University of Michigan - NAME 568/EECS 568/ROB 530 **Winter 2022**

Note: Particle Filtering

Lecturer and Author: Maani Ghaffari

1 Nonlinear Dynamic Systems and Uncertainty Propagation

We consider nonlinear systems. In the deterministic case, the system is described using the nonlinear process model, $f(u_k, x_{k-1})$, and measurement model, $h(x_k)$, as

$$\begin{aligned} x_k &= f(u_k, x_{k-1}), \\ z_k &= h(x_k). \end{aligned} \tag{1}$$

In addition, we consider the nonlinear dynamic system excited by noise, w_k and v_k, as

$$\begin{aligned} x_k &= f(u_k, x_{k-1}, w_k), \\ z_k &= h(x_k, v_k). \end{aligned} \tag{2}$$

In nonlinear Kalman filtering, we explored two ideas for uncertainty propagation. The first was using linearization via Taylor expansion, which led to the Extended Kalman Filter (EKF). The second idea was based on the unscented transform (deterministic sampling) that led to the Unscented Kalman Filter (UKF).

In this note, we explore a class of methods based on random sampling known as Monte Carlo methods. In particular, we study sequential Monte Carlo methods (Particle Filters).

We will use the delta function in this note. The Dirac delta function, for $x \in \mathbb{R}$, is defined by the properties

$$\delta(x) = \begin{cases} \infty & \text{if } x = 0 \\ 0 & \text{if } x \neq 0 \end{cases} \quad \text{and} \quad \int_{-\infty}^{\infty} \delta(x) \mathrm{d}x = 1.$$

For any smooth function f and $a \in \mathbb{R}$, we have:

$$\int_{-\infty}^{\infty} \delta(x - a) f(x) = f(a),$$

which is a Lebesgue integral with respect to the measure δ (thought as a point mass). This can be generalized to $\mathbb{R}^n$ or any set with a similar idea to define δ measure or unit mass concentrated at a point.

2 Sequential Monte Carlo Methods

Sequential Monte Carlo (SMC) methods [1] are a set of simulation-based methods for computing posterior distributions. In this framework, observations arrive sequentially in time, and we wish to perform online inference. The posterior distribution is updated as data become available (recursive Bayesian estimation/learning). SMC methods are used when dealing with non-Gaussian, high-dimensionality, and nonlinearity where often obtaining an analytical solution is not possible. In addition, SMC methods can be used for inferring both filtering and smoothing posterior distributions.

2.1 Perfect Monte Carlo Sampling

Suppose we can simulate n independent and identically distributed (i.i.d.) random samples (particles), $\{x^i_{0:k}\}^n_{i=1}$ according to $p(x_{0:k} \mid z_{1:k})$. An empirical estimate of this distribution is given by

$$p(x_{0:k} \mid z_{1:k}) \approx p_n(x_{0:k} - x^{1:n}_{0:k} \mid z_{1:k}) = \frac{1}{n}\sum_{i=1}^{n} \delta(x_{0:k} - x^i_{0:k}).$$

We note that each delta term places infinite density at the corresponding random sample. The integration of this empirical density results in n, hence, normalization by $\frac{1}{n}$. Then, the following integral can be computed:

$$\mathrm{I}_n(f) = \int f(x_{0:k}) p_n(x_{0:k} - x^{1:n}_{0:k} \mid z_{1:k}) = \frac{1}{n}\sum_{i=1}^{n} f(x^i_{0:k}).$$

If the posterior variance is bounded, i.e., $\sigma^2_f := \mathbb{E}_{p(x_{0:k}|z_{1:k})}[f^2(x_{0:k})] - \mathrm{I}^2_n(f) < \infty$, then $\mathbb{V}[\mathrm{I}_n(f)] = \frac{\sigma^2_f}{n}$. From the law of large numbers, $n \to \infty$, almost surely, $\mathrm{I}_n(f) \to \mathrm{I}(f)$. Therefore, This estimate is unbiased. Moreover, if $\sigma^2_f < \infty$, then a central limit theorem holds; that is $n \to \infty$, $\sqrt{n}(\mathrm{I}_n(f) - \mathrm{I}(f))$ converges in distribution to $\mathcal{N}(0, \sigma^2_f)$.

This result is important because we can easily estimate any quantity $\mathrm{I}(f)$ while the rate of convergence is independent of the integrand dimension. In particular, any deterministic numerical integration method has a rate of convergence that decreases as the dimension of the integrand increases. However, in practice, it is usually impossible to sample efficiently from the posterior distribution $p(x_{0:k} \mid z_{1:k})$. In the next section, we use the idea of importance sampling to resolve the problem that we do not know the posterior distribution, or we cannot sample from it.

2.2 Importance Sampling

We introduce an *importance sampling distribution* (also called *proposal distribution*), $\pi(x_{0:k} \mid z_{1:k})$. We also assume the support of $\pi(x_{0:k} \mid z_{1:k})$ includes the support of $p(x_{0:k} \mid z_{1:k})$. We get

$$\mathrm{I}(f) = \frac{\int f(x_{0:k}) w(x_{0:k}) \pi(x_{0:k} \mid z_{1:k}) \mathrm{d}x_{0:k}}{\int w(x_{0:k}) \pi(x_{0:k} \mid z_{1:k}) \mathrm{d}x_{0:k}} = \frac{\int f(x_{0:k}) p(x_{0:k}|z_{1:k}) \mathrm{d}x_{0:k}}{\int p(x_{0:k}|z_{1:k}) \mathrm{d}x_{0:k}} = \int f(x_{0:k}) p(x_{0:k}|z_{1:k}) \mathrm{d}x_{0:k},$$

where $w(x_{0:k})$ is known importance weight:

$$w(x_{0:k}) = \frac{p(x_{0:k} \mid z_{1:k})}{\pi(x_{0:k} \mid z_{1:k})}$$

Next, by drawing n i.i.d. particles according to $\pi(x_{0:k}|z_{1:k})$ (we know π and we can draw samples from it),

$$\hat{\mathrm{I}}_n(f) = \frac{\frac{1}{n}\sum_{i=1}^{n} f(x^i_{0:k}) w(x^i_{0:k})}{\frac{1}{n}\sum_{i=1}^{n} w(x^i_{0:k})} = \sum_{i=1}^{n} f(x^i_{0:k}) \tilde{w}^i_k,$$

where the *normalized importance weights* are given by

$$\tilde{w}^i_k = \frac{w(x^i_{0:k})}{\sum_{i=1}^{n} w(x^i_{0:k})}.$$

However, this is not adequate for recursive estimation.

3 Sequential Importance Sampling (SIS)

Let us factor the importance sampling distribution as follows.

$$\pi(x_{0:k}|z_{1:k}) = \pi(x_{0:k-1}|z_{1:k-1})\pi(x_k|x_{0:k-1}, z_{1:k}) = \pi(x_0)\prod_{j=1}^{k}\pi(x_j|x_{0:j-1}, z_{1:j}).$$

Then

$$\tilde{w}_k^i \propto \tilde{w}_{k-1}^i \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{\pi(x_k^i|x_{0:k-1}^i, z_{1:k})} \quad \left(w(x_{0:k}) = \frac{p(x_{0:k}|z_{1:k})}{\pi(x_{0:k}|z_{1:k})}\right).$$

Remark 1. *Recall that* $p(x_{0:k}^i|z_{1:k}) = p(x_{0:k-1}^i|z_{1:k-1})\frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{p(z_k|z_{1:k-1})}$*, where we used the Markov assumption and causality of the models in the Bayes' rule.*

In filtering, we often set $\pi(x_k|x_{0:k-1}, z_{1:k}) = \pi(x_k|x_{k-1}, z_k)$ so that the importance sampling distribution only depends on x_{k-1} and z_k.

3.1 SIS Particle Filter

At this point, we have successfully derived the SIS particle filter algorithm.

Algorithm 1 `sis-particle-filter`

Require: particles $\mathcal{X}_{k-1} = \{x_{k-1}^i, w_{k-1}^i\}_{i=1}^n$, measurement z_k;

1: $\mathcal{X}_k \leftarrow \varnothing$
2: **for** each $x_{k-1}^i \in \mathcal{X}_{k-1}$ **do**
3: $\quad x_k^i \sim \pi(x_k^i|x_{k-1}^i, z_k)$ ▷ sample from proposal distribution
4: $\quad w_k^i \leftarrow w_k^i \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{\pi(x_k^i|x_{k-1}^i, z_k)}$ ▷ update importance weights
5: $\quad \mathcal{X}_k \leftarrow \mathcal{X}_k \cup \{x_k^i, w_k^i\}$ ▷ add i-th weighted sample to the new set
6: **end for**
7: **return** $\mathcal{X}_k$

Unfortunately, the `sis-particle-filter` algorithm suffers from the *degeneracy problem*. As time increases, the distribution of the of the weights, $\tilde{w}_k^i$ becomes more and more skewed, in practice, reducing to one particle with non-zero weight after a few iterations. We can define the following measure of degeneracy using the effective sample size as

$$n_{\mathsf{eff}} = \frac{1}{\sum_{i=1}^n (\tilde{w}_k^i)^2} \quad 1 < n_{\mathsf{eff}} < n.$$

At the two extreme cases we have: 1) all particles have the same weights (uniform): $\forall i \in \{1:n\}$, $\tilde{w}_k^i = \frac{1}{n} \Longrightarrow n_{\mathsf{eff}} = n$; 2) the entire distribution mass is placed in one particle (singular): $\forall i \in \{1:j-1, j+1:n\}$, $\tilde{w}_k^i = 0$ and $\tilde{w}_k^j = 1 \Longrightarrow n_{\mathsf{eff}} = 1$.

3.2 Resampling and Generic Particle Filter Algorithm

The *resampling* idea was introduced to fix the degeneracy problem. Resampling eliminates particles with low weights and multiplies particles with high weights. Although the resampling step reduces the effect of degeneracy, it introduces a new problem known as *sample impoverishment*. It limits the parallel implementation of the algorithm since all particles must be combined. In addition, the particles with high weights are selected many times, leading to the loss of diversity, i.e., loss of alternative hypotheses. This is because eventually, most particles are initiated from a few initial parent particles.

SIS particle filter combined with the resampling idea leads to the Sample Importance Resampling (SIR) algorithm. A generic version of the SIR particle filter is shown in Algorithm 2.

Algorithm 2 `generic-particle-filter`

Require: particles $\mathcal{X}_{k-1} = \{x^i_{k-1}, \tilde{w}^i_{k-1}\}^n_{i=1}$, measurement z_k, resampling threshold n_{t} (e.g., $n/3$);

$\mathcal{X}_k \leftarrow \varnothing$
for each $x^i_{k-1} \in \mathcal{X}_{k-1}$ **do**
    draw $x^i_k \sim \pi(x^i_k | x^i_{k-1}, z_k)$ ▷ sample from proposal distribution
    $w^i_k \leftarrow \tilde{w}^i \frac{p(z_k|x^i_k)p(x^i_k|x^i_{k-1})}{\pi(x^i_k|x^i_{k-1}, z_k)}$ ▷ update importance weights
end for
$w_{\mathsf{total}} \leftarrow \sum_{i=1}^n w^i_k$ ▷ compute total weight to normalize importance weights
$\mathcal{X}_k \leftarrow \mathcal{X}_k \cup \{x^i_k, w^i_k / w_{\mathsf{total}}\}^n_{i=1}$ ▷ add weighted samples to the new set
$n_{\mathsf{eff}} \leftarrow 1/\sum_{i=1}^n (\tilde{w}^i_k)^2$ ▷ compute effective sample size
if $n_{\mathsf{eff}} < n_{\mathsf{t}}$ **then**
    $\mathcal{X}_k \leftarrow$ resample using $\mathcal{X}_k$ ▷ use a resampling algorithm to draw particles with higher weights
end if
return $\mathcal{X}_k$

The low-variance resampling is a simple and effective algorithm to use in practice. Its algorithmic implementation is shown in Algorithm 3.

Algorithm 3 `low-variance-resampling`

Require: particles $\mathcal{X}_k = \{x_k^i, \tilde{w}_k^i\}_{i=1}^n$;

$w_{\mathsf{c}} \leftarrow$ compute the vector of cumulative sum of the weights using $\{\tilde{w}_k^i\}_{i=1}^n$ ▷ w_{c} is the Cumulative Distribution Function (CDF)
$r \leftarrow \texttt{rand}(0, n^{-1})$ ▷ draw a uniform random number between 0 and n^{-1}
$j \leftarrow 1$ ▷ dummy index to climb the CDF and select particles
for all $i \in \{1 : n\}$ **do**
    $u \leftarrow r + (i-1)n^{-1}$ ▷ move along the CDF
    while $u > w_{\mathsf{c}}^j$ **do**
        $j \leftarrow j + 1$
    end while
    $x_k^i \leftarrow x_k^j$ ▷ replicate the survived particle
    $\tilde{w}_k^i \leftarrow n^{-1}$ ▷ set the weight to n^{-1} (uniform distribution)
end for
return $\mathcal{X}_k$

3.3 SIR Particle Filter Algorithm in Robotics

A special case is when the importance sampling distribution is chosen to be the prior distribution, or in robotics the robot motion model $p(x_k|x_{k-1}, u_k)$. Note that u_k is deterministic, and the motion model does not depend on the measurement z_k. Then the weights can be computed using

$$\tilde{w}_k^i \propto \tilde{w}_{k-1}^i p(z_k|x_k^i).$$

Algorithm 4 `sir-particle-filter`

Require: particles $\mathcal{X}_{k-1} = \{x_{k-1}^i, \tilde{w}_{k-1}^i\}_{i=1}^n$, action u_k, measurement z_k, resampling threshold n_{t} (e.g., $n/3$);

$\mathcal{X}_k \leftarrow \varnothing$
for each $x_{k-1}^i \in \mathcal{X}_{k-1}$ **do**
    draw $x_k^i \sim p(x_k|x_{k-1}^i, u_k)$ ▷ sample from motion model
    $w_k^i \leftarrow \tilde{w}_{k-1}^i p(z_k|x_k^i)$ ▷ update importance weights
end for
$w_{\mathsf{total}} \leftarrow \sum_{i=1}^n w_k^i$ ▷ compute total weight to normalize importance weights
$\mathcal{X}_k \leftarrow \mathcal{X}_k \cup \{x_k^i, w_k^i / w_{\mathsf{total}}\}_{i=1}^n$ ▷ add weighted samples to the new set
$n_{\mathsf{eff}} \leftarrow 1/\sum_{i=1}^n (\tilde{w}_k^i)^2$ ▷ compute effective sample size
if $n_{\mathsf{eff}} < n_{\mathsf{t}}$ **then**
    $\mathcal{X}_k \leftarrow$ resample using $\mathcal{X}_k$ ▷ use a resampling algorithm to draw particles with higher weights
end if
return $\mathcal{X}_k$

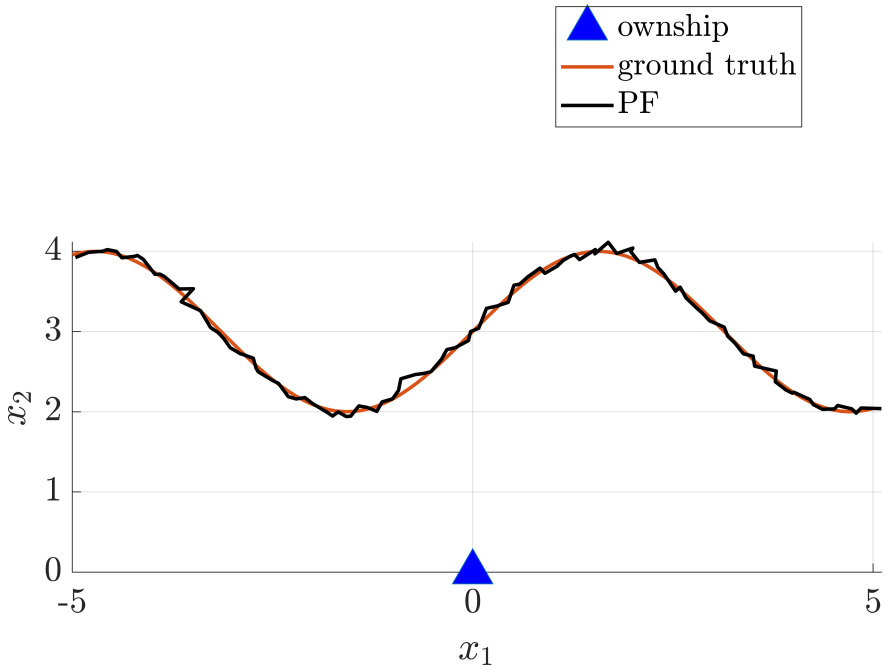

Figure 1: 2D target tracking using a PF.

Example 1 (PF Target Tracking)**.** *A target is moving in a 2D plane. The ownship position is known and fixed at the origin. We have access to relative noisy range and bearing measurements of the target position at any time step.*

$$x_k = f(u_k, x_{k-1}) + w_k = x_{k-1} + w_k,$$

$$z_k = h(x_k) + v_k = \begin{bmatrix} \sqrt{{x_k^1}^2 + {x_k^2}^2} \\ \text{atan2}(x_k^1, x_k^2) \end{bmatrix} + v_k,$$

$$Q_k = 0.1\, I_2,\ R_k = \text{diag}(0.05^2, 0.01^2).$$

We estimate the target position using a Particle Filter (PF) as shown in Figure 1. See `pf_single_target.m` *(or Python version) for code. Try to change the parameters and study the filter's behavior.*

Example 2 (PF Target Tracking, Constant Velocity Motion Model)**.** *There is no knowledge of the target motion, but this time, we assume a constant velocity random walk motion model and estimate the target velocity along with the position.*

$$x_k = f(u_k, x_{k-1}) + w_k = F_k x_{k-1} + w_k,$$

$$F_k = \begin{bmatrix} I & \Delta t I \\ 0 & I \end{bmatrix} \qquad \Delta t : \textit{sampling time},$$

$$z_k = h(x_k) + v_k = \begin{bmatrix} \sqrt{{x_k^1}^2 + {x_k^2}^2} \\ \text{atan2}(x_k^1, x_k^2) \end{bmatrix} + v_k,$$

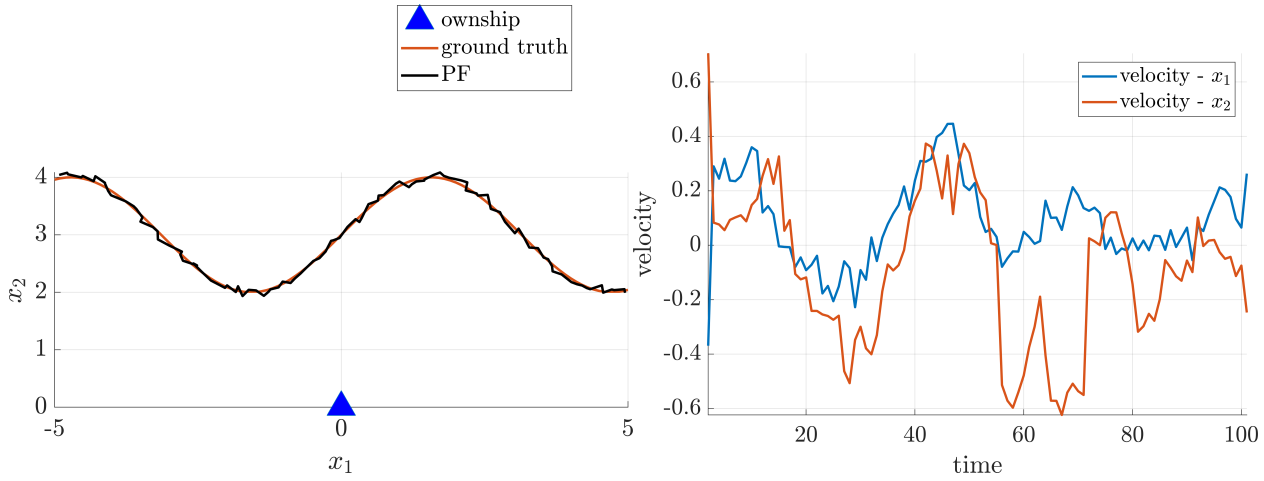

Figure 2: 2D target tracking with a constant velocity motion model using a PF.

$$Q_k = \text{diag}(0.1^2, 0.1^2, 0.01^2, 0.01^2), \; R_k = \text{diag}(0.05^2, 0.01^2).$$

We estimate the target position and velocity using a PF as shown in Figure 2. See `pf_single_target_cv.m` *(or Python version) for code. Try to change the parameters and study the filter's behavior.*

4 Summary

SMC methods can solve complex nonlinear, non-Gaussian online estimation problems. For example, dealing with global uncertainty in robot localization and solving the "kidnapped robot" problem (see [2, Chapters 7 and 8]). The algorithms are applicable to a very large class of models and are often straightforward to implement. The price to pay for this simplicity is inefficiency in some application domains. For further reading, see also [3, Chapter 4].

References

[1] A. Doucet, N. De Freitas, N. J. Gordon *et al.*, *Sequential Monte Carlo methods in practice*. Springer, 2001, vol. 1, no. 2.

[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005, vol. 1.

[3] T. D. Barfoot, *State estimation for robotics*. Cambridge University Press, 2017.