Estimation in Information Fusion

FUSION Bootcamp, 2025

Gustaf Hendeby

Linköping University

gustaf.hendeby@liu.se

LINKÖPING UNIVERSITY

Presentation Overview

1. Introduction to Estimation
2. Parameter Estimation
3. State Estimation
4. Summary

Introduction to Estimation

LINKÖPING UNIVERSITY

Presentation Overview

1. Introduction to Estimation
2. Parameter Estimation
3. State Estimation
4. Summary

The Estimation Problem

Parameter Estimation

Based on measurements $y_{1:N} = (y_1, \ldots, y_N)$ estimate the parameter x.

Example:

- Sensor positions.
- Camera intrinsics.
- Target in sensor network.

State Estimation

Based on measurements $y_{1:N} = (y_1, \ldots, y_N)$ estimate dynamic state x_k, which changes over time.

Example:

- Your own current position.
- The movements of a target.
- The battery charge over time.

Statistical Inference: Bayes vs Fisher (frequentist)

Fisher approach

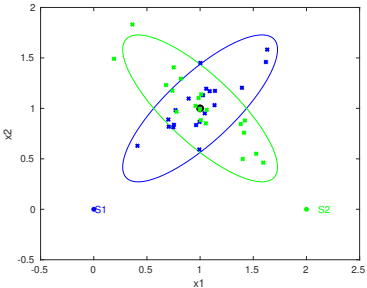
Parameter/State:	Deterministic but unknown value.
Result:	Point estimate, $\hat{x}$ and covariance P.
Prior knowledge:	—
Key concept:	Likelihood, $p(y\|x)$.

Bayes approach

Parameter/State:	Random distribution.
Result:	Posterior distribution, $p(x\|y)$.
Prior knowledge:	$p(x)$
Key concept:	Bayes' theorem $p(x\|y) = p(y\|x)p(x)/p(y)$.

Example: localization using a sensor network

Triangulation, as used by seamen for a long time. Assume two sensors that each measure bearing to target accurately, and range less accurately (or *vice versa*). How to fuse these measurements?

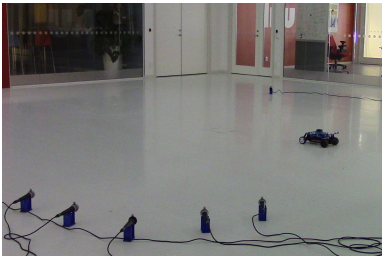


- Use all four measurements with the same weight. Poor range information may destroy the estimate.
- Discard uncertain range information, gives a triangulation approach with algebraic solution.
- Weigh the information according to its precision.

Example: acoustic localization

Vehicle sends out regular acoustic 'pings'.
Time synchronized microphones detect ping arrival times.

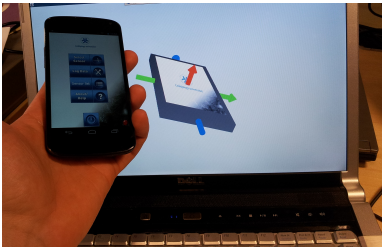
Try it yourself:



- **Localization:** Can the vehicle be located if the microphone positions are known? If the ping times are known? If the ping times are unknown?
- **Mapping:** Can the microphone positions be estimated if the vehicle position is known?
- **Simultaneous Localization and Mapping (SLAM):** Can both the target and microphone positions be estimated?

Example: smartphone orientation

Smartphones constantly collects acceleration, gyroscope, and magnetometer measurements, to provide virtual orientation measurements.



Try it yourself:

- **Noise:** How do real measurements behave?
- **Calibration:** How can the sensors be calibrated?
- **Outliers:** What are outliers? How to deal with them?

Parameter Estimation

LINKÖPING UNIVERSITY

Presentation Overview

Observation Models

- Relate parameter and observation:

$$y_k = h_k(x) + e_k$$

- Observation and model imperfections are modeled as noise, $e_k \sim p_e(e)$.
  - Zero mean, *i.e.*, $\mathsf{E}(e_k) = 0$.
  - Assumed variance, $\mathsf{var}(e_k) = R_k$.
  - Independent over time.

Examples:

- Radar: range, bearing, azimuth, Doppler
- Accelerometer: specific force
- Time of arrival: detecting pulse arrivals

Observation Models

- Relate parameter and observation:

$$y_k = h_k(x) + e_k$$

- Observation and model imperfections are modeled as noise, $e_k \sim p_e(e)$.
  - Zero mean, *i.e.*, $\mathrm{E}(e_k) = 0$.
  - Assumed variance, $\mathrm{var}(e_k) = R_k$.
  - Independent over time.

Examples:

- Radar: range, bearing, azimuth, Doppler
- Accelerometer: specific force
- Time of arrival: detecting pulse arrivals

Special Case: linear Gaussian model

$$y_k = H_k x + e_k, \quad e_k \sim \mathcal{N}(0, R_k)$$

Least Squares (LS) Point Estimate

General Point Estimate Problem:

$$\hat{x} = \arg\min_{x} V(x)$$

Least Squares (LS) Point Estimate

General Point Estimate Problem:

$$\hat{x} = \arg\min_x V(x)$$

- *Least squares* (LS) minimizes the loss function

$$\begin{aligned} V^{\text{LS}}(x) &= \textstyle\sum_{k=1}^{N} \big(y_k - h(x)\big)^T \big(y_k - h(x)\big) \\ &= \big(\mathbf{y}_N - \mathbf{h}_N(x)\big)^T \big(\mathbf{y}_N - \mathbf{h}_N(x)\big). \end{aligned}$$

- In the following sloppy notation **bold face** is used for stacking:

$$\mathbf{y}_N = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} \qquad \mathbf{h}_N(x) = \begin{pmatrix} h_1(x) \\ \vdots \\ h_N(x) \end{pmatrix} \qquad \mathbf{R}_N = \begin{pmatrix} R_1(x) & & 0 \\ & \ddots & \\ 0 & & R_N \end{pmatrix}$$

- Solve with suitable optimization method.

Least Squares (LS) Point Estimate

General Point Estimate Problem:

$$\hat{x} = \arg\min_x V(x)$$

- *Weighted Least squares* (WLS) minimizes the loss function

$$\begin{aligned} V^{\text{WLS}}(x) &= \textstyle\sum_{k=1}^{N} \big(y_k - h(x)\big)^T R_k^{-1} \big(y_k - h(x)\big) \\ &= \big(\mathbf{y}_N - \mathbf{h}_N(x)\big)^T \mathbf{R}_N^{-1} \big(\mathbf{y}_N - \mathbf{h}_N(x)\big). \end{aligned}$$

  The weights compensates for different trust in the observations.
- In the following sloppy notation **bold face** is used for stacking:

$$\mathbf{y}_N = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} \qquad \mathbf{h}_N(x) = \begin{pmatrix} h_1(x) \\ \vdots \\ h_N(x) \end{pmatrix} \qquad \mathbf{R}_N = \begin{pmatrix} R_1(x) & & 0 \\ & \ddots & \\ 0 & & R_N \end{pmatrix}$$

- Solve with suitable optimization method.

Special Case: linear least squares

Linear model

$$y_k = H_k x + e_k, \qquad \mathsf{cov}(e_k) = R_k, \quad k = 1, \dots, N,$$
$$\mathbf{y}_N = \mathbf{H}_N x + \mathbf{e}_N, \qquad \mathsf{cov}(\mathbf{e}_N) = \mathbf{R}_N.$$

WLS loss function

$$V^{WLS}(x) = \sum_{k=1}^{N} (y_k - H_k x)^T R_k^{-1} (y_k - H_k x) = (\mathbf{y}_N - \mathbf{H}_N x)^T \mathbf{R}_N^{-1} (\mathbf{y}_N - \mathbf{H}_N x).$$

Solution in batch form

$$\hat{x} = \left(\sum_{k=1}^{N} H_k^T R_k^{-1} H_k \right)^{-1} \sum_{k=1}^{N} H_k^T R_k^{-1} y_k \qquad = (\mathbf{H}_N^T \mathbf{R}_N^{-1} \mathbf{H}_N)^{-1} \mathbf{H}_N^T \mathbf{R}_N^{-1} \mathbf{y}_N,$$

$$P = \left(\sum_{k=1}^{N} H_k^T R_k^{-1} H_k \right)^{-1} \qquad = (\mathbf{H}_N^T \mathbf{R}_N^{-1} \mathbf{H}_N)^{-1}.$$

Least Squares Estimator: properties

- Assuming R_k is chosen as $\operatorname{cov}(e_k)$:
  - WLS estimator is the *best linear unbiased estimator* (BLUE) for linear problems.
  - WLS estimator is efficient for linear Gaussian problems.
- For linear Gaussian problems with independent noise, the WLS estimate is the equivalent to the *maximum likelihood* (ML) estimator

$$\hat{x}^{\mathrm{ML}} = \arg\max_x p(\mathbf{y}_N|x)$$

- The linear WLS is fast and easy to solve, the nonlinear problem might become a tricky optimization problem.

Example: fusion formula

The fusion formula for two independent estimates is

$$\begin{cases} \mathsf{E}(\hat{x}_1) = \mathsf{E}(\hat{x}_2) = x \\ \mathsf{cov}(\hat{x}_1) = P_1 \\ \mathsf{cov}(\hat{x}_2) = P_2 \end{cases}$$

Identify and plug into WLS equations:

$$\mathbf{y}_2 = \begin{pmatrix} \hat{x}_1 \\ \hat{x}_2 \end{pmatrix} \qquad \mathbf{H}_2 = \begin{pmatrix} I \\ I \end{pmatrix} \qquad \mathbf{R}_2 = \begin{pmatrix} P_1 & 0 \\ 0 & P_2 \end{pmatrix}$$

The estimate becomes:

$$\begin{aligned} \hat{x} &= (\mathbf{H}_2^T \mathbf{R}_2 \mathbf{H}_2)^{-1} \mathbf{H}_2^T \mathbf{R}_2 \mathbf{y}_2 = P(P_1^{-1}\hat{x}_1 + P_2^{-1}\hat{x}_2) \\ P &= (\mathbf{H}_2^T \mathbf{R}_2 \mathbf{H}_2)^{-1} = (P_1^{-1} + P_2^{-1})^{-1} \end{aligned}$$

Proof: ML equals WLS for independent Gaussian noise

For independent Gaussian measurement noise, $e_k \sim \mathcal{N}(0, R_k)$, R_k assumed known,

$$p(y_k|x) \propto \exp\Big(-\tfrac{1}{2}\big(y_k - h(x)\big)^T R_k^{-1}\big(y_k - h(x)\big)\Big)$$

$$p(y_{1:N}|x) = \prod_{k=1}^{N} p(y_k|x)$$

Note that $\log$ is a strictly increasing function yields, ignoring constants,

$$\arg\max_x p(y_{1:N}|x) = \arg\min_x -\log p(y_{1:N}|x) = \arg\min_x -\sum_{k=1}^{N} \log p(y_k|x)$$

$$= \arg\min_x -\sum_{k=1}^{N} \Big(-\tfrac{1}{2}\big(y_k - h(x)\big)^T R_k^{-1}\big(y_k - h(x)\big)\Big) = \arg\min_x V^{WLS}(x).$$

Hence, *the ML and WLS estimates are the same* in this case!

Iterative Estimation (Bayes approach)

The WLS works well for batch problems, but what if we want to improve our measurements as new measurements arrive? That is, $\hat{x}_1(\mathbf{y}_1)$, $\hat{x}_2(\mathbf{y}_2)$, ..., do we have to resolve the full problem each time?

Iterative Estimation (Bayes approach)

The WLS works well for batch problems, but what if we want to improve our measurements as new measurements arrive? That is, $\hat{x}_1(\mathbf{y}_1)$, $\hat{x}_2(\mathbf{y}_2)$, ..., do we have to resolve the full problem each time?

Bayes' theorem

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

Assume $p(x)$ represents the knowledge after k measurements (the estimate $\hat{x}_{k-1}$), $p(y|x)$ the likelihood for the next measurement y_k, then $p(x|y)$ represents the new estimate $\hat{x}_k$.

Iterative Estimation: a solution assuming Gaussian noise

Assume a prior of $x \sim \mathcal{N}(\hat{x}_{k-1}, P_{k-1})$, and measurement y_k:

1. Form the stochastic variable

$$\begin{pmatrix} x \\ e_k \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \hat{x}_{k-1} \\ 0 \end{pmatrix}, \begin{pmatrix} P_{k-1} & 0 \\ 0 & R_k \end{pmatrix}\right).$$

2. Find the (approximately) mapped Gaussian stochastic variable

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \\ h(x, e_k) \end{pmatrix} \overset{\text{approx.}}{\sim} \mathcal{N}\left(\begin{pmatrix} \hat{x}_k \\ \hat{y}_k \end{pmatrix}, \begin{pmatrix} P_k & P^{xy} \\ P^{yx} & P^{yy} \end{pmatrix}\right).$$

3. Apply the formula for conditional Gaussian distributions

$$\begin{aligned} \hat{x}_k &= \hat{x}_{k-1} + P^{xy}(P^{yy})^{-1}(y_k - \hat{y}_k) \\ P_k &= P_{k-1} - P^{xy}(P^{yy})^{-1}P^{yx}. \end{aligned}$$

(Approximate) Mapping Methods (1/2)

Problem formulation:

Given the mapping $z = g(u)$ of a Gaussian variable $u \sim \mathcal{N}(\hat{u}, P_u)$. Approximate the output with a new Gaussian distribution $z \overset{\text{approx.}}{\sim} \mathcal{N}(\hat{z}, P_z)$.

Analytic methods

Linear model

For $z = Gu$ and $u \sim \mathcal{N}(\hat{u}, P_u)$ the exact solution is

$$z \sim \mathcal{N}(G\hat{u}, GP_uG^T).$$

First order Taylor approximation

Approximate $z \approx g(\hat{u}) + \nabla_u g(u)(u - \hat{u})$, yielding

$$z \overset{\text{approx.}}{\sim} \mathcal{N}\big(g(\hat{u}), \nabla_u g(\hat{u}) P_u \nabla_u^T g(\hat{u})\big).$$

(Approximate) Mapping Methods (2/2)

Sample based methods

Monte Carlo simulations

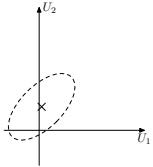
Draw $N \gg 0$ random samples of u, $u^{(i)} \sim p_u(u^{(i)})$ and compute the approximation $z \overset{\text{approx.}}{\sim} \mathcal{N}(\hat{z}, P_z)$ using a sample mean and covariance

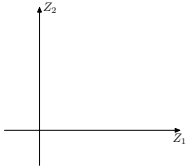
$$z^{(i)} = g(u^{(i)}) \qquad \hat{z} = \frac{1}{N}\sum_{i=1}^{N} z^{(i)} \qquad P_z = \frac{1}{N-1}\sum_{i=1}^{N}\big(z^{(i)} - \hat{z}\big)\big(z^{(i)} - \hat{z}\big)^T.$$

Unscented transformation (UT) Similar to above, but use deterministic samples and other (non-intuitive) weights.

Unscented Transform

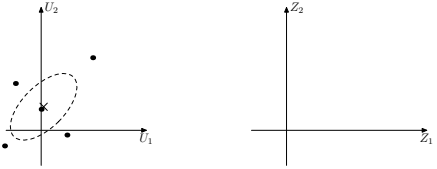
Fundamental idea:





Unscented Transform

Fundamental idea:



- Generate sigma points

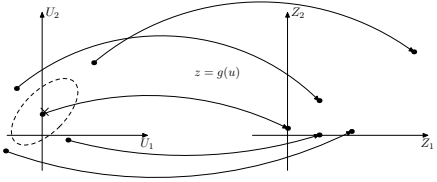
Generate $2n_u + 1$ *sigma points*:

$$u^{(0)} = \hat{u} \qquad \omega^{(0)} = \frac{\lambda}{n_u + \lambda}, \quad \omega_c^{(0)} = (1 - \alpha^2 + \beta) + \omega^{(0)}$$

$$u^{(\pm i)} = \hat{u} \pm \sqrt{n_u + \lambda}\, P_{:,i}^{1/2}, \qquad \omega^{(\pm i)} = \omega_c^{(\pm i)} = \frac{1}{2(n_u + \lambda)}, \quad i = 1, 2, \ldots, n_u$$

Unscented Transform

Fundamental idea:



• Generate • Transform sigma points

Generate $2n_u + 1$ *sigma points*, transform these:

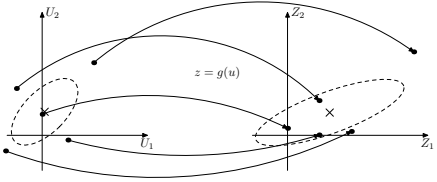
$$u^{(0)} = \hat{u} \qquad \omega^{(0)} = \frac{\lambda}{n_u + \lambda}, \quad \omega_c^{(0)} = (1 - \alpha^2 + \beta) + \omega^{(0)}$$

$$u^{(\pm i)} = \hat{u} \pm \sqrt{n_u + \lambda}\, P_{:,i}^{1/2}, \qquad \omega^{(\pm i)} = \omega_c^{(\pm i)} = \frac{1}{2(n_u + \lambda)}, \quad i = 1, 2, \ldots, n_u$$

$$z^{(i)} = g(u^{(i)})$$

Unscented Transform

Fundamental idea:



• Generate • Transform • Recombine sigma points

Generate $2n_u + 1$ *sigma points*, transform these, and fit a Gaussian distribution:

$$u^{(0)} = \hat{u} \qquad \omega^{(0)} = \frac{\lambda}{n_u+\lambda}, \quad \omega_c^{(0)} = (1 - \alpha^2 + \beta) + \omega^{(0)}$$

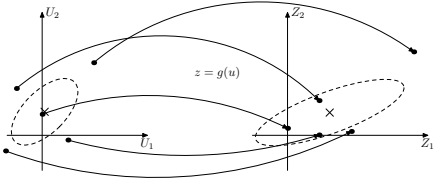
$$u^{(\pm i)} = \hat{u} \pm \sqrt{n_u + \lambda}\, P^{1/2}_{:,i}, \qquad \omega^{(\pm i)} = \omega_c^{(\pm i)} = \frac{1}{2(n_u+\lambda)}, \quad i = 1, 2, \ldots, n_u$$

$$z^{(i)} = g(u^{(i)})$$

$$\hat{z} = \sum_{i=-n_u}^{n_u} \omega^{(i)} z^{(i)} \qquad P_z = \sum_{i=-n_u}^{n_u} \omega_c^{(i)} \big(z^{(i)} - \mathsf{E}(z)\big)\big(z^{(i)} - \mathsf{E}(z)\big)^T$$

Unscented Transform

Fundamental idea:



- Generate
- Transform
- Recombine sigma points

UT parameters

- λ is defined by $\lambda = \alpha^2(n_x + \kappa) - n_x$.
- α controls the spread of the sigma points and is suggested to be chosen around 10^{-3}.
- β compensates for the distribution, and should be chosen to $\beta = 2$ for Gaussian distributions.
- κ is usually chosen to zero.

Generate $2n_u + 1$ *sigma points*, transform these, and fit a Gaussian distribution:

$$u^{(0)} = \hat{u} \qquad \omega^{(0)} = \frac{\lambda}{n_u + \lambda}, \quad \omega_c^{(0)} = (1 - \alpha^2 + \beta) + \omega^{(0)}$$

$$u^{(\pm i)} = \hat{u} \pm \sqrt{n_u + \lambda}\, P_{:,i}^{1/2}, \qquad \omega^{(\pm i)} = \omega_c^{(\pm i)} = \frac{1}{2(n_u + \lambda)}, \quad i = 1, 2, \ldots, n_u$$

$$z^{(i)} = g(u^{(i)})$$

$$\hat{z} = \sum_{i=-n_u}^{n_u} \omega^{(i)} z^{(i)} \qquad P_z = \sum_{i=-n_u}^{n_u} \omega_c^{(i)} \big(z^{(i)} - \mathsf{E}(z)\big)\big(z^{(i)} - \mathsf{E}(z)\big)^T$$

Cramér-Rao Lower Bound (CRLB)

Theorem (Cramér-Rao Lower Bound)

For any unbiased $\hat{x}$ ($\mathsf{E}(\hat{x}) = x^0$), the estimate satisfies

$$\mathsf{cov}(\hat{x}) \succeq \mathcal{I}^{-1}(x^0),$$

where $\mathcal{I}(x)$ is the Fisher information matrix (FIM),

$$\begin{aligned}\mathcal{I}(x) &= -\mathsf{E}\Big(\nabla_x^2 \log p_{\mathbf{e}}\big(\mathbf{y} - \mathbf{h}(x)\big)\Big) \\ &= \mathsf{E}\Big\{\nabla_x^T \Big(\log p_{\mathbf{e}}\big(\mathbf{y} - \mathbf{h}(x)\big)\Big) \nabla_x \Big(\log p_{\mathbf{e}}\big(\mathbf{y} - \mathbf{h}(x)\big)\Big)\Big\}\end{aligned}$$

under mild regularity conditions.

Cramér-Rao Lower Bound (CRLB): results

- $\hat{x}^{\mathrm{ML}}$ is efficient: asymptotically $\mathsf{E}(\hat{x}) = x^0$ and $\mathsf{cov}(\hat{x}) = \mathcal{I}^{-1}(x^0)$.
- If $\mathbf{h}(x) = \mathbf{H}x$, then:
  - $\hat{x}^{\mathrm{WLS}}$ is a *best linear unbiased estimator* (BLUE).
  - $\mathsf{cov}(\hat{x}^{\mathrm{WLS}}) = (\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1} \succeq \mathcal{I}^{-1}(x^0)$
    with equality if and only if $\mathbf{e}$ is Gaussian.
- If $p_{\mathbf{e}}(x) = \mathcal{N}(\mathbf{e}; \mathbf{0}, \mathbf{R})$, then

$$\mathcal{I}(x) = \big(\nabla_x^T \mathbf{h}(x)\big)\mathbf{R}^{-1}\big(\nabla_x \mathbf{h}(x)\big).$$

Summary Parameter Estimation (1/2)

- **Observation Model**

$$\mathbf{y}_N = \mathbf{h}_N(x) + \mathbf{e}_N, \quad \operatorname{cov}(\mathbf{e}_N) = \mathbf{R}_N.$$

  The linear Gaussian case is an important special case, $\mathbf{y}_N = \mathbf{H}x + \mathbf{e}_N$, $\mathbf{e}_N \sim \mathcal{N}(0, \mathbf{R}_N)$.

- **Weighted Least Squares** (WLS) minimizes the cost function

$$\hat{x}^{\text{NWLS}} = \arg\min_x V^{\text{NWLS}}(x) = \arg\min_x \big(\mathbf{y}_N - \mathbf{h}_N(x)\big)^T \mathbf{R}_N^{-1} \big(\mathbf{y}_N - \mathbf{h}_N(x)\big).$$

- Closed form for **linear case**:

$$\hat{x} = \left(\sum_{k=1}^N H_k^T R_k^{-1} H_k\right)^{-1} \sum_{k=1}^N H_k^T R_k^{-1} y_k \qquad = (\mathbf{H}_N^T \mathbf{R}_N^{-1} \mathbf{H}_N)^{-1} \mathbf{H}_N^T \mathbf{R}_N^{-1} \mathbf{y}_N,$$

$$P = \left(\sum_{k=1}^N H_k^T R_k^{-1} H_k\right)^{-1} \qquad = (\mathbf{H}_N^T \mathbf{R}_N^{-1} \mathbf{H}_N)^{-1}.$$

Summary Parameter Estimation (2/2)

- **Iterative solution (Bayes)** Assume a prior of $x \sim \mathcal{N}(\hat{x}_{k-1}, P_{k-1})$, and measurement y_k:

$$\begin{pmatrix} x \\ e_k \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \hat{x}_{k-1} \\ 0 \end{pmatrix}, \begin{pmatrix} P_{k-1} & 0 \\ 0 & R_k \end{pmatrix}\right) \to \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \\ h(x, e_k) \end{pmatrix} \overset{\text{approx.}}{\sim} \mathcal{N}\left(\begin{pmatrix} \hat{x}_k \\ \hat{y}_k \end{pmatrix}, \begin{pmatrix} P_k & P^{xy} \\ P^{yx} & P^{y}y \end{pmatrix}\right).$$

  Updated estimate:

$$\begin{aligned} \hat{x}_k &= \hat{x}_{k-1} + P^{xy}(P^{yy})^{-1}(y_k - \hat{y}_k) \\ P_k &= P^{k-1} - P^{xy}(P^{yy})^{-1}P^{yx}. \end{aligned}$$

- Many **different methods exist** to perform the mapping, Taylor expansion, unscented transform, Monte Carlo simulation...
- The **Cramér-Rao lower bound (CRLB)** provides a lower bound for the covariance of an estimate.

State Estimation

LINKÖPING UNIVERSITY

Presentation Overview

1. Introduction to Estimation
2. Parameter Estimation
3. State Estimation
   - Filtering Problem
   - Motion Models
   - Kalman Filter
   - Particle Filter
   - Cramér-Rao Lower Bound
4. Summary

The State Estimation Problem

Estimate the sequence of states x_k as they evolve over time.

- The unknown state, x_k, changes over time
- Given measurements, y_k.
- Objective: estimate x_k given measurements $y_{1:m} = (y_1, \ldots, y_m)$
  - Prediction: $k > m$
  - Filtering: $k = m$
  - Smoothing: $k < m$

Notation: The index $_{k|m}$ will be used to indicate an estimate/property at time k based on observations up until time m.

Bayesian Problem Formulation and Solution

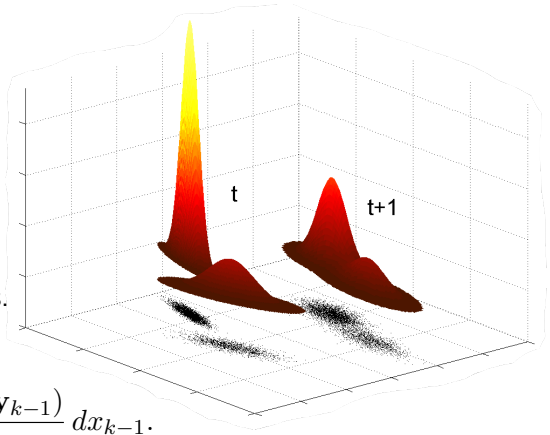
- The state x_t of interest.
- Given measurements/observations $\mathbf{y}_k = \{y_1, \ldots, y_k\}$.
- System model:

$$x_k = f(x_{k-1}, w_{k-1}) \quad \longleftrightarrow \quad p(x_k|x_{k-1})$$
$$y_k = h(x_k) + e_k \quad \longleftrightarrow \quad p(y_k|x_k)$$

  where w_{t-1} and e_t are stochastic processes.

- Bayesian solution

$$p(x_k|\mathbf{y}_k) = \int \frac{p(y_k|x_k)p(x_k|x_{k-1})p(x_{k-1}|\mathbf{y}_{k-1})}{p(y_k|\mathbf{y}_{k-1})}\, dx_{k-1}.$$



Bayesian Problem Formulation and Solution

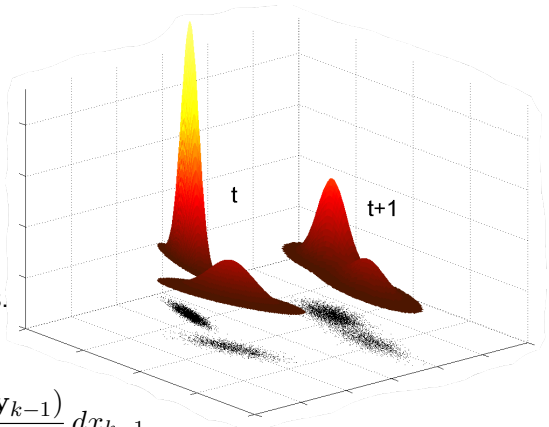
- The state x_t of interest.
- Given measurements/observations $\mathbf{y}_k = \{y_1, \ldots, y_k\}$.
- System model:

$$x_k = f(x_{k-1}, w_{k-1}) \quad \longleftrightarrow \quad p(x_k|x_{k-1})$$
$$y_k = h(x_k) + e_k \quad \longleftrightarrow \quad p(y_k|x_k)$$

  where w_{t-1} and e_t are stochastic processes.

- Bayesian solution

$$p(x_k|\mathbf{y}_k) = \int \frac{p(y_k|x_k)p(x_k|x_{k-1})p(x_{k-1}|\mathbf{y}_{k-1})}{p(y_k|\mathbf{y}_{k-1})}\, dx_{k-1}.$$



Bayesian Problem Formulation and Solution

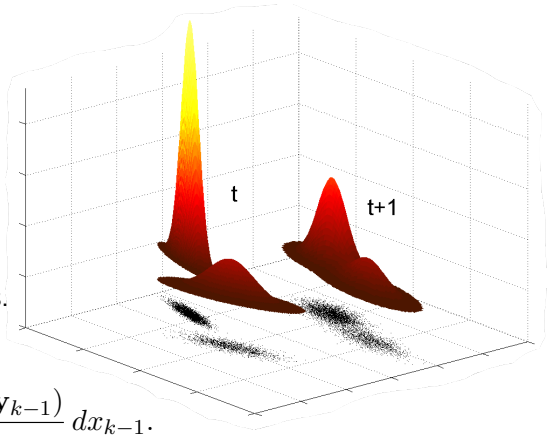
- The state x_t of interest.
- Given measurements/observations $\mathbf{y}_k = \{y_1, \ldots, y_k\}$.
- System model:

$$x_k = f(x_{k-1}, w_{k-1}) \quad \longleftrightarrow \quad p(x_k|x_{k-1})$$
$$y_k = h(x_k) + e_k \quad \longleftrightarrow \quad p(y_k|x_k)$$

  where w_{t-1} and e_t are stochastic processes.

- Bayesian solution

$$p(x_k|\mathbf{y}_k) = \int \frac{p(y_k|x_k)p(x_k|x_{k-1})p(x_{k-1}|\mathbf{y}_{k-1})}{p(y_k|\mathbf{y}_{k-1})}\, dx_{k-1}.$$



Bayesian Problem Formulation and Solution

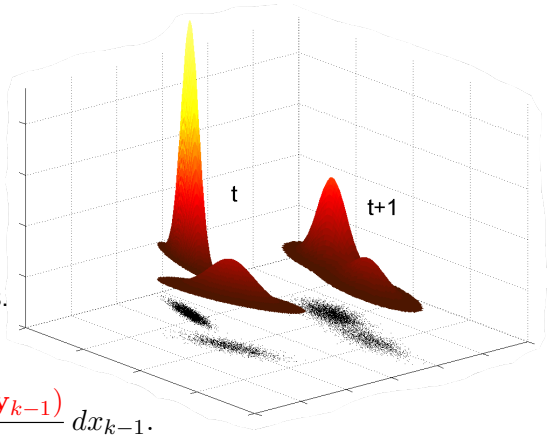
- The state x_t of interest.
- Given measurements/observations $\mathbf{y}_k = \{y_1, \ldots, y_k\}$.
- System model:

$$x_k = f(x_{k-1}, w_{k-1}) \quad \longleftrightarrow \quad p(x_k|x_{k-1})$$
$$y_k = h(x_k) + e_k \quad \longleftrightarrow \quad p(y_k|x_k)$$

  where w_{t-1} and e_t are stochastic processes.

- Bayesian solution

$$p(x_k|\mathbf{y}_k) = \int \frac{p(y_k|x_k)p(x_k|x_{k-1})p(x_{k-1}|\mathbf{y}_{k-1})}{p(y_k|\mathbf{y}_{k-1})}\, dx_{k-1}.$$



Bayesian Problem Formulation and Solution

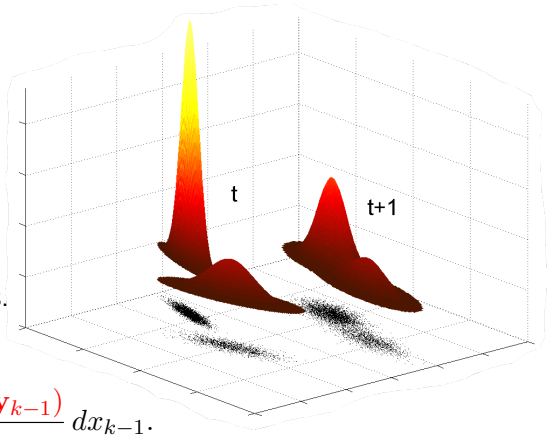
- The state x_t of interest.
- Given measurements/observations $\mathbf{y}_k = \{y_1, \ldots, y_k\}$.
- System model:

$$x_k = f(x_{k-1}, w_{k-1}) \quad \longleftrightarrow \quad p(x_k|x_{k-1})$$
$$y_k = h(x_k) + e_k \quad \longleftrightarrow \quad p(y_k|x_k)$$

  where w_{t-1} and e_t are stochastic processes.

- Bayesian solution

$$p(x_k|\mathbf{y}_k) = \int \frac{p(y_k|x_k)p(x_k|x_{k-1})p(x_{k-1}|\mathbf{y}_{k-1})}{p(y_k|\mathbf{y}_{k-1})} \, dx_{k-1}.$$



Bayesian Framework for Estimation

- Bayesian solution

$$p(x_k|\mathbf{y}_{k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|\mathbf{y}_{k-1})\,dx_{k-1} \qquad \text{(TU)}$$

$$p(x_k|\mathbf{y}_k) = \frac{p(y_k|x_k)p(x_k|\mathbf{y}_{k-1})}{p(y_k|\mathbf{y}_{k-1})} \qquad \text{(MU)}$$

- Two-stroke engine:
  - Time update (TU): Predict the future
  - Measurement update (MU): Correct prediction based on measurement
- Only a few analytic solutions:
  - Linear Gaussian model $\Rightarrow$ Kalman filter (KF)
  - Hidden Markov model (HMM)

Bayesian Framework for Estimation

- Bayesian solution

$$p(x_k|\mathbf{y}_{k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|\mathbf{y}_{k-1})\,dx_{k-1} \qquad \text{(TU)}$$

$$p(x_k|\mathbf{y}_k) = \frac{p(y_k|x_k)p(x_k|\mathbf{y}_{k-1})}{p(y_k|\mathbf{y}_{k-1})} \qquad \text{(MU)}$$

- Two-stroke engine:
  - Time update (TU): Predict the future
  - Measurement update (MU): Correct prediction based on measurement
- Only a few analytic solutions:
  - Linear Gaussian model $\Rightarrow$ Kalman filter (KF)
  - Hidden Markov model (HMM)
- In most cases approximations are needed:
  - Analytic
  - Stochastic

State-Space Models

Nonlinear model:

$$\begin{aligned} x_{k+1} &= f(x_k, v_k) \quad &\longleftrightarrow \quad & x_{k+1}|x_k \sim p(x_{k+1}|x_k) \\ y_k &= h(x_k, e_k) \quad &\longleftrightarrow \quad & y_k|x_k \sim p(y_k|x_k) \end{aligned}$$

Nonlinear model with additive noise:

$$\begin{aligned} x_{k+1} &= f(x_k) + v_k \quad &\longleftrightarrow \quad & x_{k+1}|x_k \sim p(x_{k+1}|x_k) = p_{v_k}\big(x_{k+1} - f(x_k)\big) \\ y_k &= h(x_k) + e_k \quad &\longleftrightarrow \quad & y_k|x_k \sim p(y_k|x_k) = p_{e_k}\big(y_k - h(x_k)\big) \end{aligned}$$

Linear model:

$$\begin{aligned} x_{k+1} &= F_k x_k + G_k v_k \\ y_k &= H_k x_k + e_k \end{aligned}$$

Gaussian model: $v_k \sim \mathcal{N}(0, Q_k)$, $e_k \sim \mathcal{N}(0, R_k)$ and $x_0 \sim \mathcal{N}(0, P_0)$

Target Motion Models: constant velocity

Newton's force law $F = ma$ gives the "nearly constant velocity model" in n dimensions:

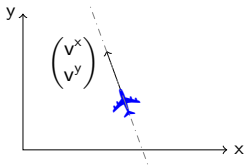
$$x_{k+1} = F_k x_k + G_k v_k = \begin{pmatrix} I_n & TI_n \\ 0_n & I_n \end{pmatrix} x_k + \begin{pmatrix} \frac{T^2}{2} I_n \\ TI_n \end{pmatrix} v_k$$

where $x_k = \begin{pmatrix} \mathsf{p}_k \\ \mathsf{v}_k \end{pmatrix}$.

Interpretation:

$$\mathsf{p}_{k+1} = \mathsf{p}_k + T\mathsf{v}_k + \tfrac{T^2}{2} v_k$$
$$\mathsf{v}_{k+1} = \mathsf{v}_k + T v_k$$

where process noise corresponds to acceleration, $v_k = a_k$.



Constant velocity

Linear transformation of independent stochastic vectors implies:

$$x_{k+1} = F_k x_k + G_k v_k, \quad x_k \sim \mathcal{N}(\hat{x}_{k|k}, P_{k|k}), \quad v_k \sim \mathcal{N}(0, Q_k)$$
$$\implies x_{k+1} \sim \mathcal{N}\big(F\hat{x}_{k|k}, F_k P_{k|k} F_k^T + G_k Q_k G_k^T\big)$$

- The nearly *constant acceleration* (CA) is also common in target tracking for agile targets. It is defined analogously to CV but the input is jerk.

Target Motion Models: coordinated turn

- (Nearly) *coordinated turn* (CT) model, *i.e.*, nearly constant speed, constant turn rate model
- State with Cartesian velocity

$$x_t = \begin{pmatrix} \mathsf{x}_t & \mathsf{y}_t & \mathsf{v}_t^{\mathsf{x}} & \mathsf{v}_t^{\mathsf{y}} & \omega_t \end{pmatrix}^T$$

Continuous time description

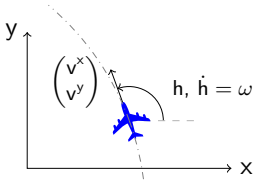
$$\dot{\mathsf{x}} = \mathsf{v}\cos(\mathsf{h}) \qquad \dot{\mathsf{y}} = \mathsf{v}\sin(\mathsf{h}),$$

from which the following differential equations are obtained

$$\ddot{\mathsf{x}} = \frac{d}{dt}\dot{\mathsf{x}} = -\mathsf{v}\dot{\mathsf{h}}\sin(\mathsf{h}) = -\omega\dot{\mathsf{y}}$$

$$\ddot{\mathsf{y}} = \frac{d}{dt}\dot{\mathsf{y}} = \mathsf{v}\dot{\mathsf{h}}\cos(\mathsf{h}) = \omega\dot{\mathsf{x}}.$$

(Exact discretization is possible.)



Coordinated turn

Time Continuous to Time Discrete Models

Linear time-invariant (LTI) state-space model:

Continuous time	**Discrete time**
$\dot{x} = Ax + Bu$	$x_{k+1} = Fx_k + Gu_k$
$y = Cx$	$y_k = Hx_k$

u is either input or process noise.

Zero-order hold (ZOH) sampling assuming the input is piecewise constant:

$$\begin{aligned} x(t+T) &= e^{AT}x(t) + \textstyle\int_0^T e^{A\tau} B u(t+T-\tau)\, d\tau \\ &= \underbrace{e^{AT}}_{F} x(t) + \underbrace{\textstyle\int_0^T e^{A\tau}\, d\tau\, B}_{G} u(t). \end{aligned}$$

Nonlinear Motion Models (1/2)

Continuous time (physical) and discrete time counterparts

$$\dot{x}(t) = a(x(t), u(t), w(t); \theta)$$
$$x(t+T) = f(x(t), u(t), w(t); \theta, T).$$

- **Kinematic models**: Do not attempt to model forces, but are 'Black-box' multi-purpose models.
  - *Translation kinematics* describes position, often based on $F = ma$.
  - *Rotational kinematics* describes orientation.
  - *Rigid body kinematics* combines translational and rotational kinematics.
  - *Constrained kinematics.* Coordinated turns (circular path motion).
- **Application specific force models**
- **Gray-box models** Parameters θ must be calibrated (estimated, identified) from data.

Nonlinear Motion Models (2/2)

There are two options:

- **Discretized linearization** (general):
  1. Linearize:

$$A = \nabla_x a(x,u) \quad B = \nabla_u a(x,u) \quad C = \nabla_x c(x,u) \quad D = \nabla_u c(x,u)$$

  2. Discretize (sample): $F = e^{AT}$, $G = \int_0^T e^{A\tau}\, d\tau\, B$, $H = C$, and $J = D$
- **Linearized discretization**:
  1. Discretize (sample nonlinear):

$$x(t+T) = f\big(x(t), u(t)\big) = x(t) + \int_t^{t+T} a\big(x(\tau), u(\tau)\big)\, d\tau$$

  2. Linearize: $F = \nabla_x f(x_k, u_k)$ and $G = \nabla_u f(x_k, u_k)$

Sampling of State Noise

Different solutions exist, they are all approximations except in the linear case:

- v_t is white noise such that its total influence during one sample interval is TQ:
$$\bar{Q} = TQ$$
- v_t is a discrete white noise sequence with variance TQ. That is, we assume that the noise enters immediately after a sample time, so $x(t+T) = f\big(x(t) + v(t)\big)$:
$$\bar{Q} = TGQG^T$$

Recommendation

In practice simple solutions often works well, but *remember to scale with* T!

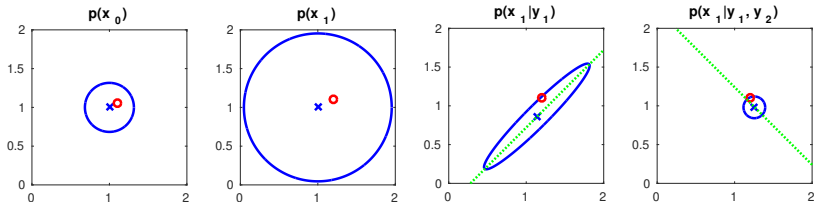
Kalman Filter (KF)

Time-varying state space model:

$$x_{k+1} = F_k x_k + G_k v_k, \qquad \operatorname{cov}(v_k) = Q_k$$
$$y_k = H_k x_k + e_k, \qquad \operatorname{cov}(e_k) = R_k$$


Rudolph E. Kalman

- Best linear unbiased estimator (BLUE).
- Optimal for Gaussian noise, reaches CRLB.
- Representation: mean $\hat{x}$ and covariance P.



Kalman Filter: algorithm

Measurement update

$$\begin{aligned}\hat{x}_{k|k} &= \hat{x}_{k|k-1} + P_{k|k-1}H_k^T S_k^{-1}(y_k - \hat{y}_k) \\ &= \hat{x}_{k|k-1} + K_k\varepsilon_k \\ P_{k|k} &= (I - K_kH_k)P_{k|k-1}(I - K_kH_k)^T + K_kR_kK_k^T \\ &= P_{k|k-1} - K_kH_kP_{k|k-1}.\end{aligned}$$

Time Update

$$\hat{x}_{k+1|k} = F_k\hat{x}_{k|k}$$

$$P_{k+1|k} = F_kP_{k|k}F_k^T + G_kQ_kG_k^T$$

Auxilary variables

$$\begin{aligned}\hat{y}_k &= H_k\hat{x}_{k|k-1} && \text{(Predicted measurement)} \\ \varepsilon_k &= y_k - H_k\hat{x}_{k|k-1} = y_k - \hat{y}_k && \text{(Innovation)} \\ S_k &= H_kP_{k|k-1}H_k^T + R_k && \text{(Innovation covariance)} \\ K_k &= P_{k|k-1}H_k^T S_k^{-1} && \text{(Kalman gain)}\end{aligned}$$

Kalman Filter Tuning

- The SNR ratio $\|Q\|/\|R\|$ is the most crucial, it sets the filter speeds. Note difference of real system and model used in the KF.
- Recommentation: fix R according to sensor specification/performance, and tune Q (motion models are anyway subjective approximations of reality).
- High SNR in the model, gives fast filter that is quick in adapting to changes/maneuvers, but with larger uncertainty (small bias, large variance).
- Conversely, low SNR in the model, gives slow filter that is slow in adapting to changes/maneuvers, but with small uncertainty (large bias, small variance).
- P_0 reflects the belief in the prior $x_1 \sim \mathcal{N}(\hat{x}_{1|0}, P_0)$. Possible to choose P_0 very large (and $\hat{x}_{1|0}$ arbitrary), if no prior information exists.
- Tune covariances in large steps (order of magnitudes)!

Robustness and Sensitivity

The following concepts are relevant for all filtering applications, but they are most explicit for KF:

- **Observability** is revealed indirectly by $P_{k|k}$; monitor its rank or better condition number.
- **Divergence tests** Monitor performance measures and restart the filter after divergence.
- **Outlier rejection** monitor sensor observations.
- **Bias error** incorrect model gives bias in estimates.
- **Sensitivity analysis** uncertain model contributes to the total covariance.
- **Numerical issues** may give complex estimates.

Approximate Kalman Filter for Nonlinear Models (1/2)

Assume Gaussian distributions in all steps in the algorithm, mean $\hat{x}_{k|l}$ and covariance $P_{k|l}$. Time and measurement step can now be approximated.

Measurement update

1. Form the stochastic variable
$$\begin{pmatrix} x_k \\ e_k \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \hat{x}_{k|k-1} \\ 0 \end{pmatrix}, \begin{pmatrix} P_{k|k-1} & 0 \\ 0 & R_k \end{pmatrix}\right).$$

2. Find the (approximately) mapped Gaussian stochastic variable
$$\begin{pmatrix} x_k \\ y_k \end{pmatrix} = \begin{pmatrix} x_k \\ h(x_k, e_k) \end{pmatrix} \overset{\text{approx.}}{\sim} \mathcal{N}\left(\begin{pmatrix} \hat{x}_{k|k-1} \\ \hat{y}_k \end{pmatrix}, \begin{pmatrix} P_{k|k-1} & P_k^{xy} \\ P_k^{yx} & S_k \end{pmatrix}\right).$$

3. Apply the formula for conditional Gaussian distributions
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + P_k^{xy} S_k^{-1}(y_k - \hat{y}_k)$$
$$P_{k|k} = P_{k|k-1} - P_k^{xy} S_k^{-1} P_k^{yx}.$$

Approximate Kalman Filter for Nonlinear Models (2/2)

Time Update

1. Form the stochastic variable
$$\begin{pmatrix} x_k \\ v_k \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \hat{x}_{k|k} \\ 0 \end{pmatrix}, \begin{pmatrix} P_{k|k} & 0 \\ 0 & Q_k \end{pmatrix}\right).$$
2. Find the (approximately) mapped Gaussian stochastic variable
$$x_{k+1} = f(x_k, v_k) \overset{\text{approx.}}{\sim} \mathcal{N}(\hat{x}_{k+1|k}, P_{k+1|k}).$$

Resulting filter:

- Linear model (exact propagation) $\Rightarrow$ normal KF.
- Linearity models with Taylor expansion $\Rightarrow$ *extended Kalman filter* (EKF).
- Propagation with unscented transform $\Rightarrow$ *unscented Kalman filter* (UKF).
- Mixing approximations is possible.

Extended Kalman Filter: algorithm (additive noise)

Measurement Update

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \varepsilon_k$$
$$P_{k|k} = P_{k|k-1} - K_k H_k P_{k|k-1}$$

$$S_k = H_k P_{k|k-1} H_k^T + R_k$$
$$K_k = P_{k|k-1} H_k^T h(\hat{x}_{k|k-1}) S_k^{-1}$$
$$\varepsilon_k = y_k - h(\hat{x}_{k|k-1})$$
$$F_k = \nabla f(\hat{x}_{k|k})$$

Time Update

$$\hat{x}_{k+1|k} = f(\hat{x}_{k|k})$$
$$P_{k+1|k} = F_k P_{k|k} F_k + Q_k$$

$$F_k = \nabla f(\hat{x}_{k|k})$$

Essentially, this is simply to use a normal Kalman filter, using the nonlinear functions for mean predictions and the Jacobian (based on the best available state estimate) where model matrices are needed in the covariance updates.

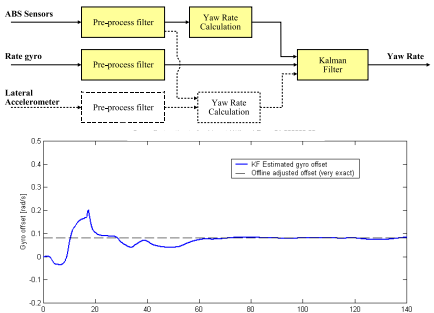
Virtual Yaw Rate Sensor

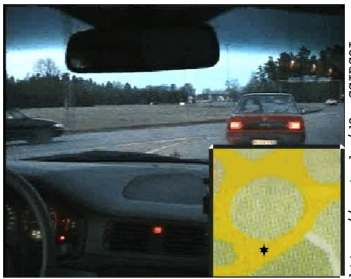
- Yaw rate subject to bias, orientation error increases linearly in time.
- Wheel speeds also give a gyro, where the orientation error grows linearly in distance.

Model, with state vector $x_k = \left(\dot{\psi}_k, \ddot{\psi}_k, b_k, \frac{r_{k,3}}{r_{k,4}}\right)$ and the measurements

$$y_k^1 = \dot{\psi}_k + b_k + e_k^1$$

$$y_k^2 = \frac{\omega_3 r_{\text{nom}} + \omega_4 r_{\text{nom}}}{2} \frac{2}{B} \frac{\frac{\omega_3}{\omega_4}\frac{r_{k,3}}{r_{k,4}} - 1}{\frac{\omega_3}{\omega_4}\frac{r_{k,3}}{r_{k,4}} + 1} + e_k^2.$$





https://youtu.be/d9rzCCIBS9I

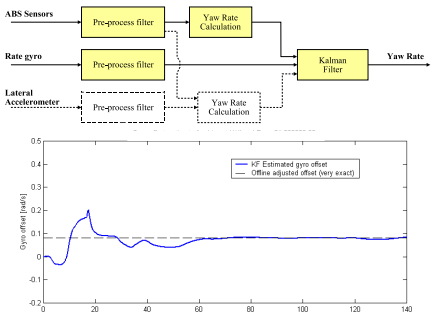
Virtual Yaw Rate Sensor

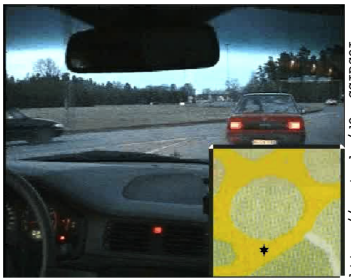
- Yaw rate subject to bias, orientation error increases linearly in time.
- Wheel speeds also give a gyro, where the orientation error grows linearly in distance.

Model, with state vector $x_k = \left(\dot{\psi}_k, \ddot{\psi}_k, b_k, \frac{r_{k,3}}{r_{k,4}}\right)$ and the measurements

$$y_k^1 = \dot{\psi}_k + b_k + e_k^1$$

$$y_k^2 = \frac{\omega_3 r_{\text{nom}} + \omega_4 r_{\text{nom}}}{2} \frac{2}{B} \frac{\frac{\omega_3}{\omega_4}\frac{r_{k,3}}{r_{k,4}} - 1}{\frac{\omega_3}{\omega_4}\frac{r_{k,3}}{r_{k,4}} + 1} + e_k^2.$$





https://youtu.be/d9rzCCIBS9I

Bayes Optimal Filter: reminder

General Bayesian recursion (time and measurement updates)

$$p(x_{k+1}|y_{1:k}) = \int p(x_{k+1}|x_k)p(x_k|y_{1:k})\,dx_k,$$
$$p(x_k|y_{1:k}) = \frac{p(y_k|x_k)p(x_k|y_{1:k-1})}{p(y_k|y_{1:k-1})}.$$

- Analytic solution available in a few special cases: linear Gaussian model (KF) and finite state-space model (HMM).
- However, for a given trajectory $x_{1:k}$, the recursion can be computed (neglect the integral).
- We can numerically evaluate different trajectories by comparing their likelihoods. But there are many possible trajectories, so Monte Carlo sampling of trajectories directly does not work.

Sample Approximation: basic idea

Postulate a **discrete approximation of the posterior**. For the predictive density, we have (sample points, $x_k^{(i)}$ and weights, $\omega^{(i)}$) :

$$\hat{p}(x_k|y_{1:k-1}) = \sum_{i=1}^{N} \omega_{k|k-1}^{(i)} \delta(x_k - x_k^{(i)}).$$

The first moments (**mean and covariance**) are simple to compute from this approximation:

$$\hat{x}_{k|k-1} = \mathsf{E}(x_k) = \sum_{i=1}^{N} \omega_{k|k-1}^{(i)} x_k^{(i)},$$

$$P_{k|k-1} = \mathsf{cov}(x_k) = \sum_{i=1}^{N} \omega_{k|k-1}^{(i)} (x_k^{(i)} - \hat{x}_{k|k-1})(x_k^{(i)} - \hat{x}_{k|k-1})^T.$$

The *maximum a posterior* (MAP) estimate can be useful:

$$\hat{x}_{k|k-1}^{\mathrm{MAP}} = \arg\max_{x_k^{(i)}} \hat{p}(x_k|y_{1:k-1}).$$

Measurement Update

The measurement follows directly, without any extra approximations

$$\hat{p}(x_k|y_{1:k}) = \sum_{i=1}^{N} \underbrace{\frac{1}{c_k} p(y_k|x_k^{(i)}) \omega_{k|k-1}^{(i)}}_{\omega_{k|k}^{(i)}} \delta(x_k - x_k^{(i)})$$

$$c_k = \sum_{i=1}^{N} p(y_k|x_k^{(i)}) \omega_{k|k-1}^{(i)}.$$

The normalization constant c_k corresponds to assuring that $\sum_{i=1}^{N} \omega_{k|k}^{(i)} = 1$.

Time Update

Bayesian time update gives a continuous distribution

$$\hat{p}(x_{k+1}|y_{1:k}) = \sum_{i=1}^{N} \omega_{k|k}^{(i)} p(x_{k+1}|x_k^{(i)}).$$

To keep the approximation form, the distribution is sampled at points $x_{k+1}^{(i)}$, and the weights are updated as

$$\omega_{k+1|k}^{(i)} = \hat{p}(x_{k+1}^{(i)}|y_{1:k}) = \sum_{j=1}^{N} \omega_{k|k}^{(j)} p(x_{k+1}^{(i)}|x_k^{(j)}), \quad i = 1, 2, \ldots, N.$$

Two principles:

- Keep the same grid, so $x_{k+1}^{(i)} = x_k^{(i)}$, which yields the point mass filter.
- Generate new samples from the posterior distribution $x_{k+1}^{(i)} \sim \hat{p}(x_{k+1}|y_{1:k})$, which yields the marginal particle filter.

Both alternatives have quadratic complexity (N weights $\omega_{k+1|k}^{(i)}$, each one involving a sum with N terms).

Point-Mass Filter (PMF)

Advantages:

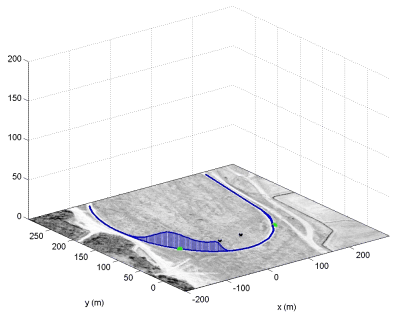
- Simple to implement.
- Works excellent when $n_x \leq 2$.
- Gives the complete posterior, not only $\hat{x}$ and P.
- Global search, no local minima.

Problems:

- Grid inefficient in higher dimensions, since the probability to be at one grid point depends on the transition probability from all other grid points.
- The grid should be adaptive (rough initially, then finer).
- Sparse matrices can be used for multi-modal distributions.

PMF Example: 2 DOA sensors, 2 targets

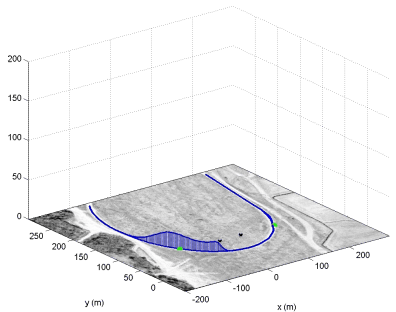
- Data from the FOCUS project (map overlay).
- Two microphone arrays (black x) compute two DOA.
- Two road-bound targets (green *).
- One grid point (stem plot) every meter on the road.



https://youtu.be/VcdTebC9uIs

PMF Example: 2 DOA sensors, 2 targets

- Data from the FOCUS project (map overlay).
- Two microphone arrays (black x) compute two DOA.
- Two road-bound targets (green *).
- One grid point (stem plot) every meter on the road.

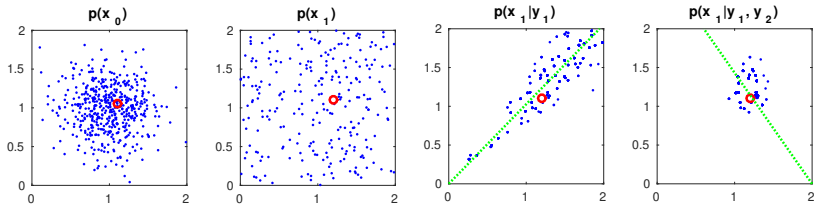


https://youtu.be/VcdTebC9uIs

Particle Filter (PF)

- Is a stochastic algorithm.
- Applicable to nonlinear and non-Gaussian models.
- Representation: randomly chosen states (particles), $\hat{p}(x) = \sum_i \omega^{(i)} \delta(x - x^{(i)})$.


Neil Gordon



Particle Filter

- Trick to avoid cubic complexity: sample trajectories, not states
- Time update for trajectory:

$$p(x^{(i)}_{1:k+1}|y_{1:k}) = \underbrace{p(x^{(i)}_{k+1}|x^{(i)}_{1:k}, y_{1:k})}_{p(x^{(i)}_{k+1}|x^{(i)}_{k})} \underbrace{p(x^{(i)}_{1:k}|y_{1:k})}_{\omega^{(i)}_{k|k}}$$
$$= \omega^{(i)}_{k|k} p(x^{(i)}_{k+1}|x^{(i)}_{k}) = \omega^{(i)}_{k+1|k}.$$

  No sum involved here!
- **Resample!** Remove unlikely particles, replicate likely ones.
- The new sample is sampled from the prior in the original PF (*sequential importance resampling* (SIR) or bootstrap PF)

$$x^{(i)}_{k+1} \sim p(x_{k+1}|x^{(i)}_{k}).$$

Basic SIR PF Algorithm

Choose the number of particles N.

Initialization: Generate $x_0^{(i)} \sim p_{x_0}, i = 1, \ldots, N$ particles, and weights $\omega_0^{(i)} = \frac{1}{N}$.

Iterate for $k = 1, 2, \ldots, t$:

1. **Measurement update:** For $k = 1, 2, \ldots,$
$$\omega_k^{(i)} = \omega_{k-1}^{(i)} p(y_k | x_k^{(i)}).$$
2. **Normalize:** $\omega_k^{(i)} := \omega_k^{(i)} / \sum_j \omega_k^{(j)}$.
3. **Estimation:** MMSE $\hat{x}_k \approx \sum_{i=1}^N \omega_k^{(i)} x_k^{(i)}$ or MAP.
4. **Resampling:** Bayesian bootstrap: Take N samples with replacement from the set $\{x_k^{(i)}\}_{i=1}^N$ where the probability to take sample i is $\omega_k^{(i)}$. Let $\omega_k^{(i)} = 1/N$.
5. **Prediction:** Generate random process noise samples
$$v_k^{(i)} \sim p_{v_k}, \quad x_{k+1}^{(i)} = f(x_k^{(i)}, v_k^{(i)}).$$

Particle Filter: pseudo-code

Input arguments

- `m` a state-space model
- measurements `y`, and
- known input `u`.

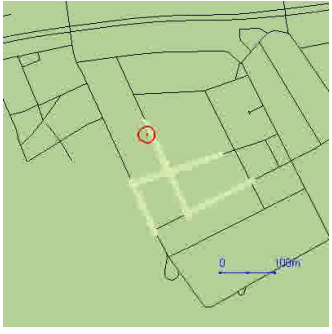
Return values

- `xhat` estimates.

```
xp = m.x0.' + rand(m.px0, Np);  % Initialization
for k = 1:N
   % Time update
   v = rand(m.pv, Np);  % Random process noise
   xp = m.f(k, xp.', u(:,k), m.th).' + v;  % State prediction
   % Measurement update
   yp = m.h(k, xp.', u(:, k).', m.th).';  % Measurement prediction
   w = pdf(m.pe, y(:,k).'-yp);  % Likelihood
   xhat(k,:) = mean(w(:).*xp);  % Estimation
   [xp, w] = resample(xp, w);  % Resampling
   xMC(:, k, :) = xp;  % MC uncertainty repr.
end
```

Example: 2D street navigation by odometry

Wheel speed sensors provide speed and yaw rate, street map provides constraints on turns.

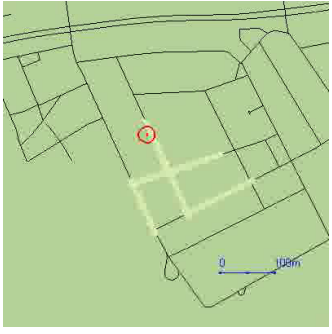


https://youtu.be/aL0Tbt9Aw6k

Video illustrates how a uniform prior over a part of the road network eventually converge to a single particle cluster when sufficient information is obtained (but many local particle clusters initially).

Example: 2D street navigation by odometry

Wheel speed sensors provide speed and yaw rate, street map provides constraints on turns.

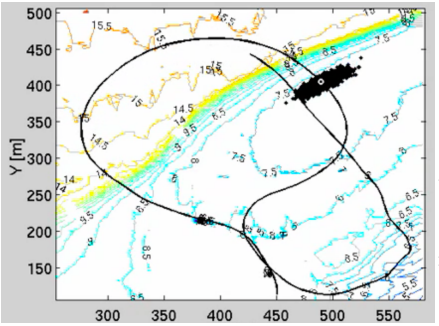


https://youtu.be/aL0Tbt9Aw6k

Video illustrates how a uniform prior over a part of the road network eventually converge to a single particle cluster when sufficient information is obtained (but many local particle clusters initially).

Example: 2D underwater navigation

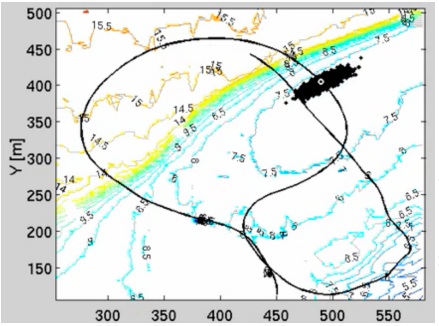
Underwater vessel measures its own depth and distance to bottom, and sea chart provides depth $h(x_k)$.



Video shows how a uniform prior quickly converges to a unimodal particle cloud. Note how the cloud changes form when passing the ridge.

Example: 2D underwater navigation

Underwater vessel measures its own depth and distance to bottom, and sea chart provides depth $h(x_k)$.



Video shows how a uniform prior quickly converges to a unimodal particle cloud. Note how the cloud changes form when passing the ridge.

CRLB: filtering

- CRLB developed for static parameter x, with many measurements $y_{1:k}$.
- The filtering CRLB concerns the case where x is replaced with $x_{1:k}$, with the constraints $x_{n+1} = f(x_n) + v_n$, $n = 1, 2, \ldots, k-1$.
- Two cases:
  - Parametric CRLB for filtering: $x_{1:k}$ is seen as a parameter with a *true value* $x^0_{1:k}$.
  - Posterior, or Bayesian, CRLB for filtering: $x_{1:k}$ is seen as a stochastic variable with a *prior* $p(x_{1:k})$.
- Parametric CRLB better in practice: easy to calculate, easy to interpret (given a certain trajectory and model, how well can a nonlinear filter estimate this trajectory?)
- Posterior CRLB useful for theoretical studies.

Parametric CRLB

- The parametric CRLB gives a lower bound on estimation error for a fixed trajectory $x_{1:k}$. That is, $\operatorname{cov}(\hat{x}_{k|k}) \succeq P^{\text{CRLB}}_{k|k}$.
- Algorithm identical to the Riccati equation (covariance update) in KF, where the gradients are evaluated along the trajectory $x_{1:k}$:

$$
\begin{aligned}
P_{k+1|k} &= F_k P_{k|k} F_k^T + G_k Q_k G_k, \\
P_{k+1|k+1} &= P_{k+1|k} - P_{k+1|k} H_k^T (H_k P_{k+1|k} H_k^T + R_k)^{-1} H_k P_{k+1|k}, \\
F_k &= \nabla_{x_k} f(x_k, v_k), \\
G_k &= \nabla_{v_k} f(x_k, v_k), \\
H_k &= \nabla_{x_k} h(x_k, e_k).
\end{aligned}
$$

Posterior CRLB

- Average FIM over all possible trajectories $x_{1:k}$ with respect to v_k.
- Much more complicated expressions.
- For linear system, the parametric and posterior CRLB coincide.

Summary State Estimation (1/3)

- **State-space model:**

$$\begin{aligned} x_{k+1} &= f(x_k, v_k) \quad &\longleftrightarrow \quad & x_{k+1}|x_k \sim p(x_{k+1}|x_k) \\ y_k &= h(x_k, e_k) \quad &\longleftrightarrow \quad & y_k|x_k \sim p(y_k|x_k) \end{aligned}$$

- Standard **motion models**: constant velocity, constant acceleration, coordinated turn, ...
- **Cramér-Rao Lower Bound** (CRLB) provides a bound of the possible performance.

Summary State Estimation (2/3)

- **Kalman filter (KF):** Is the *best linear unbiased estimator* (BLUE) estimator for linear state-space models, and is optimal for linear Gaussian state-space models.

Measurement update:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \varepsilon_k$$
$$P_{k|k} = P_{k|k-1} - K_k H_k P_{k|k-1}.$$

Time update:

$$\hat{x}_{k+1|k} = F_k \hat{x}_{k|k}$$
$$P_{k+1|k} = F_k P_{k|k} F_k^T + G_k Q_k G_k^T$$

- **Extended/Unscented Kalman filter (UKF/EKF):** KF approximations for nonlinear problems.

Summary State Estimation (3/3)

- **Particle filter (PF):** Applies to nonlinear non-Gaussian models.
  Initialization: Generate $x_0^{(i)} \sim p_{x_0}, i = 1, \ldots, N$ particles, and weights $\omega_0^{(i)} = \frac{1}{N}$.
  Iterate for $k = 1, 2, \ldots, t$:
  1. **Measurement update:** For $k = 1, 2, \ldots,$
  $$\omega_k^{(i)} = \omega_{k-1}^{(i)} p(y_k | x_k^{(i)}).$$
  2. **Normalize:** $\omega_k^{(i)} := \omega_k^{(i)} / \sum_j \omega_k^{(j)}$.
  3. **Estimation:** MMSE $\hat{x}_k \approx \sum_{i=1}^N \omega_k^{(i)} x_k^{(i)}$ or MAP.
  4. **Resampling:** Bayesian bootstrap: Take N samples with replacement from the set $\{x_k^{(i)}\}_{i=1}^N$ where the probability to take sample i is $\omega_k^{(i)}$. Let $\omega_k^{(i)} = 1/N$.
  5. **Prediction:** Generate random process noise samples
  $$v_k^{(i)} \sim p_{v_k}, \quad x_{k+1}^{(i)} = f(x_k^{(i)}, v_k^{(i)}).$$

Summary

Presentation Overview

Concluding Remarks: estimation problems

Theoretical aspects

- Powerful methods for parameters and states estimation exist.
- Using models allows for predictable results.
- The linear and Gaussian special case result in analytic solutions.

Concluding Remarks: estimation problems

Theoretical aspects

- Powerful methods for parameters and states estimation exist.
- Using models allows for predictable results.
- The linear and Gaussian special case result in analytic solutions.

Practical aspects

- Try simple things first.
- Balance details and ease of use.
- Real solutions is a craft, with theoretical support.

Resources (1/2)

Sensor Fusion course at Linköping University

- Course material
- Labs (can be done at home)
- Short videos explaining all topic
- Course book: Fredrik Gustafsson. **Statistical Sensor Fusion**, 3 ed. Studentlitteratur, 2018.
- URL: `https://www.control.isy.liu.se/student/tsrt14`

Resources (2/2)

Relevant books

- Simo Särkkä and Lennart Svensson. **Bayesian Filtering and Smoothing**, 2nd ed. Cambridge University Press. 2023.
  URL: `https://users.aalto.fi/~ssarkka/pub/bfs_book_2023_online.pdf`
- Yaakov Bar-Shalom, Peter K. Willett, and Xin Tian. **Tracking and Data Fusion: A Handbook of Algorithms**. YBS Publishing. 2011.
- Robert Popoli and Samuel Blackman. **Design and Analysis of Modern Tracking Systems**. Artech House Publishers. 1999.

Gustaf Hendeby

gustaf.hendeby@liu.se

www.liu.se

LINKÖPING UNIVERSITY