UiO : University of Oslo

FYS3240

PC-based instrumentation and microcontrollers

Data fusion, estimation and control

Spring 2017 – Lecture #13

Bekkeng 27.03.2017

Digital filter examples

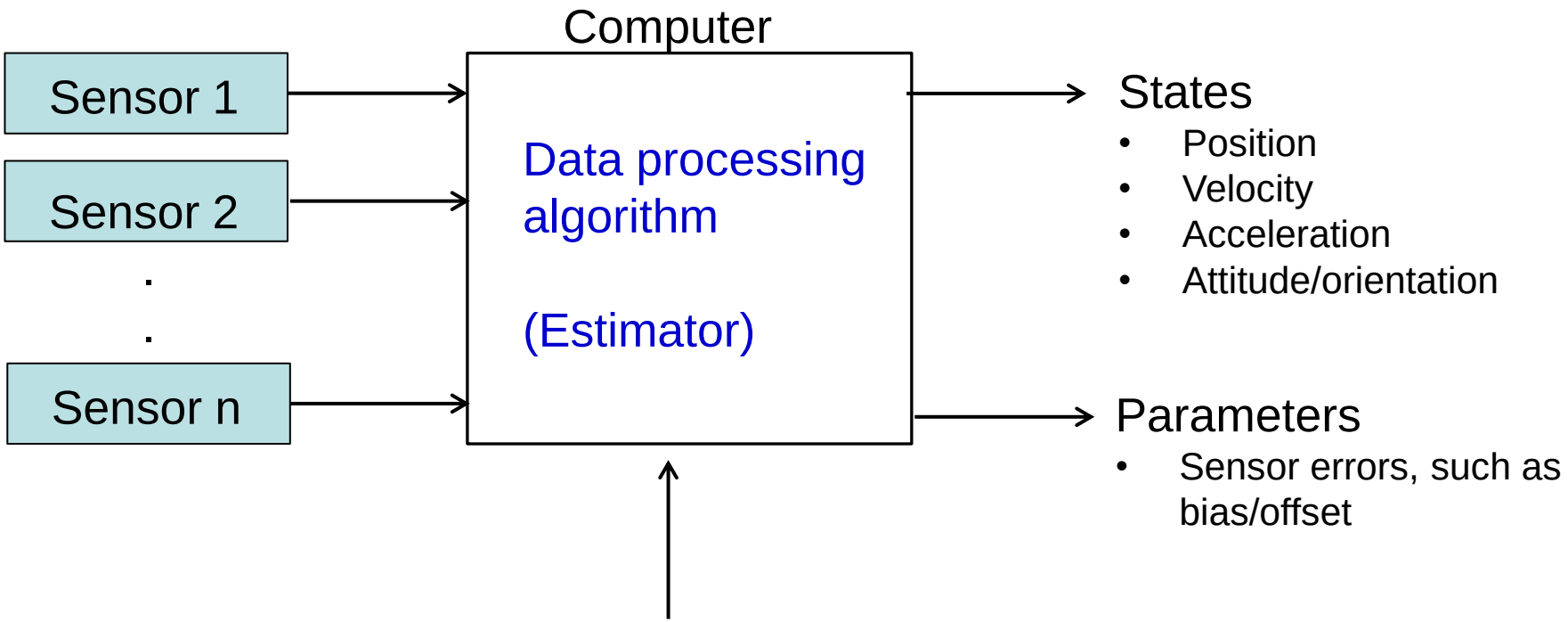
- Moving average filter: $y[n] = \frac{1}{3}(x[n] + x[n-1] + x[n-2])$

- 1. order low pass filter: y[n] = (1-a)*y[n-1] + a*x[n]
  - E.g. with a = 0.2

- Note: moving average and low pass filtering will result in a delay (lag) in the output!

Note: x[n] is the measurement at time n
y[n] is the filter output at time n

Multi-sensor systems



Can be implemented in real-time on an embedded system, or as part of post-processing of sensor data on a PC

Two-sensor data fusion example

- Both sensors take a measurement z of a constant but unknown parameter x, in the presence of noise v with standard deviation σ
- $z_1 = x + v_1$ and $z_2 = x + v_2$

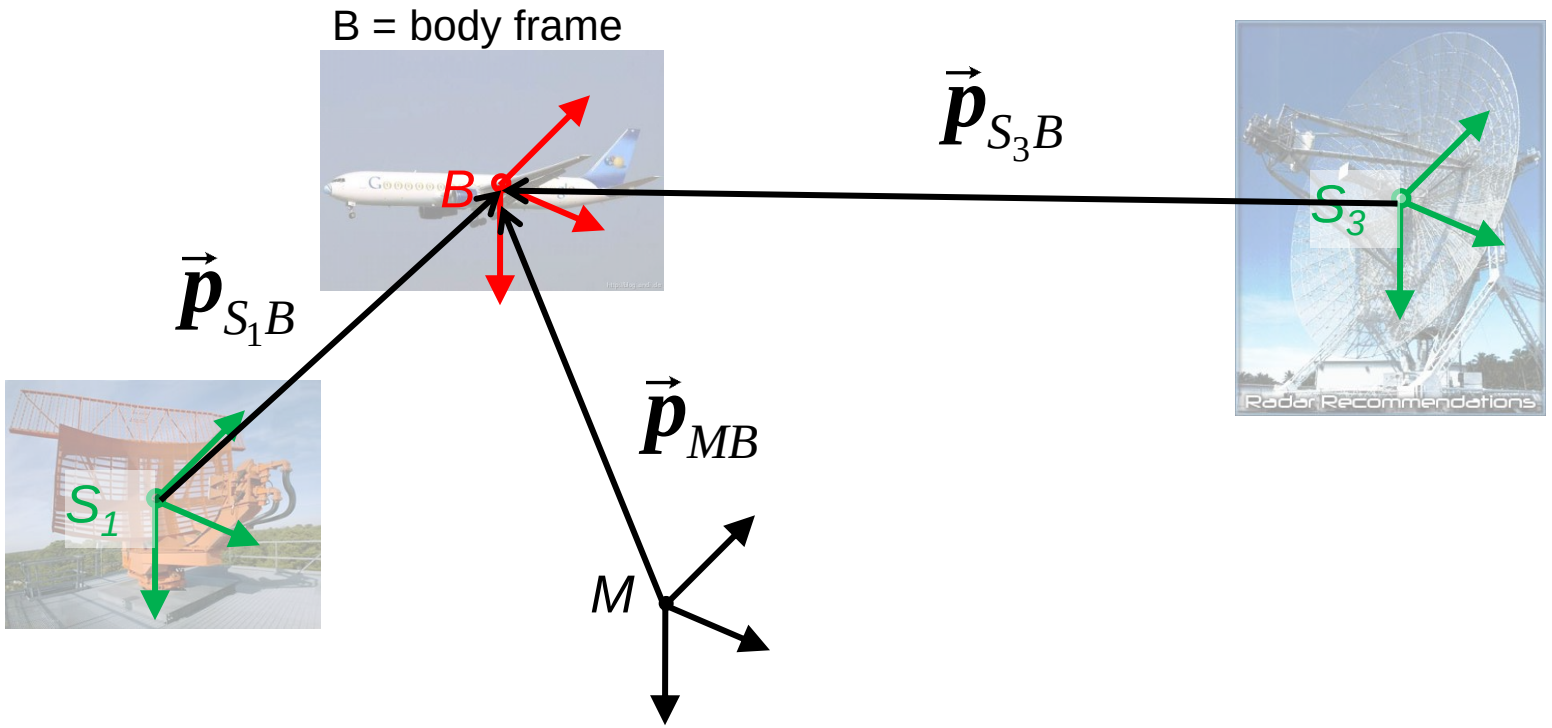
Question: How to combine the two measurements to produce an optimal estimate of $\hat{x}$ of the unknown parameter x?

Answer:

- $\hat{x} = k_1 z_1 + (1 - k_1) z_2$ (the estimate is a linear combination of the measurements)
- $\hat{x} = \left(\frac{{\sigma_2}^2}{{\sigma_1}^2 + {\sigma_2}^2}\right) z_1 + \left(\frac{{\sigma_1}^2}{{\sigma_1}^2 + {\sigma_2}^2}\right) z_2$
- Check: What happens if ${\sigma_1}^2 = {\sigma_2}^2$, or if σ_1 or σ_2 is equal to zero?

Reference frame for multi-sensor fusion

- Data measured in different coordinate frames (S1 and S3 in example)
- Before data fusion all vector measurements must be transformed into a common coordinate system M



Important coordinate frames

Frame symbol	Description
I	Inertial
E	Earth-fixed
B	Body-fixed
N	North-East-Down (local level)

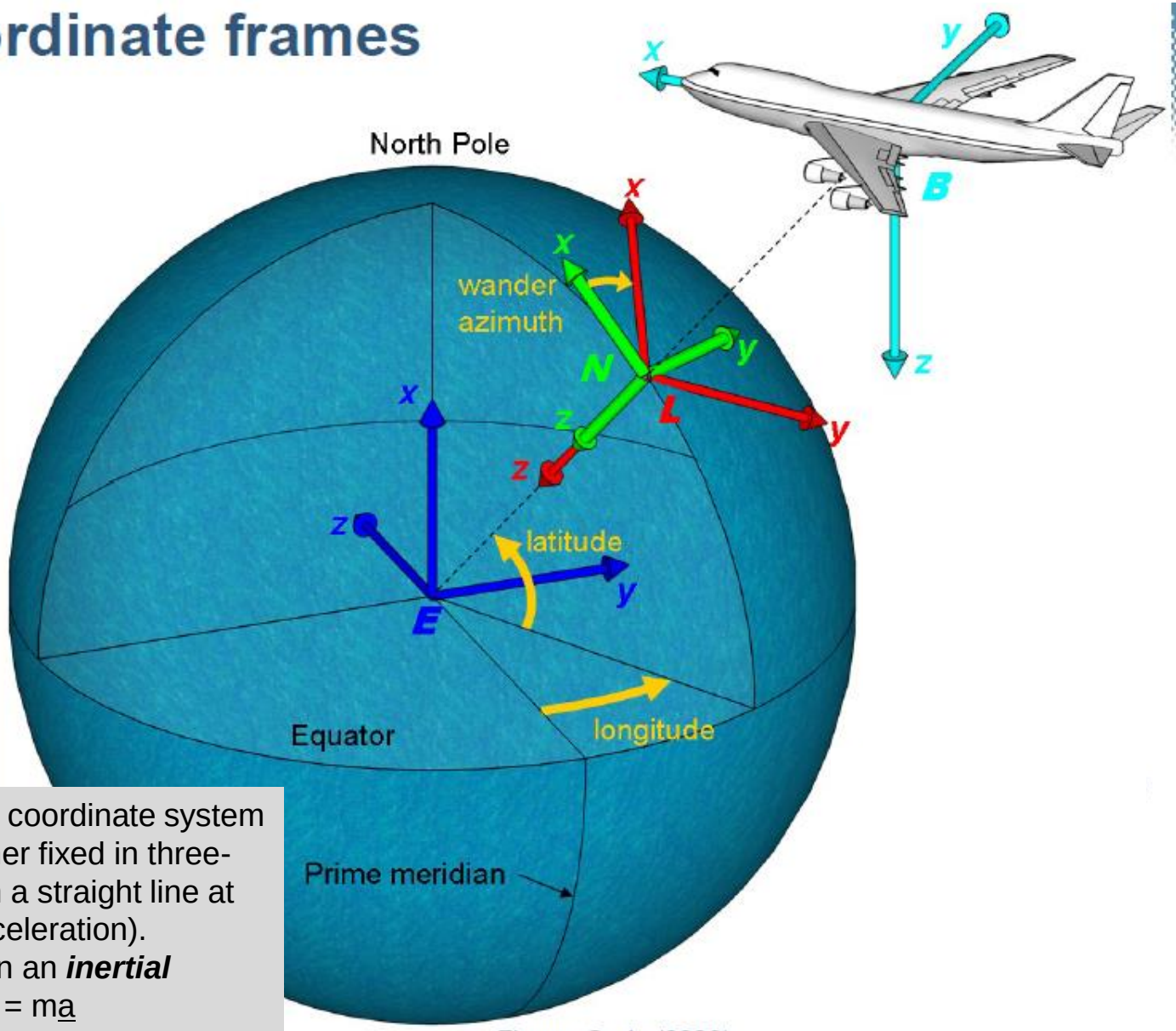


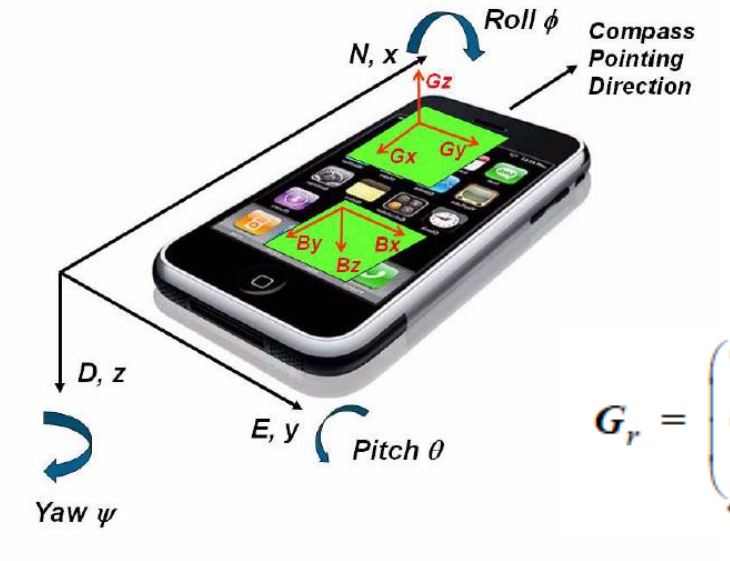
Figure: Gade (2008)

An inertial reference frame is a coordinate system that does not rotate, and is either fixed in three-dimensional space or moves in a straight line at constant velocity (with zero acceleration).
Newton's laws are valid only in an ***inertial reference frame***; remember F = ma

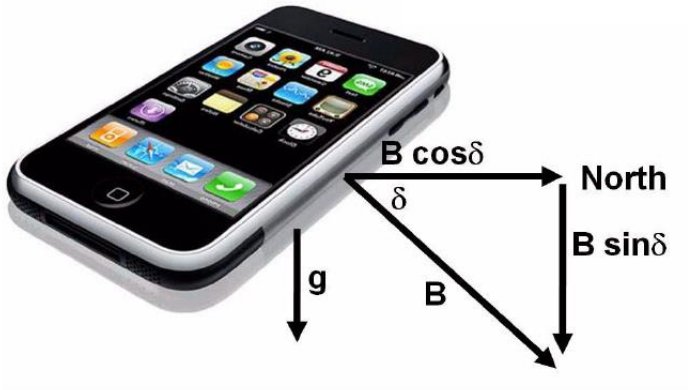
Example: Tilt-compensated compass

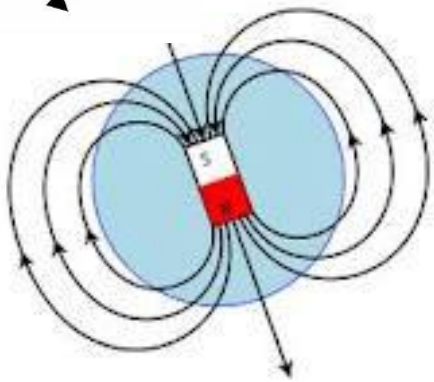
The Horizontal component of the Earth's magnetic field is used for compassing, since it points to North

$$B_r = B \begin{pmatrix} \cos\delta \\ 0 \\ \sin\delta \end{pmatrix} \begin{matrix} \text{N} \\ \text{E} \\ \text{D} \end{matrix}$$



$$G_r = \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} \begin{matrix} \text{N} \\ \text{E} \\ \text{D} \end{matrix}$$





1) Rotate measurements to a common phone frame (NED):

$$G_p = R_x(\phi)R_y(\theta)R_z(\psi)G_r \qquad B_p = R_x(\phi)R_y(\theta)R_z(\psi)B_r$$

2) Calculate the roll and pitch angles φ and θ from the accelerometer

3) De-rotate magnetometer readings (level the sensor) to correct for phone orientation:

$B_{corrected} = R_y(-\theta)R_x(-\phi)\ B_p$

Application Note

Example of multipoint measurements

GNC: Unmanned Aircraft System (UAS)

GNC : Guidance, Navigation and Control

OBC: Onboard computer

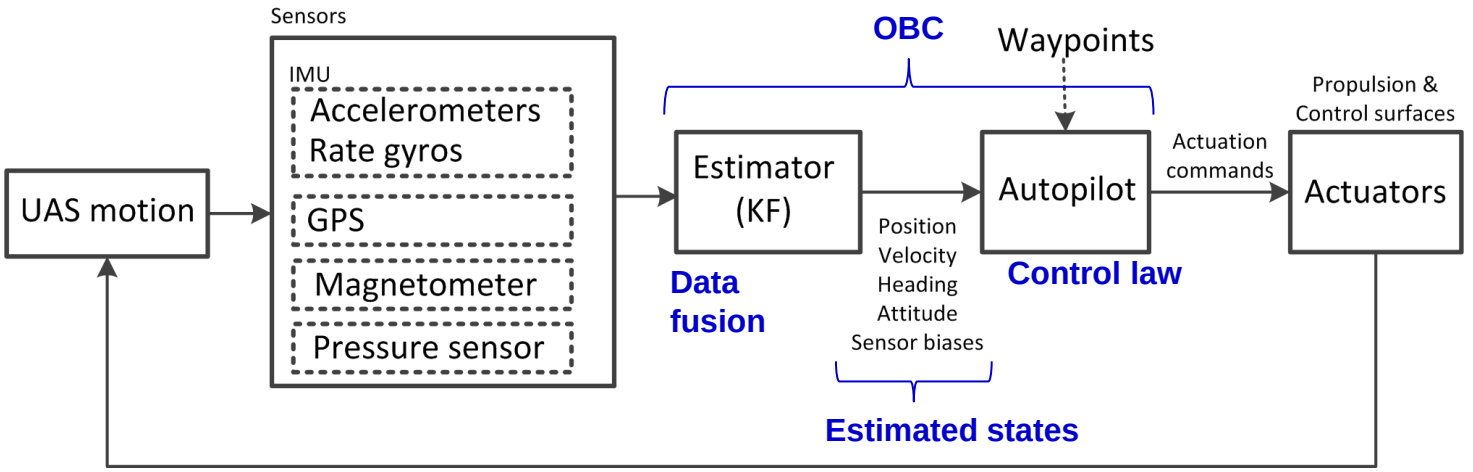
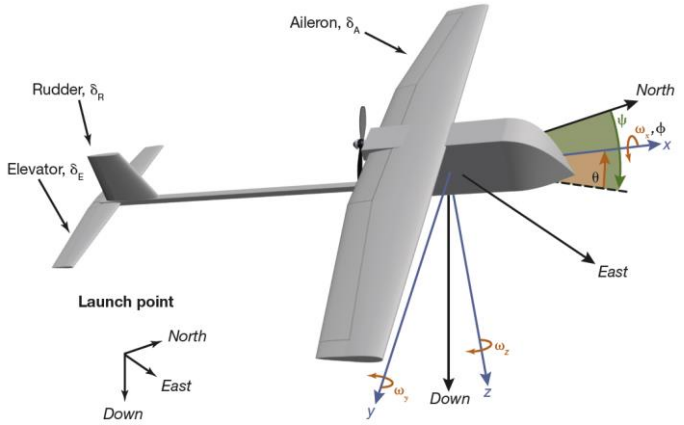


Illustration 5: Waypoints for flight over seattle



JOHNS HOPKINS APL TECHNICAL DIGEST, VOLUME 31, NUMBER 2 (2012)

Guidance example

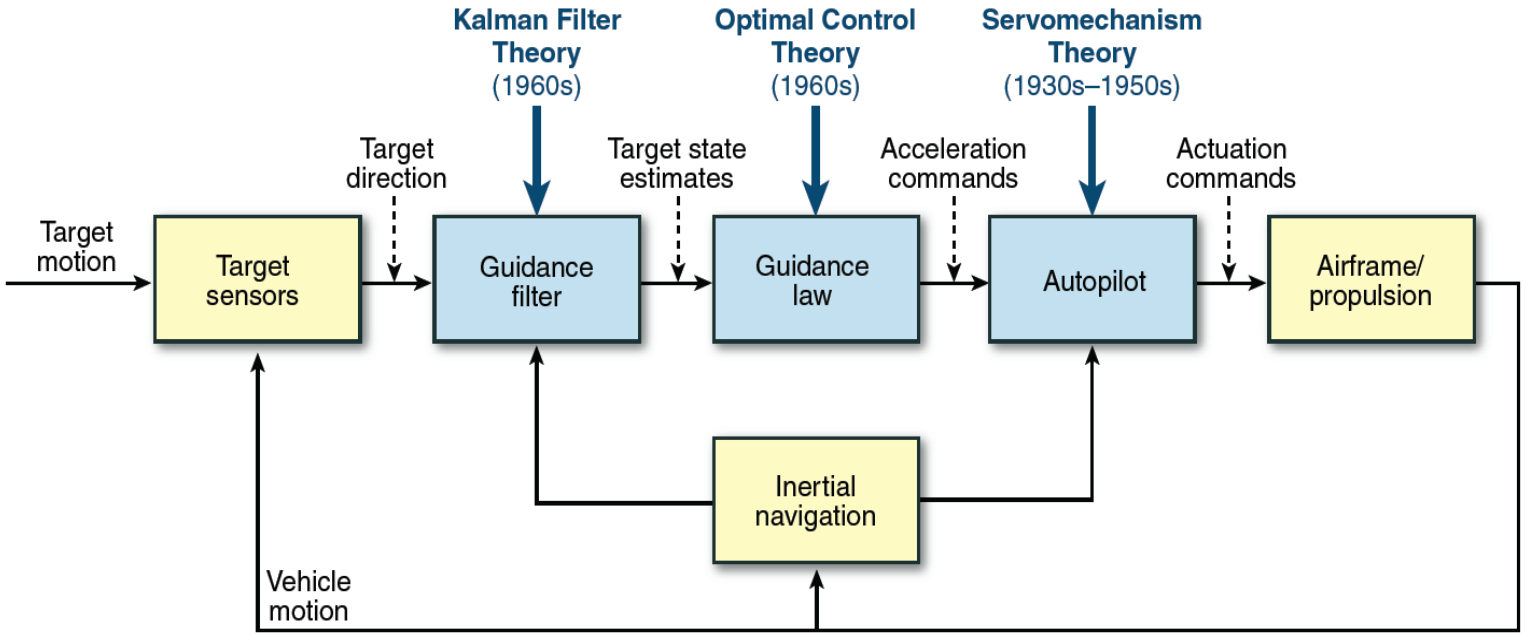


Figure 1. The traditional guidance, navigation, and control topology for a guided missile comprises guidance filter, guidance law, autopilot, and inertial navigation components. Each component may be synthesized by using a variety of techniques, the most popular of which are indicated here in blue text.

Figure from Palumbo, *Johns Hopkins APL Technical Digest*

Estimation

- In dynamic systems (systems which vary with time) the variables are called **states**.
- **Parameters** are variables that do not vary with time.
- Sometimes the states/parameters of a system are not or cannot be measured directly.
- Any measurements are corrupted by noise and other sensor errors
- In addition to finding and estimate of the unknown parameters we also want to estimate the uncertainty in our estimate

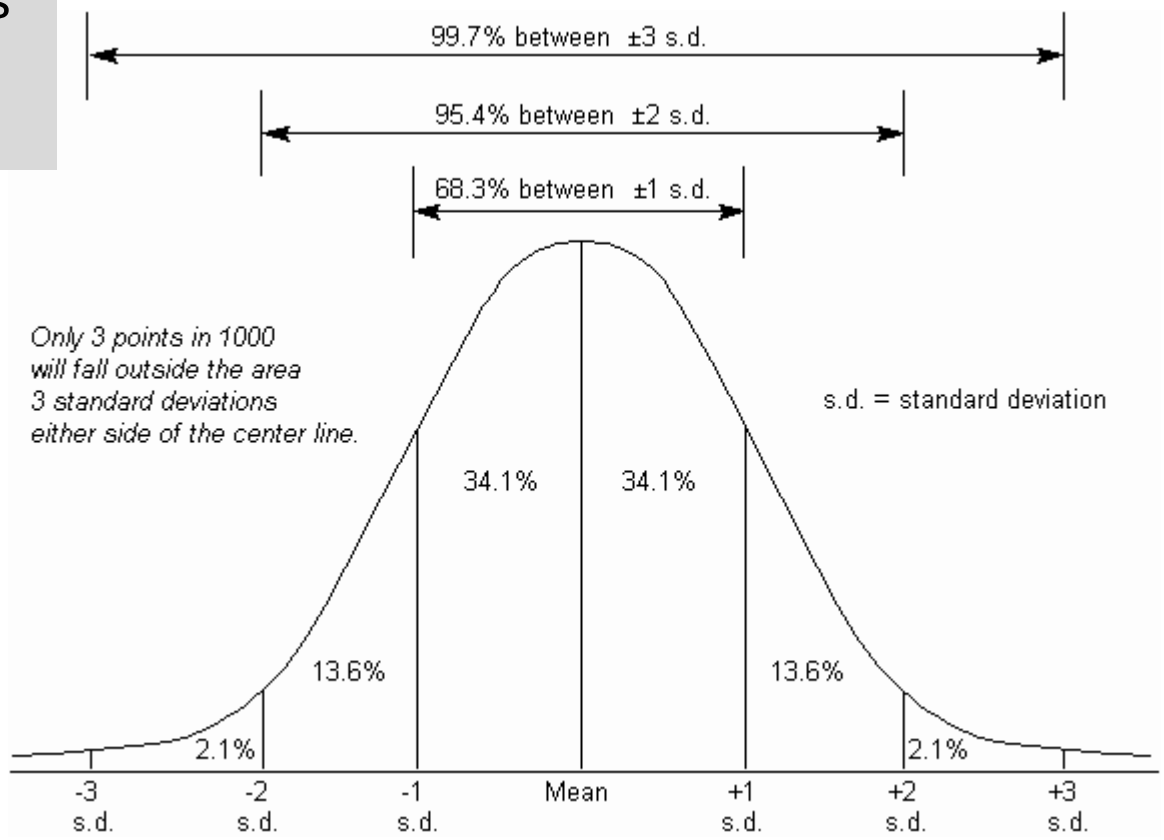
Estimator

- An estimator is a data processing algorithm.
- Often a need to combine measurements from different sensors with different data sample rates and accuracies.
- An optimal estimator combines all the available information (about the system and sensors).
- The estimator is optimal when it minimizes the estimation error in a well-defined statistical sense, based on a criterion of optimality.

Standard deviation

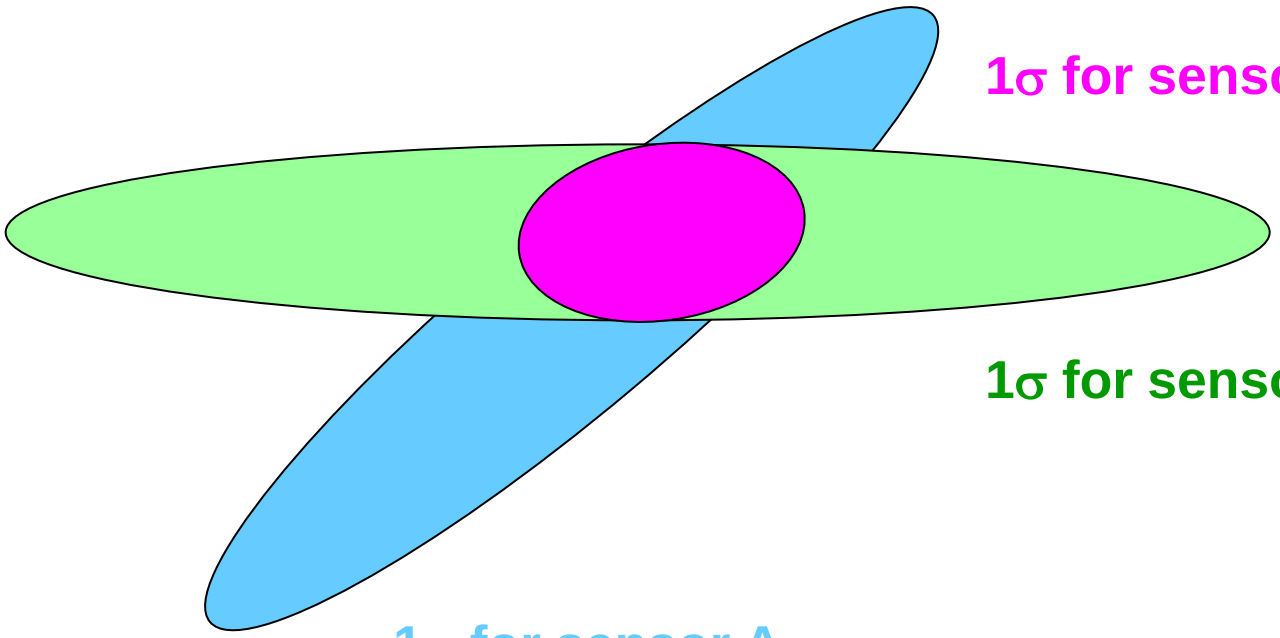
$$\sigma = s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

The standard deviation is the amount of variation from the mean



Multi-sensor data fusion

- Gives reduced uncertainty!
- Makes the system more robust!



Covariance

$$\mathbf{R} = \begin{bmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_\alpha^2 & 0 \\ 0 & 0 & \sigma_\varepsilon^2 \end{bmatrix}$$

- Covariance provides a measure of the strength of the correlation between two or more sets of random variables
- For uncorrelated variables: cov(X,Y) = 0
- cov(X,X) = var(X) = σ^2

- Covariance matrix:
  - A covariance matrix is a matrix whose element in the *i*, *j* position is the covariance between the *i* th and *j* th elements of a random vector
  - The diagonal elements of the covariance matrix is the variances
  - The off-diagonal elements represent the covariance

System of linear equations I

- A general system of *m* linear equations with *n* unknowns x_i can be written as:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \vdots \qquad\quad & \vdots \qquad\qquad\quad \vdots \quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m. \end{aligned}$$

- This can be written as a matrix equation of the form:

$$A\mathbf{x} = \mathbf{b}$$

where $$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

System of linear equations II

- If the number of measurements (number of equations) is equal to the number of unknowns x_i (m = n), the unknowns can be found from the inverse solution:

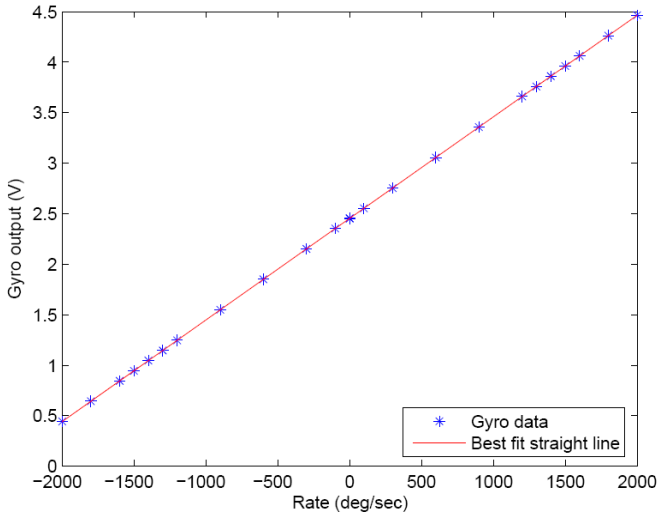
  $\mathbf{x} = A^{-1}\mathbf{b}$ (In Matlab: x = A\b, or x = inv(A)*b)

- In the more common case, there are more measurements (equations) than unknown (m > n). This is called an over determined systems. Then, a Least squares method can be used to estimate the unknown parameters.

Batch vs. recursive estimator

- Batch processing:
  - All available measurements are processed at one time
- Recursive processing:
  - Measurements are processed as they become available
    - Required computer storage is kept at a minimum

$$z(\omega) = a \cdot \omega + b$$



Linear Least-Squares (LS) Estimation

Model (static system) - The Measurement equation:

$$\mathbf{z}_k = H_k \mathbf{x}_k + \mathbf{v}_k$$

LS Cost function to be minimized:

$$J = \frac{1}{2} \sum_{k=0}^{N} (\boldsymbol{z}_k - H\hat{\boldsymbol{x}}_k)^T R^{-1} (\boldsymbol{z}_k - H\hat{\boldsymbol{x}}_k)$$

$$\boldsymbol{x} = [x_1 \quad x_2 \quad \dots \quad x_n]^T$$

$\mathbf{z}$: measurement (column) vector
H : measurement matrix
$\mathbf{x}$: (column) vector of unknowns
$\mathbf{v}$: noise vector
T : The transpose of a matrix/vector

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ . \\ . \\ x_n \end{bmatrix}$$

R : Measurement covariance matrix (weighting)
$\hat{\mathbf{x}}$: Estimate (solution)
P : Covariance matrix of the estimate (solution)

Explicit Solution for the optimal estimate:

$$\hat{\boldsymbol{x}} = (H^T R^{-1} H)^{-1} H^T R^{-1} \boldsymbol{z}$$

$$P = (H^T R^{-1} H)^{-1}$$

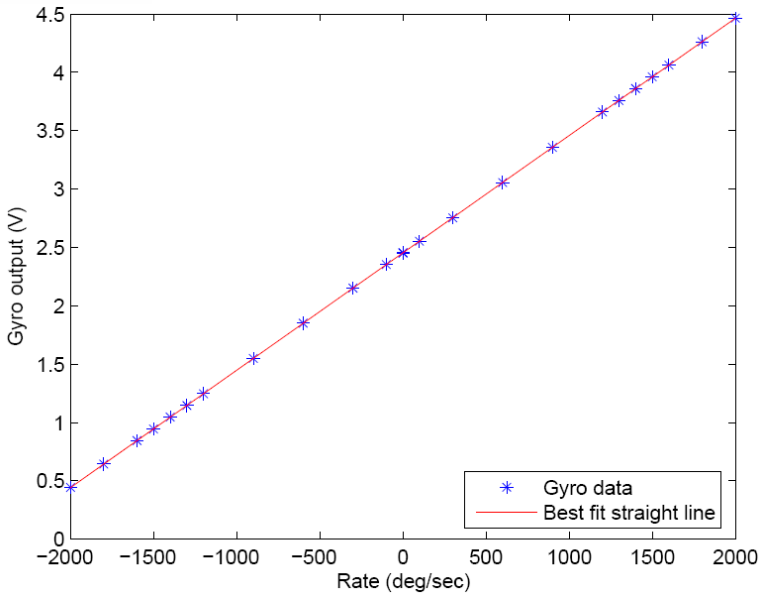
← The estimated error of the estimate

This is a batch estimator - all the measurements are processed at the same time. LS used for nonrandom (deterministic), time-invariant parameters

1D model example – curve fitting

- Measured gyro rotation rate z (in voltage) as a function of applied rate table rotation rate ω (in deg/sec) .

$z(\omega)$



1D Model:

$$z(\omega) = a \cdot \omega + b$$

Model as a matrix equation:

$$z = H\boldsymbol{x}$$

$$H = [\omega \;\; 1]$$

$$\boldsymbol{x} = \begin{bmatrix} a \\ b \end{bmatrix}$$

Simplest case: R = I (identity matrix) ⇨ R falls out of the LS-equations

Weighted Least Squares

- The traditional least squares solution places equal emphasis on each measurement.
- However, measurements are often made with unequal precision (e.g. due to different sensor accuracies). Therefore, we want to add a weight such that the more precise measurements are given more importance.

$$\hat{\boldsymbol{x}} = (H^T R^{-1} H)^{-1} H^T R^{-1} \boldsymbol{z}$$

$$P = (H^T R^{-1} H)^{-1}$$

$$\mathrm{R} = \begin{bmatrix} \sigma_r^{\,2} & 0 & 0 \\ 0 & \sigma_\alpha^{\,2} & 0 \\ 0 & 0 & \sigma_\varepsilon^{\,2} \end{bmatrix}$$

- The R matrix (weighting matrix) can be selected as a diagonal matrix with the variance of the sensor measurements on the diagonal.
- If the R matrix is selected to be an identity matrix (similar to not include the R matrix in the equations), all measurements are weighted equally.
- R is the measurement error covariance matrix.

The reqursive LS estimator

- Batch algorithms are not suitable for realtime applications.
- A recursive estimator, on the other hand, processes the measurements sequentially as they become available, and stores only the value of the last state ("Parameter" is used for time-invariant (or quasi-static) variables, while "state" is used for time-varying (dynamic) variables)
- The reqursive least squares estimator:

Measurement Predicted measurement

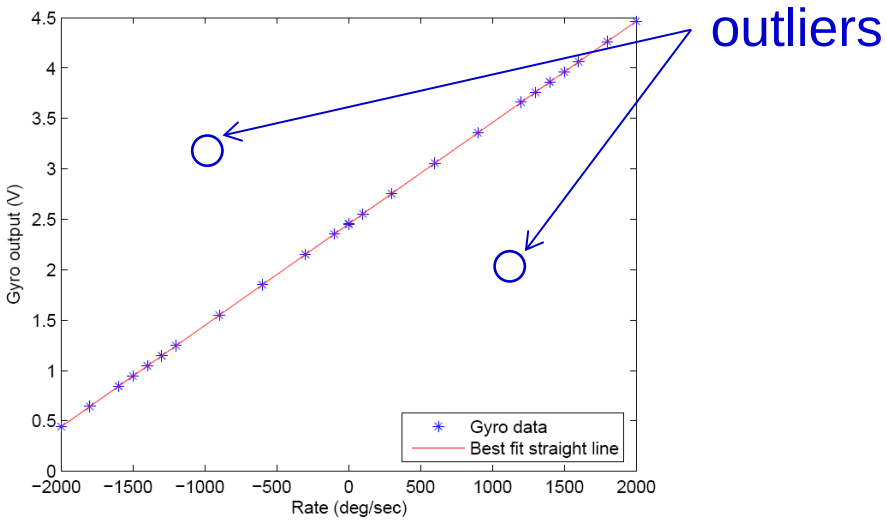
$$\hat{x}(k+1) = \hat{x}(k) + W(k+1)[z(k+1) - H(k+1)\hat{x}(k)]$$

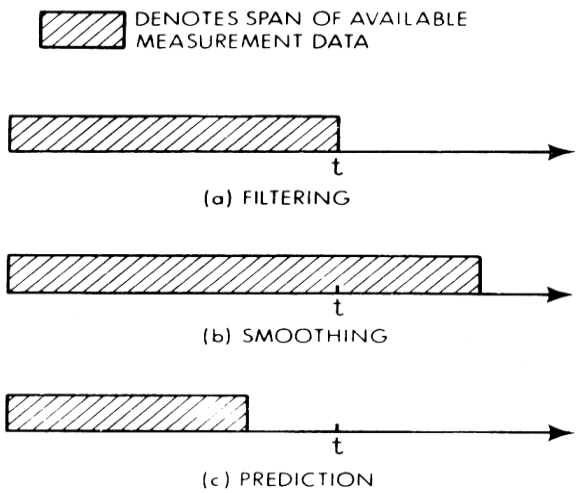
The new estimate $\hat{x}$ at time (k+1) is equal to the estimate at time (k) pluss a correction term. The correction term consist of a gain W(k+1) that multiplies the difference between the measurement z(k+1) and the predicted value of this measurement given by $H(k+1)\hat{x}(k)$

Filtering

- In estimation the term **filtering** refers to estimating parameters (the state vector) describing the system at the current time, based on all past measurements. So, filtering in estimation theory includes more than just filtering noise, such as a low pass filter.

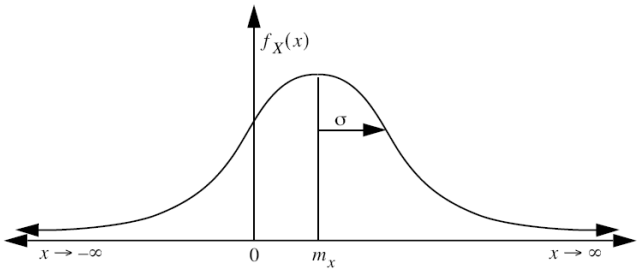
- Offline (non RT) processing makes it possible to obtain more accurate estimates. A **smoother** produces improved estimates by making use of data both before and after any given time point of interest.

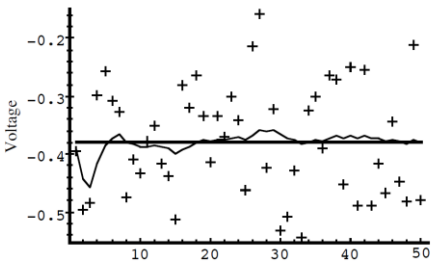




Kalman filter (KF) I

- One of the most widely used estimation algorithms.
- The Kalman filter is **used for random parameters (which can be time varying).**
- In the 1960s, the Kalman filter was applied to navigation for the Apollo Project, which required estimates of the trajectories of manned spacecraft going to the Moon and back.
- Later the Kalman filter has been applied for all kinds of navigation and tracking applications.
- The Kalman filter is a **recursive estimator**
- The Kalman filter is **the optimal minimum mean square error (MMSE) estimator for linear, Gaussian systems**.
- MMSE one possible (and very often used) optimization criteria
- “Gives a best fit (to observed measurements) in a statistical sense”





Kalman filter (KF) II

- Two sorts of information are utilized:
  - ***Measurements*** from sensors.
  - ***Mathematical models*** of the system
    - describing how the different states depend on each other, and how the measurements depend on the states.

Model (dynamic system):

$$\begin{aligned} \mathbf{x}_{k+1} &= \Phi_k \mathbf{x}_k + \Lambda_k \mathbf{u}_k + \Gamma_k \mathbf{w}_k \\ \mathbf{z}_k &= H_k \mathbf{x}_k + \mathbf{v}_k \end{aligned}$$

- In addition the *accuracy* of the measurements and the model must be specified.

Kalman filter III

- The Kalman filter produces estimates of the true values of measurements by predicting a value, estimating the uncertainty of the predicted value, and computing a weighted average of the predicted value and the measured value. The most weight is given to the value with the least uncertainty. The estimates produced by the method tend to be closer to the true values than the original measurements because the weighted average has a better estimated uncertainty than either of the values that went into the weighted average. From Wikipedia

Kalman filter equations

Predicted error (covariance matrix)

Model time-predicted value of the states

Estimate at the previous time step

$$\bar{\mathbf{x}}_{k+1} = \Phi_k \hat{\mathbf{x}}_k + \Lambda_k \mathbf{u}_k$$

$$\bar{P}_{k+1} = \Phi_k \hat{P}_k \Phi_k^T + \Gamma_k Q_k \Gamma_k^T$$

$$K_k = \bar{P}_k H_k^T (H_k \bar{P}_k H_k^T + R_k)^{-1}$$

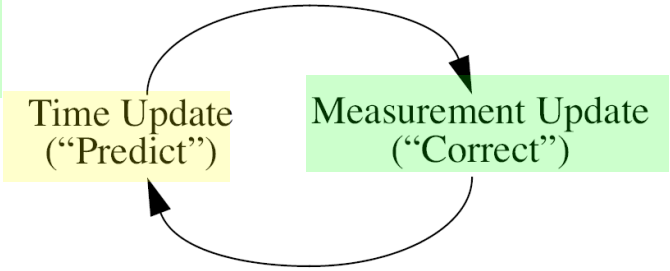
Optimal gain (K) calculation

$$\hat{\mathbf{x}}_k = \bar{\mathbf{x}}_k + K_k(\mathbf{z}_k - H_k \bar{\mathbf{x}}_k)$$

$$\hat{P}_k = (I - K_k H_k)\bar{P}_k$$

Estimated error (covariance matrix)

Estimated states



Estimation in nonlinear systems

- Based on **linearization** (taylor series expanion) of the non-linear equations
- **Requires an initial estimate of the parameters close to the true parameter values, in order to ensure that the data processing algorithm converges to the true solution**
- This makes non-linear (in the unknown parameters/states) problems much more complicated!
- Most real-world problems are non-linear!

Example: GPS position calculation

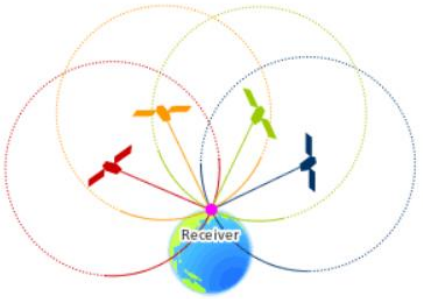
- The measured pseudorange $\tilde{P}^k$ from a satellite k can be expressed as (since we can assume no clock error in the satellite):

$$\tilde{P}^k = \sqrt{(X^k - x)^2 + (Y^k - y)^2 + (Z^k - z)^2} + d + v = \rho^k + d + v$$

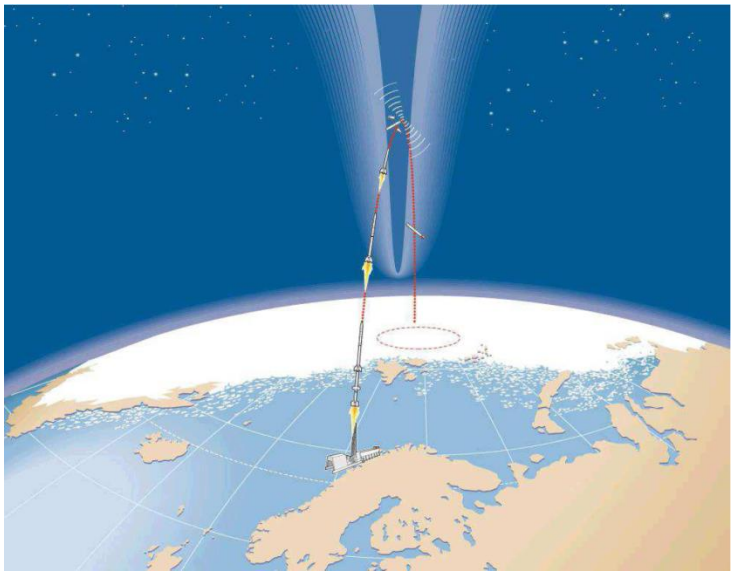
- $c\tau$ = d is the position error due to receiver clock error, (X^k,Y^k,Z^k) is the known position of satellite k, (x,y,c) is the true receiver position, and v is zero mean Gaussian white noise with variance σ^2
- This is a **nonlinear problem** on the form: $\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k$ where

$$h(\boldsymbol{x}) = \sqrt{(X^k - x)^2 + (Y^k - y)^2 + (Z^k - z)^2} + d$$

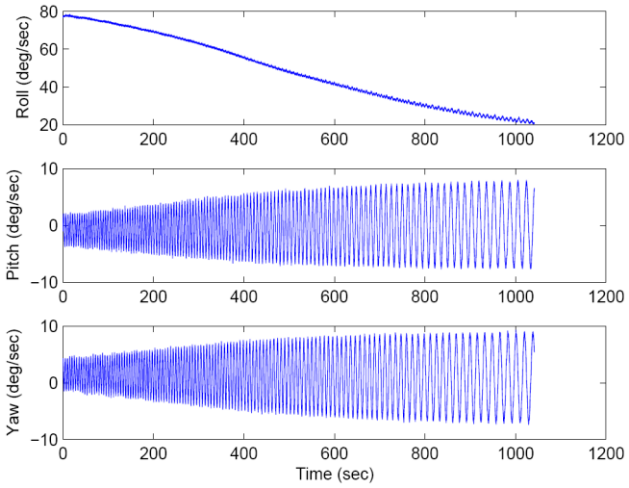
- **x** = [x, y, z , d]^T are the unknown parameters to be estimated.

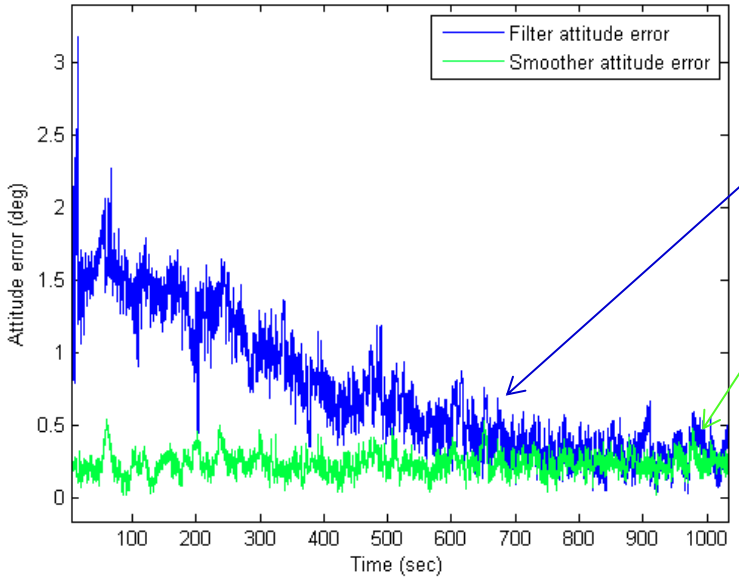


Example: Attitude determination



Simulated rocket angular rates





"Real-time" filter result

Post processing (non real-time) result

Post-processing using a smoother will give more accurate results then what is possible using a real-time filter, since more information is available!

Alternatives to the Kalman filter

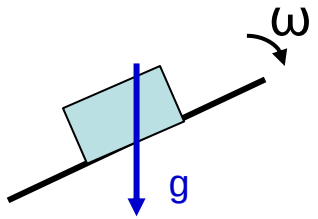
NOT THE OPTIMAL SOLUTION, BUT EASIER TO IMPLEMENT

Alpha filter – a first-order approach

- If you have a measurement $\tilde{x}_k$, you can apply a first order filter:

$$\hat{x}_k = (1 - \alpha)\bar{x}_k + \alpha\tilde{x}_k$$

- $\hat{x}_k$ is the updated (from measurements) estimate at time k
- $\bar{x}_k$ is the predicted (time propagated) estimate at time k, from a model: $\bar{x}_k = f(\hat{x}_{k-1})$
  - Data fusion example: rate gyro measurement used for predicting a rotation angle and an accelerometer used as an inclinometer to measure the absolute angle.
- α is a scalar gain between 0 and 1 (typically constant)
- If no measurements $\tilde{x}_k$ are available, α is set to 0 → only prediction
- This approach will filter out noise, but a good α must be found from "trial and error" (possibly with some "guidelines")
- Not as good as a Kalman filter!
  - Not an optimal solution!



Complimentary filter for data fusion

- Another simpler alternative to the Kalman filter
  - Not an optimal solution for a properly modelled random process.
  - Can be a good solution if the signals are not well-modelled, and/or the signal-to-noise ratio in the measurements are high.

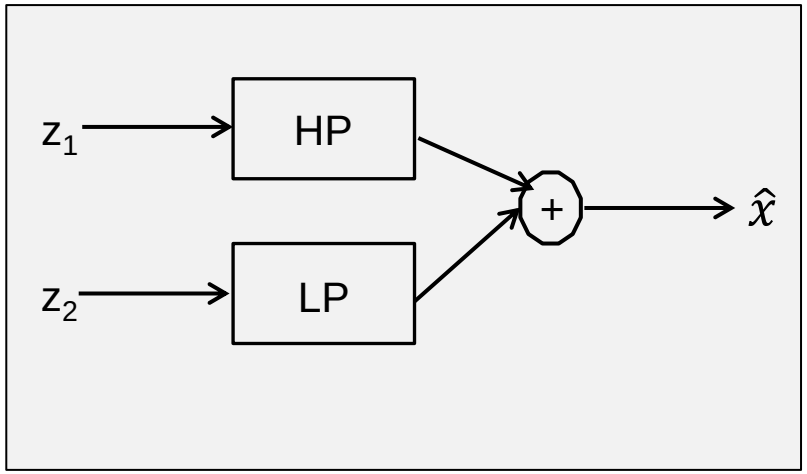
- **The idea behind the complementary filter is to take slow moving signals and fast moving signals and combine them.**
  - **The filter is based on an analysis in the frequency domain.**

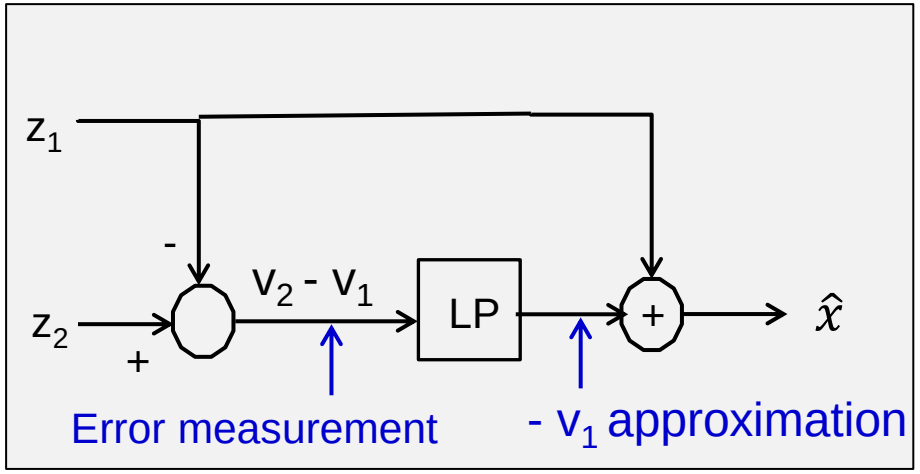
- The complementary filter fuses the sensor1 and sensor2 data by passing the former through a 1^{st}-order low pass and the latter through a 1^{st}-order high pass filter and adding the outputs.

- Possibly easy to implement on a embedded processor.

Complimentary filter architectures

- Assume two sensors that take a measurement z of a constant but unknown parameter x, in the presence of noise v.
- $z_1 = x + v_1$ and $z_2 = x + v_2$
- Assume that the noise in z_2 is mostly high frequency, and the noise in z_1 is mostly low frequency.
- Two possible complimentary filter architectures to estimate x:





Video YouTube

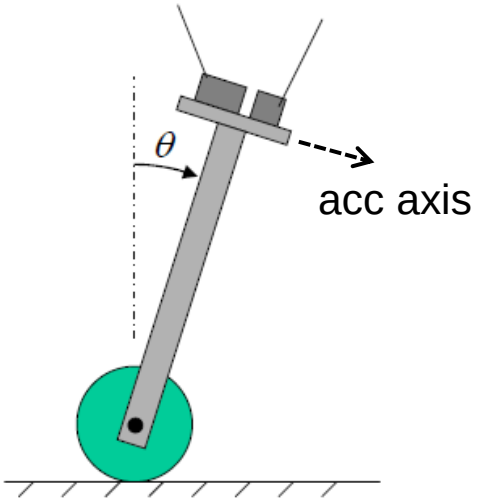
Complementary filter - balance robot

Often, there are cases where you have *two* different measurement sources for estimating *one* variable and the noise properties of the two measurements are such that one source gives good information only in low frequency region while the other is good only in high frequency region.

→ You can use a complementary filter !

Example from MIT

Example : Tilt angle estimation using accelerometer and rate gyro

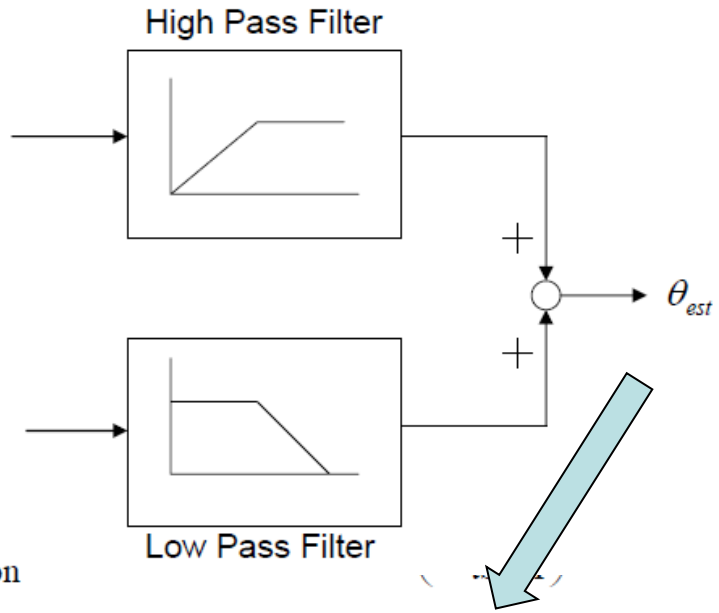


$\theta \approx \int (\text{angular rate})\, dt$

- not good in long term due to integration

$\theta \approx \sin^{-1}\left(\frac{\text{accel. output}}{g}\right)$

- only good in long term
- not proper during fast motion

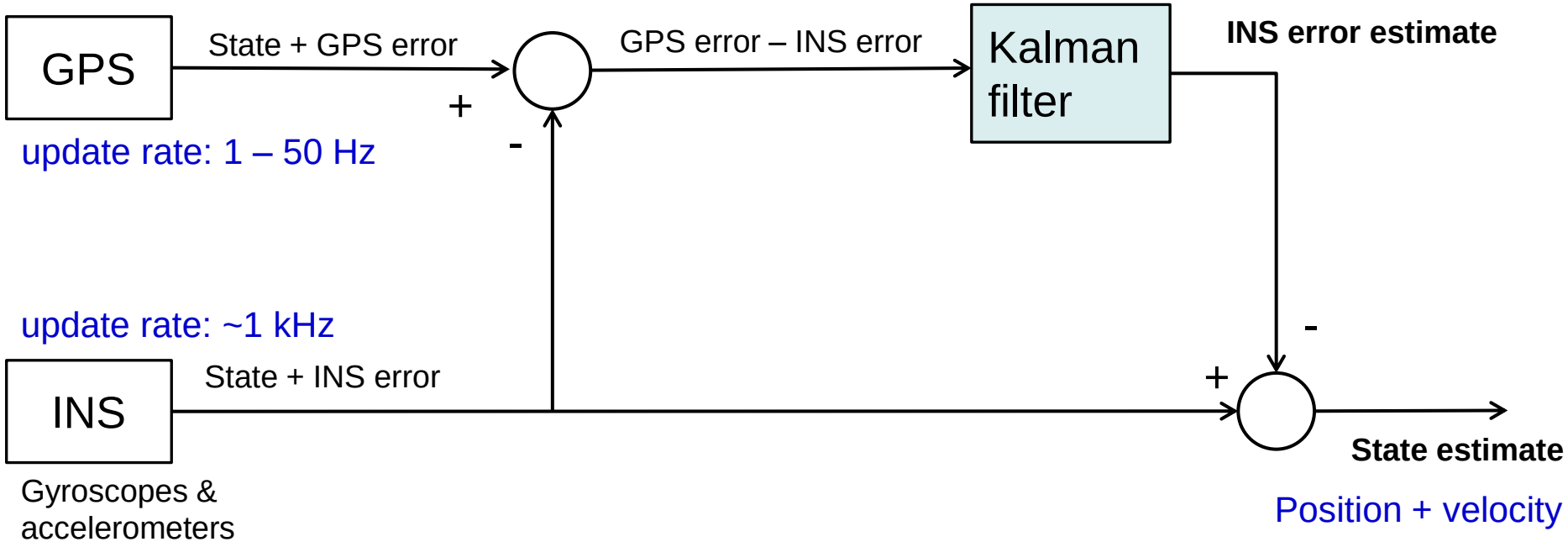


$$\hat{\theta}_k = \alpha(\theta_{k-1} + \omega_k \Delta t) + (1-\alpha)a_k$$

Example: α = 0.98

To limit gyro drift

Complimentary INS/GPS integration with KF



- A very common INS/GPS integration
- INS solution only when GPS not available
  - *Error grows with time without GPS data*

Control

On/off (bang-bang) controller

- A controller that switches between two states; e.g. either completely on or completely off
- Examples:
  - Most common residential thermostats are bang–bang controllers.

PID controller

- Proportional-Integral-Derivative (PID) algorithm is the most common control algorithm
  - Used for heating and cooling systems, fluid level monitoring, flow control, and pressure control.
- Calculates a term **proportional to the error** - the P term.
- Calculates a term **proportional to the integral of the error** - the I term.
- Calculates a term **proportional to the derivative of the error** - the D term.
- The three terms - the P, I and D terms, are added together to produce a control signal that is applied to the system being controlled
- Sometimes only a PI controller is used

PID controller II

- A PID controller continuously calculates **an *error* *value*** as the difference between **a measured process variable** and a desired **setpoint**.
- The controller attempts to minimize the error over time, by adjustment of a *control variable u(t)* , such as the position of a control valve.

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt}$$

- *P* accounts for present values of the error.
- I accounts for past values of the error, accumulates over time.
- *D* accounts for possible future values of the error, based on its current rate of change.
- Must tune the coefficients Kp, Ki og Kd
  - E.g. with Ziegler–Nichols method

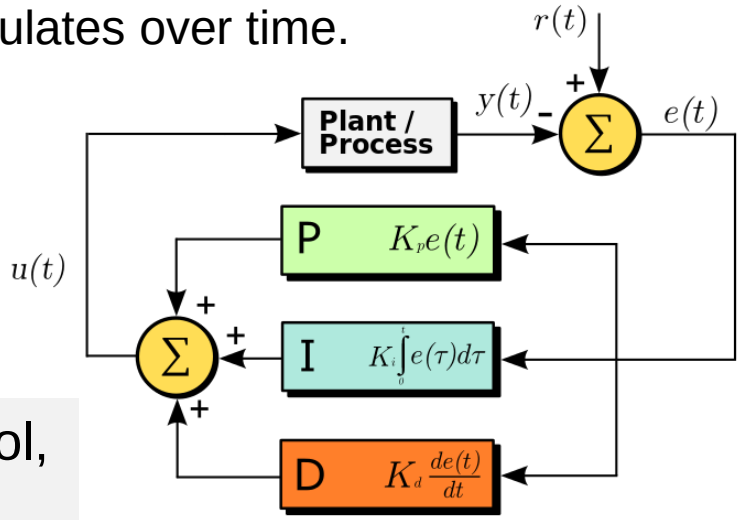


Figure from wikipedia

In general PID does not provide *optimal* control, since no modelling of the process is used

PID controller tuning example