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

Note: Nonlinear Kalman Filtering

Lecturer and Author: Maani Ghaffari

1 Nonlinear Dynamic Systems

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 general, we may consider the nonlinear dynamic system excited by multiplicative noise.

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

Then the additive noise case can be consider as a special case of (2).

$$\begin{aligned} x_k &= f(u_k, x_{k-1}) + w_k, \\ z_k &= h(x_k) + v_k. \end{aligned} \tag{3}$$

2 Uncertainty Propagation

A central problem in dealing with nonlinear systems is that, in general, we cannot propagate belief exactly. Therefore, one can ask the question that how to map belief (a probability distribution) through a nonlinear function?

In this note we explore two ideas:

1. linearization via Taylor expansion that leads to the Extended Kalman Filter (EKF);
2. and unscented transform (deterministic sampling) that leads to the Unscented Kalman Filter (UKF).

2.1 EKF: Linearization via Taylor Expansion

Linearization of a function $f : \mathbb{R}^n \to \mathbb{R}^m$ around point a is

$$\begin{aligned} f(x) &\approx f(a) + \frac{\partial f}{\partial x}\Big|_{x=a}(x-a) = (f(a) - \frac{\partial f}{\partial x}\Big|_{x=a} a) + \frac{\partial f}{\partial x}\Big|_{x=a} x \\ &=: x_0 + Fx \end{aligned}$$

This is an affine map and we know how to propagate a Gaussian distribution through an affine map. In particular, suppose $x \sim \mathcal{N}(\mu, \Sigma)$ and $y = Ax + b$, then $y \sim \mathcal{N}(A\mu + b, A\Sigma A^\mathsf{T})$.

We modify the linear Kalman filter to use the linearized models around the current operating point at every step. The following algorithm summarizes the EKF algorithm where $F_k = \frac{\partial f}{\partial x}\Big|_{x=\mu_{k-1}}$, $W_k = \frac{\partial f}{\partial w}\Big|_{x=\mu_{k-1}}$, $H_k = \frac{\partial h}{\partial x}\Big|_{x=\mu_k^-}$, and $V_k = \frac{\partial h}{\partial v}\Big|_{x=\mu_k^-}$. Note that the linearized model must be re-evaluated at each iteration.

Algorithm 1 `Extended-Kalman-filter`

Require: belief mean μ_{k-1}, belief covariance Σ_{k-1}, action u_k, measurement z_k;

$\mu_k^- \leftarrow f(u_k, \mu_{k-1})$ ▷ predicted mean

$\Sigma_k^- \leftarrow F_k \Sigma_{k-1} F_k^\mathsf{T} + W_k Q_k W_k^\mathsf{T}$ ▷ predicted covariance

$\nu_k \leftarrow z_k - h(\mu_k^-)$ ▷ innovation

$S_k \leftarrow H_k \Sigma_k^- H_k^\mathsf{T} + V_k R_k V_k^\mathsf{T}$ ▷ innovation covariance

$K_k \leftarrow \Sigma_k^- H_k^\mathsf{T} S_k^{-1}$ ▷ filter gain

$\mu_k \leftarrow \mu_k^- + K_k \nu_k$ ▷ corrected mean

$\Sigma_k \leftarrow (I - K_k H_k)\Sigma_k^-$ ▷ corrected covariance

$//\Sigma_k \leftarrow (I - K_k H_k)\Sigma_k^-(I - K_k H_k)^\mathsf{T} + K_k R_k K_k^\mathsf{T}$ ▷ numerically stable form

return μ_k, Σ_k

2.2 EKF Summary

- Highly efficient; polynomial time in measurement dimensionality n_z and state dimensionality n_x.
- Not optimal.
- Can diverge if nonlinearities are large.
- Can work well in practice for many problems despite violating all the underlying assumptions.

Example 1 (EKF 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.*

$$x_k = f(u_k, x_{k-1}) + w_k = x_{k-1} + w_k$$
$$z_k = h(x_k) + v_k = \begin{bmatrix} \sqrt{{x_k^1}^2 + {x_k^2}^2} \\ \mathrm{atan2}(x_k^1, x_k^2) \end{bmatrix} + v_k$$

$$F_k = I_2,\ G_k = 0_2,\ Q_k = 0.001\ I_2,\ R_k = \mathrm{diag}(0.05^2, 0.01^2)$$
$$H_k = \begin{bmatrix} \frac{x_k^1}{\sqrt{{x_k^1}^2+{x_k^2}^2}} & \frac{x_k^2}{\sqrt{{x_k^1}^2+{x_k^2}^2}} \\ \frac{x_k^2}{{x_k^1}^2+{x_k^2}^2} & \frac{-x_k^1}{{x_k^1}^2+{x_k^2}^2} \end{bmatrix}$$

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

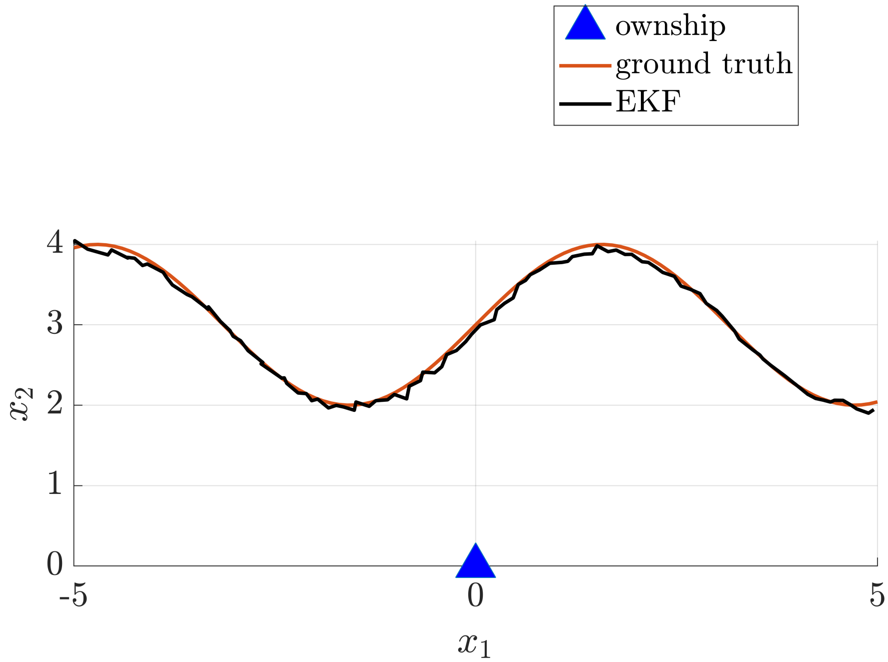

Figure 1: 2D target tracking using an EKF.

2.3 UKF: Unscented Transform

The unscented transform uses deterministic samples to propgate a Gaussian distribution through a nonlinear map. It consists of the following steps.

1. Compute a set of sigma points (samples) $\mathcal{X}$;
2. Each sigma point has a weight w;
3. Transform the point through the nonlinear function $g(x)$;
4. Compute a Gaussian distribution from weighted points using weighted sample mean and covariance.

The first sigma point is the mean, and the other $2n$ sigma points are generated using the columns of the scaled Cholesky factor of the covariance.

$$
\begin{aligned}
x_0 &= \mu, \\
x_i &= \mu + \ell'_i \quad i = 1, \dots, n, \\
x_i &= \mu - \ell'_{i-n} \quad i = n+1, \dots, 2n.
\end{aligned}
$$

ℓ'_i is the i-th column of L' where $L' = \sqrt{(n+\kappa)}L$ and $\Sigma = LL^\mathsf{T}$ can be computed using the Cholesky decomposition. n is the dimension of the state and κ is a user-definable parameter.

Given a nonlinear map $g : \mathbb{R}^n \to \mathbb{R}^m$, we compute a Gaussian distribution using the weights and transformed points.

$$
\mu' = \sum_{i=0}^{2n} w_i g(x_i), \quad \Sigma' = \sum_{i=0}^{2n} w_i (g(x_i) - \mu')(g(x_i) - \mu')^\mathsf{T}. \tag{4}
$$

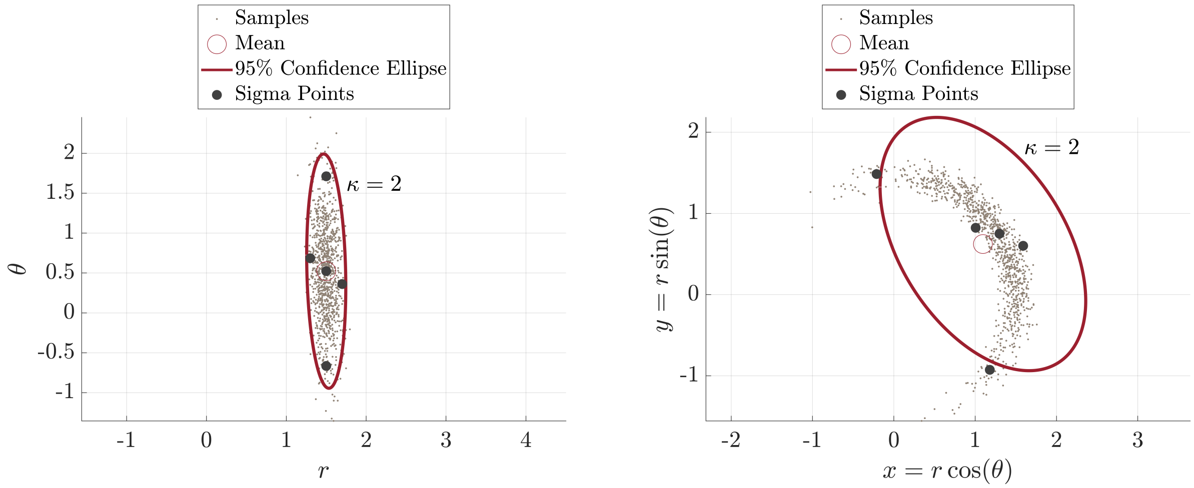

Figure 2: Left: Given distribution in the polar coordinates. Right: Transformed distribution using the unscented transform.

$$w_0 = \frac{\kappa}{n+\kappa}, \quad w_i = \frac{1}{2(n+\kappa)} \quad i = 1, \dots, 2n. \tag{5}$$

The user-defined parameter, κ, can be tuned to adjust the weight for a particular transformation. For instance $\kappa = 2$; see Barfoot [1, Ch. 4.2.7.].

Remark 1. *If the noise is additive, we simply add the noise covariance to the propagated sample covariance. If the noise is multiplicative, we augment the state with noise by adding zeros of appropriate dimension to the mean and constructing a block-diagonal covariance matrix of the state and noise covariances. Note that in the latter case the dimension of the augmented state is increased to the sum of the dimensions of the state and noise vectors; hence, more samples need to be drawn. See Barfoot [1, Ch. 4.2.9.].*

Example 2 (Unscented Transform)**.** *Transform a Gaussian distribution from polar to Cartesian coordinates.*

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r\cos(\theta) \\ r\sin(\theta) \end{bmatrix}, \qquad \begin{bmatrix} r \\ \theta \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} 1.5 \\ \pi/6 \end{bmatrix}, \begin{bmatrix} 0.1^2 & -0.09^2 \\ -0.09^2 & 0.6^2 \end{bmatrix})$$

Figure 2 shows the result before and after applying the unscented transform to the given distribution. See `unscented_transform_example.m` *(or Python version) for code.*

Algorithm 2 shows the the implementation of UKF using the unscented transform for uncertainty propagation.

Algorithm 2 `Unscented-Kalman-filter`

Require: belief mean μ_{k-1}, belief covariance Σ_{k-1}, action u_k, measurement z_k;
1: $\mathcal{X}_{k-1} \leftarrow$ compute the set of $2n+1$ sigma points using μ_{k-1} and Σ_{k-1}
2: $w^- \leftarrow$ compute the set of $2n+1$ weights
3: $\mu_k^- = \sum_{i=0}^{2n} w_i^- f(u_k, x_{k-1,i})$ ▷ predicted mean
4: $\Sigma_k^- \leftarrow \sum_{i=0}^{2n} w_i^- (f(u_k, x_{k-1,i}) - \mu_k^-)(f(u_k, x_{k-1,i}) - \mu_k^-)^\mathsf{T} + Q_k$ ▷ predicted covariance
5: $\mathcal{X}_k^- \leftarrow$ compute the set of $2n+1$ sigma points using μ_k^- and Σ_k^-
6: $w \leftarrow$ compute the set of $2n+1$ weights
7: $z_k^- = \sum_{i=0}^{2n} w_i h(x_{k,i}^-)$ ▷ predicted measurement
8: $\nu_k \leftarrow z_k - z_k^-$ ▷ innovation
9: $S_k \leftarrow \sum_{i=0}^{2n} w_i (h(x_{k,i}^-) - z_k^-)(h(x_{k,i}^-) - z_k^-)^\mathsf{T} + R_k$ ▷ innovation covariance
10: $\Sigma_k^{xz} \leftarrow \sum_{i=0}^{2n} w_k^{[i]} (x_{k,i}^- - \mu_k^-)(h(x_{k,i}^-) - z_k^-)^\mathsf{T}$ ▷ state and measurement cross covariance
11: $K_k \leftarrow \Sigma_k^{xz} S_k^{-1}$ ▷ filter gain
12: $\mu_k \leftarrow \mu_k^- + K_k \nu_k$ ▷ corrected mean
13: $\Sigma_k \leftarrow \Sigma_k^- - K_k S_k K_k^\mathsf{T}$ ▷ corrected covariance
14: **return** μ_k, Σ_k

Example 3 (UKF 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.*

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

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

$$F_k = I_2,\ G_k = 0_2,\ Q_k = 0.001\ I_2,\ R_k = \mathrm{diag}(0.05^2, 0.01^2)$$

We estimate the target position using a UKF as shown in Figure 3. See `ukf_single_target.m` *(or Python version) for code. Try to change the parameters and study the filter's behavior.*

3 UKF vs. EKF

In the following we summarize the similarities and differences between the two approach.

- Same results as EKF for linear models;
- Better approximation than EKF for nonlinear models;
- Differences often "somewhat small";
- No Jacobians needed for the UKF;
- UKF has the same complexity class;

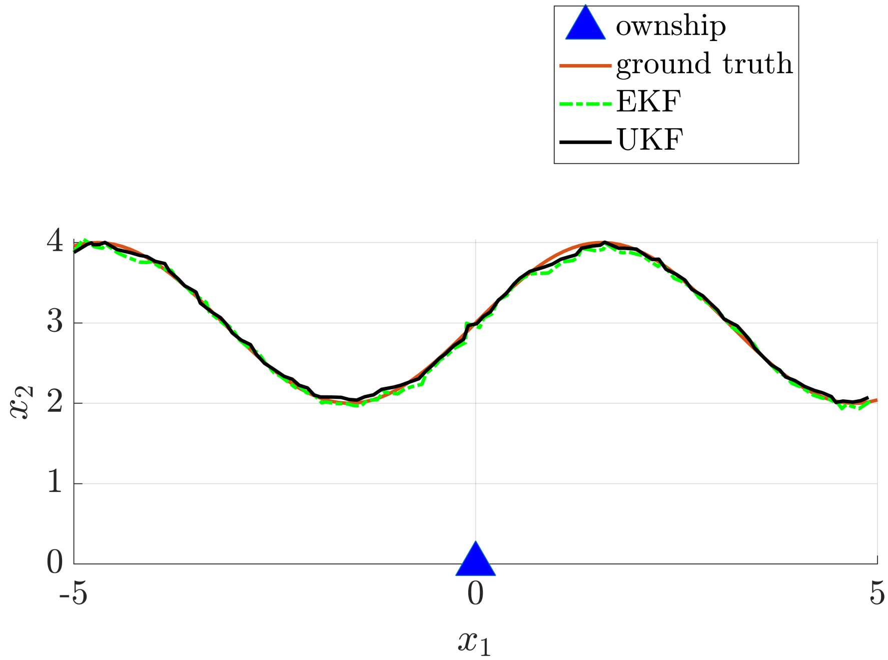

Figure 3: 2D target tracking using a UKF.

- UKF is slightly slower than the EKF.

For in-depth reading of the discussed topics, see Chapters 3 and 4 of Barfoot [1].

References

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