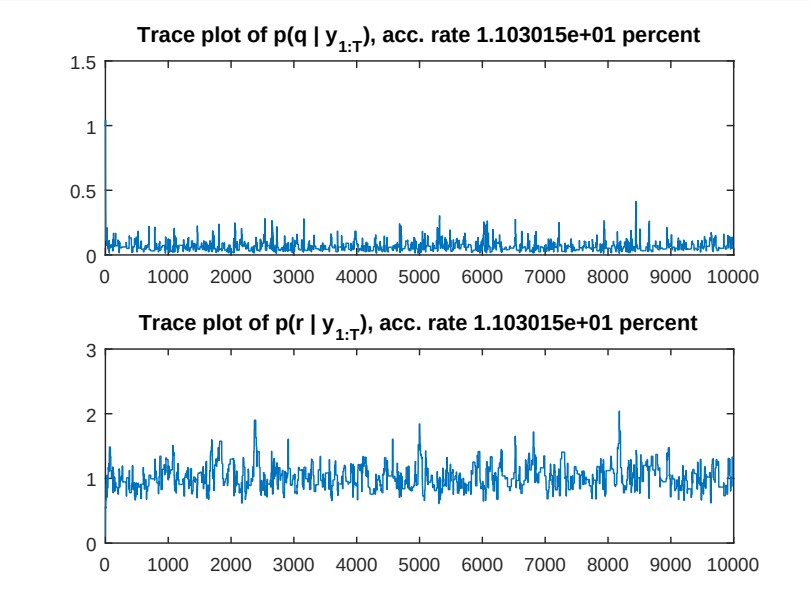
Recall true values are $q = 0.1$ and $r = 1$. The chosen priors cover both.

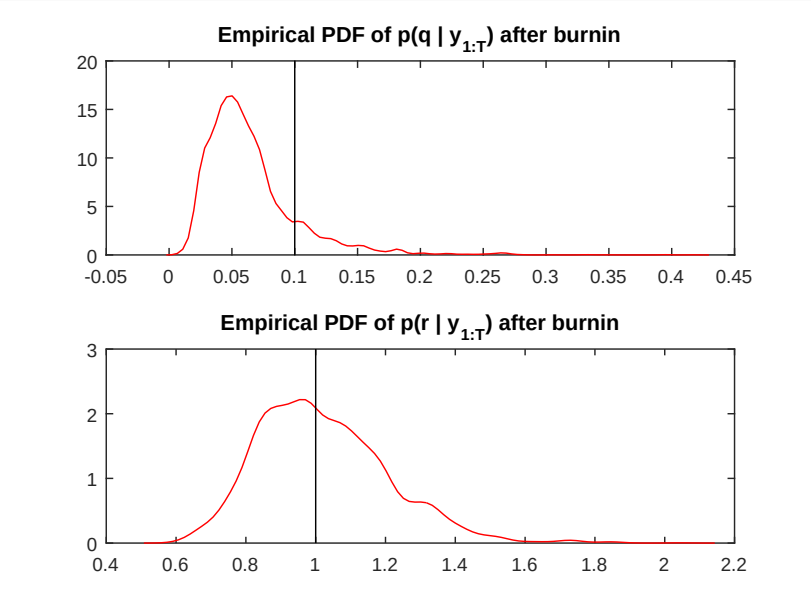
Results anticipation...

Perform an MCMC *(how about the likelihood? we'll see this later)*.

$R = 10,000$ iterations (generous burnin $= 3,000$ iterations).

Proposals: $q_{new} \sim N(q_{old}, 0.2^2)$, and similarly for r_{new}.





How do we deal with general SSM for which the exact likelihood $p(y_{1:T}|\theta)$ is unavailable? That's the case of the previous example.

For example, the previously analysed model is nonlinear in x_t and therefore the exact Kalman filter can't be used.

Also, if we were to assume that v_t and e_t are non-Gaussian, also in that case Kalman couldn't be used.

But we introduce more general methods that are able to deal with states nonlinearities and non-Gaussianity.

General Monte Carlo integration

Recall our likelihood function:

$$p(y_{1:T}|\theta) = p(y_1|\theta)\prod_{t=2}^{T} p(y_t|y_{1:t-1},\theta)$$

We can write $p(y_t|y_{1:t-1},\theta)$ as

$$p(y_t|y_{1:t-1},\theta) = \int p(y_t|x_t,\theta)p(x_t|y_{1:t-1},\theta)dx_t = \mathbb{E}(p(y_t|x_t,\theta))$$

Use Monte Carlo integration: generate N draws from $p(x_t|y_{1:t-1},\theta)$, then invoke the law of large numbers.

- produce N independent draws $x_t^i \sim p(x_t|y_{1:t-1},\theta), \quad i = 1,...,N$
- for each x_t^i compute $p(y_t|x_t^i,\theta)$
- and by LLN we have

$$\frac{1}{N}\sum_{i=1}^{N} p(y_t|x_t^i,\theta) \to \mathbb{E}(p(y_t|x_t,\theta)), \qquad N \to \infty$$

- error term is $O(N^{-1/2})$ *regardless the dimension of* x_t ☺

SMC and the bootstrap filter

But how to generate "good" draws (*particles*) $x_t^i \sim p(x_t|y_{1:t-1}, \theta)$?

Here "good" means that we want particles such that the values of $p(y_t|x_t^i, \theta)$ are not negligible ("explain" a large fraction of the integrand).

For SSM, *sequential Monte Carlo* (SMC) is the winning strategy.

We will NOT give a thorough introduction to SMC methods. We only use a few notions to solve our parameter inference problem.

[the term *particle filters* can be used interchangeably with SMC.]

Importance sampling

Let's remove for a moment the dependence on θ.

$$
\begin{aligned}
p(y_t|y_{1:t-1}) &= \int p(y_t|x_{0:t})p(x_{0:t}|y_{1:t-1})dx_{0:t} \\
&= \int p(y_t|x_t)p(x_t|x_{t-1})p(x_{0:t-1}|y_{1:t-1})dx_{0:t} \\
&= \int \frac{p(y_t|x_t)p(x_t|x_{t-1})p(x_{0:t-1}|y_{1:t-1})}{h(x_{0:t}|y_{1:t})} h(x_{0:t}|y_{1:t})dx_{0:t}
\end{aligned}
$$

where $h(\cdot)$ is an arbitrary (positive) density function called "importance density". Choose an $h(\cdot)$ "easy to simulate from".

1. simulate N samples: $\quad x^i_{0:t} \sim h(x_{0:t}|y_{1:t}), \quad i = 1, ..., N$
2. construct importance weights $w^i_t = \frac{p(y_t|x^i_t)p(x^i_t|x^i_{t-1})p(x^i_{0:t-1}|y_{1:t-1})}{h(x^i_{0:t}|y_{1:t})}$
3. $p(y_t|y_{1:t-1}) = \mathbb{E}(p(y_t|x_t)) \approx \frac{1}{N}\sum_{i=1}^N w^i_t$

Importance Sampling

Importance Sampling is an appealing and revolutionary idea for Monte Carlo integration.

However generating at each time a "cloud of particles" $x^i_{0:t}$ is not really computationally appealing, and it's not clear how to do so.

Much better to try to split the problem into a *sequential* mechanism, as t increases.

Sequential Importance Sampling

When $h(\cdot)$ is chosen in an intelligent way, an important property is the one that allows *sequential* update of weights. After some simple derivation , we have (see p. 121 in Särkkä[2] and p. 5 in Cappe et al.[3])

$$w_t^i \propto \frac{p(y_t|x_t^i)p(x_t^i|x_{t-1}^i)}{h(x_t^i|x_{0:t-1}^i, y_{1:t})}\tilde{w}_{t-1}^i, \qquad i = 1, ..., N$$

where the proportionality symbol means that we will not particularly care of the (unknown) proportionality constant, and $\tilde{w}$ denotes **normalised weights**, i.e.

$$\tilde{w}_{t-1}^i = \frac{w_{t-1}^i}{\sum_{i=1}^N w_{t-1}^i}$$

[2] Särkkä, *Bayesian filtering and smoothing*, available here.

[3] Cappe, Godsill and Moulines. Available here.

Sequential importance sampling

We can now write a sequential algorithm:

1. $t = 0$ (initialize) $x_0^i \sim \pi(x_0)$, assign $\tilde{w}_0^i = 1/N$, $i = 1, ..., N$
2. at the current t assume we have the particles x_t^i
3. From your model *propagate forward* $x_{t+1}^i \sim h(x_{t+1}|x_{0:t}^i, y_{1:t+1})$, $i = 1, ..., N$.
4. Compute (unnormalised weights)

$$w_{t+1}^i \propto \frac{p(y_{t+1}|x_{t+1}^i)p(x_{t+1}^i|x_t^i)}{h(x_{t+1}^i|x_{0:t}^i, y_{1:t+1})} \times \tilde{w}_t^i.$$

   and normalise weights $\tilde{w}_{t+1}^i = w_{t+1}^i / \sum_{i=1}^N w_{t+1}^i$
5. we can finally approximate (Creal, p. 253)

$$p(y_{t+1}|y_{1:t}) \approx \sum_{i=1}^{N} w_{t+1}^i \times \tilde{w}_t^i$$

6. set $t := t + 1$ and if $t < T$ go to step 2.

(simplified) Sequential importance sampling

Two huge problems:

- sometimes **we do not know** the transition density $p(x_t|x_{t-1})$ hence the weights cannot be computed.
- We may be unable to find a useful proposal function $h(x_t|x^i_{0:t-1}, y_{1:t})$ the depends on data.

An often useful approach (sometimes the only thing we can do) is to take

$$h(x_t|x^i_{0:t-1}, y_{1:t}) \equiv p(x_t|x_{t-1})$$

this implies a huge simplification

$$w^i_t \propto \frac{p(y_t|x^i_t)p(x^i_t|x^i_{t-1})}{h(x^i_t|x^i_{0:t-1}, y_{1:t})}\tilde{w}^i_{t-1} = \frac{p(y_t|x^i_t)p(x^i_t|x^i_{t-1})}{p(x^i_t|x^i_{t-1})}\tilde{w}^i_{t-1} = p(y_t|x^i_t)\tilde{w}^i_{t-1}$$

So we got read of the (often unavailable) transition density. **We only need to simulate from the model, not to evaluate the transition density.**

The particle degeneracy problem

Particle degeneracy occurs when at time t all but one of the importance weights w_t^i are close to zero. This implies a poor approximation to $p(y_t|y_{1:t-1})$.

Notice that when a particle gets a zero weight (or a "small" positive weight that your computer sets to zero $\rightarrow$ *numerical underflow*) that particle is doomed! Since

$$w_t^i \propto \frac{p(y_t|x_t^i)p(x_t^i|x_{t-1}^i)}{h(x_t^i|x_{0:t-1}^i, y_{1:t})}\tilde{w}_{t-1}^i.$$

if for a given i we have $\tilde{w}_{t-1}^i = 0$ particle i will have zero weight for *all subsequent times*.

The particle degeneracy problem

- For example, suppose a new data point y_t is encountered and this is an outlier.
- Then most particles will end up far away from y_t, so $p(y_t|x_t^i) \approx 0$ and hence weights will be ≈ 0.
- Those particles will *die*.
- Eventually, for a time horizon T which is long enough, all but one particle will have zero weight.
- This means particle degeneracy has affected this methodology for decades.

The Resampling idea

A life saving solution is to use *resampling with replacement* (Gordon et al.[4]).

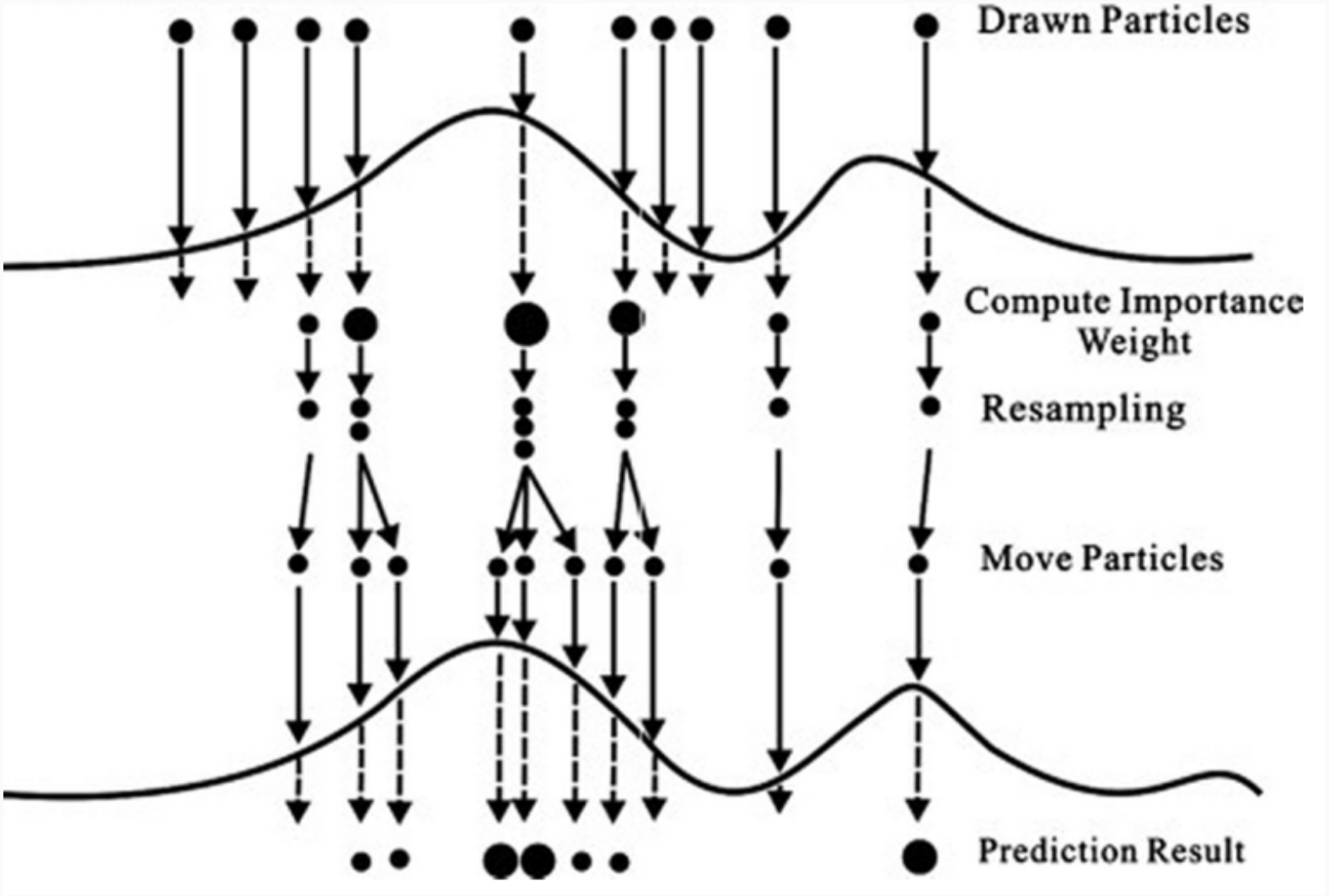
1. (normalization) set $\tilde{w}_t^i := w_t^i / \sum_i w_t^i$;
2. interpret $\tilde{w}_t^i$ as the probability to sample x_t^i from the **weighted set** $\{x_t^i, \tilde{w}_t^i, i = 1, ..., N\}$.
3. draw N times with replacement from the weighted set. Replace the old particles with the new ones $\{\tilde{x}_t^1, ..., \tilde{x}_t^N\}$.
4. Reset weights $w_t^i = 1/N$.

Since resampling is done with replacement, a particle with a large weight is likely to be drawn multiple times.

Particles with very small weights are not likely to be drawn at all. Nice!

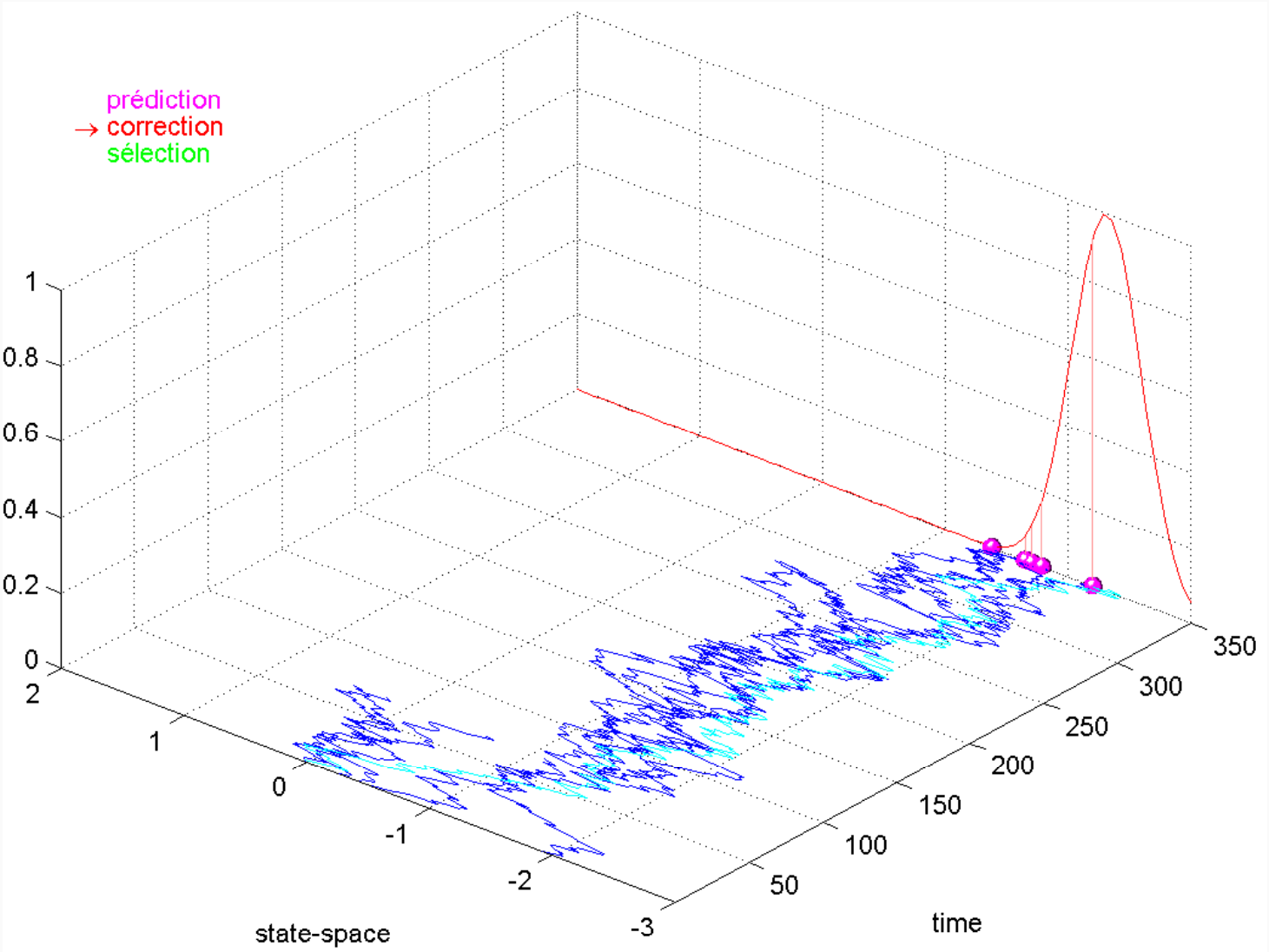
[4] Gordon, Salmond and Smith. IEEE Proceedings F. 140(2) 1993.

Propagation→weighting→resampling→propagation of resampled particles→weighting→etc



The next animation illustrates the concept of sequential importance sampling resampling with $N = 5$ particles.

- Light blue: observed trajectory (data)
- dark blue: simulation of the latent process $\{X_t\}$
- pink balls: particles x_t^i
- green balls: selected particles $\tilde{x}_t^i$ from resampling
- red curve: density $p(y_t|x_t)$


prédiction
→ correction
sélection
1
0.8
0.6
0.4
0.2
0
2
1
0
-1
-2
-3
state-space
50
100
150
200
250
300
350
time

Sequential Importance Sampling Re-sampling (SISR)

Introducing the resmpling trick into sequential importance sampling produces the SISR (sequential importance sampling with resampling) which will save our day.

Because of time, we only describe the simplest example of SISR, namely the **bootstrap filter**.

Bootstrap filter

The bootstrap filter is the simplest example of SISR.

- Choose $h(x_t^i|x_{0:t-1}^i, y_{1:t}) := p(x_t^i|x_{t-1}^i)$
- **Important**: we do NOT need to know $p(x_t^i|x_{t-1})$, we only need to be able to sample from it (forward model simulation!).
- propagate forward the particles $\rightarrow$ compute unnormalised weights w_t^i $\rightarrow$ normalise the weights $\rightarrow$ resample the particles according to normalised weights $\rightarrow$ propagate forward etc.
- In SISR **after** performing resampling reset all weights $\rightarrow \tilde{w}_{t-1}^i = \frac{1}{N}$

Thus

$$\begin{aligned} w_t^i &\propto \frac{p(y_t|x_t^i)p(x_t^i|x_{t-1}^i)}{h(x_t^i|x_{0:t-1}^i, y_{1:t})}\tilde{w}_{t-1}^i \\ &= \frac{1}{N}p(y_t|x_t^i) \propto p(y_t|x_t^i), \qquad i = 1, ..., N \end{aligned}$$

Bootstrap filter in detail

1. $t = 0$ (initialize) $x_0^i \sim \pi(x_0)$, assign $\tilde{w}_0^i = 1/N$, $i = 1, ..., N$
2. at the current t assume we have the N weighted particles $\{x_t^i, \tilde{w}_t^i\}_{i=1}^N$
3. from the current $\{x_t^i, \tilde{w}_t^i\}_{i=1}^N$, resample particles with replacement N times to obtain $\{\tilde{x}_t^i, i = 1, ..., N\}$.
4. From your model *propagate forward* $x_{t+1}^i \sim p(x_{t+1}|\tilde{x}_t^i)$, $i = 1, ..., N$.
5. Compute $w_{t+1}^i = p(y_{t+1}|x_{t+1}^i)$ and normalise weights $\tilde{w}_{t+1}^i = w_{t+1}^i / \sum_{i=1}^N w_{t+1}^i$
6. set $t := t + 1$ and if $t < T$ go to step 2.

So the bootstrap filter by et al. (1993)[5] easily provide what we need!

- $\hat{p}(y_t|y_{1:t-1};\theta) = \frac{1}{N}\sum_{i=1}^{N} w_t^i$
- Finally a **likelihood approximation**:

$$\hat{p}(y_{1:T}|\theta) = \hat{p}(y_1)\prod_{t=2}^{T}\hat{p}(y_t|y_{1:t-1};\theta)$$

We obtain:

- **approximate** maximum likelihood

$$\theta_{mle} = \text{argmax}_\theta \hat{p}(y_{1:T};\theta)$$

  or

- **Bayesian inference** by using $\hat{p}(y_{1:T}|\theta)$ inside Metropolis-Hastings.

[5] Gordon, Salmond and Smith. IEEE Proceedings F. 140(2) 1993.

Resampling particles using some software

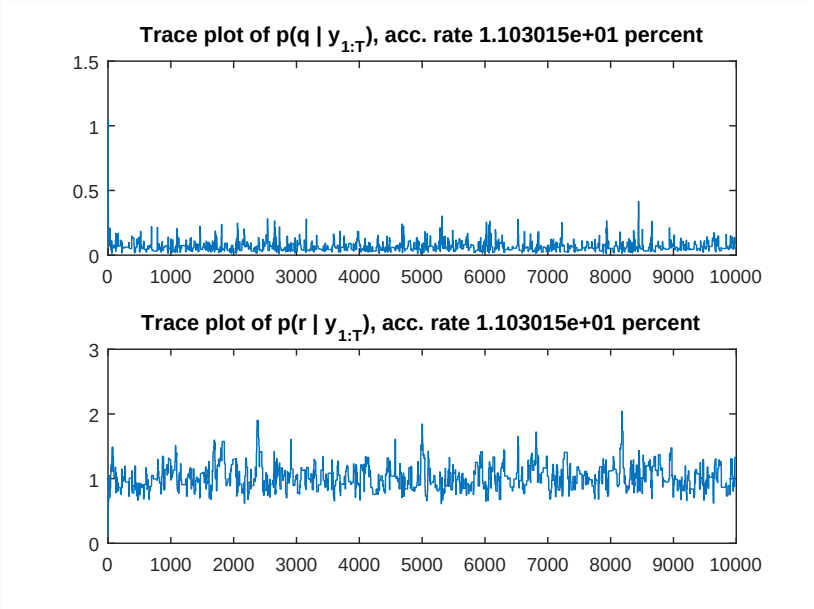
To resample particles you can make use of built-in routines. Always remember that in this context we wish to sample **with replacement**. Below `normw` denotes the normalised weights $\tilde{w}$ at a given time and `xres` is the vector of resampled particles obtained from the current `x`.

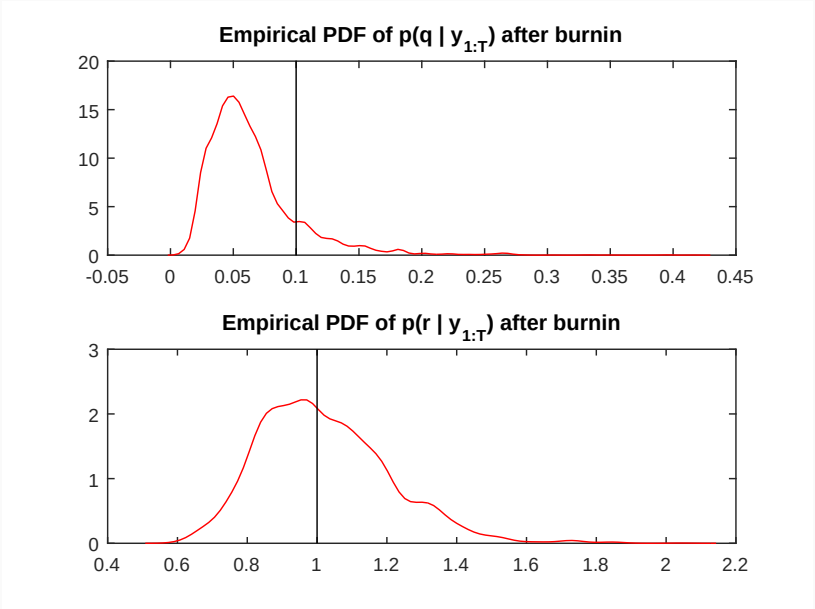
- In Matlab you can use, say, `xres = randsample(x,N,1,normw)`, this will create a new vector xres of length N of values resampled from x with replacement.
- similarly, in R you can use `xres <- sample(x, size = N, replace = TRUE, prob = normw)`

Back to the nonlinear SSM example

We can now comment on how we obtained previously shown results (reproposed here). We used the bootstrap filter with $N = 500$ particles and $R = 10,000$ MCMC iterations.

- forward propagation: $x_{t+1} = 0.5x_t + 25\frac{x_t}{(1+x_t^2)} + 8\cos(1.2t) + v_{t+1}$
- $w_{t+1}^i = N(0.05(x_{t+1}^i)^2, r)$





Numerical issues with particle degeneracy

When coding your algorithms try to consider the following **before normalizing weights**

- code unnormalised weights on log-scale: e.g. when $w_t^i = N(y_t|x_t^i)$ the `exp()` in the Gaussian pdf will likely produce an underflow ($w_t^i = 0$) for x_t^i far from y_t.
  Solution: reason in terms of `logw` instead of `w`.
- However afterwards we necessarily have to go back to `w:=exp(logw)` then normalize and the above might still not be enough.
  Solution: subtract the maximum (log)weight from each (log)weight, e.g. set `logw:=logw-max(logw)`. This is totally ok, the importance of particles is unaffected, weights are only scaled by the same constant `c=max(logw)`.

Safe (log)likelihood computation

As usual, better compute the (approximate) loglikelihood instead of the likelihood:

$$\log \hat{p}(y_{1:T}|\theta) = \sum_{t=1}^{T} \left(-\log N + \log \sum_{i=1}^{N} w_t^i \right)$$

At time t set $c_t = \max\{\log w_t^i, i = 1, ..., N\}$ and set
$w_t^{i*} := \exp(\log w_t^i - c_t)$

Once all the w_t^{i*} are available compute
$\log \sum_{i=1}^{N} w_t^i = c_t + \log \sum_{i=1}^{N} w_t^{i*}$.

(the latter follows from $w_t^i = w_t^{i*} \exp\{c_t\}$ which you don't need to evaluate)

Appendix

Multinomial resampling

This is the simplest to explain (not the most efficient though) resampling scheme.

At time t we wish to sample from a population of weighted particles $(x_t^i, \tilde{w}_t^i, i = 1, ..., N)$. What we actually do is to sample N times particle *indeces* with replacement from the population $(i, \tilde{w}_t^i, i = 1, ..., N)$. This will be a sample of size N from a *multinomial distribution*.

Pick at particle from the "urn", the larger its probability $\tilde{w}_t^i$ the more likely it will be picked. Record its index i and put it back in the urn. Repeat for a total of N times.

To code the sampling procedure we just need to recall the **inverse transform method**.
For a generic random variable X, let F_X be an invertible cdf. We can sample an x from F_X using $x := F_X^{-1}(u)$, with $u \sim U(0, 1)$.

For example let's start from a simple example of multinomial distribution, the Bernoulli distribution.

$X \in \{0, 1\}$ with $p = \mathbb{P}(X = 1)$, $1 - p = \mathbb{P}(X = 0)$. Then

$$F_X(x) = \begin{cases} 0 & x < 0 \\ 1 - p & 0 \leqslant x < 1 \\ 1 & x \geqslant 1 \end{cases} \tag{1}$$

Draw the "stair" represented by the plot of F_X. Generate a $u \sim U(0, 1)$ and "hit the stair's steps". If $0 < u \leqslant 1 - p$ then set $x := 0$ and if $u > 1 - p$ set $x := 1$.

For the multinomial case it is a simple generalization. Drop time t and set $\tilde{w}^i = p_i$. F_X is a stair with N steps. Shoot a $u \sim U(0, 1)$ and return index i

$$i := \min\Big\{i' \in \{1, ..., N\}; \big(\sum_{i=1}^{i'} p_i\big) - u \geqslant 0\Big\}.$$

Cool reads on Bayesian methods (titles are linked)

- You have an engineeristic/signal processing background: check S. Särkkä "Bayesian Filtering and Smoothing" (free PDF from the author!)
- You are a data-scientist: check K. Murphy "Machine Learning: a probabilistic perspective".
- You are a theoretical statistician: C. Robert "The Bayesian Choice".
- You are interested in bioinformatics/systems biology: check D. Wilkinson "Stochastic Modelling for Systems Biology, 2ed.".
- You are interested in inference for SDEs with applications to life sciences: check the book by Wilkinson above and C. Fuchs "Inference for diffusion processes".

Cool reads on Bayesian methods (titles are linked)

- You are a computational statistician: check "Handbook of MCMC". Older (but excellent) titles are: J. Liu "Monte Carlo strategies in Scientific Computing" and Casella-Robert "Monte Carlo Statistical Methods".
- You want a practical hands-on and (almost) maths free introduction: check "The BUGS Book" and "Doing Bayesian Data Analysis".
- "Statistical Rethinking"

Here follow appendix slides to refresh some concepts if needed.

Markov property of hidden states

Briefly: X_t (and actually the whole future $X_{t+1}, X_{t+2}, \ldots$) given X_{t-1} is independent from anything that has happened before time $t-1$:

$$p(x_t|x_{0:t-1}, y_{1:t-1}) = p(x_t|x_{t-1}), \qquad t = 1, \ldots, T$$

Also, *past is independent of the future given the present*:

$$p(x_{t-1}|x_{t:T}, y_{t:T}) = p(x_{t-1}|x_t)$$

Conditional independence of measurements

The current measurement y_t given x_t is conditionally independent of the measurement and state histories:

$$p(y_t|x_{0:t}, y_{1:t-1}) = p(y_t|x_t).$$

The Markov property on $\{X_t\}$ and the conditional independence on $\{Y_t\}$ are the key ingredients for defining an HMM or SSM.