UiO : University of Oslo

FYS3240

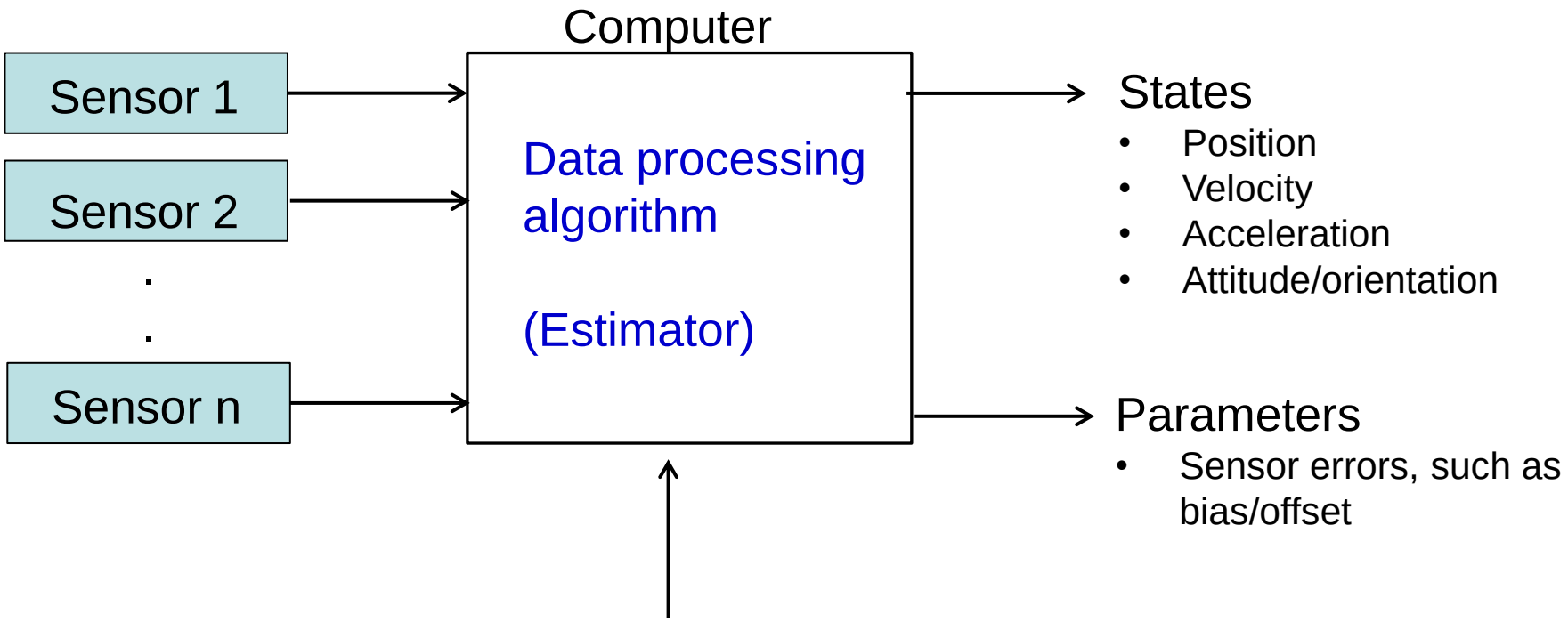
PC-based instrumentation and microcontrollers

Data fusion, estimation and sensor calibration

Spring 2015 – Lecture #13

Bekkeng 29.3.2015

Multisensor 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?

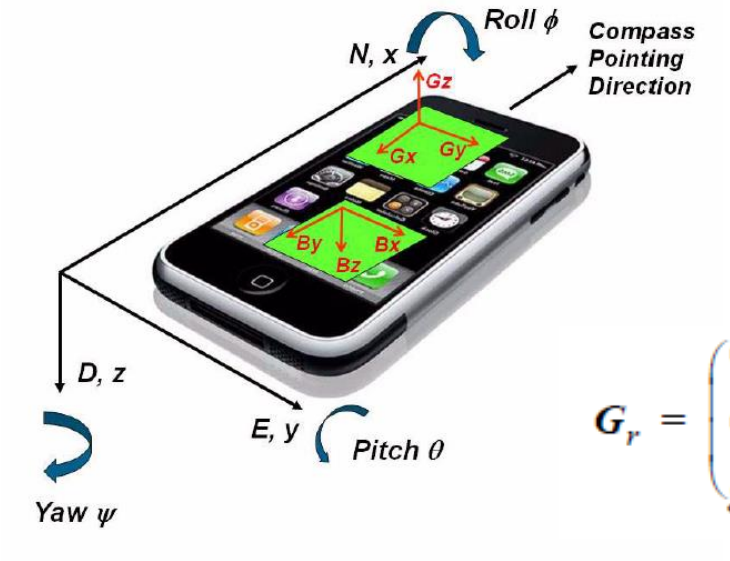
Answer:

- $\hat{x} = k_1 z_1 + (1 - k_1) z_2$ (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?

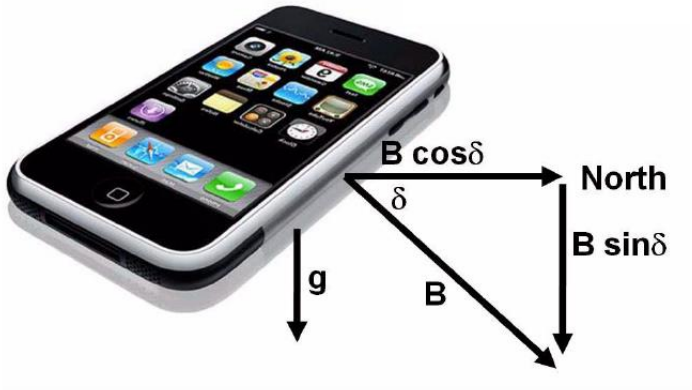
Example: Tilt-compensated compass

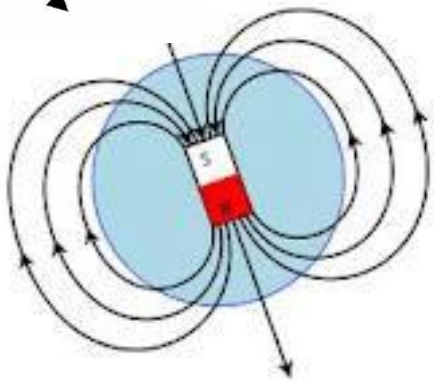
The Horizontal component of the Earth's magnetic field is used for compassing, since it points to North

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



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





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

GNC: Unmanned Aircraft System (UAS)

GNC : Guidance, Navigation and Control

OBC: Onboard computer

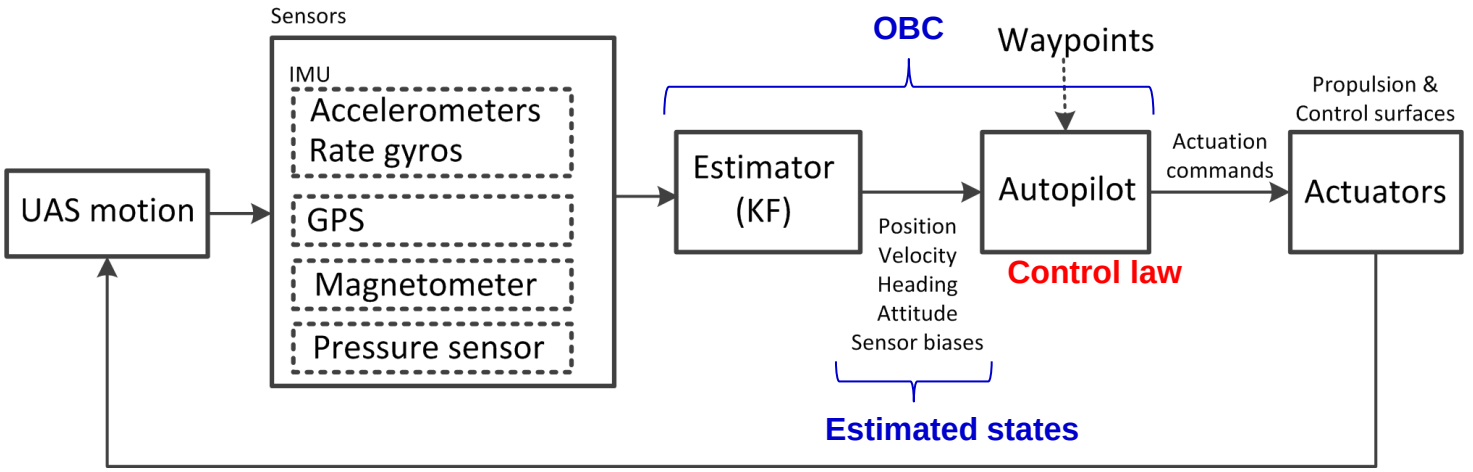
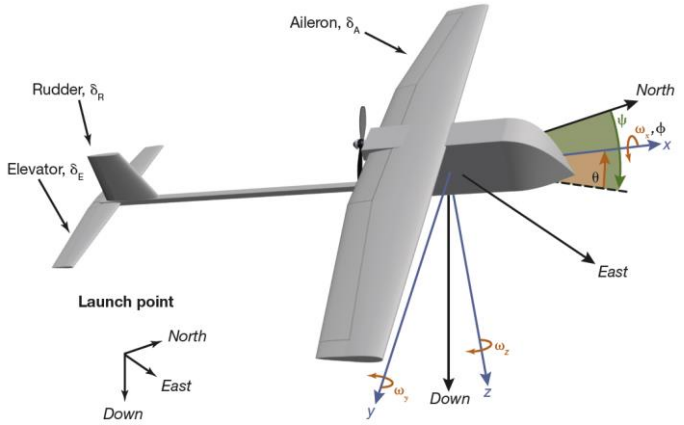
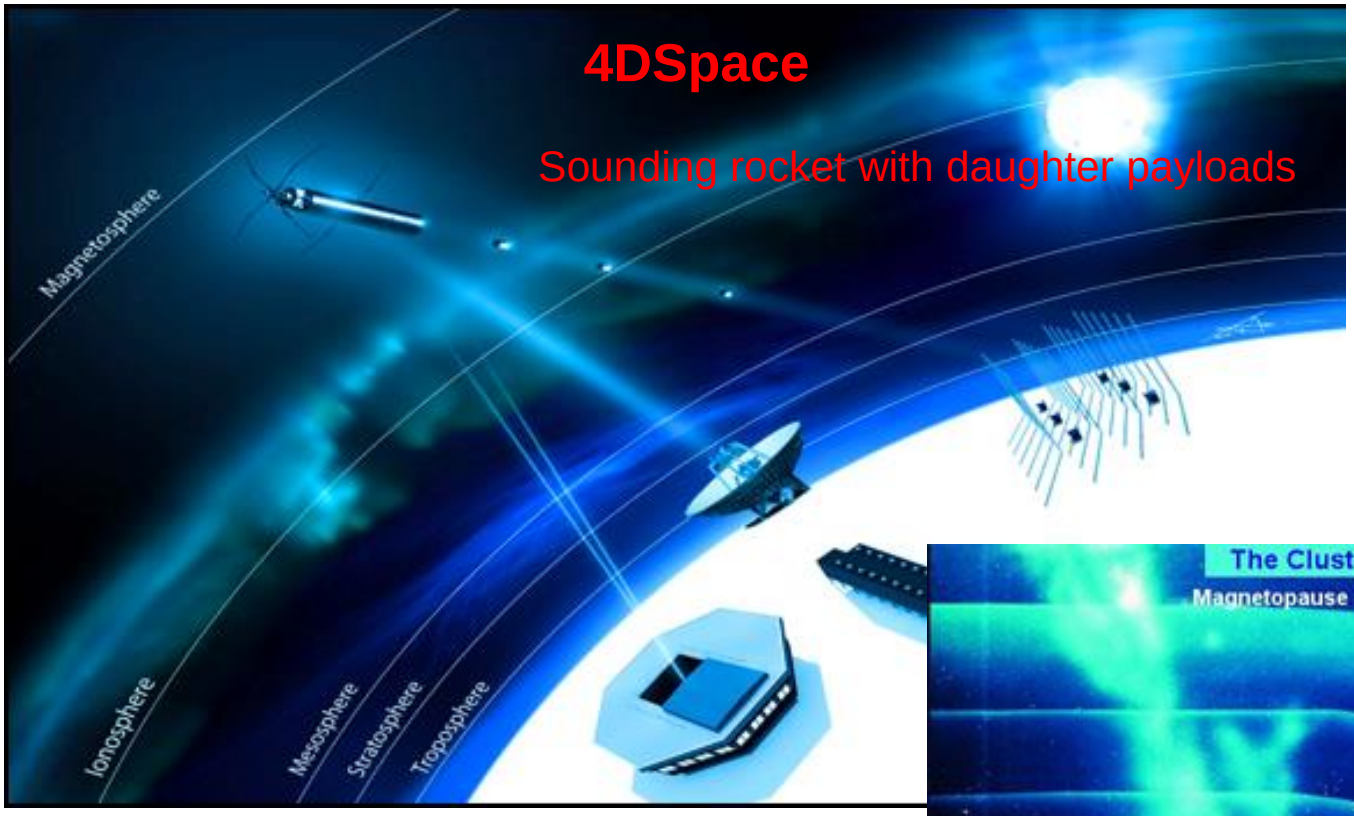


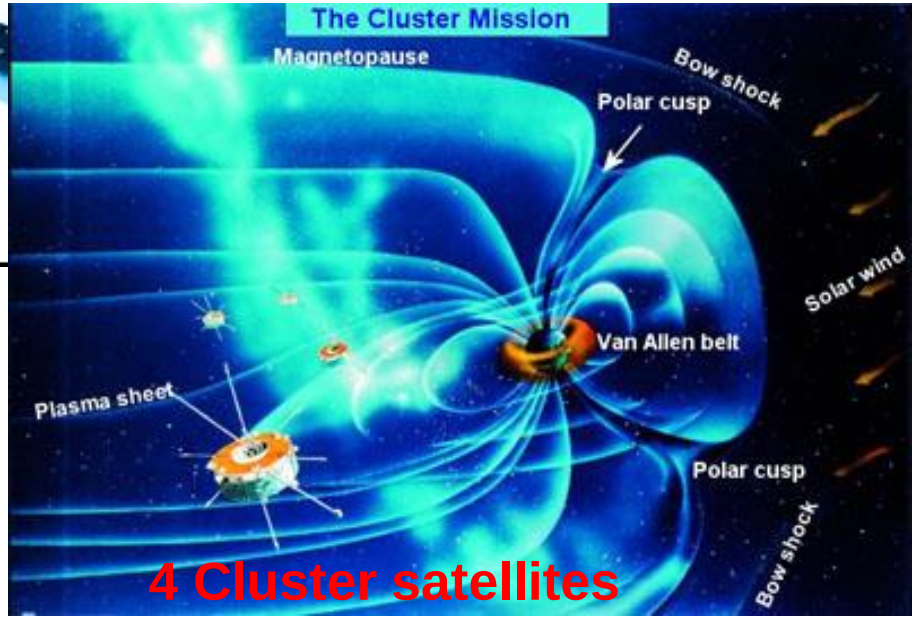
Illustration 5: Waypoints for flight over seattle



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

Examples of coordinated measurements





Coordinated multi-point measurements

⇩

Temporal and spatial information

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

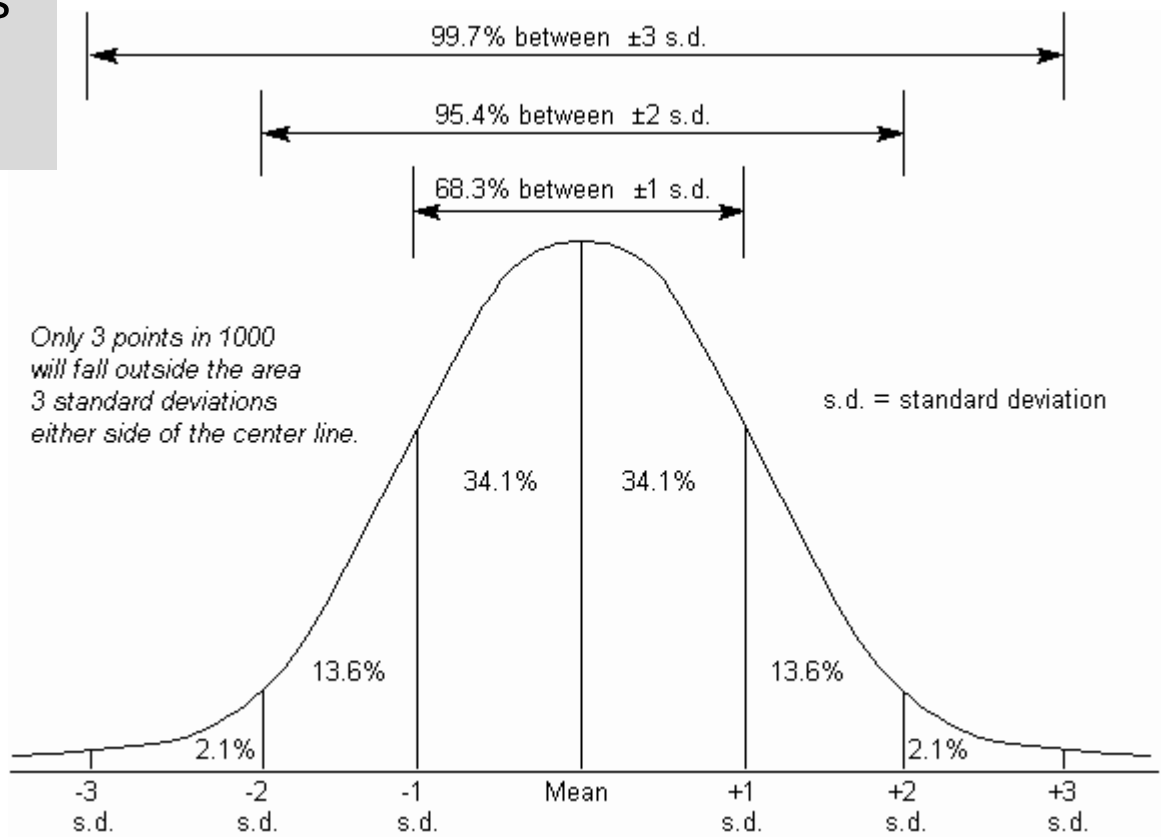
Estimator

- 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



Covariance

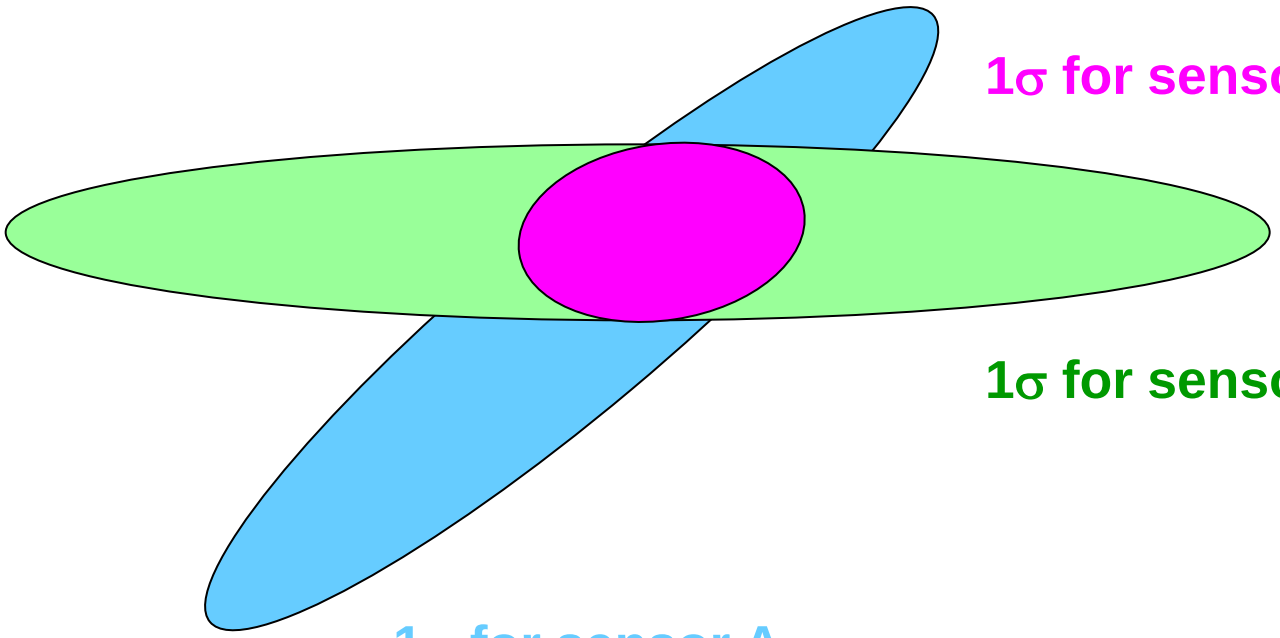
$$\mathbf{R} = \begin{bmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_\alpha^2 & 0 \\ 0 & 0 & \sigma_\varepsilon^2 \end{bmatrix}$$

- 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 covariances

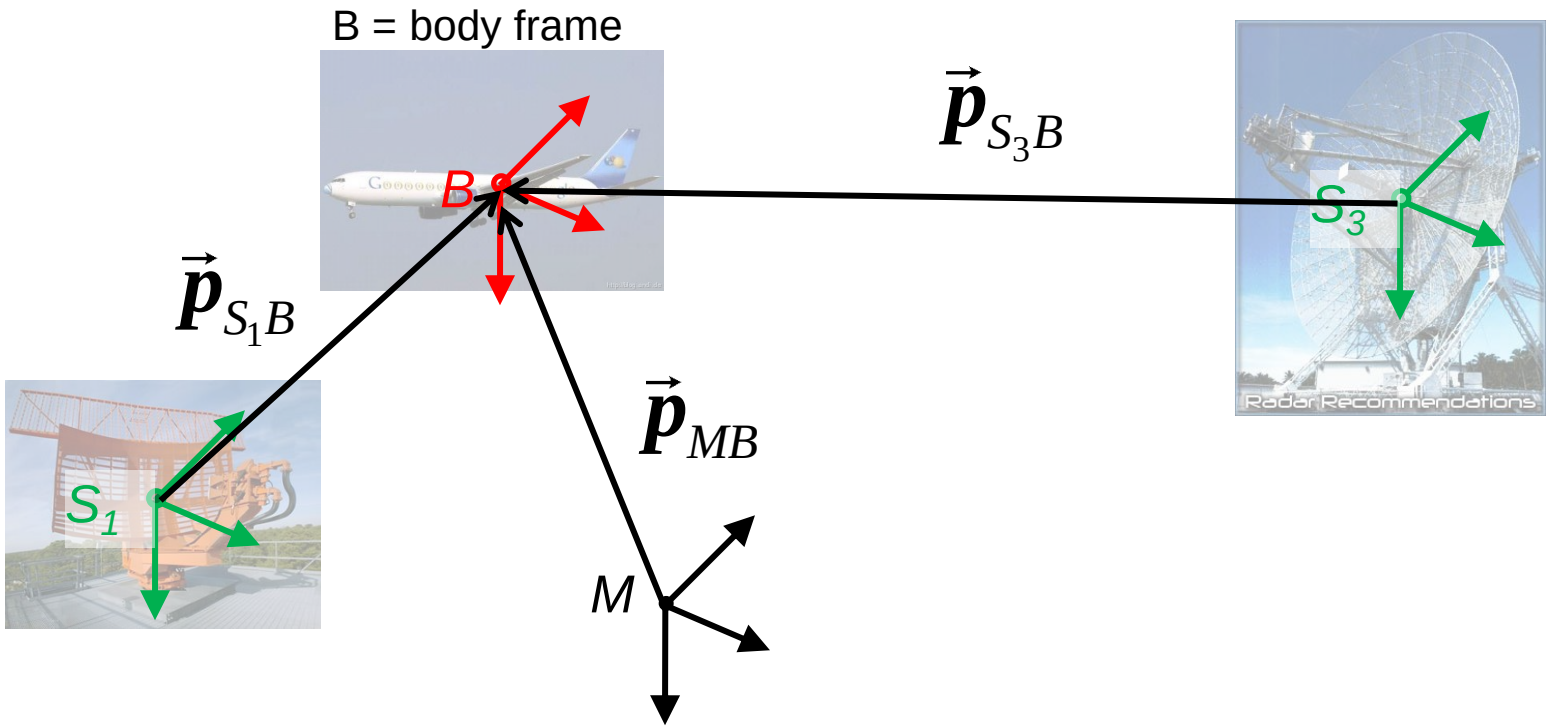
Multi-sensor fusion

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



Reference frame for multi-sensor fusion

- Data measured in different coordinate frames S1 and S3
- 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)

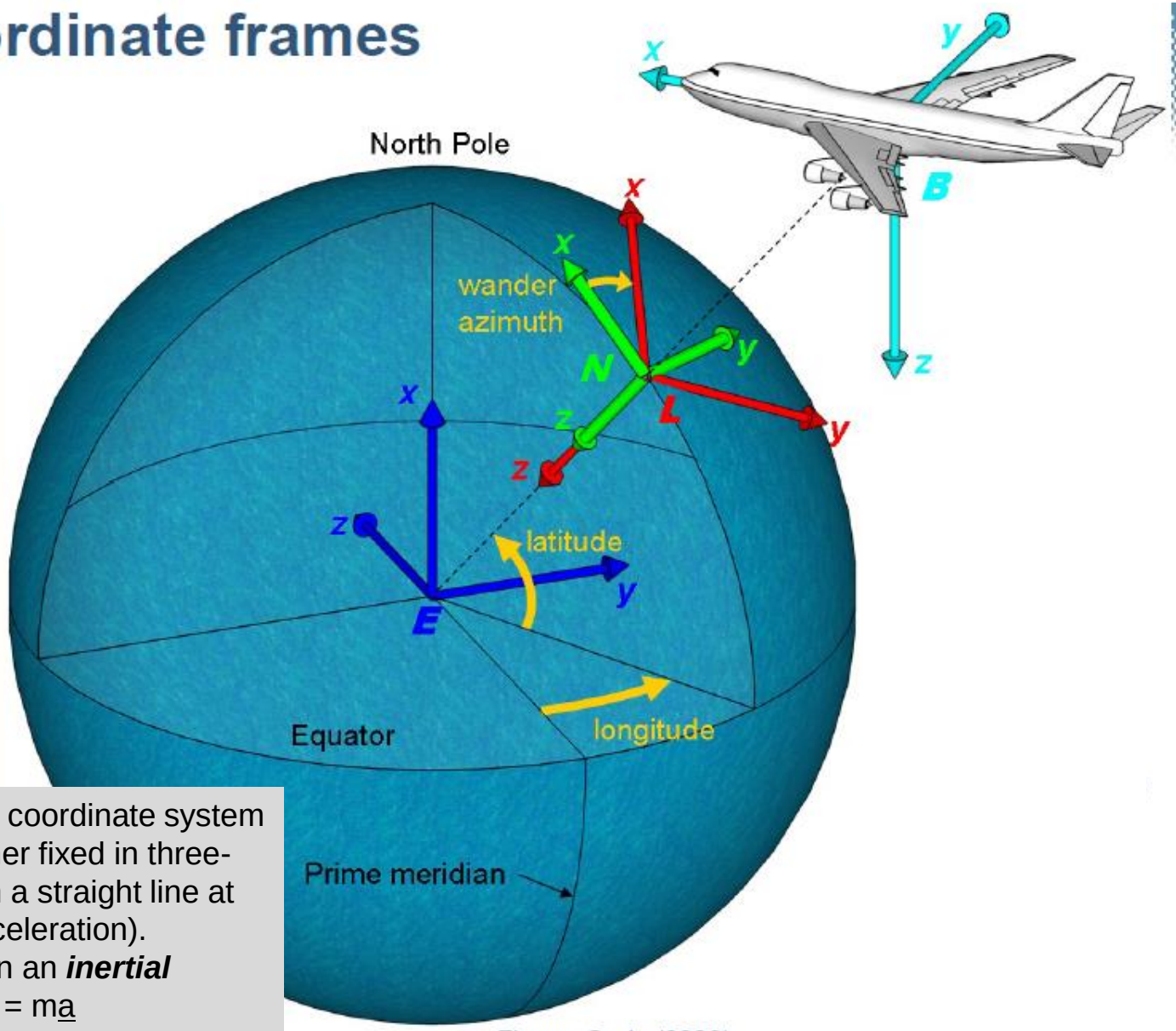
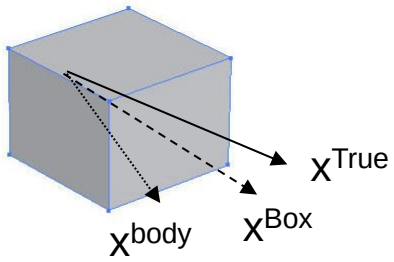


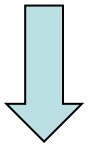
Figure: Gade (2008)

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

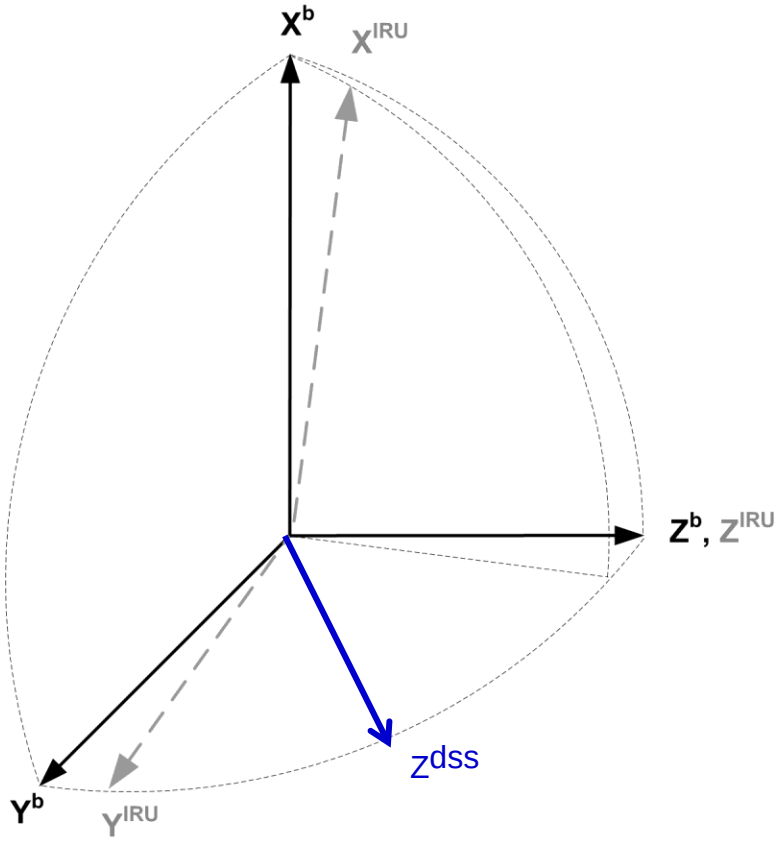


Alignment

- Mounting errors (internal and external) are present when **multiple sensor** are installed in a vehicle.
- Sensors can also be mounted in different directions with purpose.



- A calibration to re-align the sensors in software can be required



Synchronization

- Combining data (data fusion) from several sensors/instruments usually give a lower measurement uncertainty (and increased system robustness). However, such a data fusion requires the data to be **synchronized in time**.
- If simultaneous sampling of the sensors is not possible, which is often the case when multiples standalone instruments are used, time stamping of the data can be used.
- If every data set or data sample are tagged with an accurate time the different data samples can be aligned in time, for instance using interpolation.
- The GPS system provides easy access to a very accurate time base globally. GPS time is the atomic time scale implemented by the atomic clocks in the GPS ground control stations and the GPS satellites themselves.

System of linear equations I

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

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

- 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)

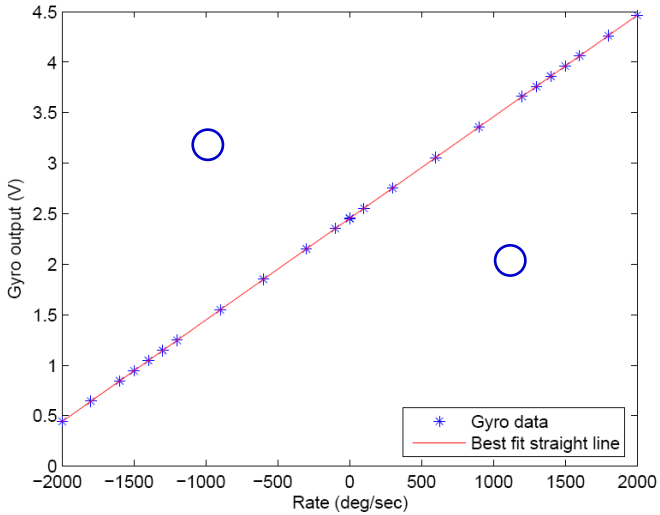
- In the more common case, there are more measurements (equations) than unknown (m > n). This is called an over determined systems. Then, a Least squares method can be used to estimate the unknown parameters.

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:

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

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

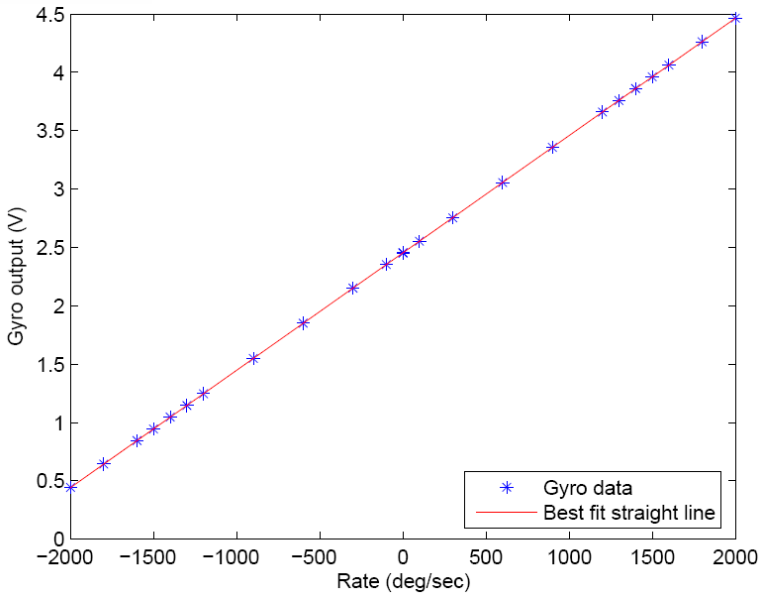
This is a batch estimator - all the measurements are processed at the same time. This means that the dimension of the estimation problem increases for each new measurement sample.

Memory hungry!

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

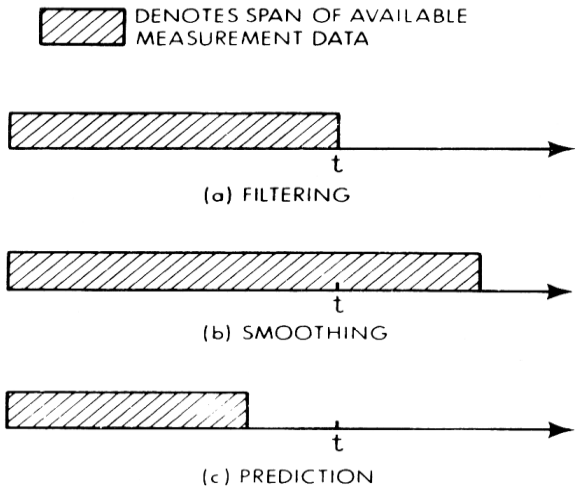
$$\hat{x}(k+1) = \hat{x}(k) + W(k+1)[z(k+1) - H(k+1)\hat{x}(k)]$$

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

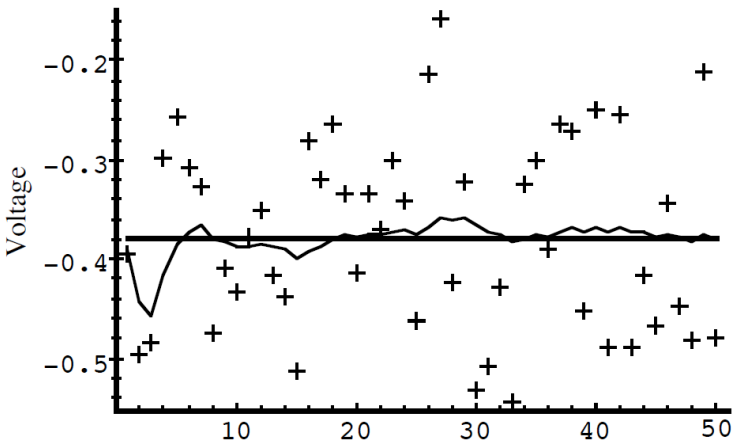
- In estimation the term **filtering** refers to estimating parameters (the state vector) describing the system at the current time, based on all past measurements. So, filtering in estimation theory includes more than just filtering noise, such as a low pass filter.

- 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

- The Kalman filter, introduced by Kalman (1960), has been one of the most widely used estimation algorithms
- 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 relevant sensors
  - ***Mathematical model*** of the system (describing how the different states depend on each other, and how the measurements depend on the states)

- 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

Linear system

The discrete state space representation of a linear system is given by:

$$\mathbf{x}_{k+1} = \Phi_k \mathbf{x}_k + \Lambda_k \mathbf{u}_k + \Gamma_k \mathbf{w}_k \tag{6.1}$$

$$\mathbf{z}_k = H_k \mathbf{x}_k + \mathbf{v}_k \tag{6.2}$$

Equation (6.1) is called the system dynamics equation or process equation. Φ_k is called the state transition matrix, and relates the state at the previous time step k to the state at the current step $k+1$. Λ is the control input matrix, $\mathbf{u}$ is a deterministic control input, Γ is the process noise matrix and $\mathbf{w}$ is the process noise, assumed to be zero-mean, white (uncorrelated), and with a Gaussian probability distribution. The index on the matrices Φ, Λ, Γ and H indicates that they might be time-variant.

Equation (6.2) is called the measurement model. $\mathbf{z}$ is the measurement vector, H is the measurement matrix and $\mathbf{v}$ is the measurement noise, assumed to be zero-mean, Gaussian white noise. The process noise and the measurement noise are also assumed to be independent/uncorrelated.

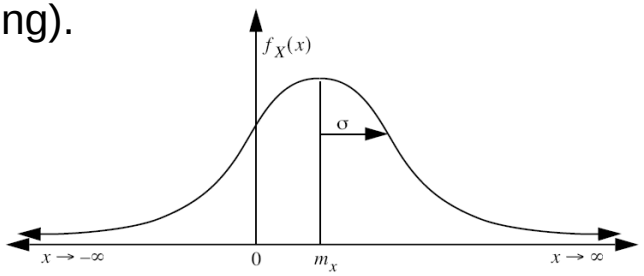
Kalman filter

Model (dynamic system):

$$\mathbf{x}_{k+1} = \Phi_k \mathbf{x}_k + \Lambda_k \mathbf{u}_k + \Gamma_k \mathbf{w}_k \quad \text{Eq.1}$$

$$\mathbf{z}_k = H_k \mathbf{x}_k + \mathbf{v}_k \quad \text{Eq.2}$$

- Eq.1 : a mathematical model of how the states **x** varies with time; called the Process Model
- Eq.2: Measurement model
- Only simple statistical information needed: mean and standard deviation.
- LS vs. Kalman filter
  - LS used for nonrandom (deterministic), time-invariant parameters.
  - The Kalman filter is used for random parameters (which can be time varying).



KF equations (in discrete time):

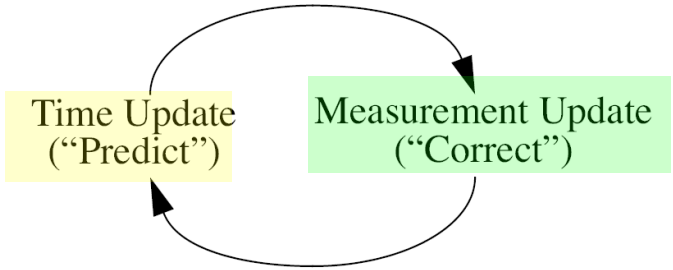
$$\bar{\mathbf{x}}_{k+1} = \Phi_k \hat{\mathbf{x}}_k + \Lambda_k \mathbf{u}_k$$

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

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

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

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



- Q is the process noise covariance matrix that represent the uncertainty in the process model in Eq. 1
- $\bar{x}$ is predicted value
- $\hat{x}$ is updated value based on measurements
- R is the measurement covariance matrix ("accuracy of the measurements")
- P is the covariance matrix of the estimate ("accuracy of the estimate")

KF equation – basic concepts

- The ratio between Q and R is important!
- Large R → small gain K
- Small R and large Q → large gain K
- Large gain K means a rapid response to the measurements
- Small gain K means a slower response to the measurements
- (z – Hx) should be zero mean and white noise (if not, the model is wrong)

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

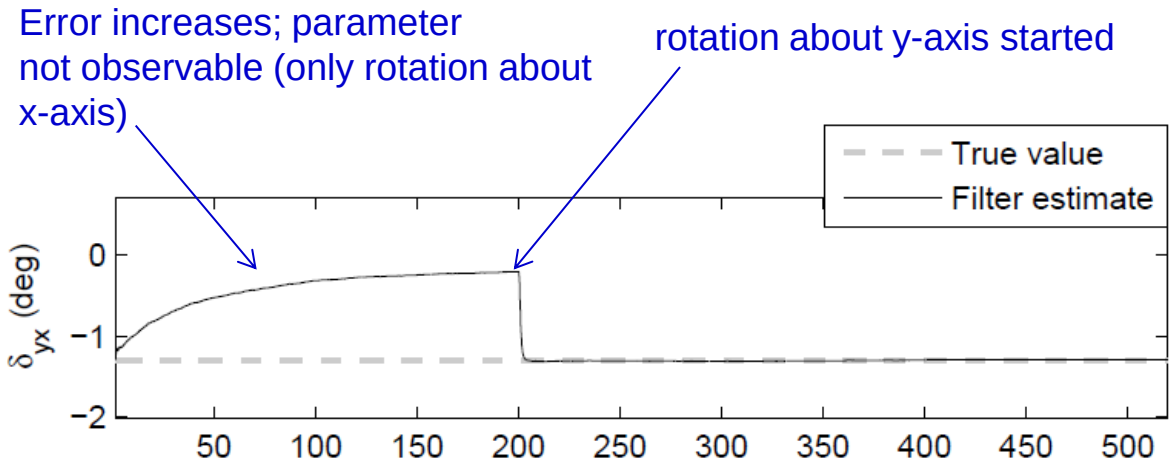
$$K_k = \bar{P}_k H_k^T (H_k \bar{P}_k H_k^T + R_k)^{-1}$$
$$\hat{\mathbf{x}}_k = \bar{\mathbf{x}}_k + K_k(\mathbf{z}_k - H_k \bar{\mathbf{x}}_k)$$
$$\hat{P}_k = (I - K_k H_k)\bar{P}_k$$

Predicted measurement

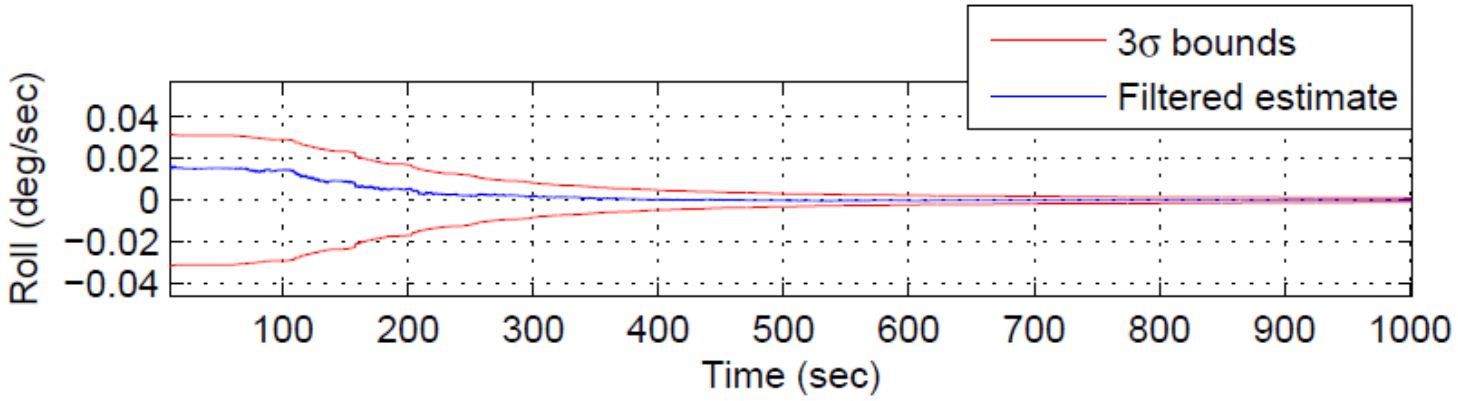
Need initial values x_0 and P_0

You should understand these basic concepts of the Kalman filter!

Estimation error vs. time for rate gyro (simulation examples)



Rate gyro calibration using KF; misalignment



KF spacecraft attitude estimator

KF equations - implementation

- The equations becomes very simple in the case of only one time-invariant unknown.
  - E.g. a single sensor axis
  - All matrices (P, H, K etc.) becomes scalar values
  - Simple to implement on an embedded processor.

- However, most real-world problems includes several unknown parameters/states to be estimated, and several of them are usually time-varying.

the major computational burden in the Kalman filter

$$
\begin{aligned}
\bar{\mathbf{x}}_{k+1} &= \Phi_k \hat{\mathbf{x}}_k + \Lambda_k \mathbf{u}_k \\
\bar{P}_{k+1} &= \Phi_k \hat{P}_k \Phi_k^T + \Gamma_k Q_k \Gamma_k^T \\
K_k &= \bar{P}_k H_k^T (H_k \bar{P}_k H_k^T + R_k)^{-1} \\
\hat{\mathbf{x}}_k &= \bar{\mathbf{x}}_k + K_k(\mathbf{z}_k - H_k \bar{\mathbf{x}}_k) \\
\hat{P}_k &= (I - K_k H_k)\bar{P}_k
\end{aligned}
$$

For a **time-invariant system** (constant H, Φ, Q, Γ and R) the covariance P and the gain K will reach a steady-state value quickly.
A **Steady state** solution of KF equations is easier to compute in real-time

- P and K fixed, and pre-computed.
- This is a sub-optimal solution

KF equations – implementation II

- The computational requirements are (usually) proportional to n^3, where n is the number of states in the filter
  - compare $[p_x\ p_y\ p_z\ v_x\ v_y\ v_z]^T$ vs. $[p_x\ p_y\ p_z\ v_x\ v_y\ v_z\ a_x\ a_y\ a_z\ b_x\ b_y\ b_z]^T$
- Can get numerical issues on less powerful hardware (less than 32 bits) using fixed point.
- A powerful floating point processor/DSP is one way to reduce the impact of computational round off errors
  - Remember that your PC use 64 bits (double precision) floating point

KalmanFilterforDummies

Example: KF for 1D simple linear gyro model

- Process model – unknown constants:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k \Rightarrow \boldsymbol{\varphi} = \mathbf{I}$$

- Measurement model:

$$z(\omega) = a \cdot \omega + b \xrightarrow{\text{matrix equation}} z = H\boldsymbol{x} \qquad H = [\omega \;\; 1]\, \boldsymbol{x} = \begin{bmatrix} a \\ b \end{bmatrix}$$

- Assume a perfect model; no process noise: $Q = \sigma^2 = 0$
- Measurement covariance matrix becomes a scalar value in 1D: $R = \sigma_v^2$
- Typically we do not use Q = 0; we always add a small amount of noise since our model is «never» perfect.

Estimation in nonlinear systems

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}_k(\mathbf{x}_k, \mathbf{u}_k) + \Gamma_k \mathbf{w}_k \\ \mathbf{z}_k &= \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k \end{aligned}$$

Non-linear functions

Noise

- 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

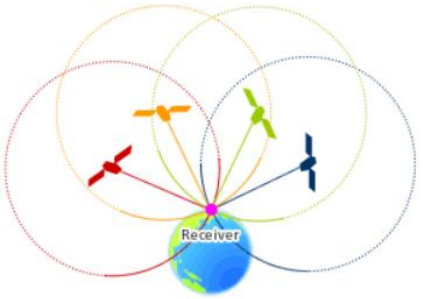
- The measured pseudorange $\tilde{P}^k$ from a satellite k can be expressed as (since we can assume no clock error in the satellite):

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

- $c\tau$ = 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.



Calibration

- Calibration is the process of comparing instrument outputs with known reference information and determining coefficients that force the output to agree with the reference information over a range of output values.

- An accurate calibration can turn low accuracy sensors into a sensor suitable for higher accuracy measurements.

- The first step in the calibration process is to make a mathematical model of the sensor.

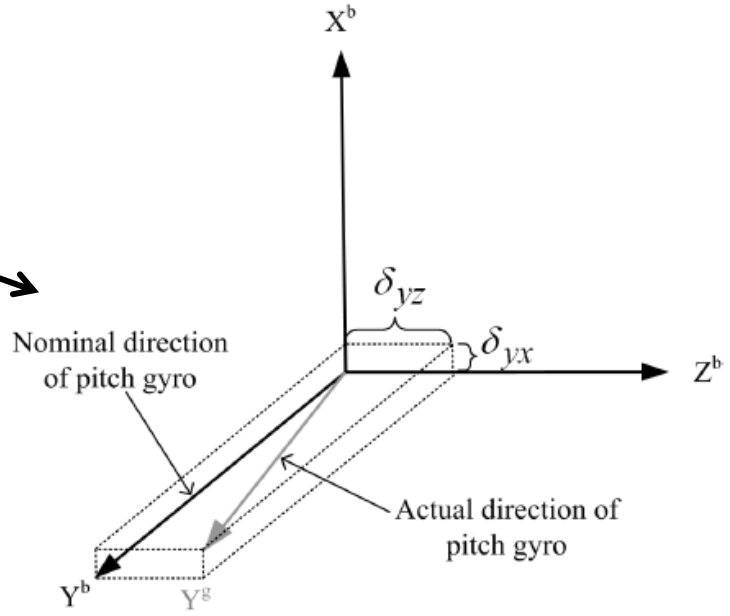
Common sensor errors

- Sensor bias (offset)
  - If the output signal is not zero when the measured property is zero, the sensor has a bias (offset).
- Sensor scale factor/sensitivity
  - Can be linear (y = ax + b; where a = sensitivity)
  - Can be non-linear
- Sensor misalignment
- Sensor Noise

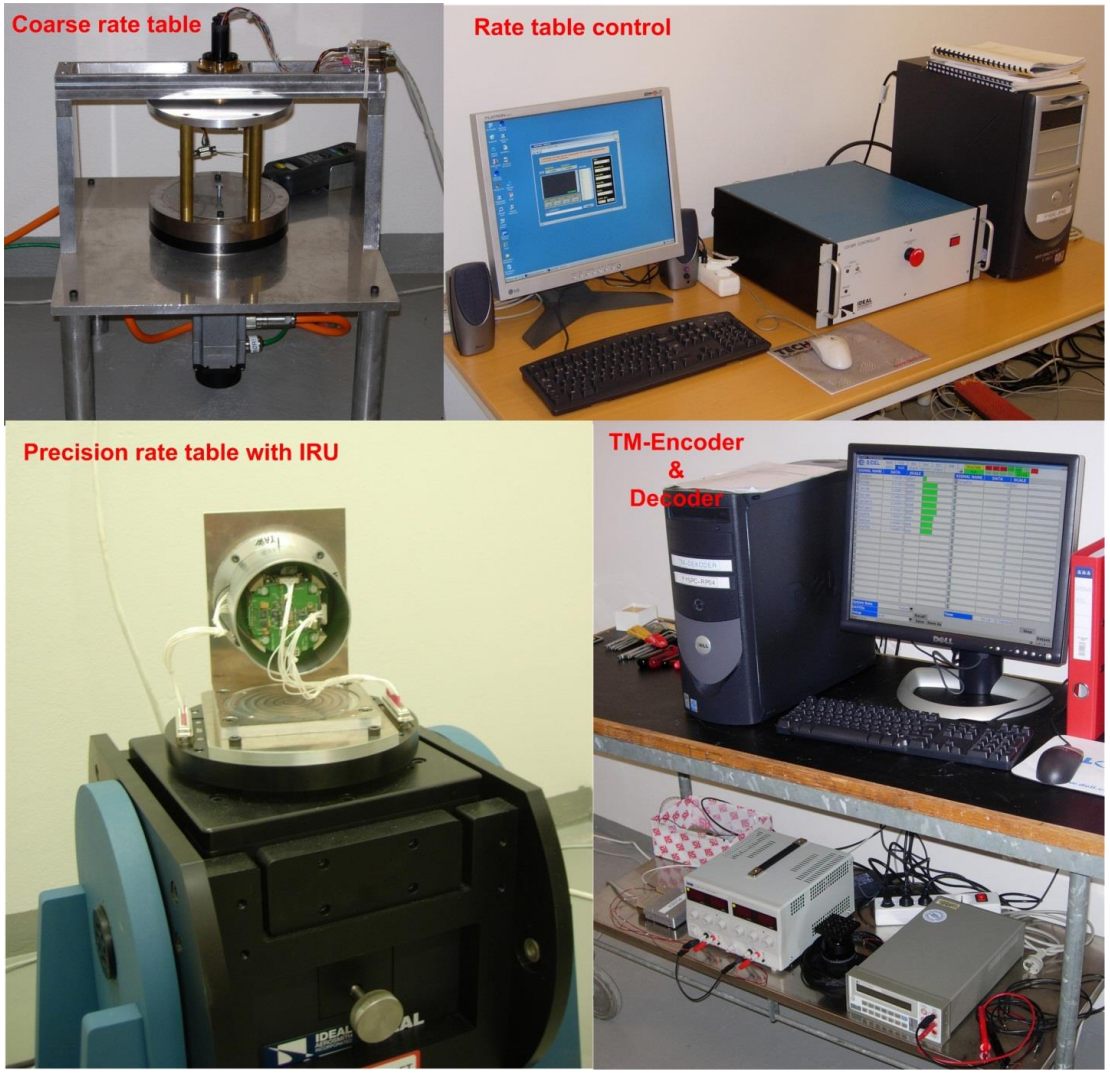
Digital level to physical unit:

[(DL * LSB) – offset] / (A * SF)

DL = digital level, A = gain, SF = scale factor,
LSB = Vref/2^n, n = number of bits

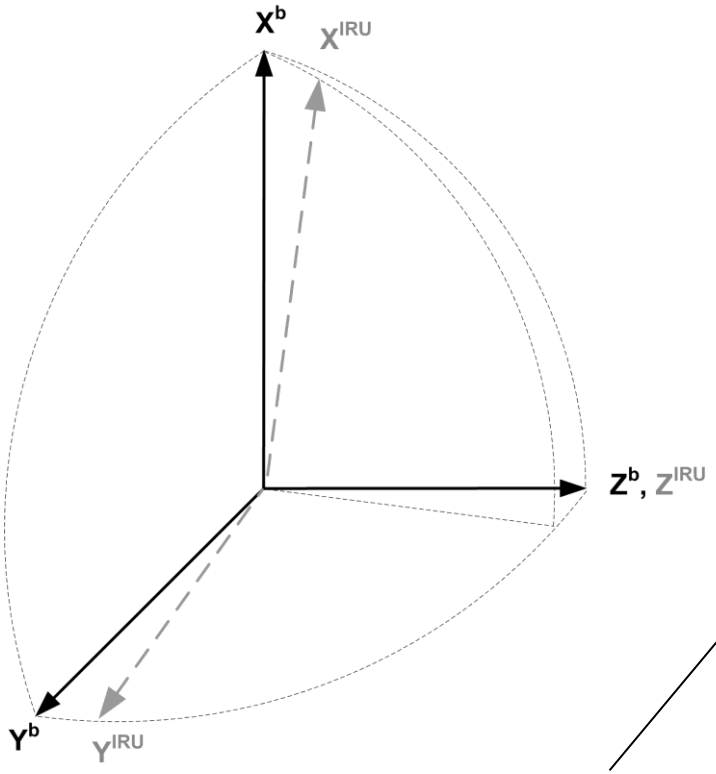


Rate gyro calibration and test facility UIO



Rate gyro sensor modelling

The sensor models include biases, misalignments and scale factor errors.



rate gyro angular rate measurement

$$\underbrace{\boldsymbol{\omega}^b}_{\text{3x1 vector true rate}} = \underbrace{S_g}_{\text{3x3 misalig. \& scale factor error matrix}}\underbrace{\tilde{\boldsymbol{\omega}}}_{\text{rate gyro angular rate measurement}} - \underbrace{\boldsymbol{\beta}^b}_{\text{Bias, 3x1 vector}} - \underbrace{\boldsymbol{\eta}_v}_{\text{Noise, 3x1 vector}}$$

$$S = I + M = \begin{bmatrix} 1+\lambda_x & \delta_{xy} & \delta_{xz} \\ \delta_{yx} & 1+\lambda_y & \delta_{yz} \\ \delta_{zx} & \delta_{zy} & 1+\lambda_z \end{bmatrix}$$

Gyro modelling – 3D model

The true angular rate ω^b in the body frame is related to the measured angular rate $\tilde{\omega}$ in the gyro frame, by

$$\omega^b = S\tilde{\omega} - \beta^b - \eta_v \tag{9.1}$$

where β^b is gyro bias (in the body frame) and η_v is zero-mean, Gaussian white noise. The measured angular rate (in deg/sec) is given by

$$\tilde{\omega}_i = \frac{(\tilde{E}_i - O_i)}{K_{0_i}} \tag{9.2}$$

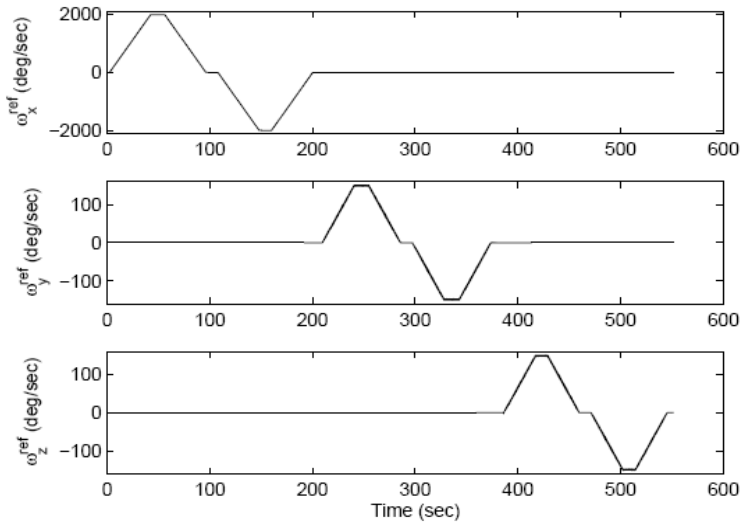
where $\tilde{E}_i$ is the raw measurement from the i'th gyro in engineering unit (V), O_i is the null offset value for the i'th gyro in engineering unit (V) and K_{0_i} is the nominal scale factor of the i'th gyro (found from the sensor data sheet

Real-time systems often only model and estimate the gyro bias due to limited processing power

S = I (identity matrix)

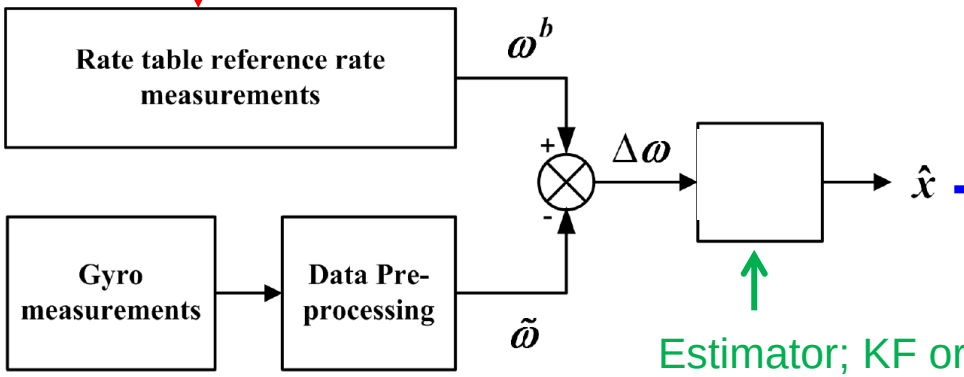
3D rate gyro calibration

Calibration maneouvre applied to the rate table



(Some) calibration parameters can be estimated “outside” the embedded system, such that the correction parameters only are applied directly in the real-time system

$$\omega^b = S_g\tilde{\omega} - \beta^b$$



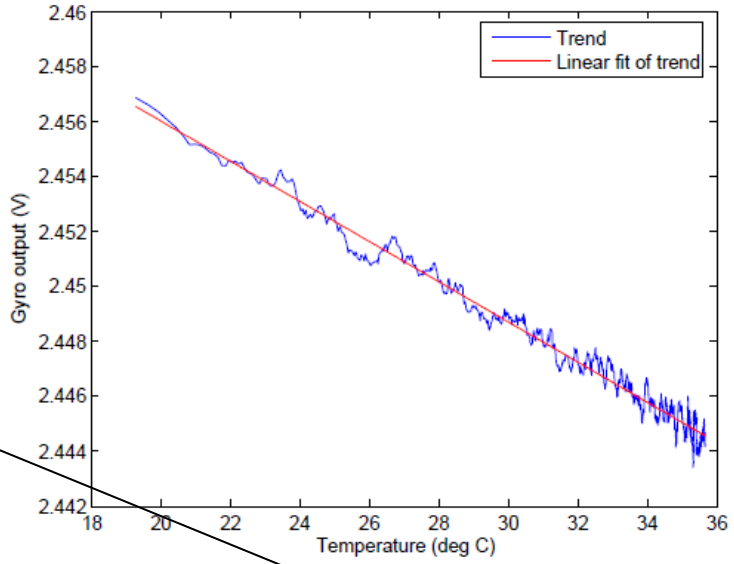
Parameter	Estimated value
δ_{xy}	0.53 deg
δ_{xz}	0.018 deg
δ_{yx}	0.29 deg
δ_{yz}	-0.60 deg
δ_{zx}	0.32 deg
δ_{zy}	0.84 deg
λ_x	-1.22 x 10^{-2}
λ_y	-6.77 x 10^{-3}
λ_z	-7.55 x 10^{-3}
β_x	-0.51 deg/sec
β_y	-9.2 x 10^{-3} deg/sec
β_z	-1.25 x 10^{-2} deg/sec

Temperature compensation

- The following linear model is used to model the offset temperature dependence:

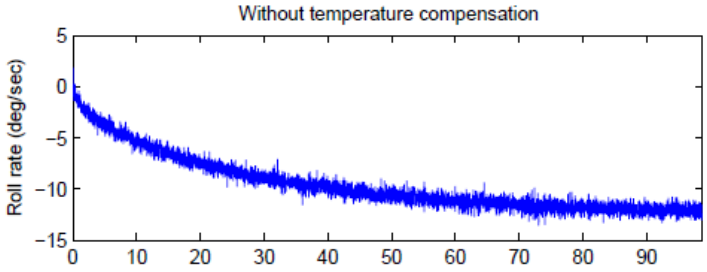
$$O(T) = O_0 + a_T \Delta T$$

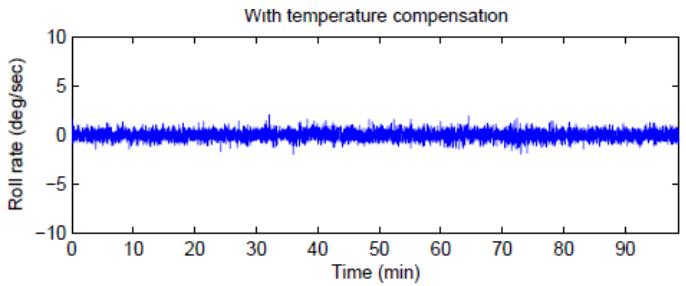
- a_T is the temperature sensitivity coefficient and ΔT is the temperature change



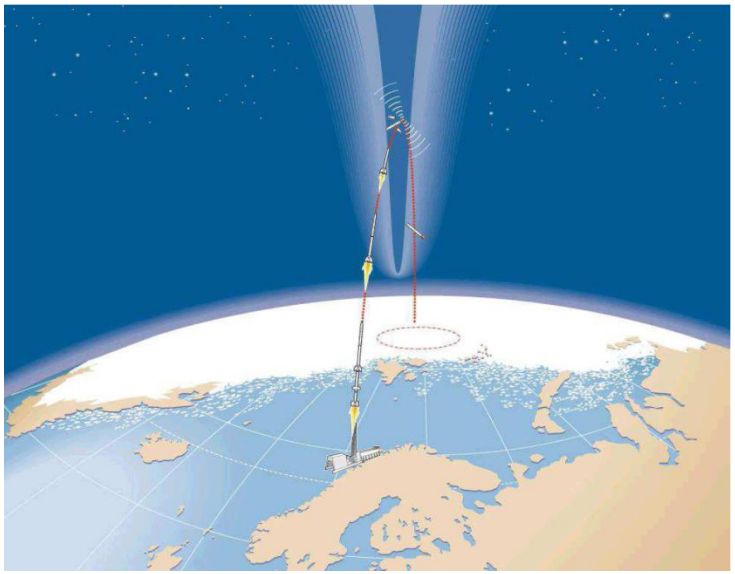
$$\tilde{\omega}_i = \frac{(\tilde{E}_i - O_i)}{K_{0_i}}$$

The gyro is not rotating in this test (Roll rate should be zero)!

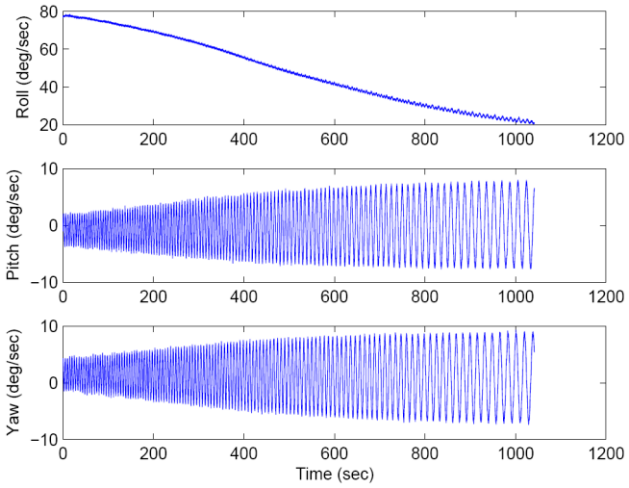


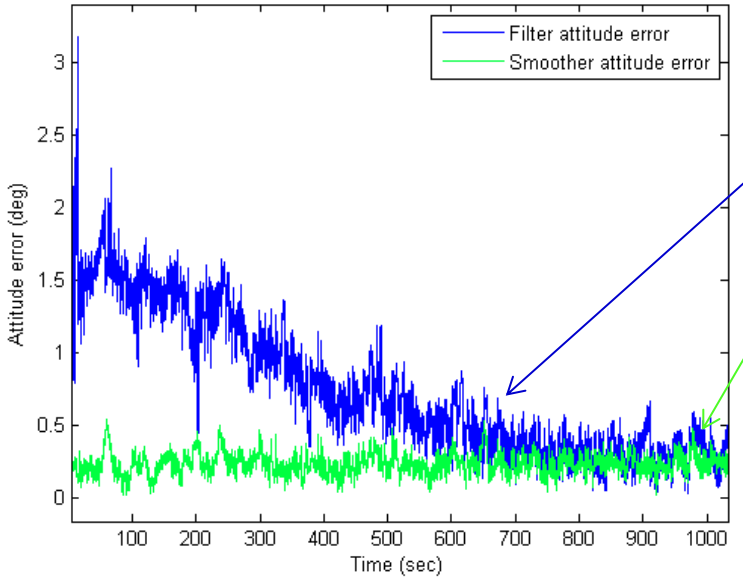


Attitude determination system simulations



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!

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!

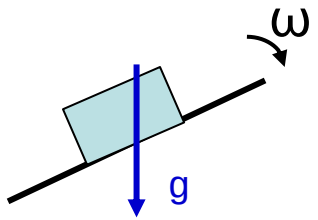
Note: x[n] is the measurement at time n
y[n] is the filter output at time n

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.

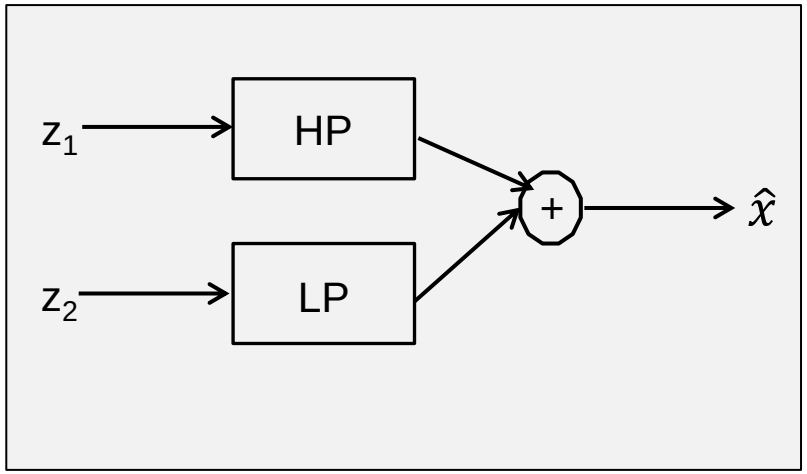
- **The idea behind the complementary filter is to take slow moving signals and fast moving signals and combine them.**
  - **The filter is based on an analysis in the frequency domain.**

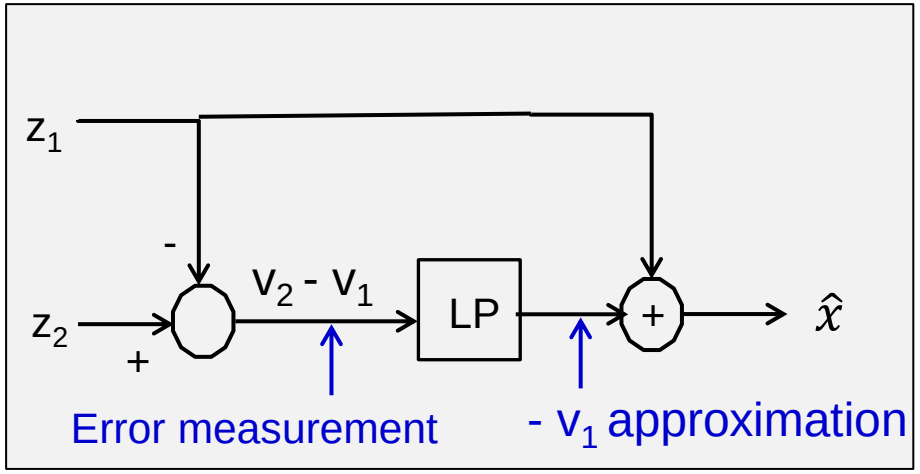
- The complementary filter fuses the sensor1 and sensor2 data by passing the former through a 1st-order low pass and the latter through a 1st-order high pass filter and adding the outputs.

- 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

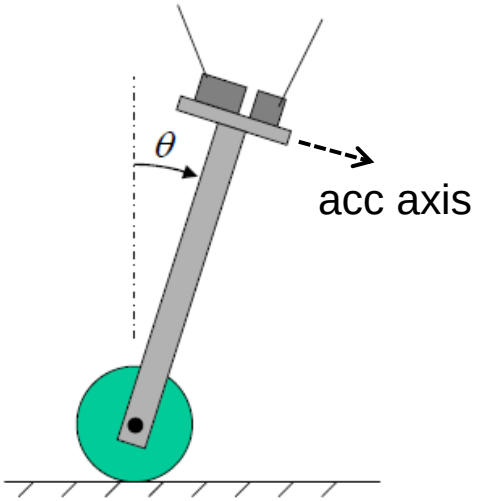
Complementary filter - balance robot

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

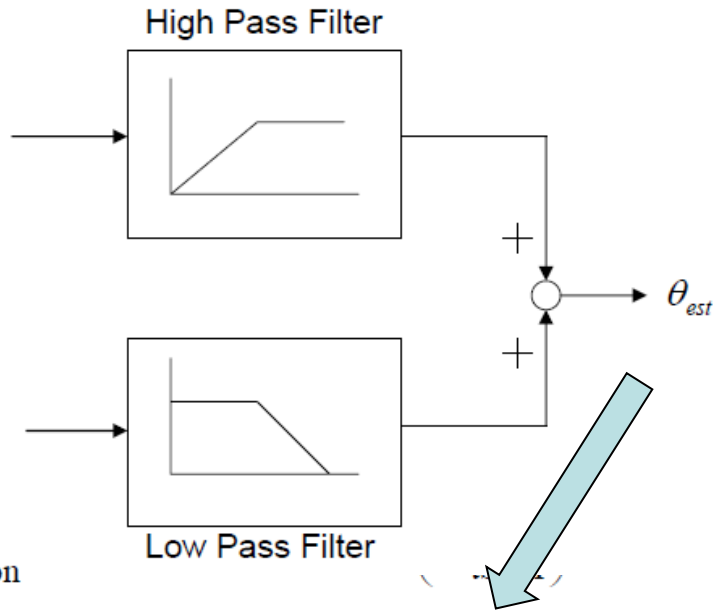


$\theta \approx \int (\text{angular rate})\, dt$

- 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

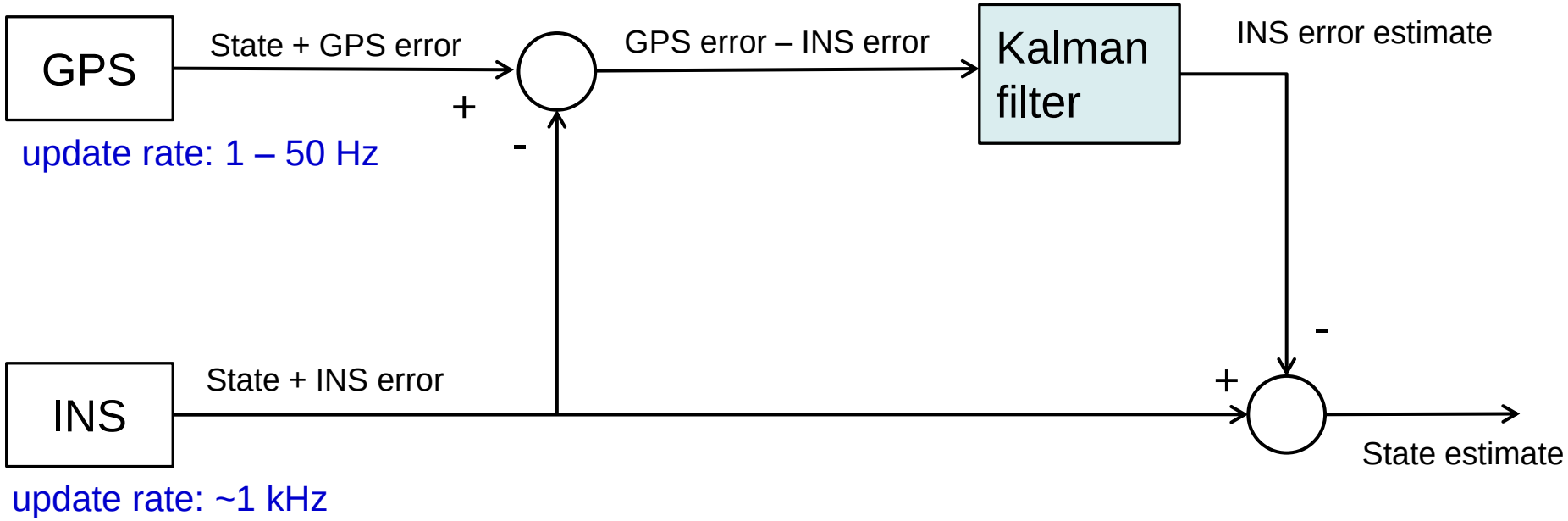


$$\hat{\theta}_k = \alpha(\theta_{k-1} + \omega_k \Delta t) + (1-\alpha)a_k$$

Example: α = 0.98

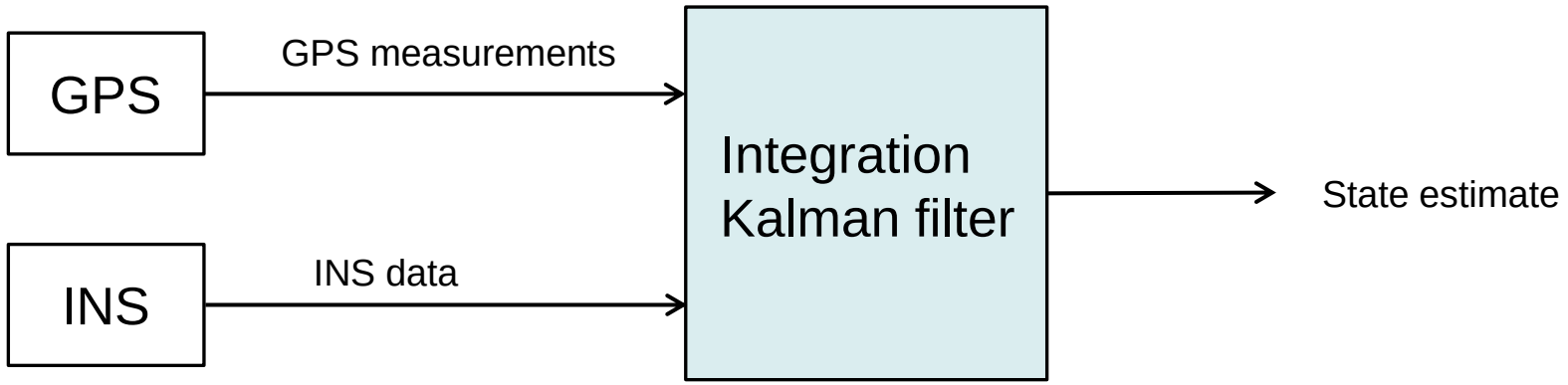
To limit gyro drift

Complimentary INS/GPS integration with KF



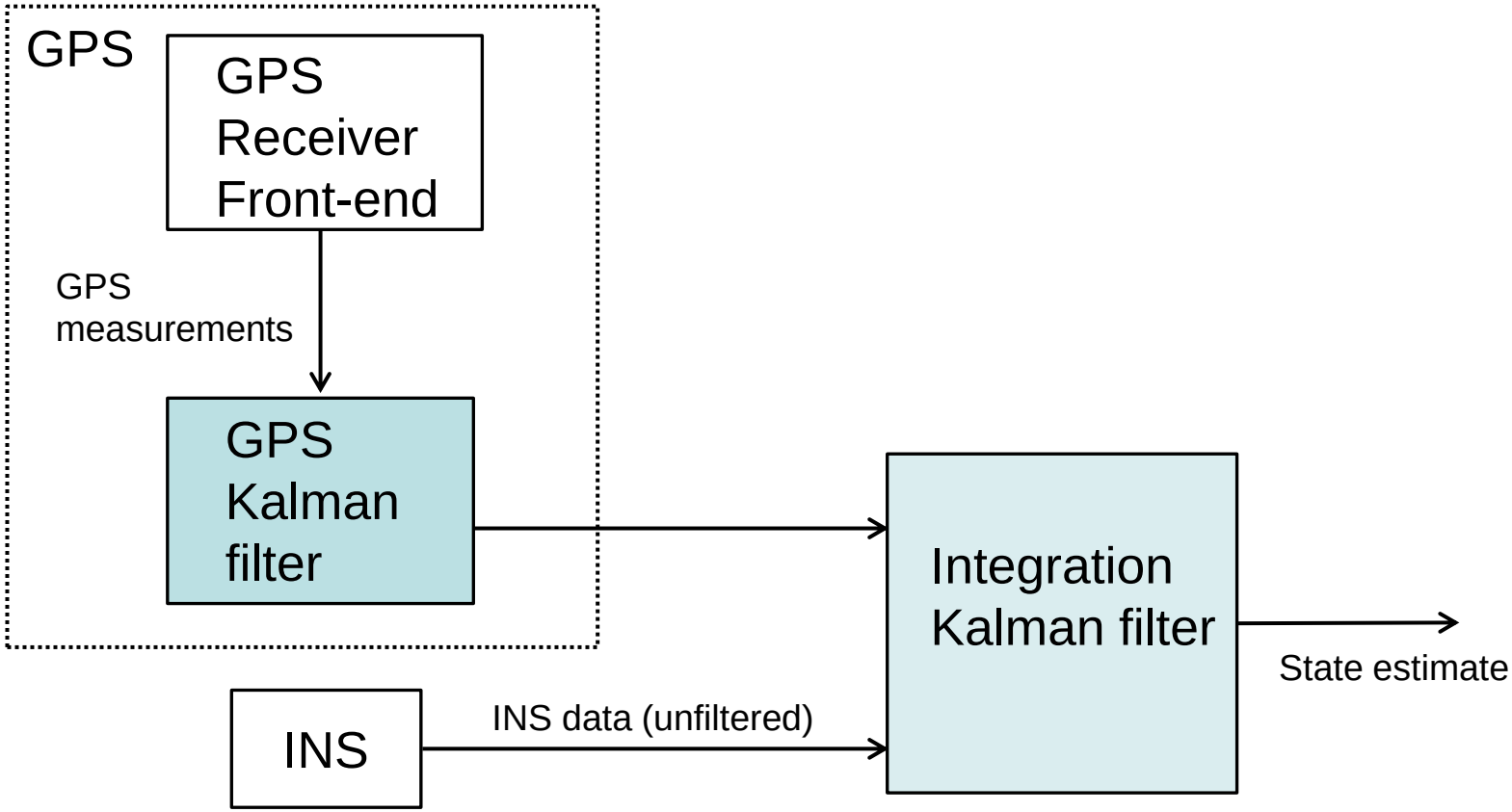
A very common INS/GPS integration

Centralized INS/GPS integration



Theoretically optimal integration, but heavy computational burden and poor fault tolerance.

Distributed INS/GPS integration



- Two filters run in parallel
- The integration filter does the combination/data fusion
- Simpler approach than the centralized solution