CHALMERS
UNIVERSITY OF TECHNOLOGY

L9.1: An introduction to particle filtering

Lars Hammarstrand

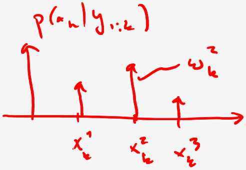
Signal Processing Group
Department of Electrical Engineering
Chalmers University of Technology, Sweden

- Gaussian filtering is a useful technique to perform nonlinear filtering.
- Limitations: Gaussian filtering methods do not perform well when
  - the models are highly nonlinear,
  - when the posterior distribution is significantly non-Gaussian, e.g., a multimodal density.
- For such problems we need a different type of approximation to the posterior density!

Basic idea

- Use a non-parametric representation

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)})$$



where $\mathbf{x}_k^{(i)}$ are particles and $w_k^{(i)}$ are associated weights.

- Filtering is (essentially) performed by
  1. propagating $\mathbf{x}_{k-1}^{(i)} \to \mathbf{x}_k^{(i)}$ over time,
  2. updating the weights, $w_k^{(i)}$.
- Basic version: $\mathbf{x}_k^{(i)} \sim p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)})$, $w_k^{(i)} \propto w_{k-1}^{(i)} p(\mathbf{y}_k|\mathbf{x}_k^{(i)})$.

After this lecture you should be able to

- explain the concepts of Monte Carlo sampling and importance sampling,
- describe what particle degeneracy is and why resampling is useful,
- and implement a particle filter.

CHALMERS
UNIVERSITY OF TECHNOLOGY

L9.2: Monte Carlo (MC) approximations and Importance Sampling (IS)

Lars Hammarstrand

Signal Processing Group
Department of Electrical Engineering
Chalmers University of Technology, Sweden

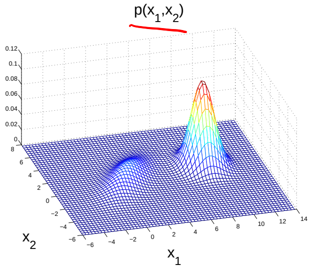
Monte Carlo approximations

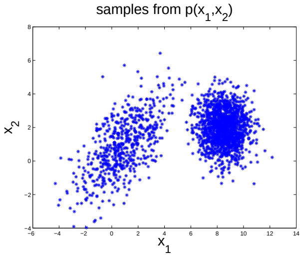
Two perspectives on Monte Carlo approximation

Given independent samples $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)} \sim p(\mathbf{x})$ we can approximate

$$\mathbb{E}[\mathbf{g}(\mathbf{x})] \approx \frac{1}{N} \sum_{i=1}^{N} \mathbf{g}(\mathbf{x}^{(i)}) \qquad (1)$$

$$p(\mathbf{x}) \approx \frac{1}{N} \sum_{i=1}^{N} \delta(\mathbf{x} - \mathbf{x}^{(i)}) \qquad (2)$$





Remarks on Monte Carlo approximations:

- non-parametric approximation to $p(\mathbf{x})$.
- approximate all kinds of densities, $p(\mathbf{x})$. Very flexible!
- does not suffer from the *curse of dimensionality*, e.g., $\hat{\mu}$

$$\text{Cov}(\hat{\mu}) = \text{Cov}\Bigg(\underbrace{\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}^{(i)}}_{\hat{\mu}}\Bigg) = \frac{1}{N}\text{Cov}(\mathbf{x})$$

  independently on dim$(\mathbf{x})$!
- Weakness: it is often difficult to generate samples from $p(\mathbf{x})$.

Importance sampling

What can we do when it is difficult to sample from $p(\mathbf{x})$?

Importance sampling

- Generate samples, $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}$, from a proposal density $q(\mathbf{x})$:

$$\mathbb{E}_{p(\mathbf{x})}[\mathbf{g}(\mathbf{x})] = \int \underbrace{g(x) \cdot \frac{p(x)}{q(x)}}_{\tilde{g}(x)} q(x)\,dx \approx \frac{1}{N} \sum_{i=1}^{N} g(x^{(i)}) \frac{p(x^{(i)})}{q(x^{(i)})}$$

$$\propto \sum_{i=1}^{N} \underbrace{\frac{p(x^{(i)})}{q(x^{(i)})}}_{\tilde{w}^{(i)}} \cdot g(x^{(i)})$$

Importance sampling approximation to $p(\mathbf{x})$

- Generate samples, $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}$, from $q(\mathbf{x})$ and set

$$p(\mathbf{x}) \approx \sum_{i=1}^{N} w^{(i)} \delta(\mathbf{x} - \mathbf{x}^{(i)})$$

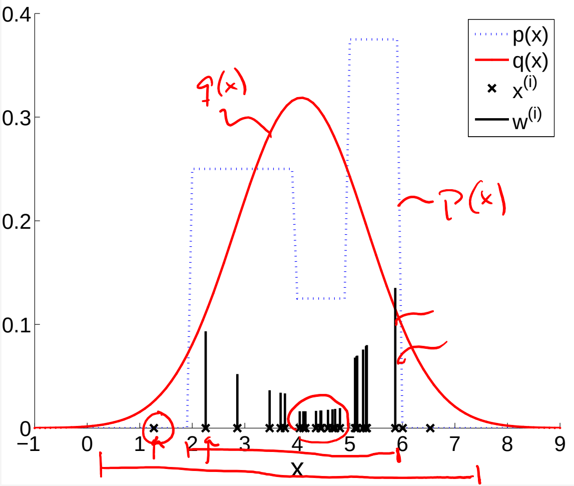
where

$$w^{(i)} = \frac{\tilde{w}^{(i)}}{\sum_{n=1}^{N} \tilde{w}^{(n)}} \quad \text{and} \quad \tilde{w}^{(i)} = \frac{p(\mathbf{x}^{(i)})}{q(\mathbf{x}^{(i)})}.$$

- Importance sampling is a flexible and powerful tool.
- It can perform very well as long as:
  1. it is easy to sample from $q(\mathbf{x})$,
  2. the support of $q(\mathbf{x})$ contains the support of $p(\mathbf{x})$,
  3. $q(\mathbf{x})$ is "similar" to $p(\mathbf{x})$.

Example – Importance sampling

- Approximate $p(x)$ using N independent samples from $q(x) = \mathcal{N}(x; 4, 1.5^2)$.



CHALMERS
UNIVERSITY OF TECHNOLOGY

L9.3: Sequential Importance Sampling (SIS)

Lars Hammarstrand

Signal Processing Group
Department of Electrical Engineering
Chalmers University of Technology, Sweden

- Objective: to recursively and accurately approximate the filtering density, $p(\mathbf{x}_k | \mathbf{y}_{1:k})$.
- Assumption: both the motion and measurement models

$$p(\mathbf{x}_k | \mathbf{x}_{k-1}) \qquad \text{and} \qquad p(\mathbf{y}_k | \mathbf{x}_k)$$

  can be easily evaluated point-wise.
- A common example is

$$\begin{aligned} \mathbf{x}_k &= f(\mathbf{x}_{k-1}) + \mathbf{q}_{k-1}, & \mathbf{q}_{k-1} &\sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{k-1}) \\ \mathbf{y}_k &= h(\mathbf{x}_k) + \mathbf{r}_k & \mathbf{r}_k &\sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k), \end{aligned}$$

  where, e.g., $p(\mathbf{x}_k | \mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{x}_k; f(\mathbf{x}_{k-1}), \mathbf{Q}_{k-1})$ is generally easy to evaluate for any values of $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$.

- Particle filters are also known as *sequential importance resampling* or *sequential Monte Carlo*.
- The basis of these methods is an algorithm called sequential importance sampling (SIS).

Standard SIS algorithm

- For $i = 1, \ldots, N$ and at each time k:
  - Draw $\mathbf{x}_k^{(i)} \sim q(\mathbf{x}_k | \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_k)$.
  - Compute weights

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(\mathbf{y}_k | \mathbf{x}_k^{(i)}) p(\mathbf{x}_k^{(i)} | \mathbf{x}_{k-1}^{(i)})}{q(\mathbf{x}_k^{(i)} | \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_k)}.$$

  - Normalize the weights.

- We then approximate $p(\mathbf{x}_k | \mathbf{y}_{1:k}) \approx \sum_{i=1}^N w_k^{(i)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)})$.

 CHALMERS

- Assuming that we describe our posterior using the following approximation $p(\mathbf{x}_k|\mathbf{y}_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)})$. What is then the MMSE estimate of $\mathbf{x}_k$?
  - $\hat{\mathbf{x}}_k = \sum_i^N w_k^{(i)} \mathbf{x}_k^{(i)}$
  - $\hat{\mathbf{x}}_k = \mathbf{x}_k^{(j)}$, where $j = \arg\max_i w_k^{(i)}$
  - $\hat{\mathbf{x}}_k = \frac{1}{N} \sum_i^N \mathbf{x}_k^{(i)}$
  - It is not possible to calculate a MMSE estimate from this approximation.

$$\hat{x}_k = E\{x_k | y_k\} = \int x_k \sum_{i=1}^{N} \omega_k^{(i)} \delta(x_k - x_k^{(i)}) dx_k$$

$$= \sum_{i=1}^{N} \omega_k^{(i)} x_k^{(i)}$$

Derivation - Basic strategy

Recursively at time $k = 1, 2, \ldots$

1. Draw particles

$$\mathbf{x}_{0:k}^{(i)} \sim q(\mathbf{x}_{0:k}|\mathbf{y}_{1:k})$$

2. Update weights

$$w_k^{(i)} \propto \frac{p(\mathbf{x}_{0:k}^{(i)}|\mathbf{y}_{1:k})}{q(\mathbf{x}_{0:k}^{(i)}|\mathbf{y}_{1:k})}$$

Comments on drawing particles:

- Let us assume that

$$q(\mathbf{x}_{0:k}|\mathbf{y}_{1:k}) = q(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{y}_k)q(\mathbf{x}_{0:k-1}|\mathbf{y}_{1:k-1}).$$

- we generate $\mathbf{x}_{0:k-1}^{(i)} \sim q(\mathbf{x}_{0:k-1}|\mathbf{y}_{1:k-1})$ at time $k-1$,
- it is sufficient to generate $\mathbf{x}_k^{(i)} \sim q(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)}, \mathbf{y}_k)$ and append that to $\mathbf{x}_{1:k-1}^{(i)}$!

- It remains to derive the expression for the weights:

$$w_k^{(i)} \propto \frac{p(\mathbf{x}_{0:k}^{(i)}|\mathbf{y}_{1:k})}{q(\mathbf{x}_{0:k}^{(i)}|\mathbf{y}_{1:k})}$$

$\propto p(x_{0:k-1}^{(i)}, x_k^{(i)}, y_k | y_{1:k-1}) = p(y_k | x_k^{(i)}) p(x_k^{(i)} | x_{k-1}^{(i)}) p(x_{0:k-1}^{(i)} | y_{1:k-1})$

$$\propto \frac{p(\mathbf{y}_k|\mathbf{x}_k^{(i)})p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})}{q(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)}, \mathbf{y}_k)} \underbrace{\frac{p(\mathbf{x}_{0:k-1}^{(i)}|\mathbf{y}_{1:k-1})}{q(\mathbf{x}_{0:k-1}^{(i)}|\mathbf{y}_{1:k-1})}}_{w_{k-1}^{(i)}}$$

$$\propto w_{k-1}^{(i)} \frac{p(\mathbf{y}_k|\mathbf{x}_k^{(i)})p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})}{q(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)}, \mathbf{y}_k)}$$

- We have thus derived the SIS algorithm:

Standard SIS algorithm

- For $i = 1, \ldots, N$ and at each time k:
  - Draw $\mathbf{x}_k^{(i)} \sim q(\mathbf{x}_k | \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_k)$.
  - Compute weights

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(\mathbf{y}_k | \mathbf{x}_k^{(i)}) p(\mathbf{x}_k^{(i)} | \mathbf{x}_{k-1}^{(i)})}{q(\mathbf{x}_k^{(i)} | \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_k)}.$$

  - Normalize the weights.

- A simple choice of importance density is

$$q(\mathbf{x}_k | \mathbf{x}_{k-1}, \cancel{\mathbf{y}_k}) = p(\mathbf{x}_k | \mathbf{x}_{k-1})$$

for which $w_k^{(i)} \propto w_{k-1}^{(i)} p(\mathbf{y}_k | \mathbf{x}_k^{(i)})$.

Example - Nonlinear filter benchmark

- The following is a common benchmark for nonlinear filters

$$x_k = \frac{x_{k-1}}{2} + \frac{25 x_{k-1}}{1 + x_{k-1}^2} + 8\cos(1.2k) + q_{k-1}$$

$$y_k = \frac{x_k^2}{20} + r_k$$

where $q_{k-1} \sim \mathcal{N}(0, 10)$ and $r_k \sim \mathcal{N}(0, 1)$.

- Let us see how the above filter performs on this challenging problem!

CHALMERS
UNIVERSITY OF TECHNOLOGY

L9.4: Sequential Importance Resampling (SIR)

Lars Hammarstrand

Signal Processing Group
Department of Electrical Engineering
Chalmers University of Technology, Sweden

- One can show that all SIS filters suffer from degeneracy:

  after a few time steps all but one particle will have negligible weight.

- Consequences of degeneracy:
  - the filter believes that it knows $\mathbf{x}_k$ exactly,
  - we obtain very poor state estimates,
  - most of our calculations are wasted on insignificant particles.

  These are very serious drawbacks!

- A key technique to improve performance is resampling.

- Challenge: we have $p(\mathbf{x}_k|\mathbf{y}_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)})$ where most weights $w_k^{(i)}$ are very small.

Idea: use Monte Carlo sampling

- Generate independent samples $\tilde{\mathbf{x}}_k^{(1)}, \ldots, \tilde{\mathbf{x}}_k^{(N)}$ from $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ and set

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \approx \sum_{i=1}^{N} \frac{1}{N} \delta(\mathbf{x}_k - \tilde{\mathbf{x}}_k^{(i)}).$$

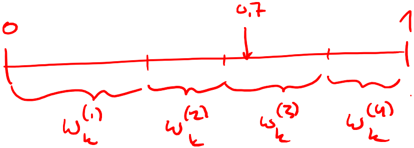
- After resampling we get
  - equal weights (they are all $1/N$),
  - multiple copies of high probability particles.

Resampling algorithm

1) Draw N samples with replacement from $\mathbf{x}_k^{(1)}, \mathbf{x}_k^{(2)}, \ldots, \mathbf{x}_k^{(N)}$, where the probability of selecting $\mathbf{x}_k^{(i)}$ is $w_k^{(i)}$.

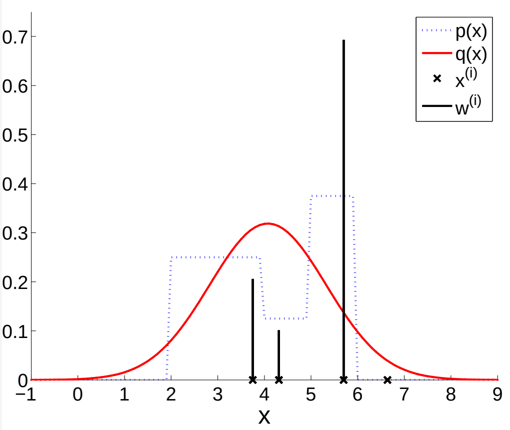
2) Replace the old sample set with the new one and set all weights to $1/N$.

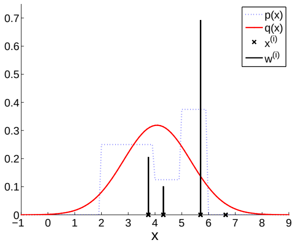
- A few remarks:
  - We use $\mathbf{x}_k^{(i)}$ and $w_k^{(i)}$ to denote the particles and their weights also after resampling.
  - We can use samples from the uniform distribution, unif[0, 1], to draw samples from the discrete distribution $p(\mathbf{x}_k | \mathbf{y}_{1:k})$.



Self-assessment – Resampling

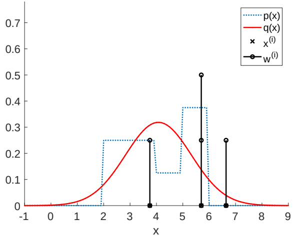
- Perform resampling on the density to the right and illustrate the result.
- Assume that the numbers 0.65, 0.03, 0.84 and 0.93 are drawn from unif[0, 1].

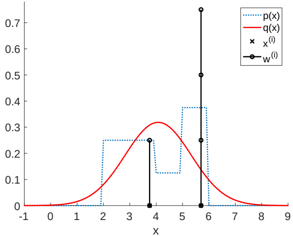


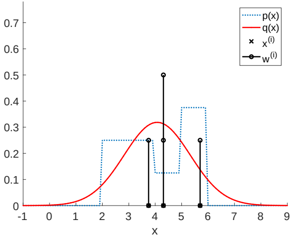


- Perform resampling on the density to the right and illustrate the result.
- Assume that the numbers 0.65, 0.03, 0.84 and 0.93 are drawn from unif[0, 1].

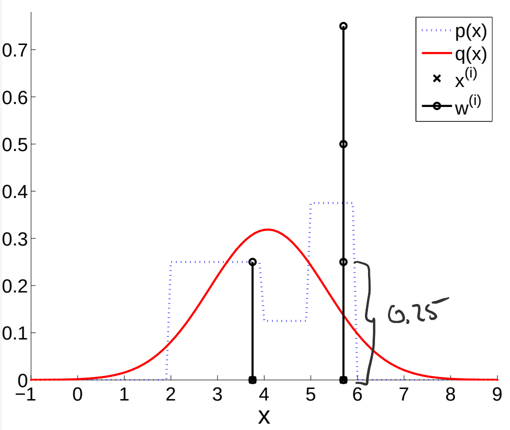
- Choose the figure below that illustrates the resampled particles:



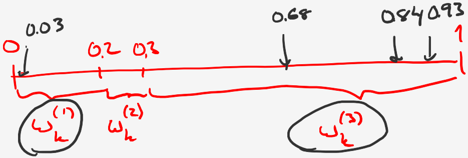




Self-assessment – Solution



- If particles where ordered in ascending order, $x^{(1)} < \cdots < x^{(4)}$, resampling gives $x^{(1)} = 3.8$ and $x^{(2)} = x^{(3)} = x^{(4)} = 5.7$.



- Resampling costs some calculations and introduces some errors, but improves performance immensely over time.
- An estimate for the *effective number of particles* is

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} \left(w_k^{(i)}\right)^2}.$$

- Many algorithms only resample when N_{eff} is below some threshold, e.g., $N/4$.

CHALMERS
UNIVERSITY OF TECHNOLOGY

L9.5: Choice of importance distribution

Lars Hammarstrand

Signal Processing Group
Department of Electrical Engineering
Chalmers University of Technology, Sweden

- A carefully selected importance distribution, $q(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{y}_k)$, can slow down the degeneracy and improve performance.

Intuition: if most particles are placed in "high probability regions" there is less need to get rid of useless particles.

Optimal importance density

- The optimal importance density is

$$q(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{y}_k) = p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{y}_k).$$

- Unfortunately, in most nonlinear settings, $p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{y}_k)$, is difficult to both draw samples from and to evaluate.

- We can approximate $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{y}_k)$ using, e.g., linearization.

- The most common choice is still, $q(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{y}_k) = p(\mathbf{x}_k|\mathbf{x}_{k-1})$, and the bootstrap algorithm.

The bootstrap PF

At each time k:

- Draw $\mathbf{x}_k^{(i)} \sim p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)})$, for $i = 1, \ldots, N$.
- Calculate $w_k^{(i)} \propto w_{k-1}^{(i)} p(\mathbf{y}_k|\mathbf{x}_k^{(i)})$ and normalize to 1.
- Resample.

- Note: if we resample at every time step, we get $w_k^{(i)} \propto p(\mathbf{y}_k|\mathbf{x}_k^{(i)})$ since $w_{k-1}^{(i)} = 1/N \;\; \forall i$ after resampling.
- Note 2: the Auxiliary PF (APF) is variation of the SIR algorithm that makes use of $\mathbf{y}_k$.

- Particle filters (PFs) can handle highly nonlinear and non-Gaussian systems.
- Particle filters are asymptotically exact as you increase N.
- The complexity is roughly $O(N)$ but the gain in performance flattens out as you increase N.
- Unfortunately, PFs suffer from the curse of dimensionality and are intractable in higher dimensions.

- The output from a PF is an approximation

$$p(\mathbf{x}_k | \mathbf{y}_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)}) \iff \Pr\{x_k = x' | y_{1:k}\} = \begin{cases} \omega_k^{(i)} & \text{if } x' = x_k^{(i)} \\ 0 & \text{otherwise} \end{cases}$$

which implies that

$$\mathbb{E}\left[\mathbf{g}(\mathbf{x}_k) | \mathbf{y}_{1:k}\right] \approx \sum_{i=1}^{N} w_k^{(i)} \mathbf{g}(\mathbf{x}_k^{(i)}).$$

CHALMERS
UNIVERSITY OF TECHNOLOGY

L9.6: Rao-Blackwellized Particle Filter

Lars Hammarstrand

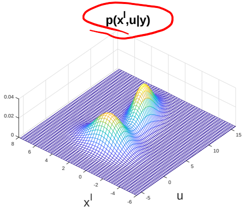
Signal Processing Group
Department of Electrical Engineering
Chalmers University of Technology, Sweden

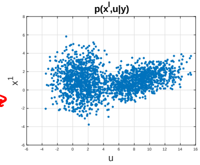
- Background:
  - particle filters are intractable in high dimensions.
  - many systems are linear in some dimensions.

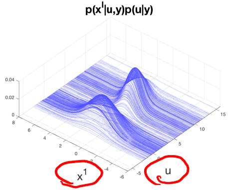
- Idea 1: *"combine a particle filter for the nonlinear states with a Kalman filter for the linear states"*.

- Idea 2: If $\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_k^l \\ \mathbf{u}_k \end{bmatrix}$ where $\mathbf{x}_k^l$ and $\mathbf{u}_k$ are the linear and nonlinear states:

$$p(\mathbf{x}_k^l, \mathbf{u}_{1:k} | \mathbf{y}_{1:k}) = \underbrace{p(\mathbf{x}_k^l | \mathbf{u}_{0:k}, \mathbf{y}_{1:k})}_{\text{Gaussian}} \underbrace{p(\mathbf{u}_{0:k} | \mathbf{y}_{1:k})}_{\text{Particle filter}}$$







- Assuming we have $\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_k^l \\ \mathbf{u}_k \end{bmatrix}$, Rao-Blackwellized particle filters are often used for models on the form

$$\mathbf{x}_k^l = f_{k-1}^l(\mathbf{u}_{k-1}) + \mathbf{A}_{k-1}^l(\mathbf{u}_{k-1})\mathbf{x}_{k-1}^l + \mathbf{q}_{k-1}^l$$

$$\mathbf{u}_k = f_{k-1}^u(\mathbf{u}_{k-1}) + \mathbf{A}_{k-1}^u(\mathbf{u}_{k-1})\mathbf{x}_{k-1}^l + \mathbf{q}_{k-1}^u$$

$$\mathbf{y}_k = h_k(\mathbf{u}_k) + \mathbf{H}_k(\mathbf{u}_k)\mathbf{x}_k^l + \mathbf{r}_k$$

where all the noises are Gaussian.

Bearing only tracking

- Bearing only tracking with a constant velocity motion in 2D.
  What is $\mathbf{x}_k^l$, $\mathbf{u}_k^l$ and $\mathbf{y}_k$ in this example?

- $\mathbf{x}_k^l$: position, $\mathbf{u}_k^l$: velocity, $\mathbf{y}_k$: bearing to target
- $\mathbf{x}_k^l$: velocity, $\mathbf{u}_k^l$: position, $\mathbf{y}_k$: bearing to target
- $\mathbf{x}_k^l$: velocity, $\mathbf{u}_k^l$: bearing to target, $\mathbf{y}_k$: position
- $\mathbf{x}_k^l$: position, $\mathbf{u}_k^l$: bearing to target, $\mathbf{y}_k$: bearing to target

Bearing only tracking – system models

- Let us denote our state vector $\mathbf{x}_k = [\underbrace{x_k^1, x_k^2}_{u_k}, \underbrace{\dot{x}_k^1, \dot{x}_k^2}_{x^l_k}]^T$, the system models can then be written as:

$$x^l_k = \begin{bmatrix} \dot{x}_k^1 \\ \dot{x}_k^2 \end{bmatrix} = \begin{bmatrix} \dot{x}_{k-1}^1 \\ \dot{x}_{k-1}^2 \end{bmatrix} + \mathbf{q}_{k-1}^l = x^l_{k-1} + q^l_{k-1}$$

$$u_k = \begin{bmatrix} x_k^1 \\ x_k^2 \end{bmatrix} = \begin{bmatrix} x_{k-1}^1 \\ x_{k-1}^2 \end{bmatrix} + T \begin{bmatrix} \dot{x}_{k-1}^1 \\ \dot{x}_{k-1}^2 \end{bmatrix} + \mathbf{q}_{k-1}^u = u_{k-1} + T \cdot x^l_k + q^u_{k-1}$$

$$\mathbf{y}_k = \operatorname{atan}_2(x_k^2, x_k^1) + \mathbf{r}_k$$

where $\mathbf{r}_k \sim \mathcal{N}(0, \sigma_r^2)$ and $\mathbf{q}_k = \begin{bmatrix} \mathbf{q}_k^u \\ \mathbf{q}_k^l \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \frac{T^2}{2}\mathbf{I} \\ T\mathbf{I} \end{bmatrix} \begin{bmatrix} \sigma_q^2 & 0 \\ 0 & \sigma_q^2 \end{bmatrix} \begin{bmatrix} \frac{T^2}{2}\mathbf{I} \\ T\mathbf{I} \end{bmatrix}^T\right)$

- One recursion of the Rao-Blackwellized particle filter contains five steps:

$p(\mathbf{x}^l_{k-1}\|\mathbf{u}_{0:k-1},\mathbf{y}_{1:k-1})p(\mathbf{u}_{0:k-1}\|\mathbf{y}_{1:k-1}) \rightarrow p(\mathbf{x}^l_k\|\mathbf{u}_{0:k},\mathbf{y}_{1:k})p(\mathbf{u}_{0:k}\|\mathbf{y}_{1:k})$	
1) PF-pred:	$p(\mathbf{u}_{1:k-1}\|\mathbf{y}_{1:k-1}) \rightarrow p(\mathbf{u}_{1:k}\|\mathbf{y}_{1:k-1})$
2) KF, dyn. upd.:	$p(\mathbf{x}^l_{k-1}\|\mathbf{u}_{1:k-1},\mathbf{y}_{1:k-1}) \rightarrow p(\mathbf{x}^l_{k-1}\|\mathbf{u}_{1:k},\mathbf{y}_{1:k-1})$
3) KF-pred:	$p(\mathbf{x}^l_{k-1}\|\mathbf{u}_{1:k-1},\mathbf{y}_{1:k-1}) \rightarrow p(\mathbf{x}^l_k\|\mathbf{u}_{1:k-1},\mathbf{y}_{1:k-1})$
4) PF, update:	$p(\mathbf{u}_{1:k}\|\mathbf{y}_{1:k-1}) \rightarrow p(\mathbf{u}_{1:k}\|\mathbf{y}_{1:k})$
5) KF, meas. upd.:	$p(\mathbf{x}^l_k\|\mathbf{u}_{1:k},\mathbf{y}_{1:k-1}) \rightarrow p(\mathbf{x}^l_k\|\mathbf{u}_{1:k},\mathbf{y}_{1:k})$

$u_k^{(i)} = u_{k-1}^{(i)} + T\cdot x_k^l + q_{k-1}^u$

$u_k^{(i)} - u_{k-1}^{(i)} = T x_k^l + q_{k-1}^u$

$x_k^l = x_{k-1}^l + q_{k-1}^l$

$w_k^{(i)} \propto w_{k-1}^{(i)} \cdot p(y_k | u_k)$

- **Note:**
  - Step 2) makes use of the motion model for $\mathbf{u}_k$ to update $\mathbf{x}^l_{k-1}$.
  - The linear states are marginalized from step 1) and 4), similarly to how we normally handle noise.

Bearing only tracking – system models

- Let us denote our state vector $\mathbf{x}_k = [x_k^1, x_k^2, \dot{x}_k^1, \dot{x}_k^2]^T$, the system models can then be written as:

$$\begin{bmatrix} \dot{x}_k^1 \\ \dot{x}_k^2 \end{bmatrix} = \begin{bmatrix} \dot{x}_{k-1}^1 \\ \dot{x}_{k-1}^2 \end{bmatrix} + \mathbf{q}_{k-1}^l$$

$$\begin{bmatrix} x_k^1 \\ x_k^2 \end{bmatrix} = \begin{bmatrix} x_{k-1}^1 \\ x_{k-1}^2 \end{bmatrix} + T \begin{bmatrix} \dot{x}_{k-1}^1 \\ \dot{x}_{k-1}^2 \end{bmatrix} + \mathbf{q}_{k-1}^u$$

$$\mathbf{y}_k = \mathrm{atan}_2(x_k^2, x_k^1) + \mathbf{r}_k$$

where $\mathbf{r}_k \sim \mathcal{N}(0, (\frac{\pi}{180})^2)$ & $\mathbf{q}_k = \begin{bmatrix} \mathbf{q}_k^u \\ \mathbf{q}_k^l \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \frac{T^2}{2}\mathbf{I} \\ T\mathbf{I} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{T^2}{2}\mathbf{I} \\ T\mathbf{I} \end{bmatrix}^T\right)$

Concluding remarks:

- Rao-Blackwellized particle filters are useful to reduce the number of particles.
- These filters enable us to handle higher dimensions than normal PFs.
- They are particularly useful if Kalman gains and posterior covariances are independent of the nonlinear states
  $\Rightarrow$ sufficient to compute them one time in each recursion.