



UiO : **University of Oslo**

FYS3240- 4240

Data acquisition & control

Data fusion & estimation

Spring 2019 – Lecture #12

Reading: Articles by Maybeck and Welch & Bishop



Bekkeng 28.12.2018

Readings

Chapter 1, "Introduction" from STOCHASTIC MODELS, ESTIMATION, AND CONTROL, Volume 1, by Peter S. Maybeck, copyright © 1979 by Academic Press, reproduced by permission of the publisher. All rights of reproduction in any form reserved.

Stochastic models, estimation, and control VOLUME 1

PETER S. MAYBECK
DEPARTMENT OF ELECTRICAL ENGINEERING
AIR FORCE INSTITUTE OF TECHNOLOGY
WRIGHT-PATTERSON AIR FORCE BASE
OHIO

An Introduction to the Kalman Filter

by
Greg Welch¹
and
Gary Bishop²

TR 95-041
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175

Abstract

In 1960, R.E. Kalman published his famous paper describing a recursive solution to the discrete-data linear filtering problem. Since that time, due in large part to advances in digital computing, the Kalman filter has been the subject of extensive research and application, particularly in the area of autonomous or assisted navigation.

The Kalman filter is a set of mathematical operations that provides an efficient computational (recursive) solution of the least-squares estimation problem. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown.

The purpose of this paper is to provide a practical introduction to the discrete Kalman filter. This introduction includes a description and some discussion of the basic discrete Kalman filter, a derivation, description and some discussion of the extended Kalman filter, and a relatively simple (tangible) example with real numbers & results.

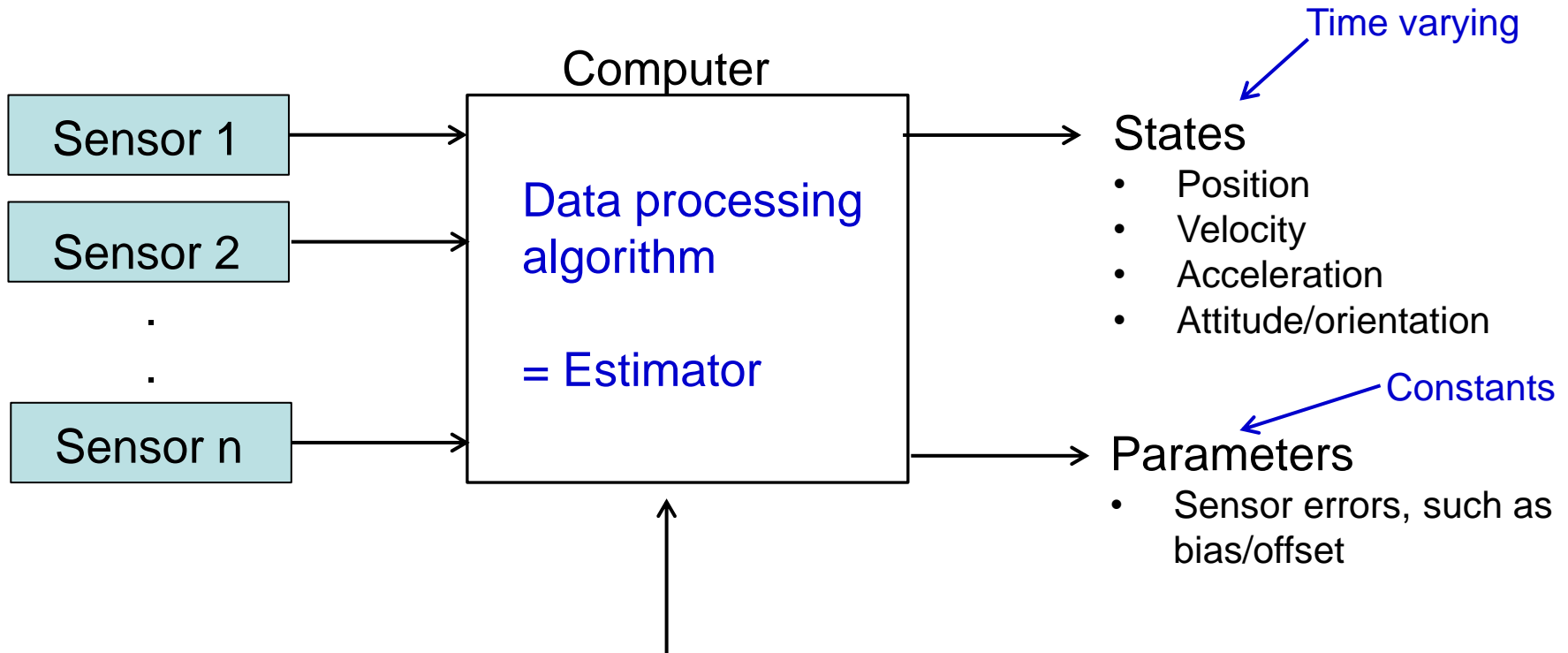
Section 1 and 3

Digital filter examples – for comparison

- 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

Data fusion in 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

GNC: Unmanned Aircraft System (UAS)

GNC : Guidance, Navigation and Control

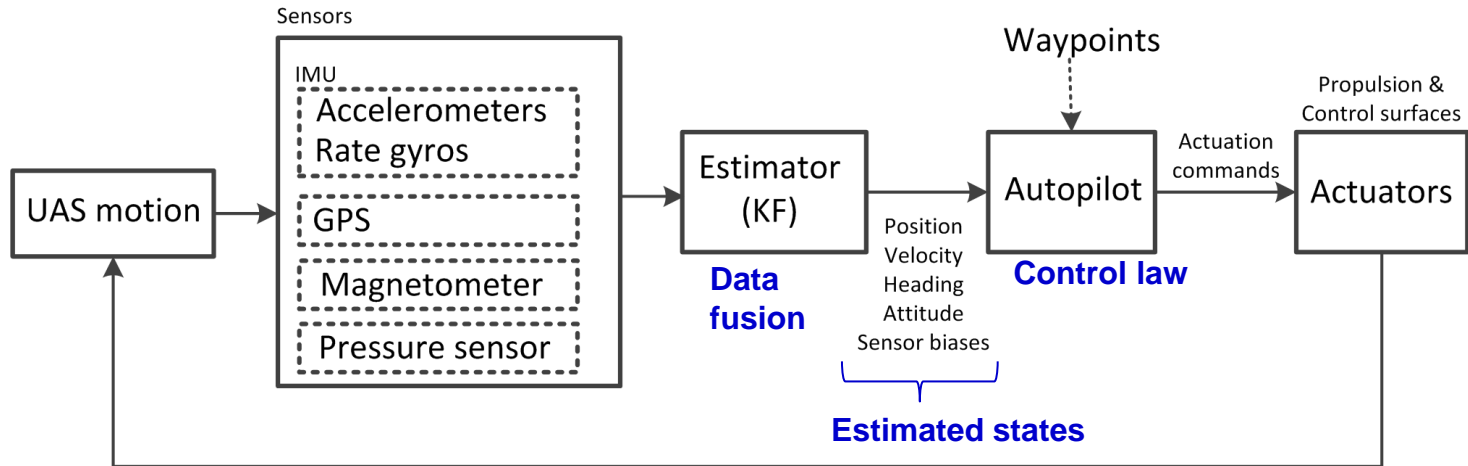
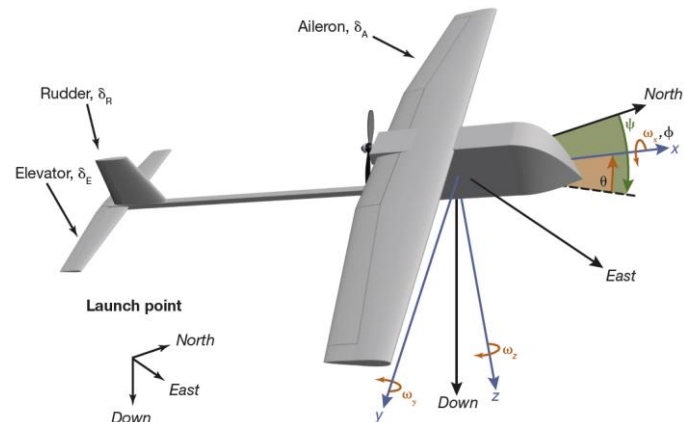


Illustration 5: Waypoints for flight over seattle 



Guidance, Navigation and Control

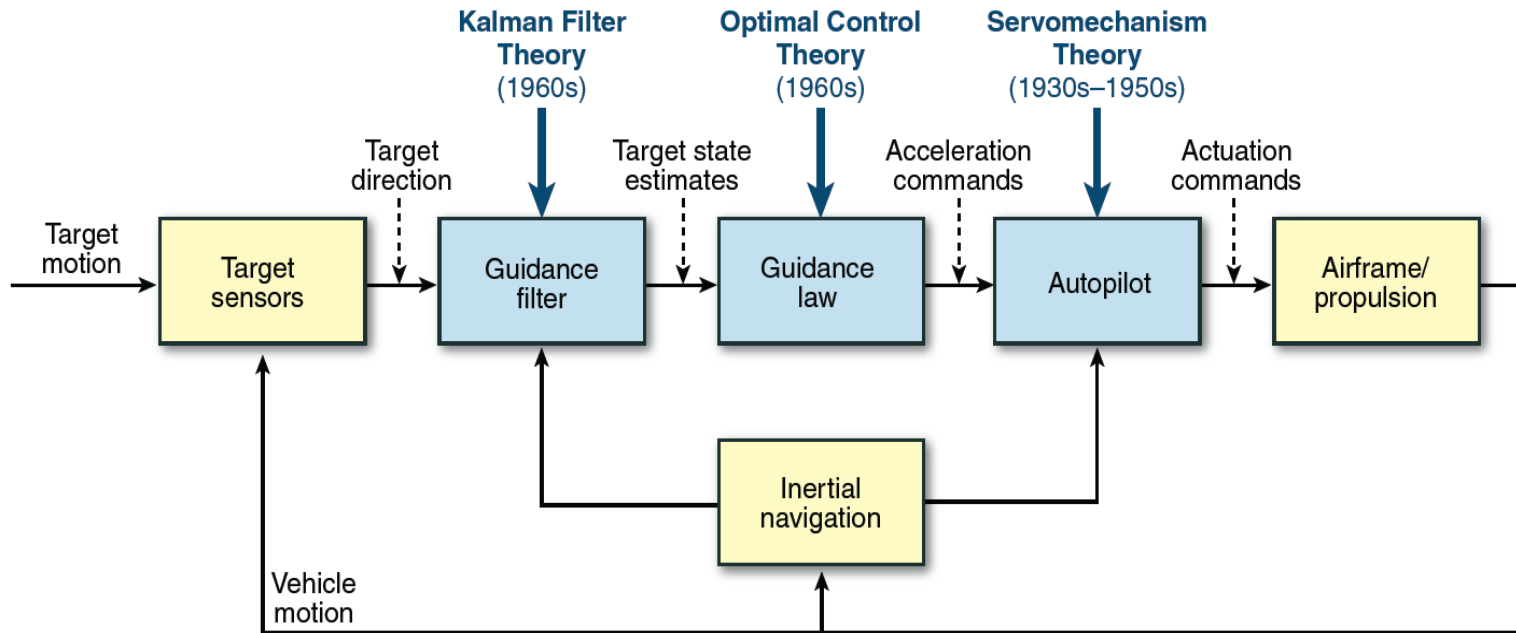


Figure 1. The traditional guidance, navigation, and control topology for a guided missile comprises guidance filter, guidance law, autopilot, and inertial navigation components. Each component may be synthesized by using a variety of techniques, the most popular of which are indicated here in blue text.

Figure from Palumbo, *Johns Hopkins APL Technical Digest*

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?

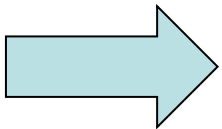
Two-sensor data fusion example II

- We assume that the measurement z_2 taken at time t_2 was taken a very short time after measurement z_1 taken at time t_1 , so that “nothing has changed”
- At time t_1 our best estimate of x is that $\hat{x}(t_1) = z_1$
- At time t_2 our best estimate of x is:

$$\hat{x}(t_2) = \left(\frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \right) z_1 + \left(\frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \right) z_2$$

- We can rewrite this equations as:

$$\begin{aligned} \hat{x}(t_2) &= [\sigma_{z_2}^2 / (\sigma_{z_1}^2 + \sigma_{z_2}^2)] z_1 + [\sigma_{z_1}^2 / (\sigma_{z_1}^2 + \sigma_{z_2}^2)] z_2 \\ &= z_1 + [\sigma_{z_1}^2 / (\sigma_{z_1}^2 + \sigma_{z_2}^2)] [z_2 - z_1] \end{aligned}$$



where

$$\hat{x}(t_2) = \hat{x}(t_1) + K(t_2)[z_2 - \hat{x}(t_1)]$$

$$K(t_2) = \sigma_{z_1}^2 / (\sigma_{z_1}^2 + \sigma_{z_2}^2)$$

Form used in the Kalman filter!

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.

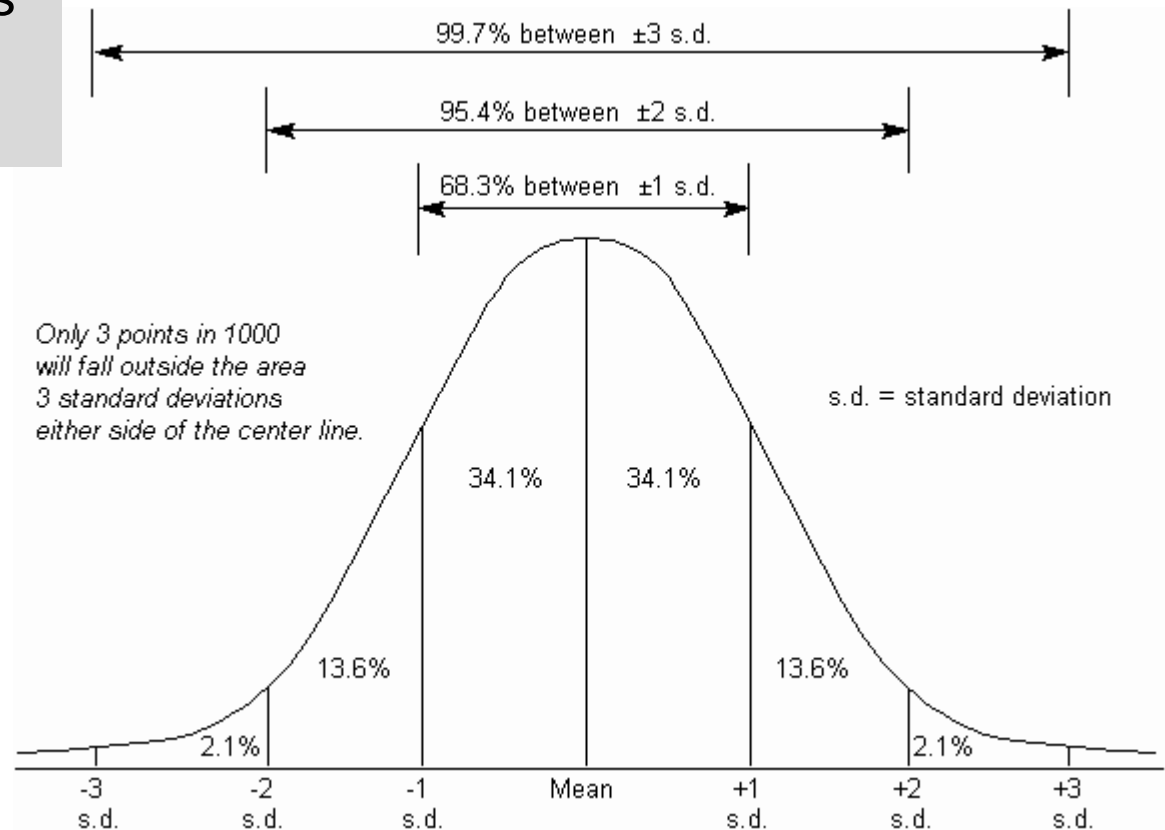
Filtering in Estimation

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

Standard deviation

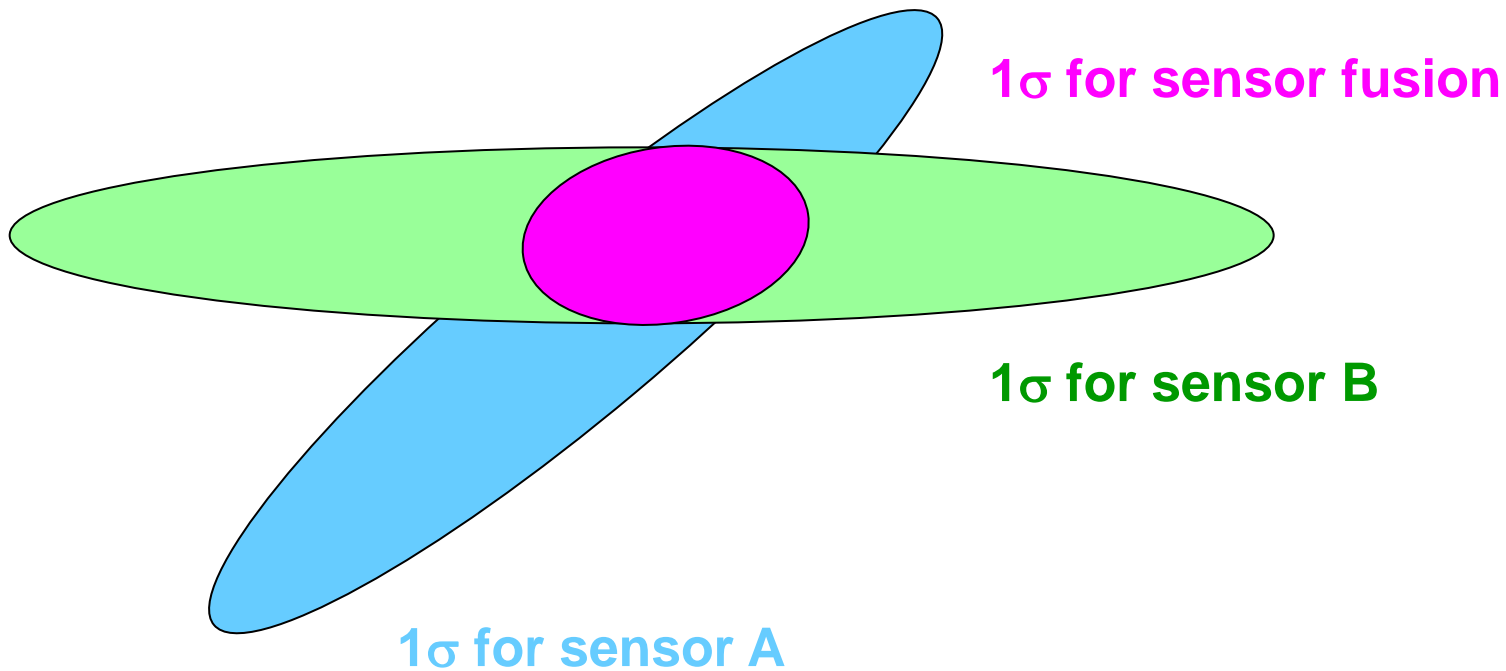
$$\sigma = s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

The standard deviation is the amount of variation from the mean



Multi-sensor data fusion

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



System of linear equations II

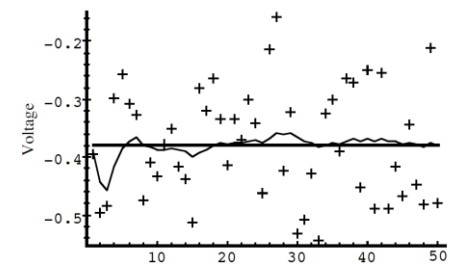
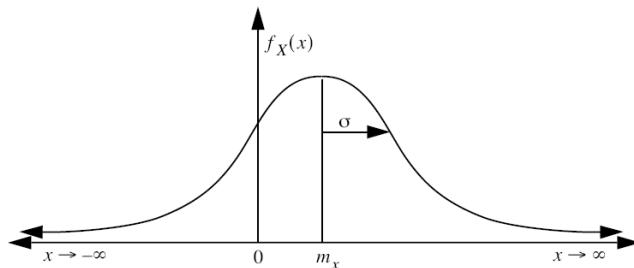
- If the number of measurements (number of equations) is equal to the number of unknowns x_i ($m = n$), the unknowns can be found from the inverse solution:

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

- 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.

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”.



Notation

\mathbf{x} : (column) vector of unknowns
 \mathbf{z} : measurement (column) vector
 \mathbf{H} : measurement matrix
 \mathbf{w} : process noise vector
 \mathbf{v} : measurement noise vector


\mathbf{R} : Measurement covariance (uncertainty) matrix
 \mathbf{Q} : Process covariance (uncertainty) matrix
 $\hat{\mathbf{x}}$: Estimate (solution)
 \mathbf{P} : Covariance (uncertainty) matrix of the estimate
 k : discrete time point / sample number

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix}$$

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

Kalman filter (KF) II

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

Process equation:
$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}$$

 deterministic input

Measurement equation:
$$z_k = Hx_k + v_k.$$

where the random variables w and v represent the process and measurement noise (respectively)

$$p(w) \sim N(0, Q)$$

$$p(v) \sim N(0, R)$$

KF: a predict-correct algorithm

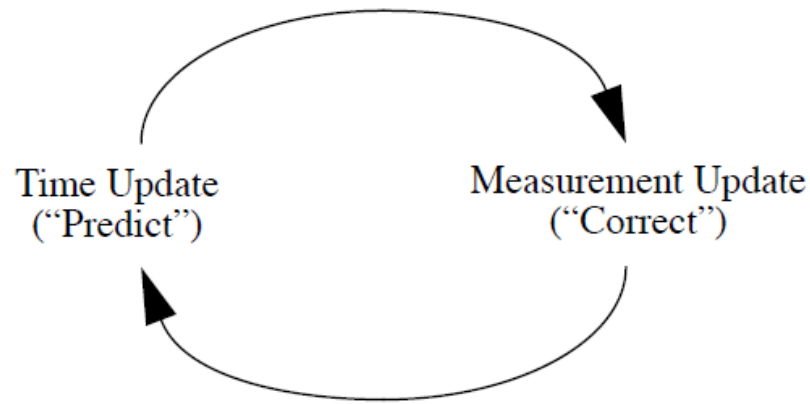


Figure 1-1. The ongoing discrete Kalman filter cycle. The *time update* projects the current state estimate ahead in time. The *measurement update* adjusts the projected estimate by an actual measurement at that time.

Kalman filter equations

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$

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

Time update (prediction)

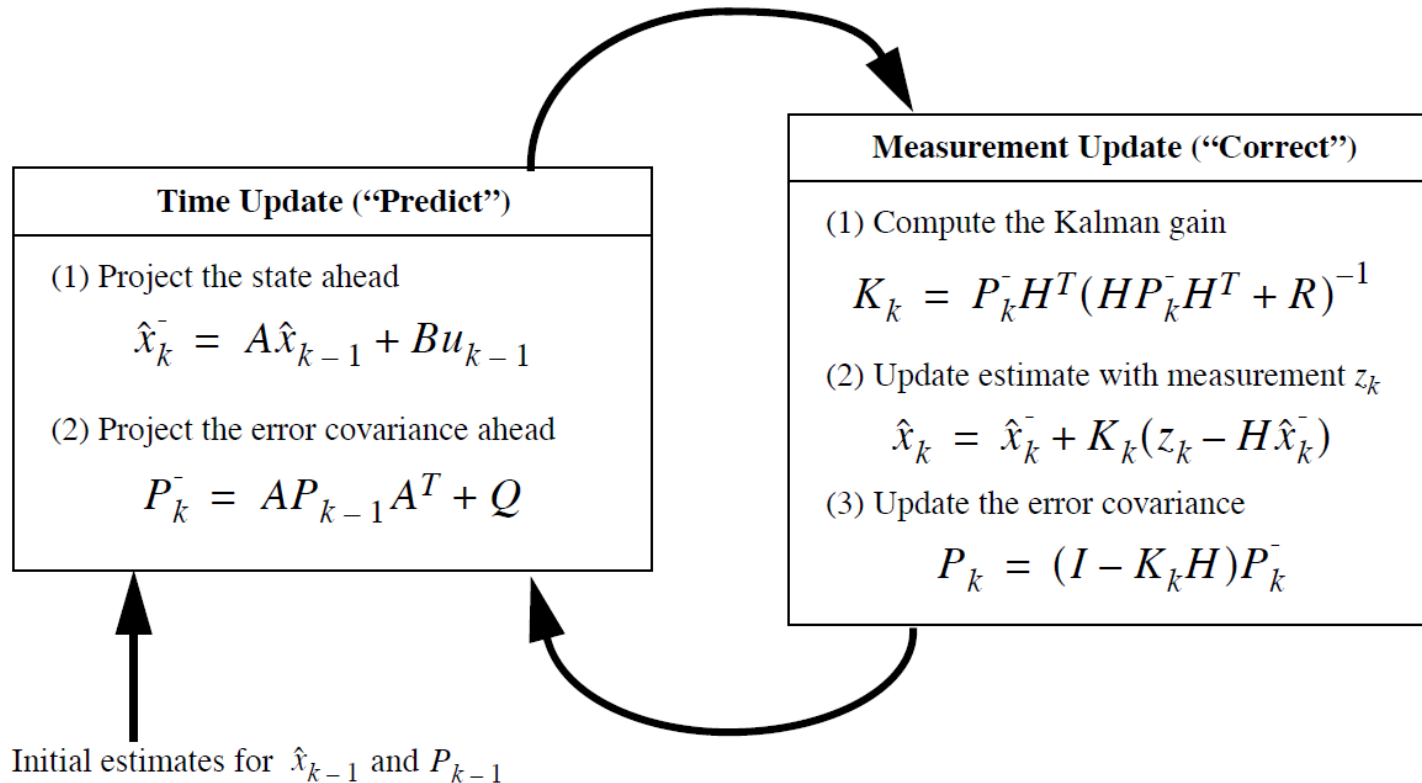
$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

Measurement update

$$P_k = (I - K_k H)P_k^-$$

The operation of the Kalman filter



The process noise covariance Q and the measurement noise covariance R must be set/tuned correctly!

Example: Estimating a random constant with KF

Models

- Voltage measurements with a 0.1 V RMS white noise
- Process model:

$$\begin{aligned}x_k &= Ax_{k-1} + Bu_{k-1} + w_k \\ &= x_{k-1} + w_k\end{aligned}$$

- Measurement model:

$$\begin{aligned}z_k &= Hx_k + v_k \\ &= x_k + v_k\end{aligned}$$



$$\begin{aligned}A &= I \text{ (identity matrix)} \\ H &= I \\ B &= 0\end{aligned}$$

Filter equations

Our time update equations are

$$\hat{x}_k^- = \hat{x}_{k-1},$$

$$P_k^- = P_{k-1} + Q,$$

and our measurement update equations are

$$\begin{aligned} K_k &= P_k^- (P_k^- + R)^{-1} \\ &= \frac{P_k^-}{P_k^- + R}, \end{aligned}$$

$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - \hat{x}_k^-),$$

$$P_k = (1 - K_k) P_k^-.$$

Initial conditions

Presuming a very small process variance, we let $Q = 1e-5$. (We could certainly let $Q = 0$ but assuming a small but non-zero value gives us more flexibility in “tuning” the filter as we will demonstrate below.) Let’s assume that from experience we know that the true value of the random constant has a standard normal probability distribution, so we will “seed” our filter with the guess that the constant is 0. In other words, before starting we let $\hat{x}_{k-1} = 0$.

Similarly we need to choose an initial value for P_{k-1} , call it P_0 . If we were absolutely certain that our initial state estimate $\hat{x}_0 = 0$ was correct, we would let $P_0 = 0$. However given the uncertainty in our initial estimate \hat{x}_0 , choosing $P_0 = 0$ would cause the filter to initially and always believe $\hat{x}_k = 0$. As it turns out, the alternative choice is not critical. We could choose almost any $P_0 \neq 0$ and the filter would *eventually* converge. We’ll start our filter with $P_0 = 1$.

Simulation results with correct R

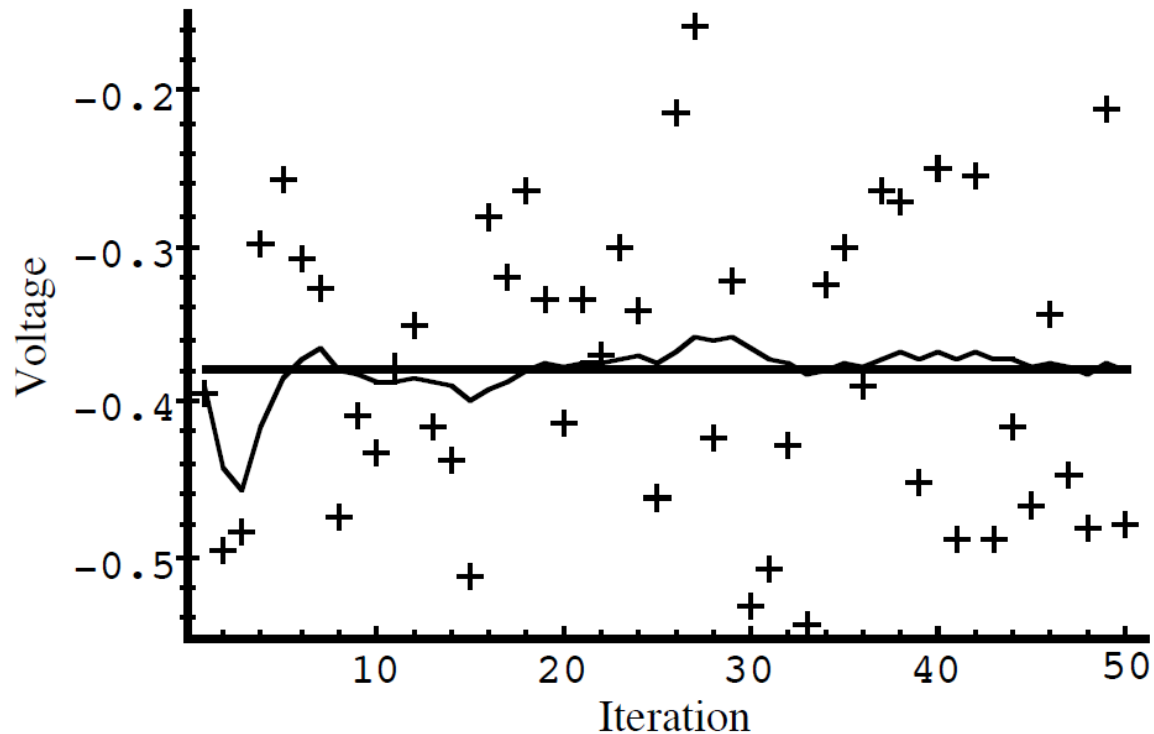


Figure 3-1. The first simulation: $R = (0.1)^2 = 0.01$. The true value of the random constant $x = -0.37727$ is given by the solid line, the noisy measurements by the cross marks, and the filter estimate by the remaining curve.

(Q and) R tuning examples

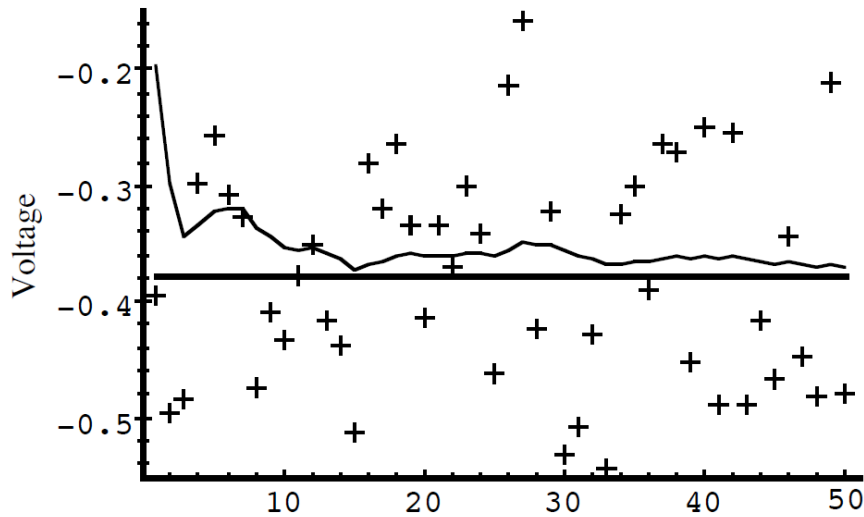


Figure 3-3. Second simulation: $R = 1$. The filter is slower to respond to the measurements, resulting in reduced estimate variance.

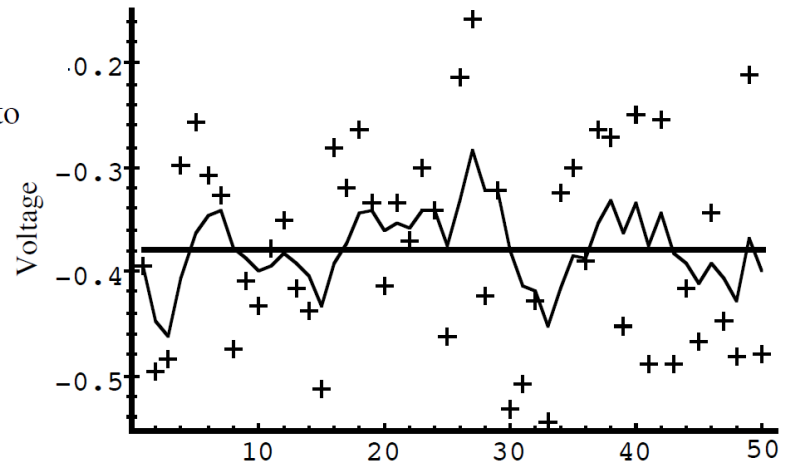


Figure 3-4. Third simulation: $R = 0.0001$. The filter responds to measurements quickly, increasing the estimate variance.

Estimation in nonlinear systems

- Based on **linearization** (taylor series expansion) of the non-linear equations → Extended Kalman Filter (EKF)
- **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

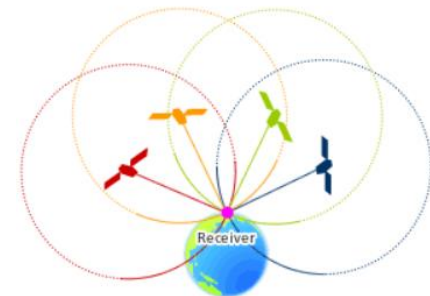
- The measured pseudorange \tilde{P}^k from a satellite k can be expressed as (since we can assume no clock error in the satellite):

$$\tilde{P}^k = \sqrt{(X^k - x)^2 + (Y^k - y)^2 + (Z^k - z)^2} + d + v = \rho^k + d + v$$
- $c\tau = d$ is the position error due to receiver clock error, (X^k, Y^k, Z^k) is the known position of satellite k , (x, y, z) 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: $z_k = h_k(x_k) + v_k$ where

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

- $\mathbf{x} = [x, y, z, d]^T$ are the unknown parameters to be estimated.



Alternatives to the Kalman filter

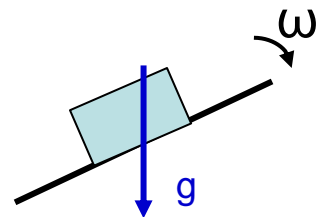
NOT THE OPTIMAL SOLUTION, BUT EASIER TO IMPLEMENT

Alpha filter – a first-order estimator

- If you have a measurement z_k , you can apply a first order filter:

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

- \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 z_k are available, α is set to 0 \rightarrow 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!

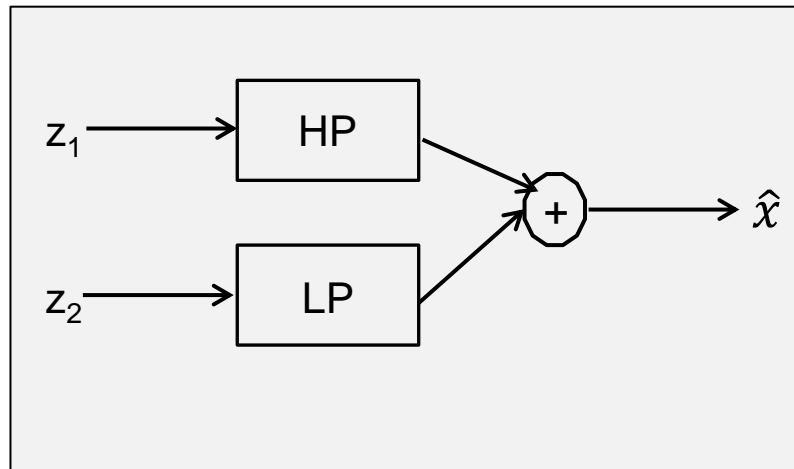


Complimentary filter for data fusion

- 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.
- Need to come up with a mathematical equation to combine the data, e.g. $\hat{\theta}_k = \alpha(\theta_{k-1} + \omega_k \Delta t) + (1 - \alpha)a_k$ where a good value of α need to be determined.



[Video YouTube](#)