COLUMBIA UNIVERSITY EEME E6911 FALL '25

TOPICS IN CONTROL : PROBABILISTIC ROBOTICS

EXTENDED KALMAN FILTER (EKF)
UNSCENTED KALMAN FILTER (UKF)
PRACTICAL CONSIDERATIONS

Instructor: Ilija Hadzic

Assumptions and Objectives

- Same as Kalman Filter, relaxed linearity assumption.
- System is not linear.
- System can be linearized in the vicinity of the operating point.
- Gaussian properties lost after applying the system model, but output forced to re-fit to the Gaussian.

System Model

$$\boldsymbol{x}[n] = \boldsymbol{g}(\boldsymbol{x}[n-1], \boldsymbol{u}[n])$$
$$\mathbf{z}[n] = \boldsymbol{h}(\boldsymbol{x}[n])$$

- System model and measurement models are general functions.
- No longer a linear system.
- We must linearize it.

How does EKF Differ, Intuitively?

- Posterior is no longer Gaussian.
- The filter tries to approximate it with a Gaussian.
- Mean can carry through the model directly.
- Covariance requires calculating the Jacobian.

Prediction

$$\overline{\boldsymbol{x}}[n] = \boldsymbol{g}(\boldsymbol{x}[n-1], \boldsymbol{u}[n])$$

$$\overline{\boldsymbol{\Sigma}}_x[n] = \boldsymbol{G}_x \boldsymbol{\Sigma}_x [n-1] \boldsymbol{G}_x^T + \boldsymbol{G}_u \boldsymbol{\Sigma}_u [n] \boldsymbol{G}_u^{\mathrm{T}}$$

$$\boldsymbol{G}_x = \left[\frac{\boldsymbol{\partial g}}{\boldsymbol{\partial x}}\right] \qquad \boldsymbol{G}_u = \left[\frac{\boldsymbol{\partial g}}{\boldsymbol{\partial u}}\right]$$

- Same principles as in KF, but we use Jacobians.
- Jacobians evaluate for present state/input.
- Side-note: textbook assumes that we have input covariance already transformed to state space.

Kalman Gain

$$\boldsymbol{K} = \overline{\boldsymbol{\Sigma}}_x[n]\boldsymbol{H}_{\boldsymbol{x}}^{\mathrm{T}}\left(\boldsymbol{H}_x\overline{\boldsymbol{\Sigma}}_x[n]\boldsymbol{H}_x^{\mathrm{T}} + \boldsymbol{\Sigma}_z[n]\right)^{-1}$$

$$\boldsymbol{H}_x = \left[\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right]$$

- Same concept as KF
- C-matrix replaced with the Jacobian.

Innovation

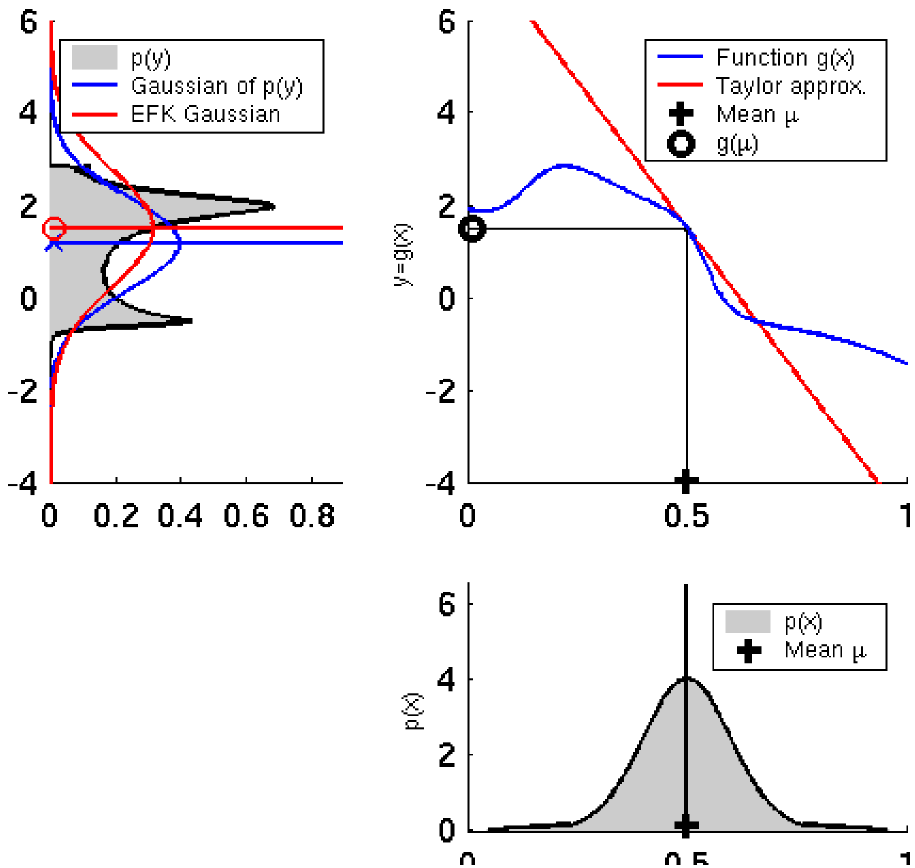
Don't need to linearize here!

$$\boldsymbol{x}[n] = \overline{\boldsymbol{x}}[n] + \mathbf{K}(\boldsymbol{z}[n] - \boldsymbol{h}(\overline{\boldsymbol{x}}[\mathrm{n}]))$$

$$\boldsymbol{\Sigma}_x[n] = (\mathbf{I} - \mathbf{K}\boldsymbol{H})\overline{\boldsymbol{\Sigma}}_x[n]$$

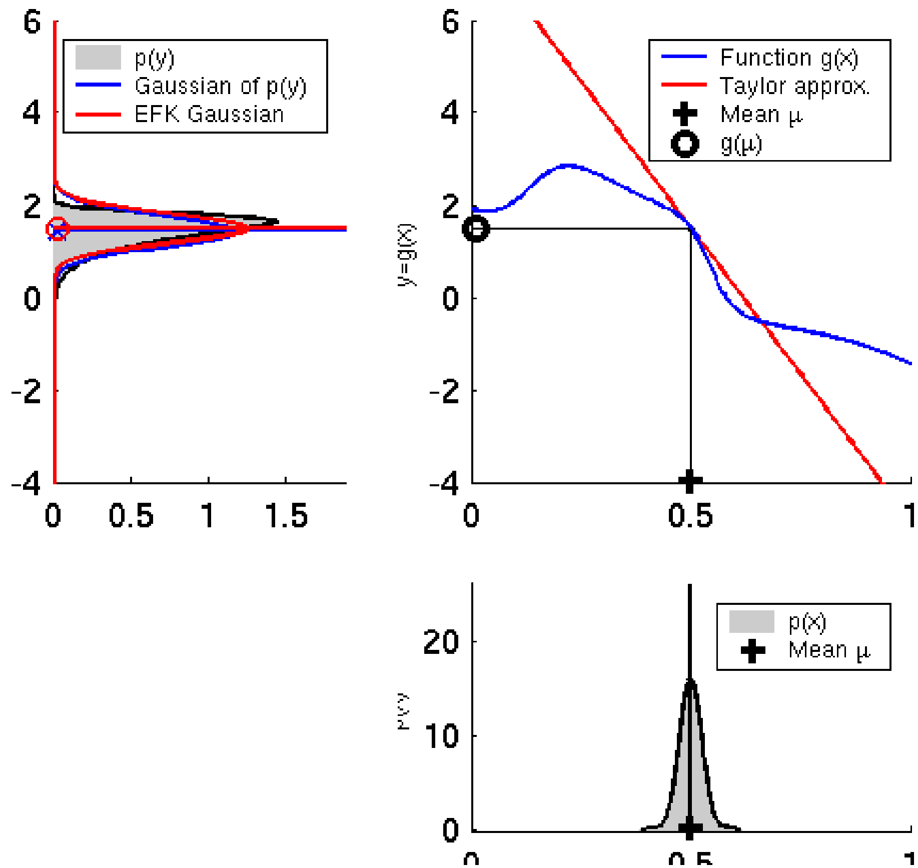
- Track the mean: use non-linear functions g and h.
- Track covariance: use Jacobians $\boldsymbol{G}$ and $\boldsymbol{H}$.
- Same principle otherwise: prediction *expands* the uncertainty, innovation *reduces* uncertainty.

Linearization – What is going on?



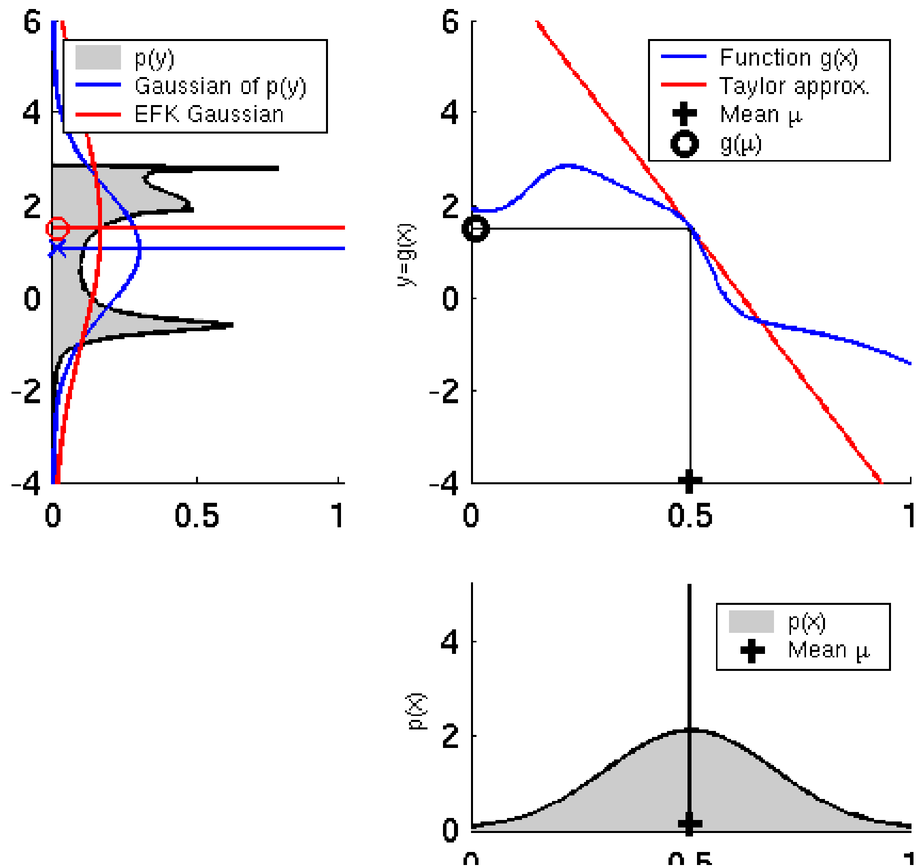
* Source: www.probabilistic-robotics.org

Linearization – High prior confidence



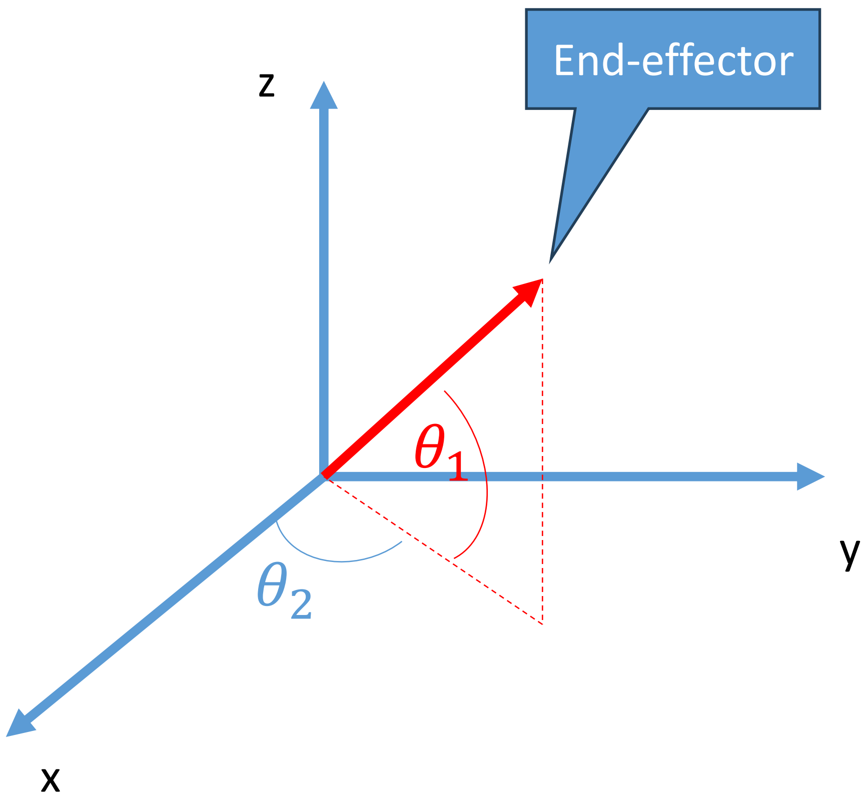
* Source: www.probabilistic-robotics.org

Linearization – Low prior confidence



* Source: www.probabilistic-robotics.org

Example: Two-Axis Gimbal "Arm"



- Two motors control gimbal angles.
- Shaft encoders can only read angular velocity.
- Camera at the tip can measure the (x, y, z) position of the tip.
- Fuse encoders and camera measurements to estimate (x, y, z).

Forward Kinematics

$$x = a \cdot \cos\theta_1 \cos\theta_2$$

$$y = a \cdot \cos\theta_1 \sin\theta_2$$

$$z = a \cdot \sin\theta_1$$

Convert to recursive form.

Forward Kinematics – Recursive

$$x[n] = a \cdot \cos(\theta_1[n-1] + \omega_1 \Delta t)\cos(\theta_2[n-1] + \omega_2 \Delta t)$$

$$\begin{aligned} x[n] = {} & \\ & x[n-1]\cos\omega_1\Delta t\cos\omega_2\Delta t \\ & -y[n-1]\cos\omega_1\Delta t\sin\omega_2\Delta t \\ & -\frac{z[n-1]x[n-1]}{\sqrt{a^2 - z^2[n-1]}}\sin\omega_1\Delta t\cos\omega_2\Delta t \\ & +\frac{z[n-1]y[n-1]}{\sqrt{a^2 - z^2[n-1]}}\sin\omega_1\Delta t\sin\omega_2\Delta t \end{aligned}$$

Forward Kinematics – Recursive

$$y[n] = a \cdot \cos(\theta_1[n-1] + \omega_1 \Delta t)\sin(\theta_2[n-1] + \omega_2 \Delta t)$$

$$\begin{aligned}
y[n] = {} & \\
& y[n-1]\cos\omega_1\Delta t\cos\omega_2\Delta t \\
& -x[n-1]\cos\omega_1\Delta t\sin\omega_2\Delta t \\
& -\frac{z[n-1]y[n-1]}{\sqrt{a^2 - z^2[n-1]}}\sin\omega_1\Delta t\cos\omega_2\Delta t \\
& +\frac{z[n-1]x[n-1]}{\sqrt{a^2 - z^2[n-1]}}\sin\omega_1\Delta t\sin\omega_2\Delta t
\end{aligned}$$

Forward Kinematics – Recursive

$$z[n] = a \cdot \sin(\theta_1[n-1] + \omega_1 \Delta t)$$

$$z[n] = z[n-1] \cos \omega_1 \Delta t + \sqrt{a^2 - z^2[n-1]} \sin \omega_1 \Delta t$$

Exercise: Write the program to simulate the estimation

- Assume that x, y, z, and covariance can be measured directly
- Assume the angular velocities are independent of each other and each is characterized by its own variance.
- Follow the same principle seen in KF simulation
- Hint: do not hand-calculate the Jacobians, use `sympy` package.

Using symbolic math package

```
import sympy as sp

x, y, z, w1, w2, dt, a = sp.symbols('x, y, z, w1, w2, dt, a')

gx = x * sp.cos(w1 * dt) * sp.cos(w2 * dt) \
    - y * sp.cos(w1 * dt) * sp.sin(w2 * dt) \
    - (z * x) / sp.sqrt(a**2 - z**2) * sp.sin(w1 * dt) * sp.cos(w2 * dt) \
    + (z * y) / sp.sqrt(a**2 - z**2) * sp.sin(w1 * dt) * sp.sin(w2 * dt) \
gy = y * sp.cos(w1 * dt) * sp.cos(w2 * dt) \
    - x * sp.cos(w1 * dt) * sp.sin(w2 * dt) \
    - (z * y) / sp.sqrt(a**2 - z**2) * sp.sin(w1 * dt) * sp.cos(w2 * dt) \
    + (z * x) / sp.sqrt(a**2 - z**2) * sp.sin(w1 * dt) * sp.sin(w2 * dt) \
gz = z * sp.cos(w1 * dt) + sp.sqrt(a**2 - z**2) * sp.sin(w1 * dt)

f = sp.Matrix([gx, gy, gz])
Hx = f.jacobian(sp.Matrix([x, y, z]))
Hu = f.jacobian(sp.Matrix([w1, w2]))
Hx_simp = Hx.applyfunc(sp.simplify)
Hu_simp = Hu.applyfunc(sp.simplify)
Hx_func = sp.lambdify((x, y, z, w1, w2, dt, a), Hx_simp)
Hu_func = sp.lambdify((x, y, z, w1, w2, dt, a), Hu_simp)
```

Corollary of KF/EKF

- Kalman filter tells us how to combine information:
  - Consider two measurements of the same physical quantity.
  - We can use Kalman gain to combine them.
  - C-matrix is an identity matrix.

$$
\begin{tikzcd}
\boldsymbol{m}_1, \boldsymbol{\Sigma}_1 \arrow[dr] & & \boldsymbol{m}_2, \boldsymbol{\Sigma}_2 \arrow[dl] \\
& \boldsymbol{m}, \boldsymbol{\Sigma} &
\end{tikzcd}
$$

$$\boldsymbol{K} = \boldsymbol{\Sigma}_1(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)^{-1}$$
$$\boldsymbol{m} = \boldsymbol{m}_1 + \boldsymbol{K}(\boldsymbol{m}_2 - \boldsymbol{m}_1)$$
$$\boldsymbol{m} = (\boldsymbol{\Sigma}_2\boldsymbol{m}_1 + \boldsymbol{\Sigma}_1\boldsymbol{m}_2)(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)^{-1}$$

$$\boldsymbol{\Sigma} = (\mathbf{I} - \mathbf{K})\boldsymbol{\Sigma}_1$$
$$\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_1\boldsymbol{\Sigma}_2(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)^{-1}$$

Generalization

- Multiple measurements

$$\boldsymbol{m}_1, \boldsymbol{\Sigma}_1$$

$$\boldsymbol{m}_2, \boldsymbol{\Sigma}_2$$

$$\boldsymbol{m}_n, \boldsymbol{\Sigma}_n$$

$$\downarrow$$

$$\boldsymbol{m}, \boldsymbol{\Sigma}$$

$$\boldsymbol{m} = \boldsymbol{\Sigma} \sum_{\boldsymbol{i}} \boldsymbol{\Sigma}_{\boldsymbol{i}}^{-1} \boldsymbol{m}_i$$

$$\boldsymbol{\Sigma} = \left(\sum_i \boldsymbol{\Sigma}_i^{-1} \right)^{-1}$$

- Careful! Data sources must be **independent**!
- Further reading: “Covariance Intersection”

Synchronization

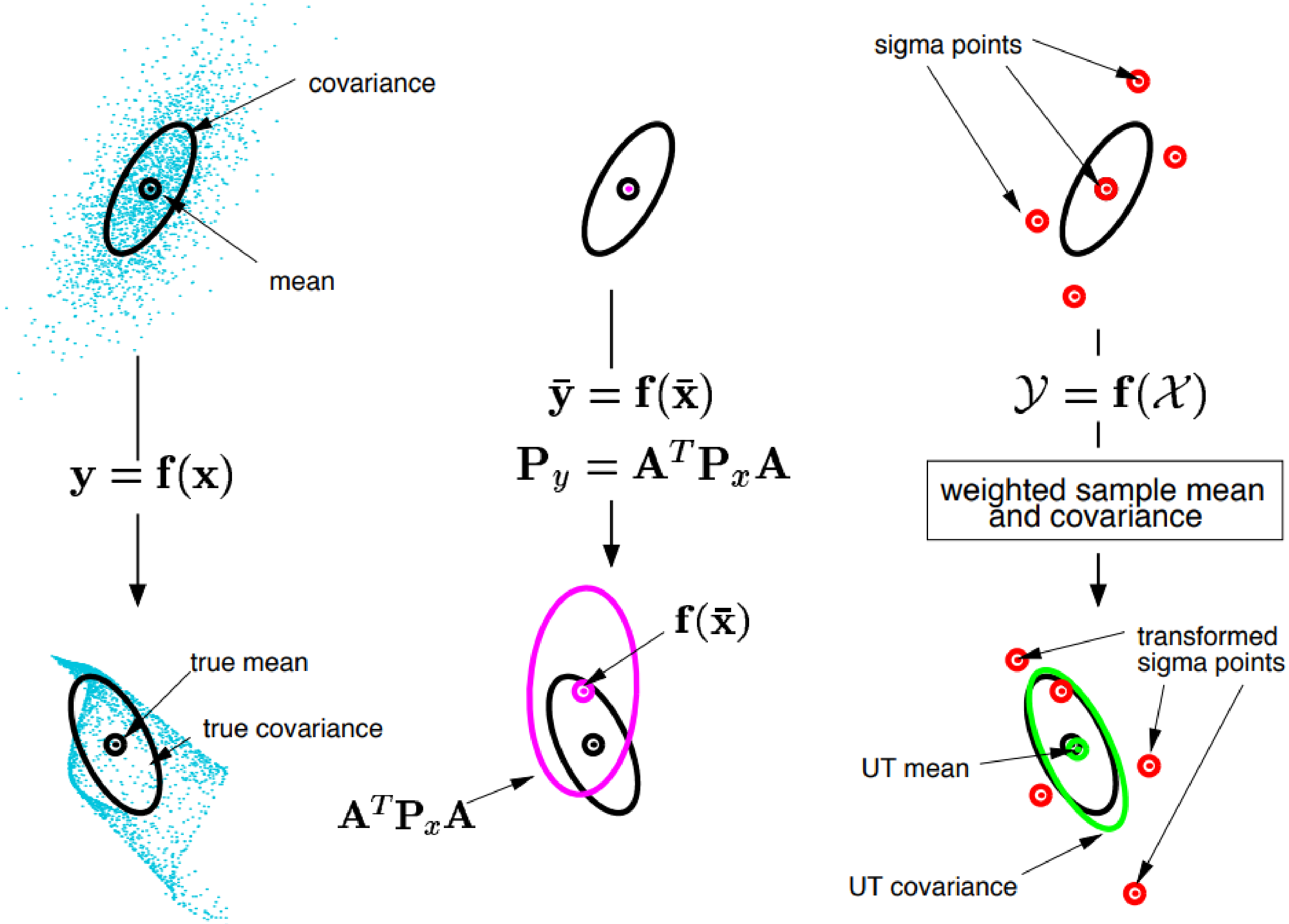
- Linear combination of two signals requires that they be sampled at the same time.
- What if they are not?
  - Oversampling.
  - Interpolation.
- Covariance of interpolated signal?
  - Use linear combination rule: $w_1 = t, w_2 = 1 - t$.
  - Careful: subsequent measurements must be *independent*!
- Synchronization is the reason why ROS messages are timestamped

$$\boldsymbol{x} = \sum_i w_i \boldsymbol{x}_i \qquad \boldsymbol{\Sigma} = \sum_i w_i^2 \boldsymbol{\Sigma}_i$$

Unscented Kalman Filter (UKF)

- Alternative linearization method to EKF.
- EKF linearizes using Taylor expansion.
- UKF linearizes by fitting key points in PDF:
  - Pick sample points on PDF.
  - Run each point through the system model.
  - Reconstruct the mean and covariance from output points.
  - Do the same for measurement model.
  - Calculate the Kalman gain and update the state as usual.
- Produces covariance and mean that are closer to true covariance and mean in highly non-linear systems.

Visualization of UKF



Source: The Unscented Filter For Non-Linear Estimation, Wan and Van Der Merwe, Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium

Generating Sample Points

$\boldsymbol{x}, \boldsymbol{\Sigma}$ - Mean and covariance of dimension L

Generate $2L + 1$ sample points

$$\boldsymbol{\chi}^{[0]} = \boldsymbol{x}$$

$$\boldsymbol{\chi}^{[i]} = \boldsymbol{x} + \left(\sqrt{(L + \lambda)\boldsymbol{\Sigma}}\right)_i \qquad i = 1, \ldots L$$

$$\boldsymbol{\chi}^{[i]} = \boldsymbol{x} - \left(\sqrt{(L + \lambda)\boldsymbol{\Sigma}}\right)_{i-L} \quad i = L + 1, \ldots 2L$$

Generating Sample Points

$$\boldsymbol{\chi}^{[i]} = \boldsymbol{x} + \left(\sqrt{(L+\lambda)\boldsymbol{\Sigma}}\right)_i \qquad i = 1,..L$$

Careful! This is the square root of the matrix, not element wise square root!

*i*th row of the matrix

Must be positive

$$\lambda = \alpha^2(L+\kappa) - L$$

Typically 1..3

Typically 0

Aside:

square root of a matrix:

$$\boldsymbol{M}^{1/2} = \boldsymbol{V}\boldsymbol{\Lambda}^{1/2}\boldsymbol{V}^{-1}$$

Mean and Covariance Reconstruction

$\boldsymbol{\Upsilon}^{[i]} = \mathbf{g}(\boldsymbol{\chi}^{[i]})$ - Point passed through system model

Reconstruct mean and covariance

$$\overline{\boldsymbol{x}} = \sum_{i=0}^{2L} w_m^{[i]} \boldsymbol{\Upsilon}^{[i]}$$

$$\overline{\boldsymbol{\Sigma}} = \sum_{i=0}^{2L} w_c^{[i]} \left(\boldsymbol{\Upsilon}^{[i]} - \overline{\boldsymbol{x}}\right)\left(\boldsymbol{\Upsilon}^{[i]} - \overline{\boldsymbol{x}}\right)^T$$

Weight Factors

Typically 2, for pure Gaussian PDF

$$w_m^{[0]} = \frac{\lambda}{L + \lambda} \qquad w_c^{[0]} = \frac{\lambda}{L + \lambda} + (1 - \alpha^2 + \beta)$$

Must be positive

$$w_m^{[i]} = w_c^{[i]} = \frac{1}{2(L + \lambda)}$$