AuE 835
Automotive Electronics Integration

PROJECT1.1: SIGNAL PROCESSING REVIEW

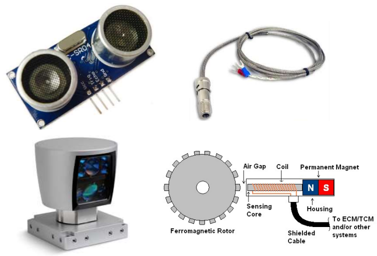
Project Schedule

- Oct 18 – Arduino and programming
- Oct 23 – Ultrasonic sensing, vehicle control & Project 1 Announcement
- Oct 25 – Signal processing review and Project 1 hands-on
- Oct 30 – Control review and Project 1 hands-on
- Nov 1 - Project 1 debugging, Q&A, Test details
- Nov 8 - Project 1 Test

- Nov 13 – Autonomous boat control & Project 2 Announcement
- Nov 15 – Project 2 debugging, Q&A, Test details
- Nov 20 – Project 2 debugging, Q&A, Test details
- Nov 27 – Project 2 Test

- Presentations and report writing

Signal Processing



Automotive sensors

Sensor output

Why do we need signal process?

1. Digital sampling 2. Noise removal 3. Calibration

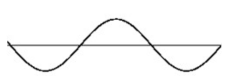
Signal Processing

1. Sampling of Signals
2. Signal Filtering
3. Calibrations

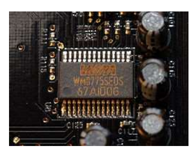
Signal Processing

1. Sampling of Signals
2. Signal Filtering
3. Calibrations

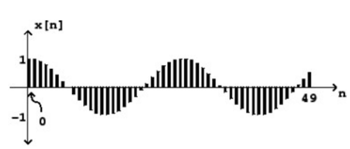
1. Sampling of Signals



Analog-to-digital (A/D) converter

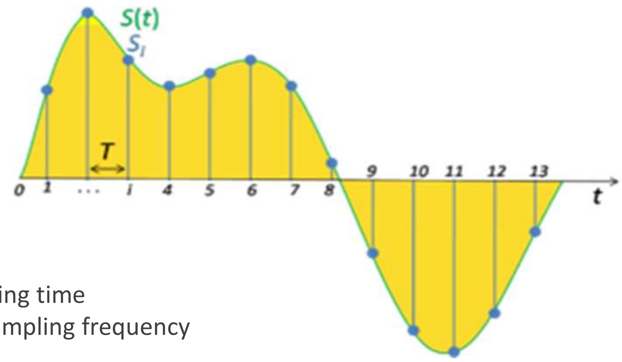


Continuous analog signals → Sampling → Quantization → Discrete digital signals



1.1 Sampling

Sampling is the reduction of a continuous signal to a discrete signal.



T: Sampling time
f=1/T: Sampling frequency

1.1 Sampling

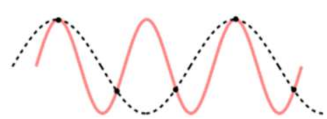
- **Nyquist–Shannon sampling theorem**

Consider a scalar signal $f(t)$ consisting of frequency components in the range $\left(-\omega_s/2, \omega_s/2\right)$. If this signal is sampled at period $T < 2\pi/\omega_s$, then the signal can be perfectly reconstructed from the samples using:

$$\boxed{y(t) = \sum_{k=-\infty}^{\infty} y[k] \frac{\sin\left[\left(\frac{\omega_s}{2}\right)(t-k\Delta)\right]}{\left(\frac{\omega_s}{2}\right)(t-k\Delta)}}$$

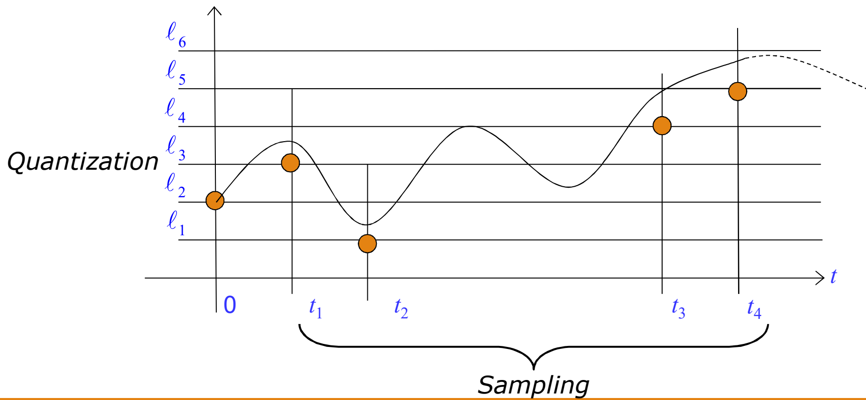
Sampling frequency should be at least two times larger than max signal frequency.

(Normally 2.56 ~ 4 times)



1.2 Quantization

Quantization is the process of mapping a large set of input values to a (countable) smaller set. Rounding and truncation are typical examples of quantization processes.



1.2 Quantization

Noises in sampling and quantization.

Signal Processing

1. Sampling of Signals
2. Signal Filtering
3. Calibrations

2. Signal Filtering

Signal filtering is a process that removes some unwanted components or features (such as noise) from a signal.

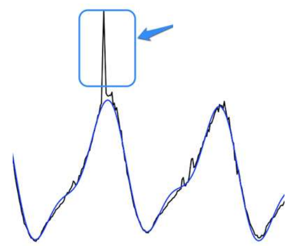
2.1 Simple time-domain filters
2.2 Frequency-domain filters
2.3 Kalman Filter

2.1 Simple time-domain filters

- **Filter by signal value range**

Given a known signal value range [min, max], if a sampled data X(k) is out of the range, then:

- Use previous sampled data
  X(k) = X(k-1)
- Use mean value
  X(k) = (X(k-1)+X(k+1))/2
- Use predicted value
  X(k) = X(k-1)+(X(k-1)-X(k-2))
- Other prediction approaches
  X(k) =F(X(k-1), X(k-2), X(k-3), …..)
- ……………



13

2.2 Frequency-domain filters

- **Frequency domain**

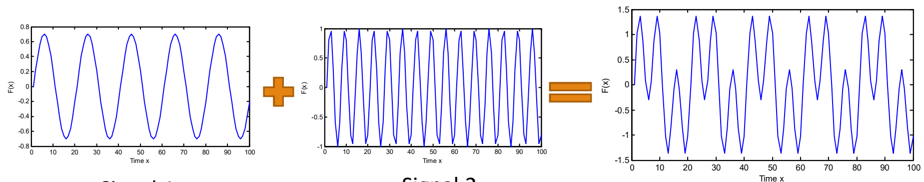
Frequency domain refers to the analysis of mathematical functions or signals with respect to frequency, rather than time.

- Time-domain signal function shows how a signal changes over time:
  f(t) or f (x),   t or x is time
- Frequency-domain signal function shows how much of the signal lies within each given frequency band over a range of frequencies:
  F(u),   u is frequency

Why do we want to analyze signals in frequency domain?

14

2.2 Frequency-domain filters

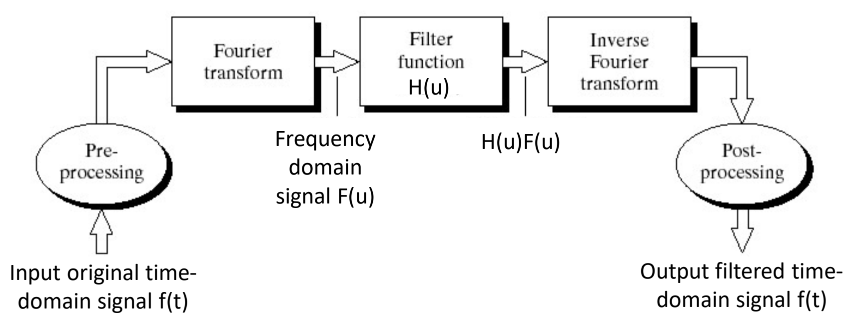


Signal 1 Signal 2 Sampled signal

2.2 Frequency-domain filters

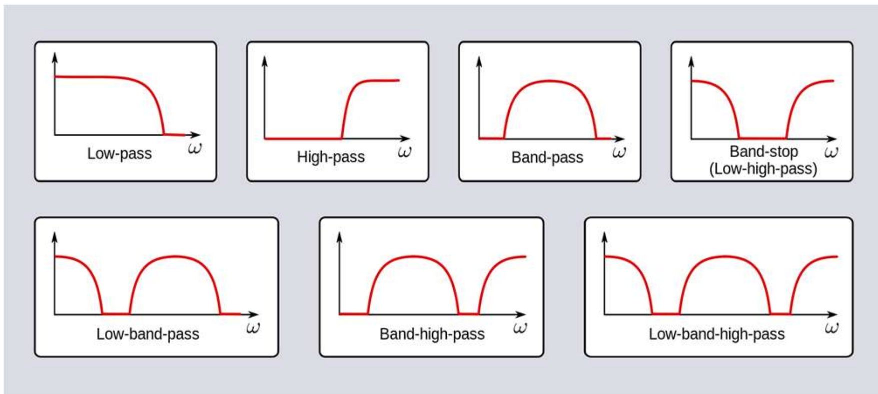
- **Frequency domain filtering process**

Given known signal frequency ranges, transfer signals to frequency domain and filter the signals whose frequencies are out of the known ranges.



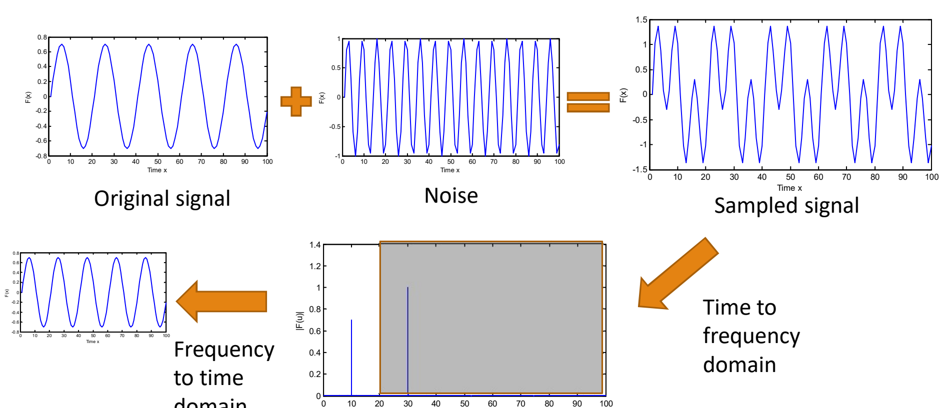
2.2 Frequency-domain filters

- **Frequency-domain filter types**



2.2 Frequency-domain filters

- **Low-pass filtering example**



2.3 Kalman Filter

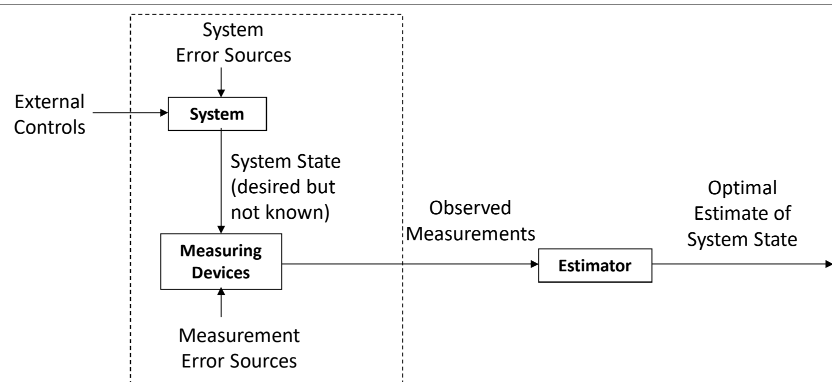
Signal filtering is a process that removes some unwanted components or features (such as noise) from a signal.

2.1 Simple time-domain filters (known value ranges)
2.2 Frequency-domain filters (known frequency ranges)

What if the noises are random, we don't know the value and frequency ranges, or we cannot distinguish the signals and noises by values and ranges?

2.3 Kalman Filter

The Problem



System state cannot be measured directly or precisely

Need to estimate "optimally" from measurements

What is a Kalman Filter?

Recursive data processing algorithm

- Doesn't need to store all previous measurements and reprocess all data each time step

Generates optimal estimate of desired quantities given the set of measurements

Optimal?

- For linear system and white Gaussian errors, Kalman filter is "best" estimate based on all previous measurements
- For non-linear system optimality is 'qualified'

Kalman Filter

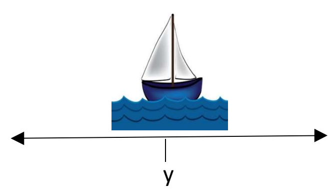
- **Model of Process**

$y_k = Ay_{k-1} + Bu_k + w_k$

$z_k = Hy_k + v_k$

k: discrete time
y: system state to be estimated
u: system input (optional)
z: system measurement
A, B, H: constant matrices

w: process noise with w$\sim N(0, Q)$
v: measurement noise with v$\sim N(0, R)$



Boat is stationary

$y_k = 1^*y_{k-1} + w_k$

$z_k = 1^*y_k + v_k$

Boat is moving with a known speed u?

$y_k = 1^*\ y_{k-1} + T^*u + w_k$

$z_k = 1^*\ y_k + v_k$

Kalman Filter

- **Kalman Filter process**

Prediction Step:
$\hat{y}^-_k$ is predicted based on measurements at previous time-step

Predicted state estimate: $\hat{y}^-_k = Ay_{k-1} + Bu_k$

Predicted error covariance: $P^-_k = AP_{k-1}A^T + Q$

Correction Step:
$\hat{y}_k$ is estimated by correcting $\hat{y}^-_k$ based on new measurement

Kalman gain: $K = P^-_k H^T (HP^-_k H^T + R)^{-1}$

Updated state estimate: $\hat{y}_k = \hat{y}^-_k + K(z_k - H\,\hat{y}^-_k)$

Updated error covariance: $P_k = (I - KH)P^-_k$

Input:
Control signal: u_k
Measurement: z_k

Output: $\hat{y}_k$

Blending Factor

- If we are sure about measurements:
  - Measurement error covariance (R) decreases to zero
  - K increases and weights residual more heavily than prediction
- If we are sure about prediction
  - Prediction error covariance (P^-_k) decreases to zero
  - K decreases and weights prediction more heavily than residual

Kalman Filter

Process model

$y_k = Ay_{k-1} + Bu_k + w_k$

$z_k = Hy_k + v_k$

Prediction (Time Update)

(1) Project the state ahead

$$\hat{y}^-_k = Ay_{k-1} + Bu_k$$

(2) Project the error covariance ahead

$$P^-_k = AP_{k-1}A^T + Q$$

Correction (Measurement Update)

(1) Compute the Kalman Gain

$$K = P^-_k H^T (HP^-_k H^T + R)^{-1}$$

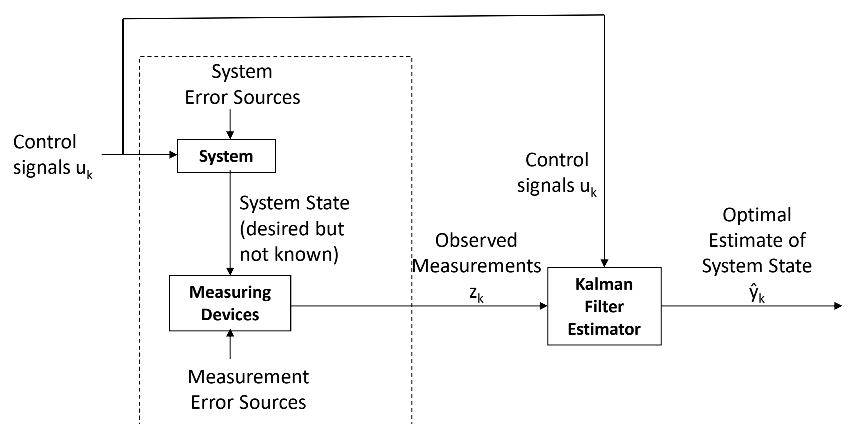
(2) Update estimate with measurement z_k

$$\hat{y}_k = \hat{y}^-_k + K(z_k - H\hat{y}^-_k)$$ OUTPUT

(3) Update Error Covariance

$$P_k = (I - KH)P^-_k$$

Kalman Filter



Quick Example – Constant Model

Kalman Filter

Process model

$y_k = Ay_{k-1} + Bu_k + w_k$

$z_k = Hy_k + v_k$

Prediction

$\hat{y}^-_k = Ay_{k-1} + Bu_k$

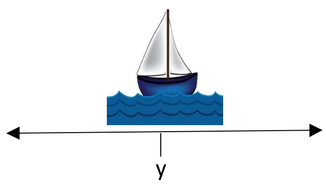
$P^-_k = AP_{k-1}A^T + Q$

Correction

$K = P^-_k H^T (HP^-_k H^T + R)^{-1}$

$\hat{y}_k = \hat{y}^-_k + K(z_k - H\,\hat{y}^-_k)$

$P_k = (I - KH)P^-_k$



Boat is stationary

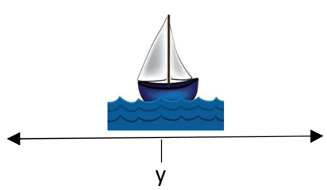
Process model

$y_k = 1{*}y_{k-1} + w_k$

$z_k = 1{*}y_k + v_k$

Design the Kalman filter for estimating y(t)

Quick Example – Constant Model



Boat is stationary

$y_k = 1{*}y_{k-1} + w_k$

$z_k = 1{*}y_k + v_k$

Prediction

$\hat{y}^-_k = y_{k-1}$

$P^-_k = P_{k-1}$

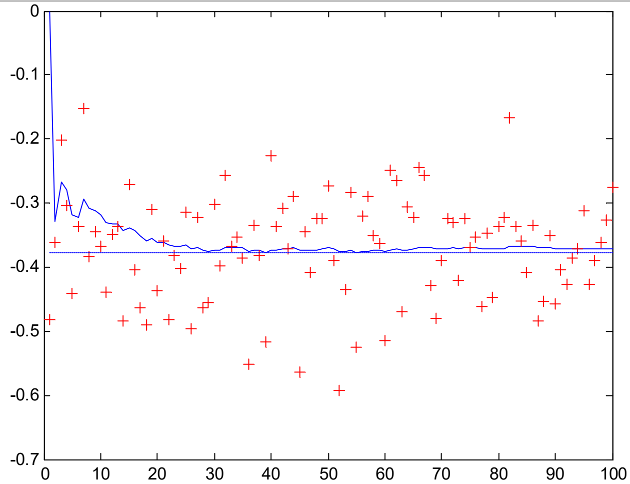
Correction

$K = P^-_k(P^-_k + R)^{-1}$

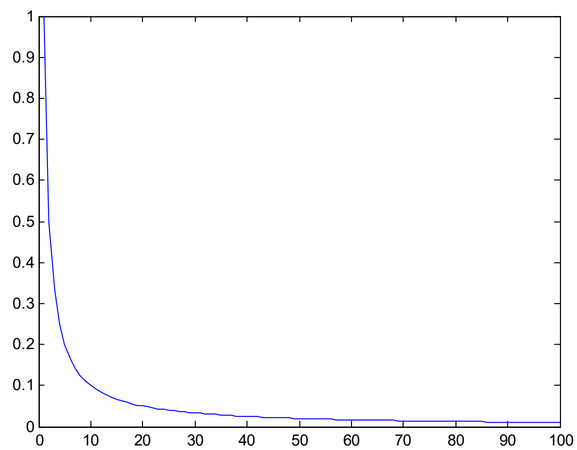
$\hat{y}_k = \hat{y}^-_k + K(z_k - H\,\hat{y}^-_k)$

$P_k = (I - K)P^-_k$

Quick Example – Constant Model

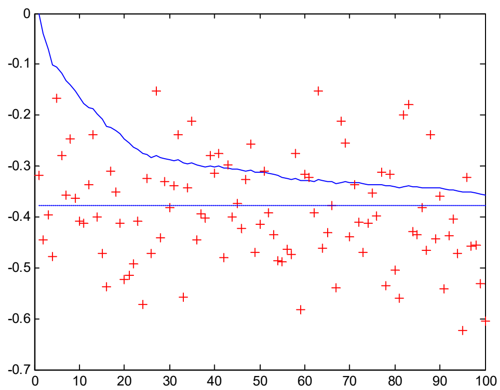


Quick Example – Constant Model



Convergence of Error Covariance - P_k

Quick Example – Constant Model



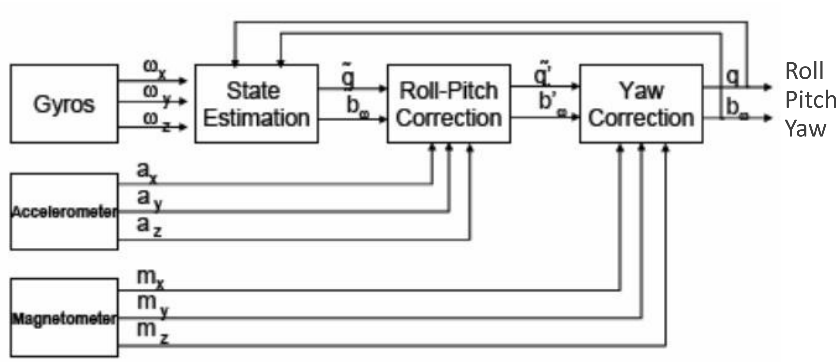
Larger value of R – the measurement error covariance (indicates poorer quality of measurements)

↓

Filter slower to 'believe' measurements – slower convergence

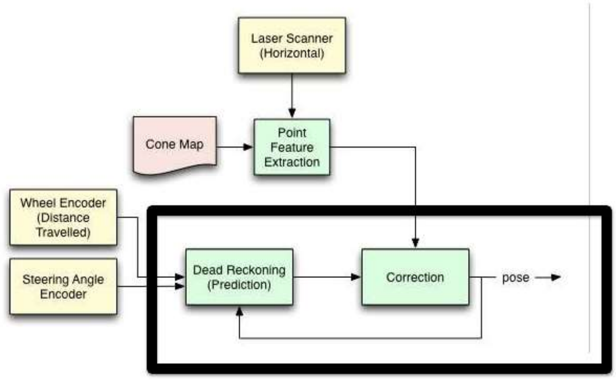
Quick Example – Sensor Fusion

- Sensor Fusion for Inertia Measurement Unit (IMU)



Quick Example – Sensor Fusion

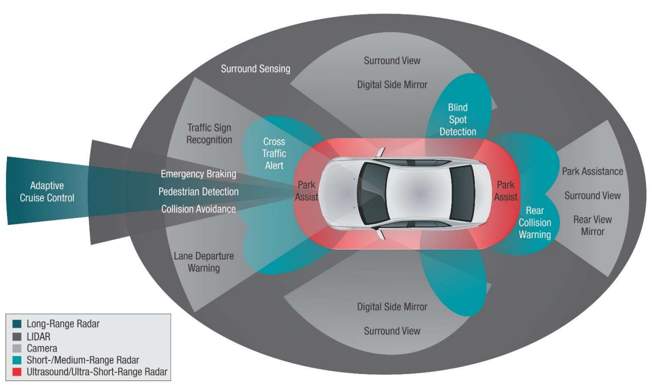
- Sensor Fusion for Vehicle State Estimation (Position, Velocity)



Simultaneous Localization and Mapping (SLAM)

Quick Example – Autonomous Driving

- Single sensor noise removal
- Multi-sensor fusion for noise removal



Kalman Filter

Reading material on Kalman Filter:
"An Introduction to the Kalman Filter"
by Greg Welchand Gary Bishop
University of North Carolina at Chapel Hill

Signal Processing

1. Sampling of Signals
2. Signal Filtering
3. Calibrations

3. Calibration

We have developed methods to measure and filter *sensing signals y*

What are the units of *y*?

- The process of development of a relationship between the measured variable (input) and the physical variable (output) for a specific sensor is known as the ***calibration*** of the sensor.
- Typically, a sensor (or an entire instrument system) is calibrated by providing a known physical variable to the system and recording the corresponding measurement.

38

Calibration

Find a function to map measured variable to physical variable:

Physical Variable = f (Measured Variable)

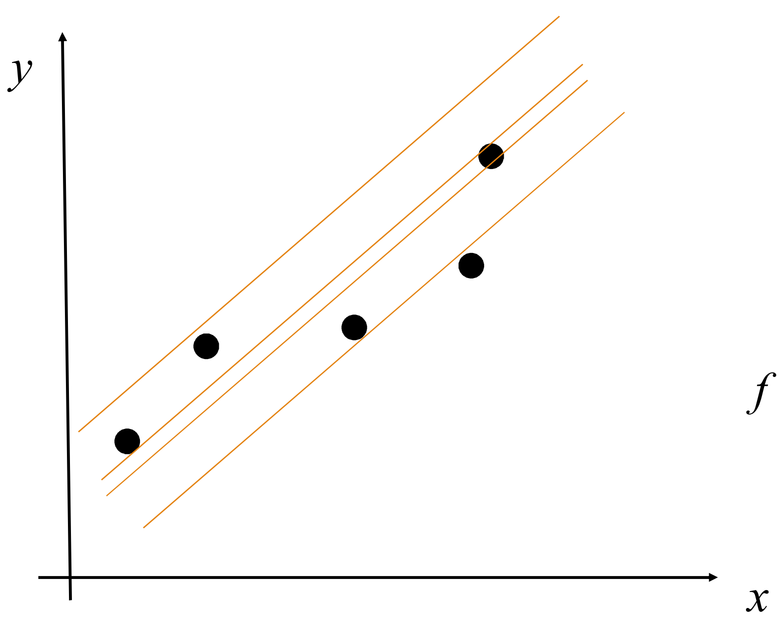
- Design the function *f*: Analytical models of phenomena (e.g. equations from physics). Create an equation with some unknown parameters.
- Determine the unknown parameters: Calculate the unknown parameters based on their measured values and their corresponding physical values.

Linear Fitting Function (linear regression)

Given the general form of a straight line, how can we find the coefficients that best fits the line to the data?

$$f(x) = ax + b$$

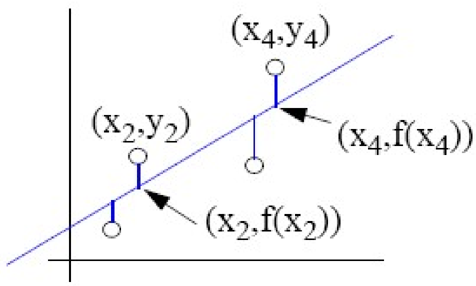
The goal is to identify the coefficients *'a'* and *'b'* such that *f(x)* 'fits' the data well



$$f(x) = ax + b$$

Define error in a curve fit

Assumptions: Positive or negative error have the same value (data point is above or below the line)



- Denote data values as (x, y)
- Name points on the fitted line as (x, f(x)), where $f(x) = ax + b$

The error is calculated at the four data points.

$$Error = \sum d_i^2 = [y_1 - f(x_1)]^2 + [y_2 - f(x_2)]^2 + [y_3 - f(x_3)]^2 + [y_4 - f(x_4)]^2$$

Our fit is a straight line, so now substitute $f(x) = ax + b$

$$Error = \sum_{i=1}^{N} [y_i - f(x_i)]^2 = \sum_{i=1}^{N} [y_i - (ax_i + b)]^2$$

•The 'best' line has **minimum error** between line and data points

$$Minimize \left\{ Error = \sum_{i=1}^{N} [y_i - (ax_i + b)]^2 \right\}$$

•This is called the **least squares approach**, since square of the error is minimized.

$$Minimize\left\{Error = \sum_{i=1}^{N}\left[y_i - (ax_i + b)\right]^2\right\}$$

Take the derivative of the error with respect to a and b, set each to zero

$$\frac{\partial(Error)}{\partial a} = -2\sum_{i=1}^{N} x_i\left[y_i - (ax_i + b)\right] = 0$$

$$\frac{\partial(Error)}{\partial b} = -2\sum_{i=1}^{N}\left[y_i - (ax_i + b)\right] = 0$$

Solve for the *a* and *b* so that the previous two equations both = 0

$$a\sum_{i=1}^{N} x_i^2 + b\sum_{i=1}^{N} x_i = \sum_{i=1}^{N} x_i\, y_i$$

$$a\sum_{i=1}^{N} x_i + bN = \sum_{i=1}^{N} y_i$$

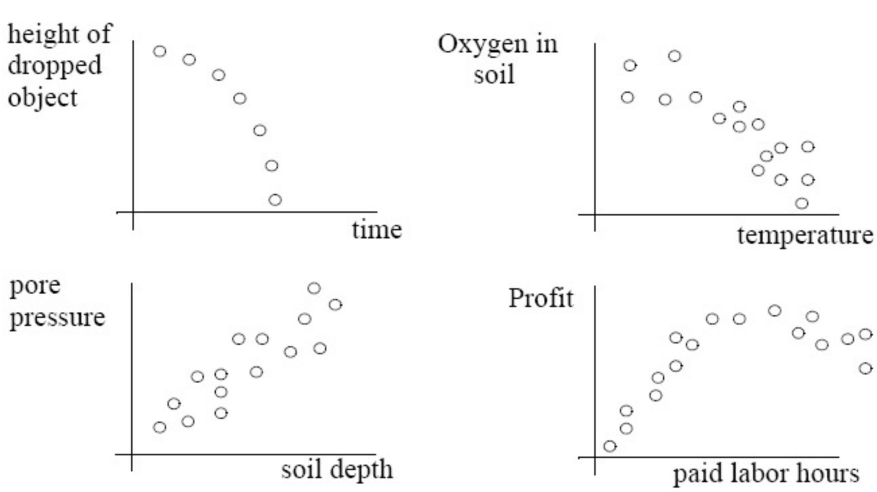
put these into **matrix form**

$$\begin{bmatrix} N & \sum_{i=1}^{N} x_i \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} x_i^2 \end{bmatrix}\begin{bmatrix} b \\ a \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} y_i \\ \sum_{i=1}^{N} x_i y_i \end{bmatrix}$$

$$a = \frac{N\sum_{i=1}^{N} x_i\, y_i - \sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i}{N\sum_{i=1}^{N} x_i^2 - \left[\sum_{i=1}^{N} x_i\right]^2}$$

$$b = \frac{\sum_{i=1}^{N} y_i \sum_{i=1}^{N} x_i^2 - \sum_{i=1}^{N} x_i \sum_{i=1}^{N} x_i\, y_i}{N\sum_{i=1}^{N} x_i^2 - \left[\sum_{i=1}^{N} x_i\right]^2}$$

Is a straight line suitable for all sensors ?



Other Fitting Functions

- Linear curve fitting: $f(x)=ax+b$
- Polynomial curve fitting: $f(x)=a_0+a_1x^1+a_1x+a_2x^2+....+a_mx^m$
- Power Law curve fitting: $f(x)=ax^b$
- Exponential curve fitting: $f(x)=ae^{bx}$
- Other functions or even piece-wise:

  $ln(f(x)) = ln(a)+bln(x)$, if $x<t$

  $ln(f(x))=ln(a)+bx$, if $x>=t$ && $x<T$

  ....

The Least-Squares m^{th} Degree Polynomials

When using an m^{th} degree polynomial

$$y = f(x) = a_0 + a_1 x + a_2 x^2 + ...a_m x^m$$

to approximate the given set of data, $(x_1,y_1), (x_2,y_2)...... (x_n,y_n)$, where $n \geq m$, the best fitting curve has the least square error, i.e.,

$$\Pi = \sum_{i=1}^{n} \{y_i - f(x_i)\}^2 = \min$$

$$\Pi = \sum_{i=1}^{n} \{y_i - [a_0 + a_1 x_i + a_2 x_i^2 + ...a_n x_i^n]\}^2 = \min$$

To obtain the least square error, the unknown coefficients a_0, a_1, …. and a_m must yield zero first derivatives.

$$
\begin{cases}
\dfrac{\partial \Pi}{\partial a_0} = 2\sum_{i=1}^{n}\left[y_i - (a_0 + a_1 x_i + a_2 x_i^{\,2} + \ldots + a_m x^m)\right] = 0 \\
\dfrac{\partial \Pi}{\partial a_1} = 2\sum_{i=1}^{n} x_i \left[y_i - (a_0 + a_1 x_i + a_2 x_i^{\,2} + \ldots + a_m x^m)\right] = 0 \\
\dfrac{\partial \Pi}{\partial a_2} = 2\sum_{i=1}^{n} x_i^{\,2}\left[y_i - (a_0 + a_1 x_i + a_2 x_i^{\,2} + \ldots + a_m x^m)\right] = 0 \\
\vdots \\
\vdots \\
\dfrac{\partial \Pi}{\partial a_m} = 2\sum_{i=1}^{n} x_i^{\,m}\left[y_i - (a_0 + a_1 x_i + a_2 x_i^{\,2} + \ldots + a_m x^m)\right] = 0
\end{cases}
$$

Expanding the previous equations, we have

$$
\begin{cases}
\sum_{i=1}^{n} y_i = a_0 \sum_{i=1}^{n} 1 + a_1 \sum_{i=1}^{n} x_i + a_2 \sum_{i=1}^{n} x_i^{\,2} + \ldots + a_m \sum_{i=1}^{n} x_i^{\,m} \\
\sum_{i=1}^{n} x_i y_i = a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^{\,2} + a_2 \sum_{i=1}^{n} x_i^{\,3} + \ldots + a_m \sum_{i=1}^{n} x_i^{\,m+1} \\
\sum_{i=1}^{n} x_i^{\,2} y_i = a_0 \sum_{i=1}^{n} x_i^{\,2} + a_1 \sum_{i=1}^{n} x_i^{\,3} + a_2 \sum_{i=1}^{n} x_i^{\,4} + \ldots + a_m \sum_{i=1}^{n} x_i^{\,m+2} \\
\vdots \\
\vdots \\
\sum_{i=1}^{n} x_i^{\,m} y_i = a_0 \sum_{i=1}^{n} x_i^{\,m} + a_1 \sum_{i=1}^{n} x_i^{\,m+1} + a_2 \sum_{i=1}^{n} x_i^{\,m+2} + \ldots + a_m \sum_{i=1}^{n} x_i^{\,2m}
\end{cases}
$$

The unknown coefficients can hence be obtained by solving the above linear equations.

AX=B

$$A = \begin{bmatrix} n & \sum x_i & \sum x_i^2 & \dots & \sum x_i^j \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \dots & \sum x_i^{j+1} \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \dots & \sum x_i^{j+2} \\ \vdots & \vdots & \vdots & & \vdots \\ \sum x_i^j & \sum x_i^{j+1} & \sum x_i^{j+2} & \dots & \sum x_i^{j+j} \end{bmatrix}, \quad X = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_j \end{bmatrix}, \quad B = \begin{bmatrix} \sum y_i \\ \sum (x_i y_i) \\ \sum \left(x_i^2 y_i\right) \\ \vdots \\ \sum \left(x_i^j y_i\right) \end{bmatrix}$$

No matter what the order j, we always get equations **LINEAR** with respect to the coefficients. We can use the following solution method:

If A is square and inversable, ***$X=A^{-1}B$***

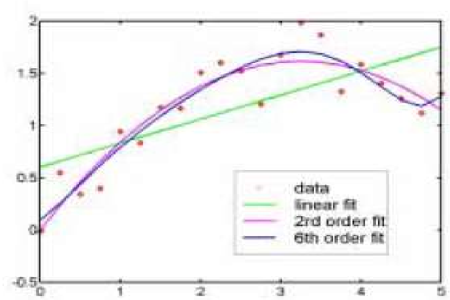
Otherwise, use pseudo-inverse of A : ***$X=A^+B$, $A^+=A^T(AA^T)^{-1}$***

Least Square solution in Matlab for any A: ***X=pinv(A)*B, or X=A\B***

Selection of Order of Fit

$$y = f(x) = a_0 + a_1 x + a_2 x^2 + \dots\dots\dots a_m x^m$$

Given a set of data (x, y), what m should we choose?



2^{nd} and 6^{th} order look similar, but 6^{th} has a 'squiggle' to it. Is it necessary?

Under Fit Vs Over Fit: Pick a Right Order

• ***Underfit*** *- If the order is too low to capture obvious trends in the data*
• ***Overfit*** *– If the order is too high to over-do the requirement for the fit to 'match' the trends in the data*
• *Polynomials become more 'squiggly' as their order increases.*

General rules:

- pick a polynomial form at least several orders lower than the number of data points.
- Start with linear and add order until trends are matched.

Signal Processing

1. Sampling of Signals
2. Signal Filtering
3. Calibrations

Reading material on Kalman Filter:
"An Introduction to the Kalman Filter"
by Greg Welchand Gary Bishop
University of North Carolina at Chapel Hill

Questions?

56