NA 568 - Winter 2026

Nonlinear Kalman Filtering

Maani Ghaffari

January 14, 2026

UNIVERSITY OF MICHIGAN

Last time (Kalman Filter):

- Linear models + Gaussian noise ⇒ belief stays Gaussian.
- We tracked only (μ_k, Σ_k) with *Predict* → *Update*.

Today (Nonlinear models):

$$x_k = f(u_k, x_{k-1}) + w_k, \qquad z_k = h(x_k) + v_k$$

矩阵乘法变成非线性函数f

- Pushing a Gaussian through $f(\cdot)$ or conditioning through $h(\cdot)$ is *not closed-form* in general.
- We still *approximate* the belief as Gaussian and propagate only (μ_k, Σ_k).

虽然非线性，但是依旧近似为高斯

From KF to Nonlinear KF (EKF/UKF)

Two approximation philosophies:

- EKF: approximate the *functions* via local linearization (Jacobians).
- UKF: approximate the *distribution* via sigma points (Unscented Transform).

Goal: understand when each works well and what can go wrong.

Nonlinear dynamic system with additive noise.

- Nonlinear process model: 加性噪声在外面

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

- Nonlinear measurement model:

$$z_k = h(x_k) + v_k$$

Nonlinear dynamic system with multiplicative noise.

- Nonlinear process model: 乘性噪声在里面

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

乘性噪声不仅仅要线性化x
，还要线性化w。

- Nonlinear measurement model:

$$z_k = h(x_k, v_k)$$

Q. How to map belief (a probability distribution) through a nonlinear function? 如何用非线性函数来映射一个概率分布

Key ideas:

- Linearization via Taylor expansion 泰勒级数近似
  $\longrightarrow$ Approximate the function.
- Unscented Transform (deterministic sampling) 无迹变换
  $\longrightarrow$ Approximate the distribution (up to the first two moments). 确定性采样
- Monte-Carlo methods (random sampling) 蒙特卡洛方法
  $\longrightarrow$ Approximate the distribution.

Q. How to map belief (a probability distribution) through a nonlinear function?

Key ideas: 三种想法对应扩展卡尔曼、无迹卡尔曼和粒子滤波

- Linearization via Taylor expansion
  $\longrightarrow$ Extended Kalman Filter (EKF)

- Unscented Transform (deterministic sampling)
  $\longrightarrow$ Unscented Kalman Filter (UKF)

- Monte-Carlo methods (random sampling)
  $\longrightarrow$ Sequential Monte-Carlo methods (Particle Filters)

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

$$f(x) \approx f(a) + \frac{\partial f}{\partial x}\Big|_{x=a}(x-a)$$

x在a附近一个小圈内，要
用f(a)来线性近似f(x)

$$= \left(f(a) - \frac{\partial f}{\partial x}\Big|_{x=a} a\right) + \frac{\partial f}{\partial x}\Big|_{x=a} x$$

$$=: x_0 + Fx$$

这就是一个一次函数，即
有关于x的仿射变换。

Affine! We know how to propagate a Gaussian through an affine map.

Recall: Affine Transformation of a Multivariate Gaussian

Suppose $x \sim \mathcal{N}(\mu, \Sigma)$ and $y = Ax + b$.

Then $y \sim \mathcal{N}(A\mu + b, A\Sigma A^\mathsf{T})$.

经过仿射变换的高斯公式

EKF: linearize then do KF

We assume the belief is Gaussian: $bel(x_k) \approx \mathcal{N}(\mu_k, \Sigma_k)$.

EKF idea: approximate nonlinear models locally by first-order Taylor expansion:

$$f(u_k, x) \approx f(u_k, \mu_{k-1}) + F_k(x - \mu_{k-1}), \quad h(x) \approx h(\mu_k^-) + H_k(x - \mu_k^-)$$

where

$$F_k = \left.\frac{\partial f}{\partial x}\right|_{x=\mu_{k-1}}, \qquad H_k = \left.\frac{\partial h}{\partial x}\right|_{x=\mu_k^-}.$$

Remark

EKF is just KF applied to these local linear models.

对 f 在 $(x, w) = (\mu_{k-1}, 0)$ 附近做一阶泰勒展开：

$$f(x, u, w) \approx f(\mu_{k-1}, u, 0) \ + \ F_k(x - \mu_{k-1}) \ + \ W_k w$$

其中

$$F_k = \left.\frac{\partial f}{\partial x}\right|_{x=\mu_{k-1}}, \qquad W_k = \left.\frac{\partial f}{\partial w}\right|_{x=\mu_{k-1}}$$

于是预测误差（对均值的偏差）近似是

$$\delta x_k \approx F_k \, \delta x_{k-1} + W_k \, w_k$$

取协方差：

$$\Sigma_k^- = \mathbb{E}[\delta x_k \delta x_k^\top] \approx F_k \Sigma_{k-1} F_k^\top + W_k Q_k W_k^\top$$

$h(x,v) \approx h(\mu_k^-, 0) + H_k(x - \mu_k^-) + V_k v$

$H_k = \left.\frac{\partial h}{\partial x}\right|_{x=\mu_k^-}, \quad V_k = \left.\frac{\partial h}{\partial v}\right|_{x=\mu_k^-}$

$$F_k = \left.\frac{\partial f}{\partial x}\right|_{x=\mu_{k-1}}, \; W_k = \left.\frac{\partial f}{\partial w}\right|_{x=\mu_{k-1}}, \; H_k = \left.\frac{\partial h}{\partial x}\right|_{x=\mu_k^-}, \; V_k = \left.\frac{\partial h}{\partial v}\right|_{x=\mu_k^-}$$

Algorithm 1 `Extended-Kalman-filter`

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

1: $\mu_k^- \leftarrow f(u_k, \mu_{k-1})$ xk|k-1均值 ▷ predicted mean
2: $\Sigma_k^- \leftarrow F_k \Sigma_{k-1} F_k^\intercal + \underline{W_k Q_k W_k^\intercal}$ xk|k-1协方差，多了个W ▷ predicted covariance
3: $\nu_k \leftarrow z_k - h(\mu_k^-)$ 观测zk和观测均值之差 ▷ innovation
4: $S_k \leftarrow H_k \Sigma_k^- H_k^\intercal + \underline{V_k R_k V_k^\intercal}$ 观测zk的协方差，多了个V ▷ innovation covariance
5: $K_k \leftarrow \Sigma_k^- H_k^\intercal S_k^{-1}$ ▷ filter gain
6: $\mu_k \leftarrow \mu_k^- + K_k \nu_k$ ▷ corrected mean
7: $\Sigma_k \leftarrow (I - K_k H_k)\Sigma_k^-$ ▷ corrected covariance
8: $//\Sigma_k \leftarrow (I - K_k H_k)\Sigma_k^-(I - K_k H_k)^\intercal + K_k R_k K_k^\intercal$ ▷ numerically stable form
9: **return** μ_k, Σ_k

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.

$$x_k = f(u_k, x_{k-1}) + w_k = x_{k-1} + w_k$$ 状态方程为线性

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

观测极坐标

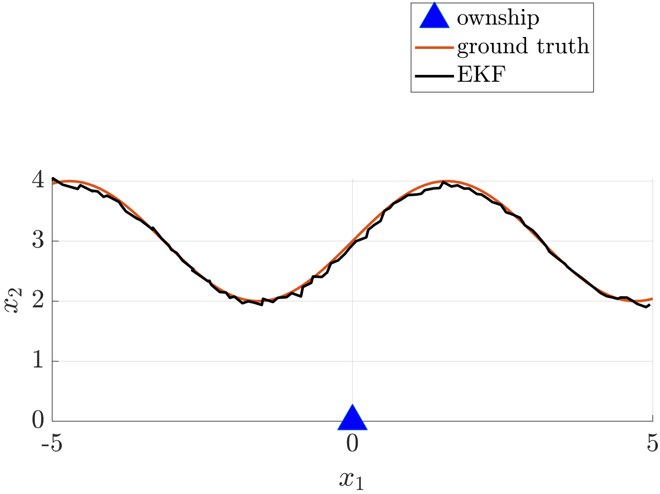
观测噪声协方差

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

状态噪声协方差

$$H_k = \begin{bmatrix} \frac{x_k^1}{\sqrt{{x_k^1}^2+{x_k^2}^2}} & \frac{x_k^2}{\sqrt{{x_k^1}^2+{x_k^2}^2}} \\ \frac{x_k^2}{{x_k^1}^2+{x_k^2}^2} & \frac{-x_k^1}{{x_k^1}^2+{x_k^2}^2} \end{bmatrix}$$ 观测值对状态值的雅可比矩阵

See `ekf_single_target.m` for code.



- 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.

无迹卡尔曼特点：1.已知均值协方差的分布

- Given a distribution of known mean and covariance matrix.

2.计算一个sigma点集合，每个点都有权重w

- Compute a set of sigma points (samples) $\mathcal{X}$, each with a weight w;

3.这种离散点分布均值和协方差和原始分布一样

- Such that the mean and covariance of the discrete distribution of points are the same as the original distribution;

4.使用非线性函数来变换这些点

- Transform the point through the nonlinear function $g(x)$;

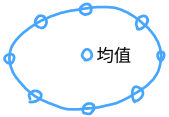
5.变换之后的集合的均值和方差作为原始分布的非线性变换计算

- The mean and covariance of the transformed ensemble are computed as the estimate of the nonlinear transformation of the original distribution.

Unscented Transform: Sigma Points

A = LU
Sigma = LL^T

The first sigma point is the mean.



外圈为sigma点

$$x_0 = \mu \qquad \text{均值}$$

$$x_i = \mu + \ell'_i \quad i = 1, \ldots, n \qquad \text{均值加向量，n个}$$

$$x_i = \mu - \ell'_{i-n} \quad i = n+1, \ldots, 2n \qquad \text{均值减向量，n个}$$

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

取二维协方差：

$$\Sigma = \begin{bmatrix} 4 & 2 \\ 2 & 3 \end{bmatrix}$$

对它做 Cholesky 分解（下三角）：

$$\Sigma = LL^\top, \quad L = \begin{bmatrix} 2 & 0 \\ 1 & \sqrt{2} \end{bmatrix}$$

（你可以自己乘一下：$LL^\top = \begin{bmatrix} 4 & 2 \\ 2 & 1+2 \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 2 & 3 \end{bmatrix}$）

设 $n = 2, \kappa = 0$，则尺度因子 $\sqrt{n+\kappa} = \sqrt{2}$：

$$L' = \sqrt{2}L = \begin{bmatrix} 2\sqrt{2} & 0 \\ \sqrt{2} & 2 \end{bmatrix}$$

所以两列向量是：

- 第1列 $\ell_1' = \begin{bmatrix} 2\sqrt{2} \\ \sqrt{2} \end{bmatrix}$
- 第2列 $\ell_2' = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$

若取均值 $\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$，sigma 点就是：

$$x_0 = \mu$$

$$x_1 = \mu + \ell_1', \quad x_2 = \mu + \ell_2'$$

$$x_3 = \mu - \ell_1', \quad x_4 = \mu - \ell_2'$$

也就是：

- $x_1 = (2\sqrt{2},\ \sqrt{2})$
- $x_2 = (0,\ 2)$
- $x_3 = (-2\sqrt{2},\ -\sqrt{2})$
- $x_4 = (0,\ -2)$

关键验证：这些点真的“代表”了 Σ

在这种写法里（$\kappa = 0$），常用权重是：

- $w_0 = 0$
- 其他 4 个点：$w_i = \frac{1}{2(n+\kappa)} = \frac{1}{4}$

计算样本协方差（由于正负对称，均值就是 0）：

$$\Sigma_{\text{sigma}} = \sum_{i=1}^{4} w_i\, x_i x_i^\top$$

因为 $x_3 = -x_1$，$x_4 = -x_2$，所以

$$\Sigma_{\text{sigma}} = \frac{1}{4}(x_1x_1^\top + x_2x_2^\top + x_3x_3^\top + x_4x_4^\top) = \frac{1}{2}(x_1x_1^\top + x_2x_2^\top)$$

算外积：

- $x_1x_1^\top = \begin{bmatrix} (2\sqrt{2})^2 & (2\sqrt{2})(\sqrt{2}) \\ (2\sqrt{2})(\sqrt{2}) & (\sqrt{2})^2 \end{bmatrix} = \begin{bmatrix} 8 & 4 \\ 4 & 2 \end{bmatrix}$
- $x_2x_2^\top = \begin{bmatrix} 0 & 0 \\ 0 & 4 \end{bmatrix}$

加起来：

$$x_1x_1^\top + x_2x_2^\top = \begin{bmatrix} 8 & 4 \\ 4 & 6 \end{bmatrix}$$

乘 $\frac{1}{2}$：

$$\Sigma_{\text{sigma}} = \begin{bmatrix} 4 & 2 \\ 2 & 3 \end{bmatrix} = \Sigma$$

L 是什么?

L 是协方差矩阵 Σ 的一个"平方根因子"，来自 Cholesky **分解**：

$$\Sigma = LL^\top$$

- $\Sigma \in \mathbb{R}^{n\times n}$ 是状态的协方差
- $L \in \mathbb{R}^{n\times n}$ 通常取 **下三角矩阵**（Cholesky 分解的标准形式）
- 直观：L 的列向量描述了分布在各个方向上的"尺度/形状"

Unscented Transform: Weights

L' 是什么?

L' 是把 L **整体放大**后的矩阵，用来决定 sigma 点离均值 μ 有多远：

$$L' = \sqrt{n+\kappa}\, L$$

所以 L' 还是 $n \times n$，只是比 L 多了一个尺度因子 $\sqrt{n+\kappa}$。

$$w_0 = \frac{\kappa}{n+\kappa}$$

权重和尺度因子的关系

$$w_i = \frac{1}{2(n+\kappa)} \quad i = 1, \ldots, 2n$$

The user-defined parameter, κ, can be tuned to adjust the weight for a particular transformation. For instance $\kappa = 2$ (see *State Estimation for Robotics*, Timothy D. Barfoot, 2018, Ch. 4.2.7.).

第 1 部分：证明加权均值为 μ

先计算加权和：

$$\sum_{i=0}^{2n} w_i x_i = w_0\mu + \sum_{i=1}^{n} w_i(\mu + \ell_i') + \sum_{i=1}^{n} w_{i+n}(\mu - \ell_i').$$

均值性质只要满足对称性+权重公式即可成立

注意 $w_i = w_{i+n} = \frac{1}{2(n+\kappa)}$，所以 ℓ_i' 会成对抵消：

$$\sum_{i=1}^{n} w_i(\mu + \ell_i') + \sum_{i=1}^{n} w_{i+n}(\mu - \ell_i') = \sum_{i=1}^{n}(w_i + w_{i+n})\mu + \sum_{i=1}^{n}(w_i - w_{i+n})\ell_i' = \sum_{i=1}^{n} \frac{1}{n+\kappa}\mu + 0 = \frac{n}{n+\kappa}\mu.$$

因此

$$\sum_{i=0}^{2n} w_i x_i = w_0\mu + \frac{n}{n+\kappa}\mu = \frac{\kappa}{n+\kappa}\mu + \frac{n}{n+\kappa}\mu = \mu.$$

均值证明完毕。

第 2 部分：证明加权协方差为 Σ

由于已证明加权均值就是 μ，我们直接算二阶中心矩：

$$\sum_{i=0}^{2n} w_i(x_i - \mu)(x_i - \mu)^\top.$$

先看 $i = 0$ 项：$x_0 - \mu = 0$，所以这一项为 0。

对 $i = 1, \ldots, n$：

$$x_i - \mu = \ell_i', \qquad x_{i+n} - \mu = -\ell_i'.$$

因此

$$(x_i - \mu)(x_i - \mu)^\top = \ell_i'\ell_i'^\top, \qquad (x_{i+n} - \mu)(x_{i+n} - \mu)^\top = (-\ell_i')(-\ell_i')^\top = \ell_i'\ell_i'^\top.$$

所以成对相加：

$$w_i\ell_i'\ell_i'^\top + w_{i+n}\ell_i'\ell_i'^\top = (w_i + w_{i+n})\ell_i'\ell_i'^\top = \frac{1}{n+\kappa}\ell_i'\ell_i'^\top.$$

对 $i=1$ 到 n 求和：

$$\sum_{i=0}^{2n} w_i(x_i-\mu)(x_i-\mu)^\top = \sum_{i=1}^{n} \frac{1}{n+\kappa} \ell_i' \ell_i'^\top = \frac{1}{n+\kappa} \sum_{i=1}^{n} \ell_i' \ell_i'^\top .$$

关键一步：$\sum_{i=1}^{n} \ell_i' \ell_i'^\top$ 恰好等于 $L'L'^\top$。

协方差计算就体现了为什么l'必须来自于L'的列。

因为 L' 的列向量就是 $\ell_1', \dots, \ell_n'$，而矩阵乘法的列展开恒等式是：

$$L'L'^\top = \sum_{i=1}^{n} \ell_i' \ell_i'^\top . \qquad L'L'^\top = [\, \ell_1' \ \ell_2' \ \cdots \ \ell_n' \,] \begin{bmatrix} \ell_1'^\top \\ \ell_2'^\top \\ \vdots \\ \ell_n'^\top \end{bmatrix}$$

因此

$$\sum_{i=0}^{2n} w_i(x_i-\mu)(x_i-\mu)^\top = \frac{1}{n+\kappa} L'L'^\top .$$

代入 $L' = \sqrt{n+\kappa} L$：

这个根号缩放大小，会被权重因子的分母消掉，最后得到Sigma

$$\frac{1}{n+\kappa} L'L'^\top = \frac{1}{n+\kappa} (\sqrt{n+\kappa} L)(\sqrt{n+\kappa} L)^\top = \frac{1}{n+\kappa} (n+\kappa) LL^\top = LL^\top = \Sigma .$$

所以这个方法叫做Sigma点！就是因为这些点的加权协方差是原分布的Sigma

Compute the mean and covariance using the sigma points and their weights. They should be the same as the original distribution.

$$\mu' = \sum_{i=0}^{2n} w_i x_i$$

$$\Sigma' = \sum_{i=0}^{2n} w_i (x_i - \mu')(x_i - \mu')^{\mathsf{T}}$$

Mean and covariance of the transformed distribution

Let $g : \mathbb{R}^n \to \mathbb{R}^m$, denoting the nonlinear transformation. Compute the mean and covariance using the transformed points and their weights.

$$\mu'' = \sum_{i=0}^{2n} w_i g(x_i)$$

变换之后，重新计算均值和协方差即可，作为变换后的分布的均值和协方差。

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

Remark

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 State Estimation for Robotics, Timothy D. Barfoot, 2018, Ch. 4.2.9.

Example: Unscented Transform

See `unscented_transform_example.m` for code.

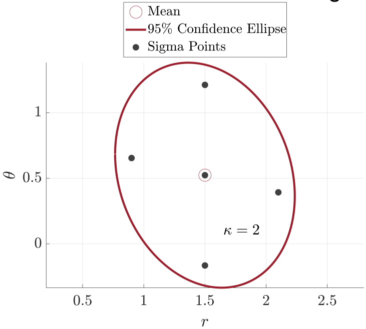
Transform a Gaussian distribution from polar to Cartesian coordinates.

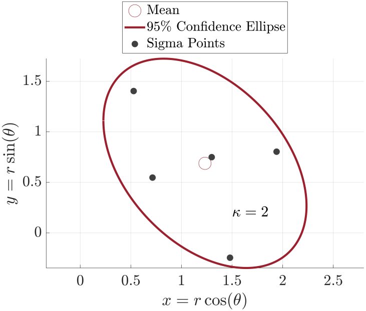
$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r\cos(\theta) \\ r\sin(\theta) \end{bmatrix}, \qquad \begin{bmatrix} r \\ \theta \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} 1.5 \\ \pi/6 \end{bmatrix}, \begin{bmatrix} 0.3^2 & -0.14^2 \\ -0.14^2 & 0.35^2 \end{bmatrix})$$

Example: Unscented Transform

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r\cos(\theta) \\ r\sin(\theta) \end{bmatrix}, \qquad \begin{bmatrix} r \\ \theta \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 1.5 \\ \pi/6 \end{bmatrix}, \begin{bmatrix} 0.3^2 & -0.14^2 \\ -0.14^2 & 0.35^2 \end{bmatrix}\right)$$

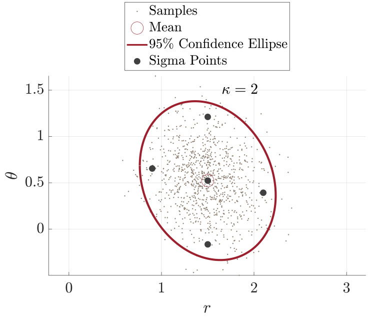
Sigma点变换之后

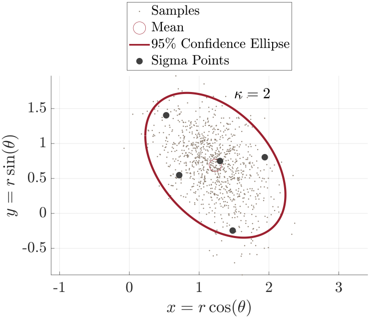




Example: Unscented Transform

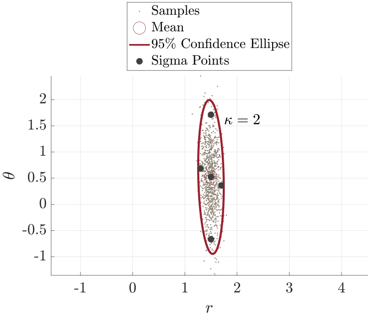
$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r\cos(\theta) \\ r\sin(\theta) \end{bmatrix}, \qquad \begin{bmatrix} r \\ \theta \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 1.5 \\ \pi/6 \end{bmatrix}, \begin{bmatrix} 0.3^2 & -0.14^2 \\ -0.14^2 & 0.35^2 \end{bmatrix}\right)$$

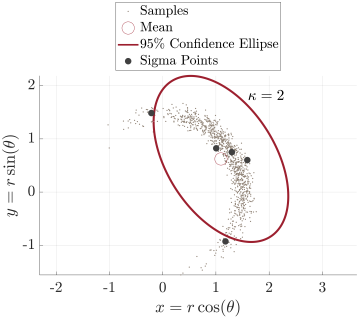




Example: Unscented Transform

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





UKF Algorithm

Algorithm 2 `Unscented-Kalman-filter`

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

$\mathcal{X}_{k-1} \leftarrow$ compute the set of $2n+1$ sigma points using μ_{k-1} and Σ_{k-1} 计算得到Sigma点

$w^- \leftarrow$ compute the set of $2n+1$ weights 计算2n+1个权重

$\mu_k^- = \sum_{i=0}^{2n} w_i^- f(u_k, x_{k-1,i})$ 计算变换之后的sigma点均值 ▷ predicted mean

$\Sigma_k^- \leftarrow \sum_{i=0}^{2n} w_i^- (f(u_k, x_{k-1,i}) - \mu_k^-)(f(u_k, x_{k-1,i}) - \mu_k^-)^{\mathsf{T}} + Q_k$ ▷ predicted covariance 计算状态方程变换之后的sigma点协方差

$\mathcal{X}_k^- \leftarrow$ compute the set of $2n+1$ sigma points using μ_k^- and Σ_k^- 根据均值方差得到新的sigma点

$w \leftarrow$ compute the set of $2n+1$ weights 计算权重

$z_k^- = \sum_{i=0}^{2n} w_i h(x_{k,i}^-)$ 计算观测方程变换之后的sigma均值 ▷ predicted measurement

$\nu_k \leftarrow z_k - z_k^-$ 观测方程变换的均值zk|k-1和zk ▷ innovation

$S_k \leftarrow \sum_{i=0}^{2n} w_i (h(x_{k,i}^-) - z_k^-)(h(x_{k,i}^-) - z_k^-)^{\mathsf{T}} + R_k$ z协方差 ▷ innovation covariance

$\Sigma_k^{xz} \leftarrow \sum_{i=0}^{2n} w_i (x_{k,i}^- - \mu_k^-)(h(x_{k,i}^-) - z_k^-)^{\mathsf{T}}$ ▷ state and measurement cross covariance x和z计算协方差

$K_k \leftarrow \Sigma_k^{xz} S_k^{-1}$ ▷ filter gain

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

$\Sigma_k \leftarrow \Sigma_k^- - K_k S_k K_k^{\mathsf{T}}$ ▷ corrected covariance

return μ_k, Σ_k

与一般卡尔曼滤波不同，这里用样本的方式计算x和z的协方差。在一般卡尔曼滤波中，因为x和z有仿射关系，可以直接用矩阵表示，而UKF是非线性映射，没有仿射关系

$\Sigma_k^{xz} S_k^{-1} S_k S_k^{-1} (\Sigma_k^{xz})^{T}$

Example: UKF Target Tracking

UKF做的是：认为 (x, z) 近似服从联合高斯

$$\begin{bmatrix} x \\ z \end{bmatrix} \approx \mathcal{N}\left(\begin{bmatrix} \mu^- \\ \bar{z} \end{bmatrix}, \begin{bmatrix} \Sigma^- & \Sigma^{xz} \\ (\Sigma^{xz})^\top & S \end{bmatrix} \right)$$

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.

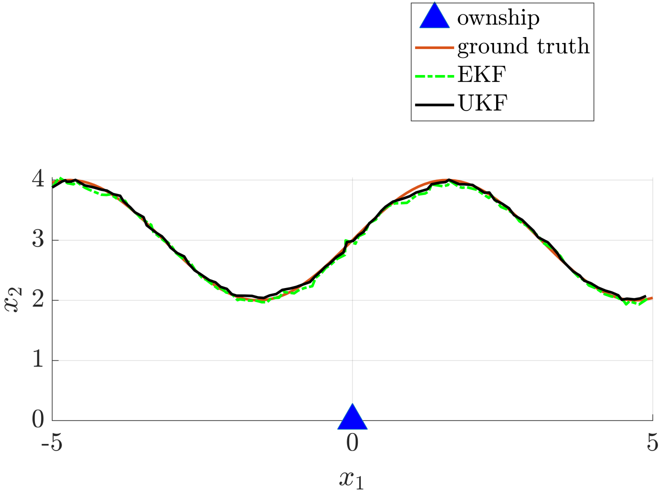
$$x_k = f(u_k, x_{k-1}) + w_k = x_{k-1} + w_k$$

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

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

Example: UKF Target Tracking

See `ukf_single_target.m` for code.



- Highly efficient: same complexity as EKF, with a constant factor slower in typical practical applications;
- Better approximation than EKF;
- Derivative-free: no Jacobians needed. UKF typically tracks curvature better without Jacobians.
- Not optimal.

UKF vs. EKF

- Same results as EKF for linear models;
- Better approximation than EKF for non-linear models;
- Differences often “somewhat small”;
- No Jacobians needed for the UKF;
- Same complexity class;
- Slightly slower than the EKF

Practical notes: when EKF/UKF fail (and what to try)

- Divergence: linearization/UT mismatch + overconfident covariance.
- Bad Jacobians: wrong derivatives or evaluated at the wrong point.
- Inconsistent noise: Q,R too small $\Rightarrow$ filter trusts itself too much.

Common fixes:

- Increase Q (model uncertainty) or R (measurement uncertainty) slightly
- Use the *Joseph form* covariance update for numerical stability
- Re-linearize (iterated EKF) when measurement model is strongly nonlinear
- Consider switching to UKF / Particle Filter for severe nonlinearities

Readings (Nonlinear Kalman Filter)

- Probabilistic Robotics: Ch. 3
- State Estimation for Robotics: Ch. 4
- Lecture Notes for Mobile Robotics: Ch. 2 and 6

Next Time

- Particle Filtering

- Readings:
  - Probabilistic Robotics: Ch. 4
  - State Estimation for Robotics: Ch. 4 (4.2.8)
  - Lecture Notes for Mobile Robotics: Ch. 8
  - Sequential Monte Carlo Methods in Practice: Ch. 1