EE5060
Smart Sensors and Instrumentation

EE5111
Industrial Control and Instrumentation

Jiang Rui, Ph.D., BEng
Adjunct Lecturer
Department of Electrical and Computer Engineering

Semester 1, AY2021/2022

Table of Contents

Learning Objectives

In this lecture, you are expected to:

1. Understand the necessity and importance of sensor fusion;
2. Design and implement a sensor fusion framework to solve practical issues in your area;
3. Understand the latest technologies, trends, and applications in smart sensing and instrumentation.

Outline

Why need sensor fusion?

- Sensors may fail

  [Copa Airlines Flight 201] On 6 June 1992, the Boeing 737-204 rolled, entered a steep dive, disintegrated in mid-air, and crashed into the jungle 29 minutes after takeoff, killing all 47 people on board. Investigation shows that one of the cause is the instrument malfunction of the ADI (Attitude Director Indicator).

Why need sensor fusion?

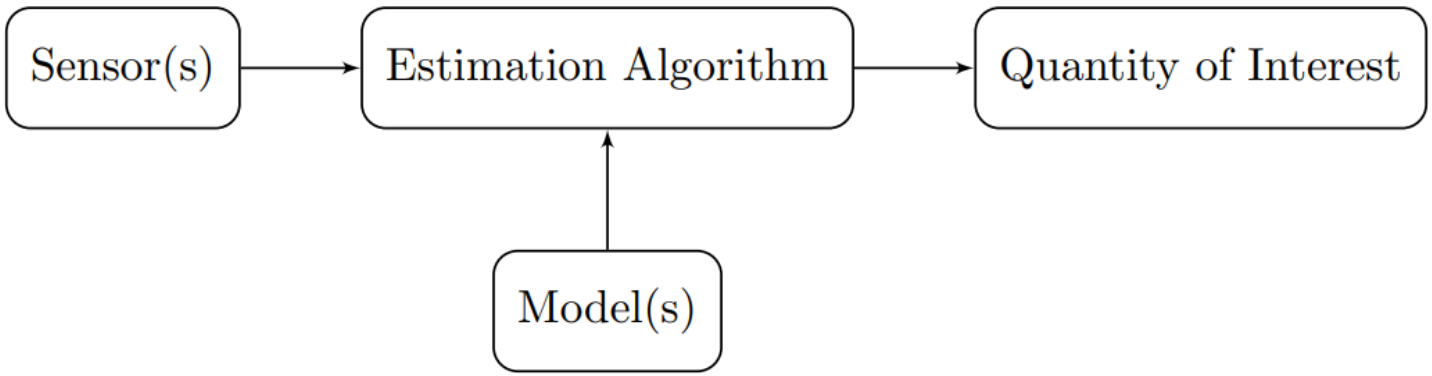
- Sensors have noises

The more, the better?

What is sensor fusion?

Sensor fusion is the process of combining sensory data or data derived from disparate sources such that the resulting information has less uncertainty than would be possible when these sources were used individually.
Other definition also possible!

- Three main components:
  1. one or more **sensor(s)** that measure an observable quantity,
  2. **model(s)** that relate the observed quantity to the quantity of interest,
  3. an **estimation algorithm**s that estimate the quantity of interest.



Sensors

- Example of common sensors

Sensor	Measurement	Application Examples
Accelerometer	Gravity, acceleration	Inertial navigation, activity tracking, screen rotation
Gyroscope	Rotational velocity	Inertial navigation, activity tracking
Magnetometer	Magnetic field strength	Inertial navigation, digital compass, object tracking
Radar	Range, bearing, speed	Target tracking, autonomous vehicles
LIDAR	Range, bearing, speed	Target tracking, autonomous vehicles, robotics
Ultrasound	Range	Robotics
Camera	Visual scene	Security systems, autonomous vehicles, robotics
Strain gauge	Strain	Condition monitoring, scales

Models

- A basic **vector** model for a single measurement is

$$\boldsymbol{y}_n = g_n(\boldsymbol{\theta}) + \boldsymbol{r}_n$$

where

n: the index of measurement (usually discrete time)
$\boldsymbol{y}_n$: the measured quantity
$g_n(\boldsymbol{\theta})$: the ideal (noise-free) function $\boldsymbol{\theta}$ that relates the quantity of interest to the measurement $\boldsymbol{y}_n$
$\boldsymbol{r}_n$: the measurement noise

- Above is called a **sensor model, measurement model**, or **observation model**.

Models

- What if you have multiple measurements?
  The vector model still works!

$$\boldsymbol{y} = g(\boldsymbol{\theta}) + \boldsymbol{r}$$

where

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{y}_N \end{bmatrix}, g(\boldsymbol{\theta}) = \begin{bmatrix} g_1(\boldsymbol{\theta}) \\ g_2(\boldsymbol{\theta}) \\ \vdots \\ g_N(\boldsymbol{\theta}) \end{bmatrix}, \boldsymbol{r} = \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \vdots \\ \boldsymbol{r}_N \end{bmatrix}$$

Models

- Gaussian measurement noise **assumption**:

  The measurement noise is zero-mean Gaussian noise

  $$\boldsymbol{r} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R})$$

  where the pdf $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for an M-dimensional random variable $\boldsymbol{z}$ is defined as

  $$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{M/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{\mu})^{\top}\boldsymbol{\Sigma}^{-1}(\boldsymbol{z}-\boldsymbol{\mu})\right).$$

  Does the assumption hold in practice? Why need such assumption?

Estimation Algorithm

- Given a cost function J, the estimation of interested quantity is to find

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta})$$

How to design the cost function J?

- Least Squares
  Define the error for each measurement $\boldsymbol{e}_n = \boldsymbol{y}_n - g_n(\boldsymbol{\theta})$,
  and the cost function as

$$\begin{aligned} J_{\mathsf{LS}}(\boldsymbol{\theta}) &= \sum_{n=1}^{N} \boldsymbol{e}_n^\top \boldsymbol{e}_n \\ &= \sum_{n=1}^{N} \left(\boldsymbol{y}_n - g_n(\boldsymbol{\theta})\right)^\top \left(\boldsymbol{y}_n - g_n(\boldsymbol{\theta})\right) \end{aligned}$$

Can you represent the J_{LS} without the $\sum$ notation?

Estimation Algorithm

- Weighted Least Squares
  Can we assume that all measurements are equally reliable?

$$J_{\text{WLS}}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \boldsymbol{e}_n^{\top} \boldsymbol{W}_n \boldsymbol{e}_n$$

where $\boldsymbol{W}_n$ is the diagonal weight matrix.

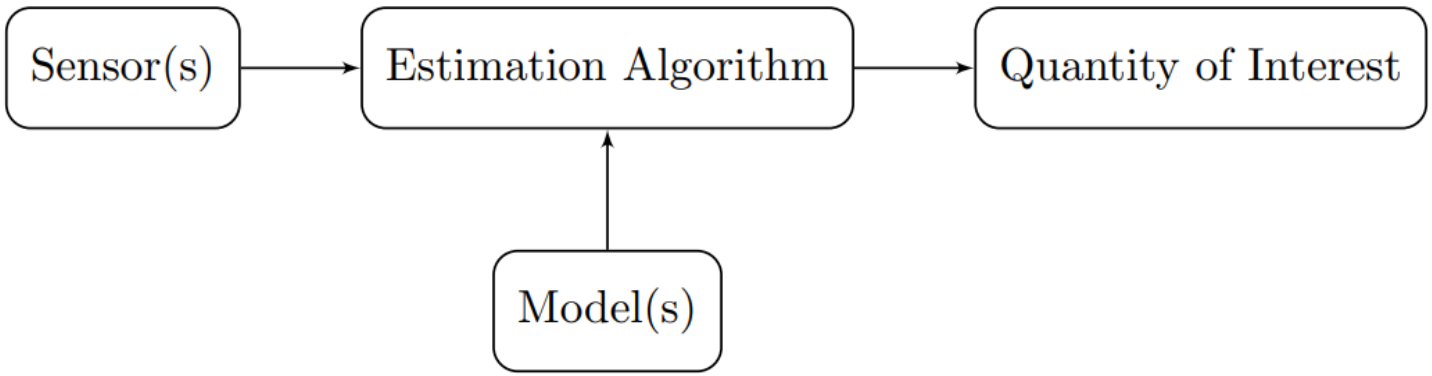
How to choose the weights?
One possible selection is to use the inverse of the measurement noise covariance matrix:

$$\boldsymbol{W}_n = \boldsymbol{R}_n^{-1}$$

What is the rationality behind above choice?

Introduction - Summary

- The reason for sensor fusion:
  To extract more robust and accurate information from sensors.
- The definition of sensor fusion.
- The main components in sensor fusion:



Static Linear Models

- What do "static" and "linear" mean?
  1. The parameter (interested quantity) does not change w.r.t. time.
  2. The measurement equation is linear.

  A single measurement $\boldsymbol{y}_n$ can be represented as

$$\begin{aligned}\boldsymbol{y}_n &= \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1K} \\ c_{21} & c_{22} & \cdots & c_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ c_{d_y 1} & c_{d_y 2} & \cdots & c_{d_y K} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_K \end{bmatrix} + \boldsymbol{r}_n \\ &= \boldsymbol{C}_n \boldsymbol{\theta} + \boldsymbol{r}_n \end{aligned}$$

  For multiple measurements:

$$\boldsymbol{y} = \boldsymbol{G}\boldsymbol{\theta} + \boldsymbol{r}$$

  where $\boldsymbol{G} = \begin{bmatrix} \boldsymbol{C}_1 & \boldsymbol{C}_2 & \cdots & \boldsymbol{C}_N \end{bmatrix}^\top$.

- Why start from here?
  Simple but effective!

- For the above linear model, what are known and unknown variables? How to solve the problem?

  $\boldsymbol{y}$: known measurement vector;
  $\boldsymbol{G}$: known model;
  $\boldsymbol{\theta}$: unknown parameters;
  $\boldsymbol{r}$: unknown noises with known statistical characteristics.

  Under-determined system: No. eqns $<$ No. unknowns

Linear Least Squares

- Idea: minimize the error cost function!

$$J_{\text{LS}}(\boldsymbol{\theta}) = (\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta})^{\top}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta})$$

1. Calculate the gradient of cost function w.r.t. $\boldsymbol{\theta}$.
2. Set the gradient to zero and solve for $\boldsymbol{\theta}$.

$$\begin{aligned}\frac{\partial J_{\text{LS}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= \frac{\partial}{\partial \boldsymbol{\theta}}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta})^{\mathsf{T}}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta}) \\ &= \frac{\partial}{\partial \boldsymbol{\theta}}(\boldsymbol{y}^{\mathsf{T}}\boldsymbol{y} - \boldsymbol{y}^{\mathsf{T}}\boldsymbol{G}\boldsymbol{\theta} - \boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{G}^{\mathsf{T}}\boldsymbol{y} + \boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G}\boldsymbol{\theta} \\ &= -2\boldsymbol{G}^{\mathsf{T}}\boldsymbol{y} + 2\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G}\boldsymbol{\theta}\end{aligned}$$

$$\Rightarrow \hat{\boldsymbol{\theta}}_{\text{LS}} = (\boldsymbol{G}^{\top}\boldsymbol{G})^{-1}\boldsymbol{G}^{\top}\boldsymbol{y}$$

- Statistical properties of the estimator: **mean**

$$\begin{aligned}\mathrm{E}\{\hat{\boldsymbol{\theta}}_{\mathrm{LS}}\} &= \mathrm{E}\{(\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathsf{T}}\boldsymbol{y}\} \\ &= \mathrm{E}\{(\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathsf{T}}(\boldsymbol{G}\boldsymbol{\theta}+\boldsymbol{r})\} \\ &= \mathrm{E}\{(\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G}\boldsymbol{\theta}+(\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathsf{T}}\boldsymbol{r}\} \\ &= \boldsymbol{\theta}+(\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathsf{T}}\,\mathrm{E}\{\boldsymbol{r}\},\end{aligned}$$

  If and only if $\mathrm{E}\{\boldsymbol{r}\}=0$, we have $\mathrm{E}\{\hat{\theta}_{\mathsf{LS}}\}=\boldsymbol{\theta}$.

- Statistical properties of the estimator: **variance**

$$\mathrm{Cov}\{\hat{\boldsymbol{\theta}}_{\mathrm{LS}}\} = (\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G})^{-1}\boldsymbol{G}^{\mathsf{T}}\boldsymbol{R}\boldsymbol{G}((\boldsymbol{G}^{\mathsf{T}}\boldsymbol{G})^{-1})^{\mathsf{T}}.$$

Could you derive the solution for weighted linear least squares?

Sequential Linear Least Squares

- Consider the scenario where **sensor data arrives sequentially**:

  At time $n-1$ Computed $\hat{\boldsymbol{\theta}}_{n-1}$ with covariance for data
  $$\boldsymbol{y} = \boldsymbol{y}_{1:n-1} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_{n-1}\};$$
  At time n New measurement $\boldsymbol{y}_n$ arrives.

  What is the least squares solution? Any iterative solution available?

  Just write the cost function as

  $$J_{\mathrm{SLS}}(\boldsymbol{\theta}) = (\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta})^{\mathsf{T}} \boldsymbol{R}^{-1} (\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta}) + (\boldsymbol{y}_n - \boldsymbol{C}_n\boldsymbol{\theta})^{\mathsf{T}} \boldsymbol{R}_n^{-1} (\boldsymbol{y}_n - \boldsymbol{C}_n\boldsymbol{\theta})$$

  When new measurement arrives, and follow the same steps of least squares estimation, we have

- Iterative solution:

$$\hat{\boldsymbol{\theta}}_n = \hat{\boldsymbol{\theta}}_{n-1} + \boldsymbol{L}_n(\boldsymbol{y}_n - \boldsymbol{C}_n\hat{\boldsymbol{\theta}}_{n-1})$$

with the gain

$$\boldsymbol{L}_n = \boldsymbol{P}_{n-1}\boldsymbol{C}_n^\top(\boldsymbol{C}_n\boldsymbol{P}_{n-1}\boldsymbol{C}_n^\top + \boldsymbol{R}_n)^{-1}$$

What are the influence on $\boldsymbol{L}_n$ and $\hat{\boldsymbol{\theta}}_n$ from a large or small $\boldsymbol{R}_n$?

- Mean: $\mathrm{E}\{\hat{\boldsymbol{\theta}}_n\} = \mathrm{E}\{\hat{\boldsymbol{\theta}}_{n-1}\} = \boldsymbol{\theta}$.
  Unbiased estimate.

- Covariance: $\mathrm{Cov}\{\hat{\boldsymbol{\theta}}_n\} = \boldsymbol{P}_{n-1} - \boldsymbol{L}_n(\boldsymbol{C}_n\boldsymbol{P}_{n-1}\boldsymbol{C}_n^\top + \boldsymbol{R}_n)\boldsymbol{L}_n^\top$
  The more, the better: adding a measurement will make covariance smaller.

Regularized Linear Least Squares

- Consider another scenario where **we have some prior information about the parameters**:

  For example, $\mathrm{E}\{\boldsymbol{\theta}\} = \boldsymbol{m}$ with covariance $\mathrm{Cov}\{\boldsymbol{\theta}\} = \boldsymbol{P}$.

  How to incorporate above prior information into the estimator?

  Add a regularization term to the cost function:

  $$J_{\mathrm{ReLS}}(\boldsymbol{\theta}) = (\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta})^{\mathsf{T}}\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{\theta}) + (\boldsymbol{\theta} - \boldsymbol{m})^{\mathsf{T}}\boldsymbol{P}^{-1}(\boldsymbol{\theta} - \boldsymbol{m})$$

  and we have

- The regularized linear least squares estimator:

  $$\hat{\boldsymbol{\theta}}_{\mathrm{ReLS}} = (\boldsymbol{G}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{G} + \boldsymbol{P}^{-1})^{-1}(\boldsymbol{G}^{\mathsf{T}}\boldsymbol{R}^{-1}\boldsymbol{y} + \boldsymbol{P}^{-1}\boldsymbol{m})$$

  The weighted average of the data and prior information!

- Another form of the estimator:

$$\hat{\boldsymbol{\theta}}_{\mathrm{ReLS}} = \boldsymbol{m} + \boldsymbol{K}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{m})$$

with the gain

$$\boldsymbol{K} = \boldsymbol{P}\boldsymbol{G}^{\mathsf{T}}(\boldsymbol{G}\boldsymbol{P}\boldsymbol{G}^{\mathsf{T}} + \boldsymbol{R})^{-1}$$

The term $\boldsymbol{y} - \boldsymbol{G}\boldsymbol{m}$ is the error between the actual measurement $\boldsymbol{y}$ and the output predicted from the prior mean $\boldsymbol{G}\boldsymbol{m}$.

"Prediction" + "Correction"

- Mean and covariance can be derived but are omitted here.

Static Linear Models - Summary

- What is a linear model?
- Why study the linear model?
- Linear least squares: solution and statistical characteristics.
- The variants of linear least squares:
  1. Sequential linear least squares: iterative form.
  2. Regularized linear least squares: prediction-correction form.

Static Nonlinear Models

- Linear models: **closed-form estimators**
- The world is nonlinear!
- We may have to develop the estimation algorithms for general models:

$$\boldsymbol{y} = g(\boldsymbol{\theta}) + \boldsymbol{r}$$

and the cost function is:

$$J_{\text{WLS}}(\boldsymbol{\theta}) = (\boldsymbol{y} - g(\boldsymbol{\theta}))^{\top} \boldsymbol{R}^{-1} (\boldsymbol{y} - g(\boldsymbol{\theta}))$$

No analytical expressions for above optimization problem...

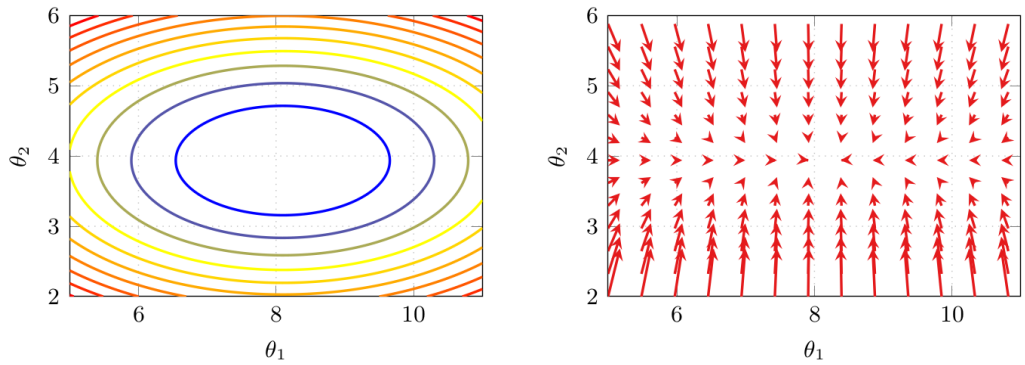
Looking for **Numerical methods.**

Gradient Descent

- Idea: Moving toward the direction of negative gradient to decrease the cost function.
- Update rule:

$$\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i)} - \gamma \nabla_{\theta} J_{\text{WLS}}(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}^{(i)}}$$

  where $\nabla_{\theta} J_{\text{WLS}}(\boldsymbol{\theta})$ denotes the gradient of the cost function w.r.t. $\boldsymbol{\theta}$; $\gamma > 0$ is a constant step length.



- Disadvantage: Gradients is small in areas where the cost function is flat. A large number of iterations required.

Gradient Descent: Exercise

- **Problem**: Derive the gradient of the naive least squares cost function $\nabla_\theta J_{\mathsf{LS}}(\boldsymbol{\theta})$, where $J_{\mathsf{LS}} = \sum_{n=1}^{N}(y_n - g_n(\boldsymbol{\theta}))^2$.
- Hints:

$$\begin{aligned}\nabla_\theta J_{\mathsf{LS}}(\boldsymbol{\theta}) &= \nabla_\theta \sum_{n=1}^{N}(y_n - g_n(\boldsymbol{\theta}))^2 \\ &= \sum_{n=1}^{N} -2(y_n - g_n(\boldsymbol{\theta}))\nabla_\theta g_n(\boldsymbol{\theta})\end{aligned}$$

Rewrite the above sum in vector form.

Gradient Descent: Exercise

- Solution:

$$\nabla_{\boldsymbol{\theta}} J_{\mathrm{LS}}(\boldsymbol{\theta}) = -2 \begin{bmatrix} \nabla g_1(\boldsymbol{\theta}) & \nabla g_2(\boldsymbol{\theta}) & \dots & \nabla g_N(\boldsymbol{\theta}) \end{bmatrix} \left(\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} - \begin{bmatrix} g_1(\boldsymbol{\theta}) \\ g_2(\boldsymbol{\theta}) \\ \vdots \\ g_N(\boldsymbol{\theta}) \end{bmatrix} \right)$$

$$= -2 \begin{bmatrix} \frac{\partial g_1(\boldsymbol{\theta})}{\partial \theta_1} & \frac{\partial g_2(\boldsymbol{\theta})}{\partial \theta_1} & \cdots & \frac{\partial g_N(\boldsymbol{\theta})}{\partial \theta_1} \\ \frac{\partial g_1(\boldsymbol{\theta})}{\partial \theta_2} & \frac{\partial g_2(\boldsymbol{\theta})}{\partial \theta_2} & & \vdots \\ \vdots & & \ddots & \frac{\partial g_N(\boldsymbol{\theta})}{\partial \theta_{K-1}} \\ \frac{\partial g_1(\boldsymbol{\theta})}{\partial \theta_K} & \cdots & \frac{\partial g_{N-1}(\boldsymbol{\theta})}{\partial \theta_K} & \frac{\partial g_N(\boldsymbol{\theta})}{\partial \theta_K} \end{bmatrix} (\boldsymbol{y} - g(\boldsymbol{\theta})),$$

- By defining the Jacobian matrix $\boldsymbol{G}_\theta$, we have the solution

$$\nabla_\theta J_{\mathrm{LS}}(\boldsymbol{\theta}) = -2\boldsymbol{G}_\theta^\top (\boldsymbol{y} - g(\boldsymbol{\theta}))$$

where the Jacobian matrix is evaluated at the $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}^{(i)}$.

Gauss-Newton Algorithm

- Idea: Locally approximate the nonlinear function $g(\boldsymbol{\theta})$ using Taylor series expansion:

$$\begin{aligned} g(\boldsymbol{\theta}) &\approx g(\hat{\boldsymbol{\theta}}^{(i)}) + \nabla_{\boldsymbol{\theta}} g(\boldsymbol{\theta})(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}) \\ &= g(\hat{\boldsymbol{\theta}}^{(i)}) + \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}) \end{aligned}$$

  Then the weighted least squares cost can be approximated by

$$\begin{aligned} J_{\mathrm{WLS}}(\boldsymbol{\theta}) &\approx \left(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}) - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right)^{\mathsf{T}} \boldsymbol{R}^{-1} \left(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}) - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right) \\ &= \left(\boldsymbol{e}^{(i)} - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right)^{\mathsf{T}} \boldsymbol{R}^{-1} \left(\boldsymbol{e}^{(i)} - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right) \end{aligned}$$

  Above approximation is now linear w.r.t. $\boldsymbol{\theta}$ and we already have the closed form solution!

- **Disadvantage**: Not suitable for highly-nonlinear problems.

Levenberg-Marquardt Algorithm

- Idea: Combining the good properties of Gauss-Newton and the gradient descent. Consider the following cost function:

$$\begin{aligned} J_{\mathrm{ReLS}}(\boldsymbol{\theta}) \approx & \left(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}) - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right)^{\mathsf{T}} \boldsymbol{R}^{-1} \left(\boldsymbol{y} - g(\hat{\boldsymbol{\theta}}^{(i)}) - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right) \\ & + \lambda(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})^{\mathsf{T}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}) \\ = & \left(\boldsymbol{e}^{(i)} - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right)^{\mathsf{T}} \boldsymbol{R}^{-1} \left(\boldsymbol{e}^{(i)} - \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})\right) + \lambda(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)})^{\mathsf{T}}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}^{(i)}) \end{aligned}$$

  which contains two parts: **Gauss-Newton term** and **regularization term with damping factor** λ.

  Small λ: Gauss-Newton algorithm;
  Large λ: scaled gradient descent method.

- For adaptation rules for λ, see this paper.

Convergence Criteria

- All above algorithms are iterative.
  When to terminate the search?

  Three mostly-used and simple rules:
  1. The absolute or relative change in the cost falls below a certain threshold,
  2. the absolute or relative change in the parameter estimate falls below a threshold,
  3. a maximum number of iterations is reached.

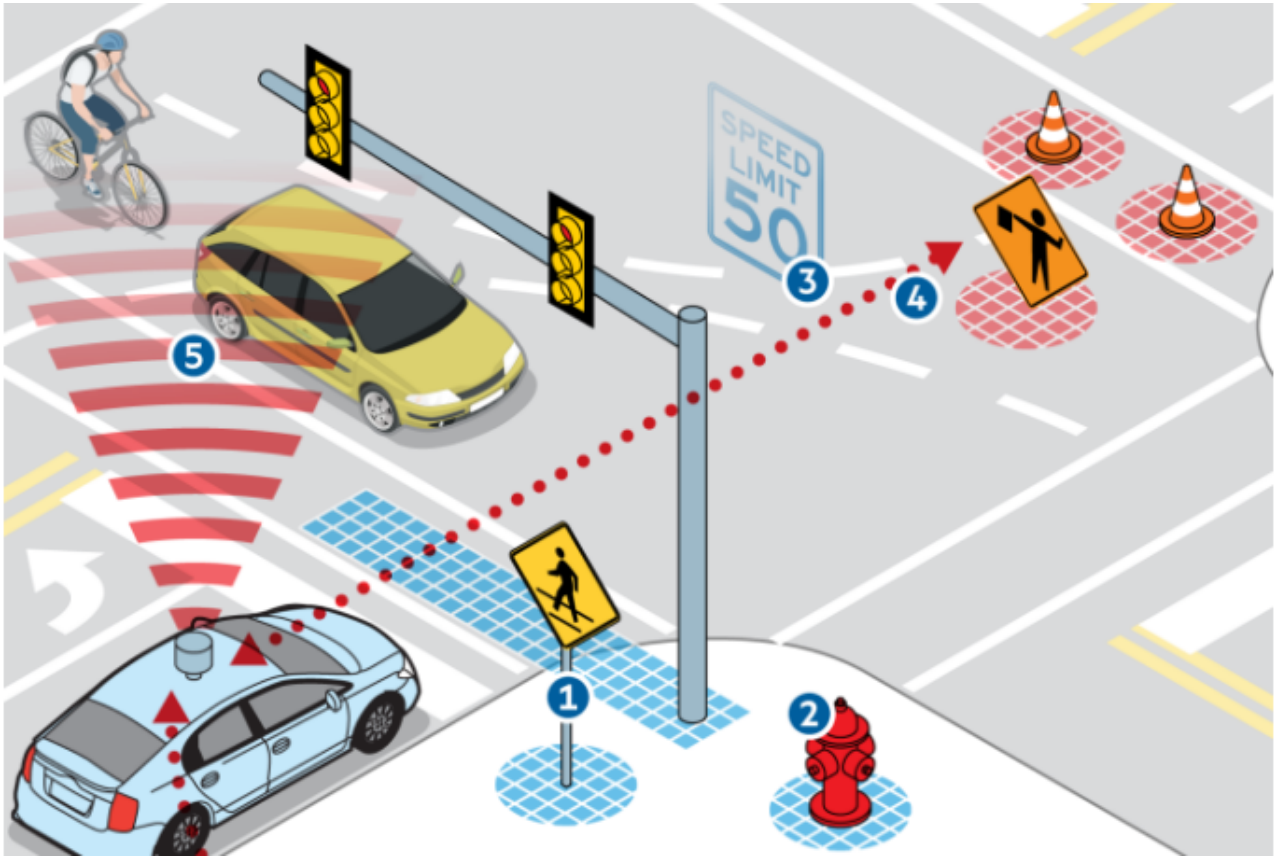
  Of course you can design your own rules!

Static Nonlinear Models - Summary

- The world is nonlinear!
- Difficulties in nonlinear problems: no closed-form solution.
- Numerical optimization methods:
  1. gradient descent,
  2. the Gauss-Newton algorithm, and
  3. the Levenberg-Marquardt algorithm.
- Convergence criteria: when to terminate the iterations.

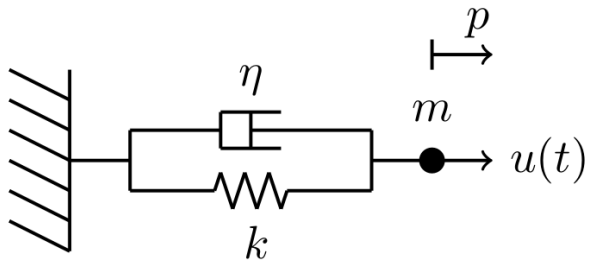
State Space Models

- We have discussed static models.
  What about the parameters are varying w.r.t. time?



Continuous State Space Models

- "State" - dynamically varying parameters.
- Consider a spring-damper system:



From Newton's second law of motion:

$$ma(t) = -kp(t) - \eta v(t) + u(t)$$

Rewrite the equation above, we have:

$$\begin{aligned} v(t) &= v(t) \\ a(t) &= -\frac{k}{m}p(t) - \frac{\eta}{m}v(t) + \frac{1}{m}u(t) \end{aligned}$$

Could you write the equation system in matrix form?

Continuous State Space Models

$$\begin{bmatrix} v(t) \\ a(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{\eta}{m} & -\frac{k}{m} \end{bmatrix} \begin{bmatrix} p(t) \\ v(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t)$$

- Rewrite above equation further gives:

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 \\ -\frac{\eta}{m} & -\frac{k}{m} \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t)$$

  where $\boldsymbol{x}(t) = \begin{bmatrix} p(t) \\ v(t) \end{bmatrix}$ is called the **state (vector)**.

- The general continuous state space model

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t)) + \boldsymbol{B}_w(\boldsymbol{x}(t))\boldsymbol{w}(t),$$
$$\boldsymbol{y}_n = g(\boldsymbol{x}_n) + \boldsymbol{r}_n,$$

  which contains state transition function (**process equation**) with process noise, and **measurement equation** with measurement noise.

Discrete-Time State Space Models

- Not all process can be represented in continuous domain:
  Sampled data, computer implementation, ...
- The general discrete state space model

$$\boldsymbol{x}_n = f(\boldsymbol{x}_{n-1}) + \boldsymbol{B}_q \boldsymbol{q}_n,$$
$$\boldsymbol{y}_n = g(\boldsymbol{x}_n) + \boldsymbol{r}_n,$$

  which also contains state transition function (**process equation**) with process noise, and **measurement equation** with measurement noise.

Continuous state space model encodes differential equations;
Discrete state space model encodes difference equations.

Discrete-Time State Space Models: Exercise

- **Problem**: Consider a truck on frictionless, straight rails. Initially, the truck is stationary at position 0, but it is buffeted by random uncontrolled forces. We measure the position of the truck every Δt seconds, but these measurements are imprecise.

  Derive the discrete state space model of the truck's position and velocity.

State vector? Process equation and measurement equation?
Covariance matrices of the process and measurement noise vectors?

Discrete-Time State Space Models: Exercise

- **Solution**:

  (Process equation) Define $\boldsymbol{x}_n = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$. From Newton's laws of motion we have:

$$\boldsymbol{x}_n = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \boldsymbol{x}_{n-1} + \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} a_n = \boldsymbol{F}\boldsymbol{x}_{n-1} + \boldsymbol{G}a_n = \boldsymbol{F}\boldsymbol{x}_{n-1} + \boldsymbol{w}_n$$

  Suppose $a_n \sim \mathcal{N}(0, \sigma_a)$, then

$$\begin{aligned} \mathrm{Cov}(\boldsymbol{w}_n) = \mathrm{Cov}(\boldsymbol{G}a_n) &= \mathrm{E}[(\boldsymbol{G}a_n)(\boldsymbol{G}a_n)^\top] \\ &= \boldsymbol{G}\mathrm{E}[a_n^2]\boldsymbol{G}^\top \\ &= \sigma_a^2 \boldsymbol{G}\boldsymbol{G}^\top \end{aligned}$$

Discrete-Time State Space Models: Exercise

- **Solution**:
  (Measurement equation) As we are measuring the position only,

$$z_n = \boldsymbol{H}\boldsymbol{x}_n + v_n = \begin{bmatrix} 1 & 0 \end{bmatrix} \boldsymbol{x}_n + v_n$$

  where v_n is assumed to be zero-mean Gaussian: $v_n \sim \mathcal{N}(0, \sigma_z^2)$.

State Space Models - Summary

- Why need state space?
  To represent dynamic process.
- Continuous/discrete state space models could be built according to the physical process itself:
  1. Continuous model: differential equation
  2. Discrete model: difference equation
- Discrete models are more common in practice because of easier implementation.
- By combining **process** and **measurement** equations, we have a state space model.

Filtering

- We just defined the dynamic model, but we do not have an estimator yet! Previous methods only deal with static models.
- Filtering approach: Combine
  1. the prior information from the dynamic model, and
  2. the measurements at different points in time.
- The discrete-time state-space model:

$$\begin{aligned} \boldsymbol{x}_n &= f(\boldsymbol{x}_{n-1}) + \boldsymbol{q}_n \\ \boldsymbol{y}_n &= g(\boldsymbol{x}_n) + \boldsymbol{r}_n \end{aligned}$$

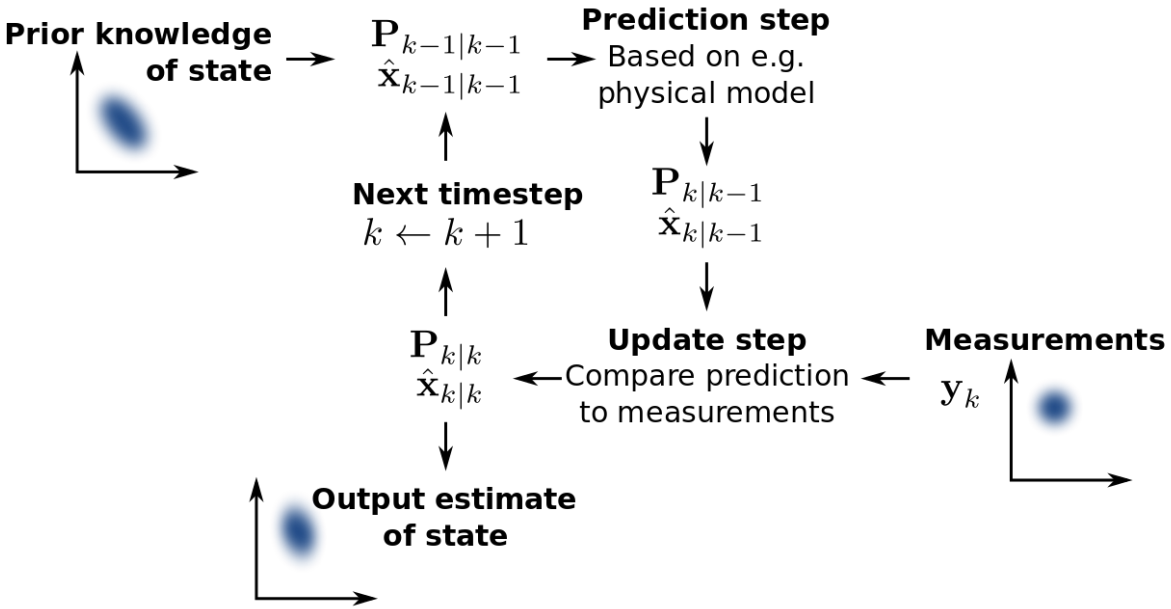
Noises zero-mean with covariance $\mathrm{Cov}\{\mathrm{q_n}\} = \boldsymbol{Q}_n$ and $\mathrm{Cov}\{\mathrm{r_n}\} = \boldsymbol{R}_n$

Model inherently discrete, or from discretizing a continuous-time model

- Initial conditions:
  Assume the probabilistic distribution of initial state $\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0)$.
  Particularly,

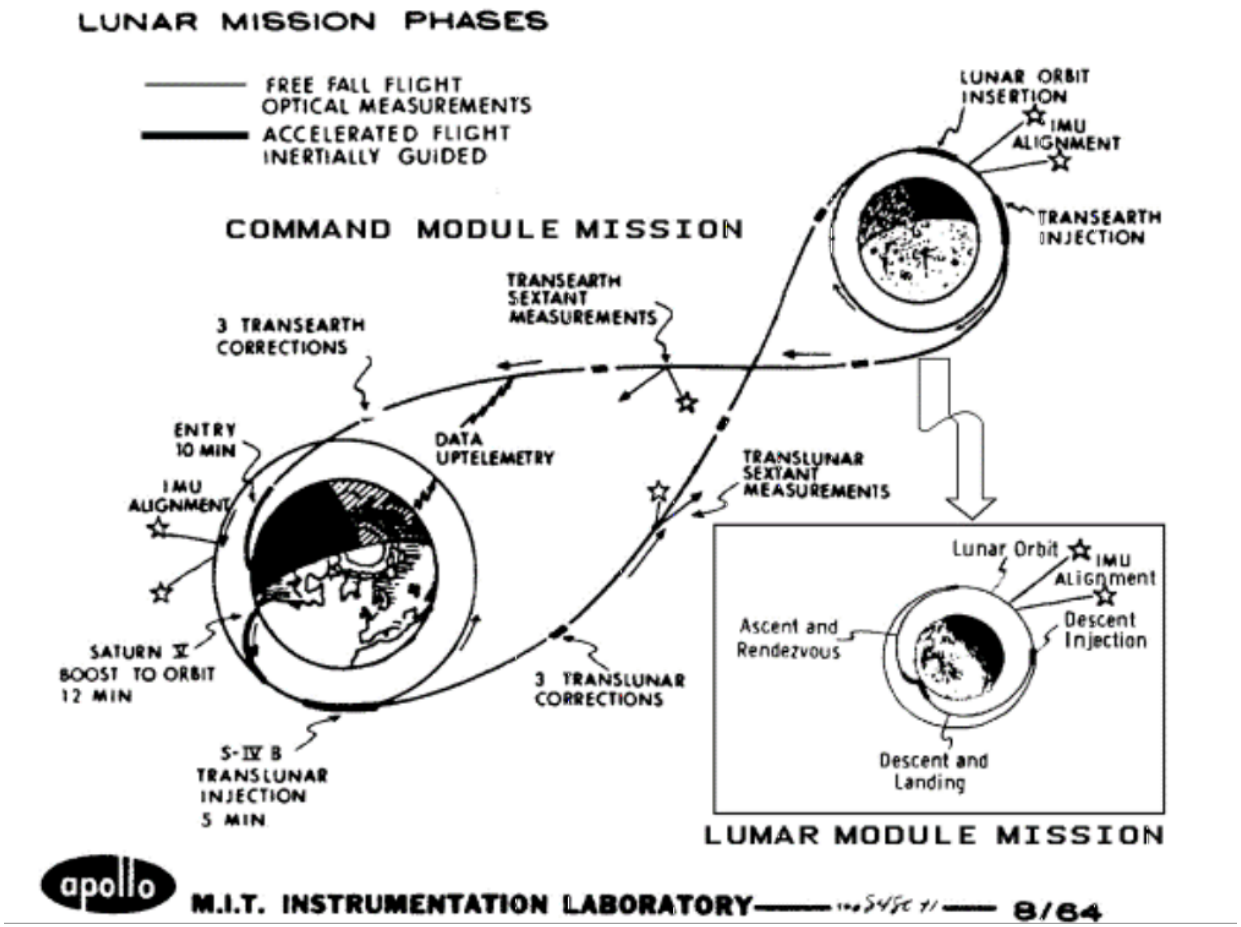
$$\begin{aligned} \mathrm{E}\{\boldsymbol{x}_0\} &= \boldsymbol{m}_0 \\ \mathrm{Cov}\{\boldsymbol{x}_0\} &= \boldsymbol{P}_0 \end{aligned}$$

- Filtering iteration: "**Prediction - Update/Correction**"



Kalman Filter

- Professor Dr. Rudolf Kalman (1930-2016)



- Let us consider linear discrete dynamic model:

$$\begin{aligned} \boldsymbol{x}_n &= \boldsymbol{F}_n \boldsymbol{x}_{n-1} + \boldsymbol{q}_n \\ \boldsymbol{y}_n &= \boldsymbol{G}_n \boldsymbol{x}_n + \boldsymbol{r}_n \end{aligned}$$

with

$$\begin{aligned} &\mathrm{E}\{\boldsymbol{x}_0\} = \boldsymbol{m}_0, \ \mathrm{Cov}\{\boldsymbol{x}_0\} = \boldsymbol{P}_0, \\ &\mathrm{E}\{\boldsymbol{q}_n\} = 0, \ \mathrm{Cov}\{\boldsymbol{q}_n\} = \boldsymbol{Q}_n, \\ &\mathrm{E}\{\boldsymbol{r}_n\} = 0, \ \mathrm{Cov}\{\boldsymbol{r}_n\} = \boldsymbol{R}_n. \end{aligned}$$

How to derive the prediction and update steps?
Regularized linear squares estimator!

- **Prediction**: Utilize the process model!
- **Measurement update**: The cost function is

$$\begin{aligned} J_{\mathrm{ReLS}}(\boldsymbol{x}_n) = &(\boldsymbol{y}_n - \boldsymbol{G}_n \boldsymbol{x}_n)^{\mathsf{T}} \boldsymbol{R}_n^{-1} (\boldsymbol{y}_n - \boldsymbol{G}_n \boldsymbol{x}_n) \\ &+ (\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1})^{\mathsf{T}} \boldsymbol{P}_{n|n-1}^{-1} (\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1}) \end{aligned}$$

We already have the solution!

- Kalman Filter: Algorithm[1]

1: Initialize $\hat{\boldsymbol{x}}_{0|0} = \boldsymbol{m}_0$, $\boldsymbol{P}_{0|0} = \boldsymbol{P}_0$
2: **for** $n = 1, 2, \ldots$ **do**
3: Prediction (time update):

$$\hat{\boldsymbol{x}}_{n|n-1} = \boldsymbol{F}_n \hat{\boldsymbol{x}}_{n-1|n-1}$$
$$\boldsymbol{P}_{n|n-1} = \boldsymbol{F}_n \boldsymbol{P}_{n-1|n-1} \boldsymbol{F}_n^{\mathsf{T}} + \boldsymbol{Q}_n$$

4: Measurement update:

$$\boldsymbol{K}_n = \boldsymbol{P}_{n|n-1} \boldsymbol{G}_n^{\mathsf{T}} (\boldsymbol{G}_n \boldsymbol{P}_{n|n-1} \boldsymbol{G}_n + \boldsymbol{R}_n)^{-1}$$
$$\hat{\boldsymbol{x}}_{n|n} = \hat{\boldsymbol{x}}_{n|n-1} + \boldsymbol{K}_n (\boldsymbol{y}_n - \boldsymbol{G}_n \hat{\boldsymbol{x}}_{n|n-1})$$
$$\boldsymbol{P}_{n|n} = \boldsymbol{P}_{n|n-1} - \boldsymbol{K}_n (\boldsymbol{G}_n \boldsymbol{P}_{n|n-1} \boldsymbol{G}_n + \boldsymbol{R}_n) \boldsymbol{K}_n^{\mathsf{T}}$$

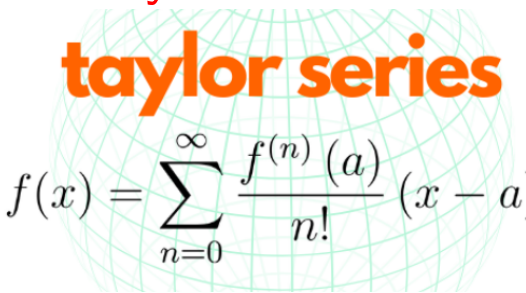
5: **end for**

- Discussions:
  1. Covariance increases in prediction while decreases in correction;
  2. Measurement update is in iterative form:
     prediction + corrected value ($\boldsymbol{y}_n - \boldsymbol{G}_n \hat{\boldsymbol{x}}_{n|n-1}$, called **innovation**) weighted by the Kalman gain.
  3. Measurement covariance $\boldsymbol{R}_n$ is large $\rightarrow$ small Kalman gain.

[1] Typos in the algorithm: the matrix $\boldsymbol{G}_n$ after $\boldsymbol{P}_{n|n-1}$ in line 4 should be transposed.

Extended Kalman Filter (EKF)

- How to deal with non-linear dynamic model?



- Linearization using **Taylor series approximation**
  The nonlinear dynamic model can be re-written as:

$$\begin{aligned}
\boldsymbol{x}_n &= f(\boldsymbol{x}_{n-1}) + \boldsymbol{q}_n \\
&\approx f(\hat{\boldsymbol{x}}_{n-1|n-1}) + \boldsymbol{F}_x(\boldsymbol{x}_{n-1} - \hat{\boldsymbol{x}}_{n-1|n-1}) + \boldsymbol{q}_n \\
\boldsymbol{y}_n &= g(\boldsymbol{x}_n) + \boldsymbol{r}_n \\
&\approx g(\hat{\boldsymbol{x}}_{n|n-1}) + \boldsymbol{G}_x(\boldsymbol{x}_n - \hat{\boldsymbol{x}}_{n|n-1}) + \boldsymbol{r}_n
\end{aligned}$$

  where $\boldsymbol{F}_x$ and $\boldsymbol{G}_x$ are Jacobian matrices of process and measurement equations. You can also use second/higher-order approximation to increase the accuracy (may have other pros and cons)!

- With above modeling, we could derive EKF estimator following the similar steps of Kalman filter.
- Extened Kalman Filter: Algorithm

1: Initialize $\hat{\boldsymbol{x}}_{0|0} = \boldsymbol{m}_0$, $\boldsymbol{P}_{0|0} = \boldsymbol{P}_0$
2: **for** $n = 1, 2, \ldots$ **do**
3: Prediction (time update):

$$\hat{\boldsymbol{x}}_{n|n-1} = f(\hat{\boldsymbol{x}}_{n-1|n-1})$$
$$\boldsymbol{P}_{n|n-1} = \boldsymbol{F}_x \boldsymbol{P}_{n-1|n-1} \boldsymbol{F}_x^{\mathsf{T}} + \boldsymbol{Q}_n$$

4: Measurement update:

$$\boldsymbol{K}_n = \boldsymbol{P}_{n|n-1} \boldsymbol{G}_x^{\mathsf{T}} (\boldsymbol{G}_x \boldsymbol{P}_{n|n-1} \boldsymbol{G}_x^{\mathsf{T}} + \boldsymbol{R}_n)^{-1}$$
$$\hat{\boldsymbol{x}}_{n|n} = \hat{\boldsymbol{x}}_{n|n-1} + \boldsymbol{K}_n (\boldsymbol{y}_n - g(\hat{\boldsymbol{x}}_{n|n-1}))$$
$$\boldsymbol{P}_{n|n} = \boldsymbol{P}_{n|n-1} - \boldsymbol{K}_n (\boldsymbol{G}_x \boldsymbol{P}_{n|n-1} \boldsymbol{G}_x^{\mathsf{T}} + \boldsymbol{R}_n) \boldsymbol{K}_n^{\mathsf{T}}$$

5: **end for**

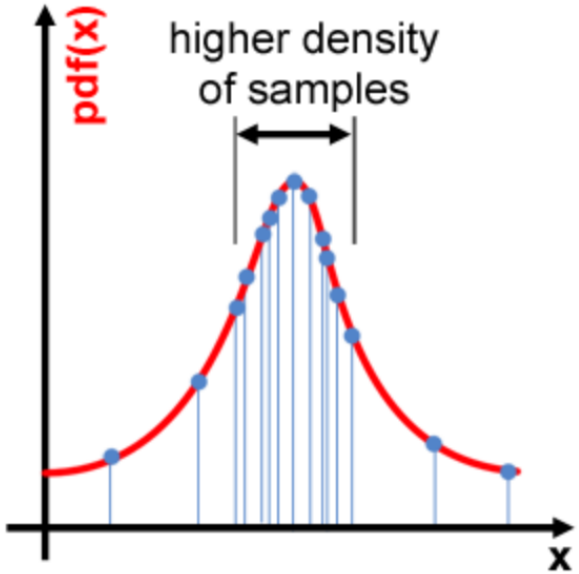
- Other variants:
  Unscented Kalman Filter (UKF),
  Kalman smoother,
  and more...

Particle Filter

- Generalizing the dynamic model
  1. What if the noises are not Gaussian?
     Mean and covariance are not enough to represent the pdf!
  2. What if the model is highly nonlinear?
     First-order approximation is not accurate enough!

EKF cannot deal with such cases, especially case 1.

New ideas for pdf representation? A set of samples.



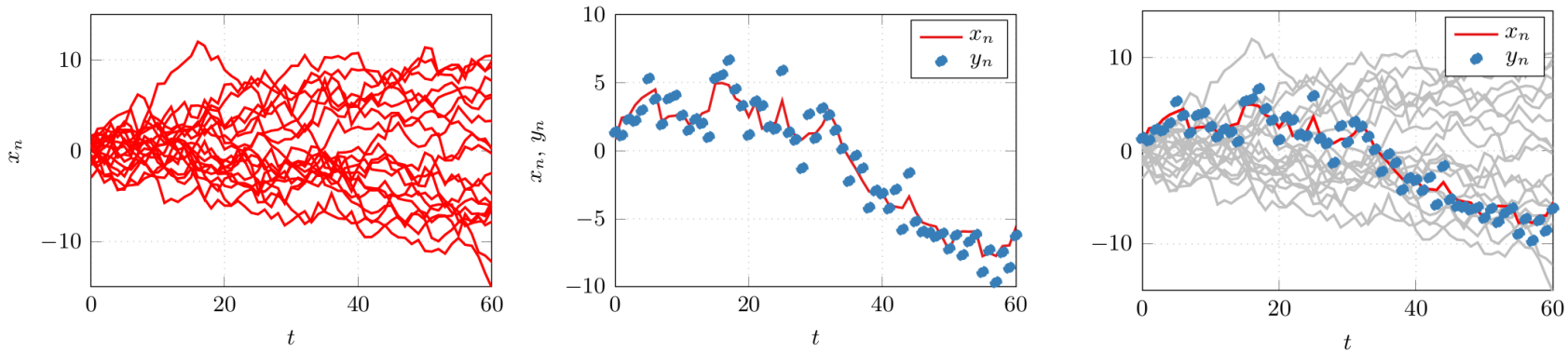
Particle Filter

- General discrete-time state-space model

$$\begin{aligned} \boldsymbol{x}_n &= f(\boldsymbol{x}_{n-1}) + \boldsymbol{q}_n \\ \boldsymbol{y}_n &= g(\boldsymbol{x}_n) + \boldsymbol{r}_n \end{aligned}$$

  with $\boldsymbol{q}_n \sim p(\boldsymbol{q}_n)$ and $\boldsymbol{r}_n \sim p(\boldsymbol{r}_n)$.

- How do process and measurement equations iterate?



$$x_n = x_{n-1} + q_n, \text{ with } x_0 \sim \mathcal{N}(0,1), q_n \sim \mathcal{N}(0,1)$$
$$y_n = x_n + r_n \text{ with } r_n \sim \mathcal{N}(0,1)$$

Particle Filter

- **Idea**: First simulate a set of trajectories and then evaluate how well each of these trajectories explain the measurements that we are observing.
  1. Prediction: Given a set of simulated states $\boldsymbol{x}_{n-1}^j (j = 1, \cdots, J)$, simulate from t_{n-1} to t_n to obtain a set of simulated states $\boldsymbol{x}_n^j (j = 1, \cdots, J)$;
  2. Update: Evaluate how well the simulated states $\boldsymbol{x}_n^j$ explain the observed measurement $\boldsymbol{y}_n$.

- **Prediction**: Use the process equation!
  Sample $\boldsymbol{q}_n^j \sim p(\boldsymbol{q}_n)$ and calculate $\boldsymbol{x}_n^j = f(\boldsymbol{x}_{n-1}^j) + \boldsymbol{q}_n^j$.
  Assign an importance weight w_n^j to each sample s.t. $\sum_{j=1}^J w_n^j = 1$.

  Why need the weight?

  To represent how well each sample explain the measurement.

Particle Filter

- **Update**: How to adjust the weights?

$$\begin{aligned} \text{measurement update: } \tilde{w}_n^j &= p(\boldsymbol{y}_n|\boldsymbol{x}_n^j) \\ \text{normalization: } w_n^j &= \frac{\tilde{w}_n^j}{\sum_{i=1}^J \tilde{w}_n^i} \end{aligned}$$

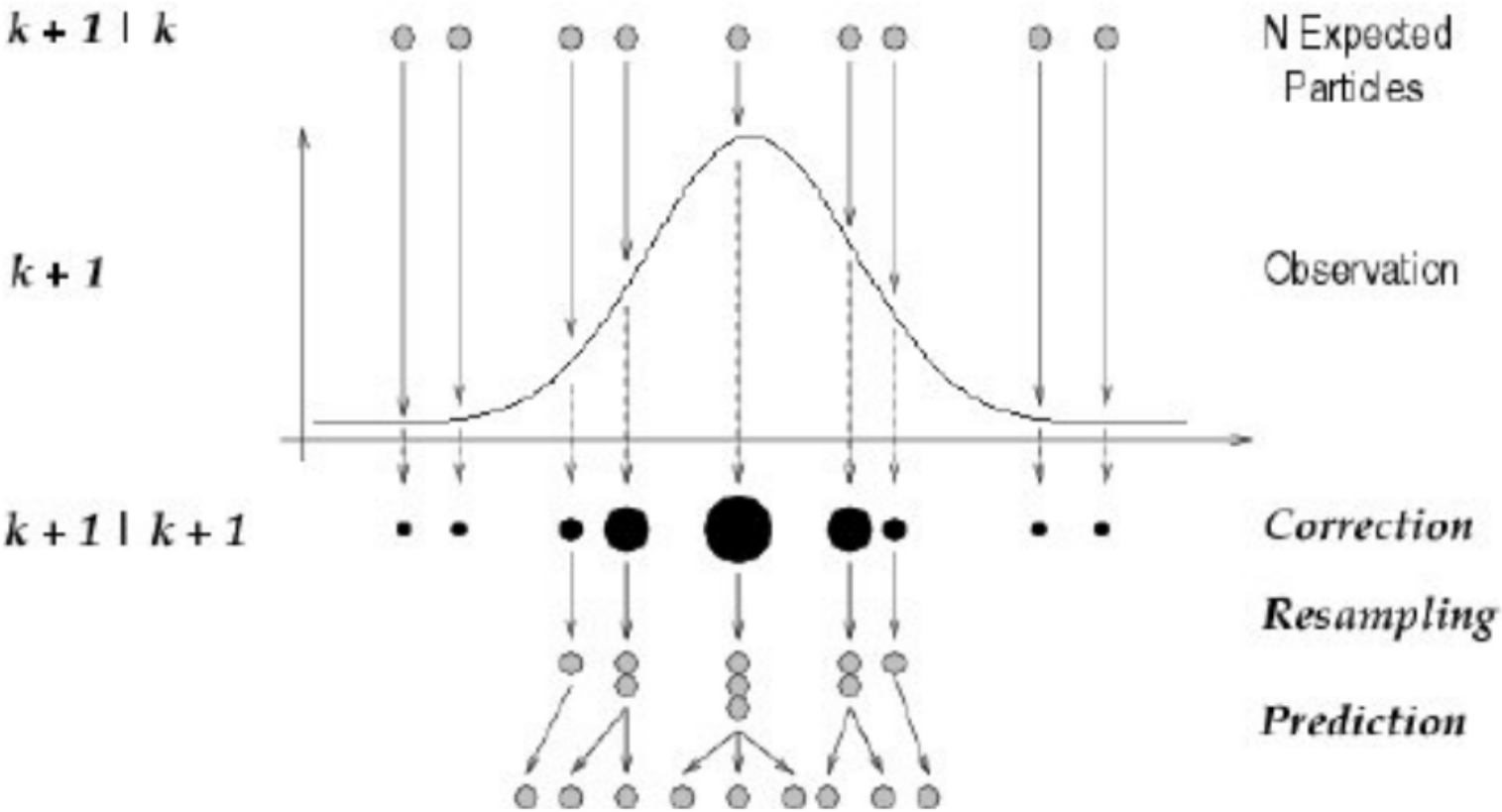
- State estimation and its covariance are represented with **particles**:

$$\hat{\boldsymbol{x}}_{n|n} = \sum_{j=1}^{J} w_n^j \boldsymbol{x}_n^j,$$

$$\boldsymbol{P}_{n|n} = \sum_{j=1}^{J} w_n^j (\boldsymbol{x}_n^j - \hat{\boldsymbol{x}}_{n|n})(\boldsymbol{x}_n^j - \hat{\boldsymbol{x}}_{n|n})^{\mathsf{T}}.$$

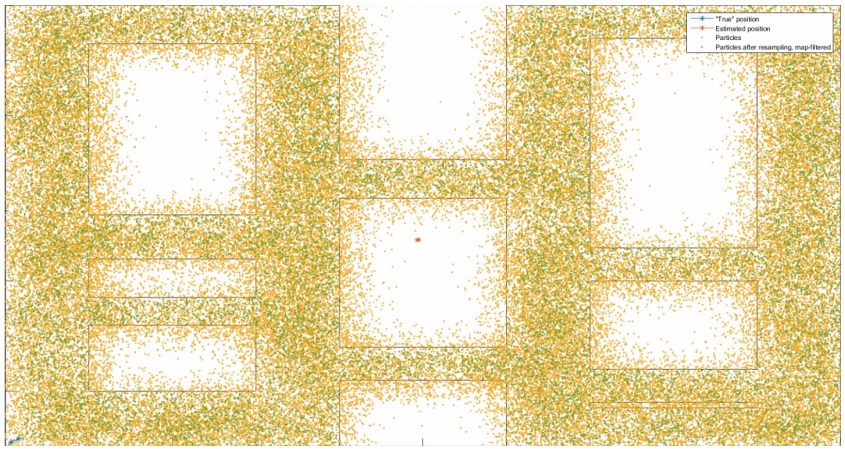
Particle Filter

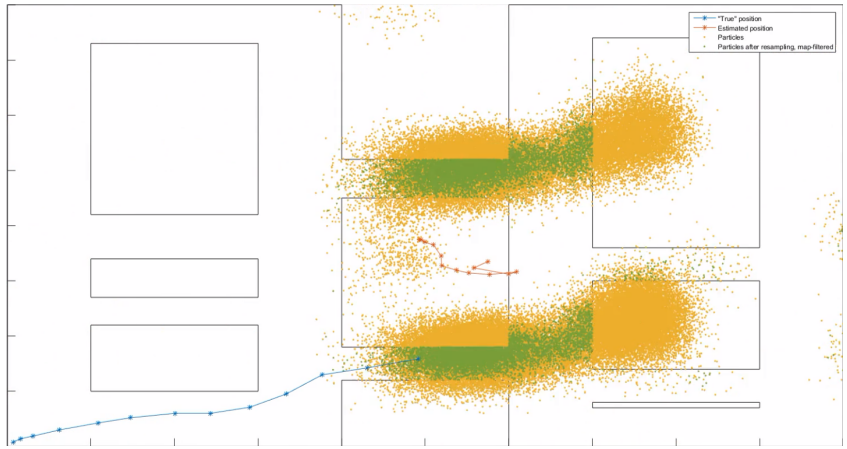
- **Resampling**: Weights concentrates on a small group of particles.
  We want to keep the diversity of particles.

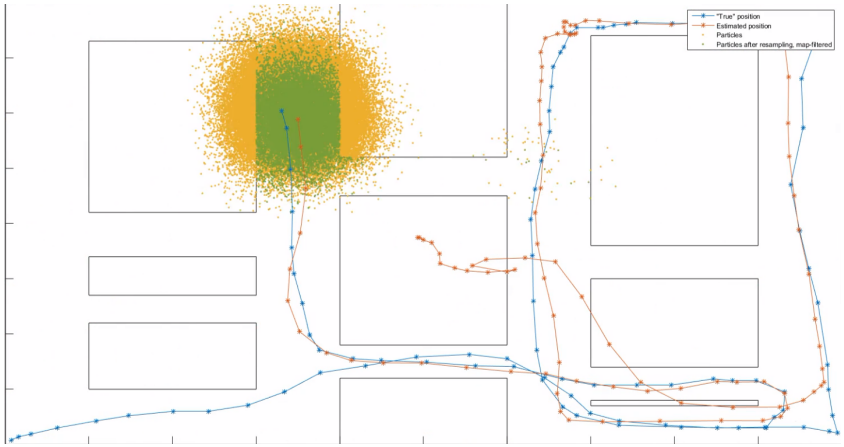


Particle Filter: Application

- Robot Localization - video







Filtering - Summary

- Filtering problem to deal with dynamic state estimation.
- "Prediction - Update" iterative process.
- Several classical filters and their applications:
  1. Kalman filter
  2. Extended Kalman filter (EKF)
  3. Particle filter