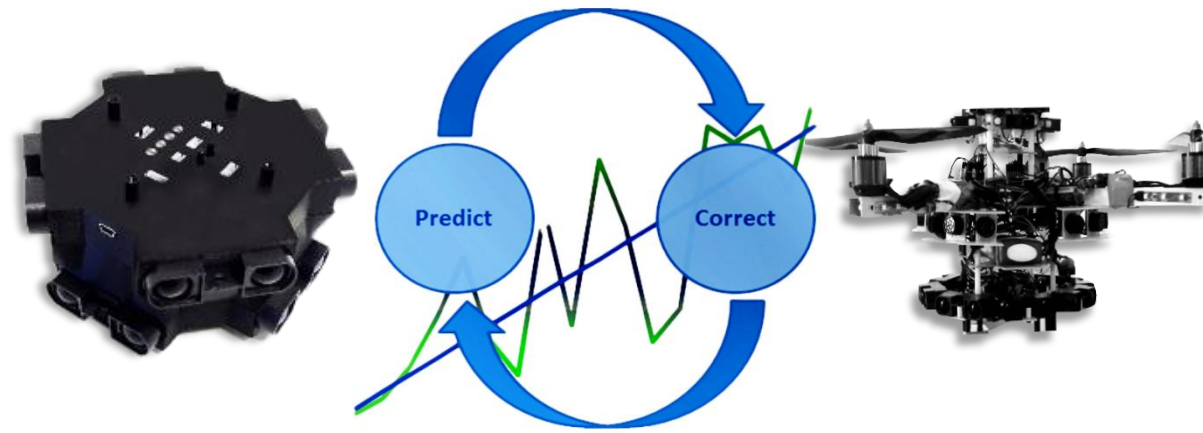


Kalman Filter

Introduction based on the example of
position determination

Lesson 3





- Motivation (definition, properties, application)
- Representation of state space
- Filter equations
- Illustrative explanation
- Derivation of the filter structure
- Application in QCS
- EMQ Library / Framework
- Exercises and instructions
- Basic mathematical operations

Time scope: approx. 2-4 hours





Definition of Kalman Filter:

- ***The Kalman Filter is a state estimator***
- Problem: Measurements are error-prone (noisy).
- Approach: These errors should be reduced by an intelligent method of overdetermined information.
- Idea: With the help of the knowledge about the current state, the (partially) predictable behavior of the system and the new measurements, the current system state is optimally estimated.
- Procedure: After initialization, alternating prediction- and correction steps (see Kalman Filter equations).



Properties of the Kalman Filter:

- Recursive algorithm (real-time capable)
- Based on the state space model
- Not a classical frequency filter, but an estimator (name only from historical reasons)
- Optimal filter with regard to mean and variance
→ follows from **LSM¹-Criterion** (see derivation)

¹: *least squares method*



Usage / application of the Kalman Filter:

- Data fusion (fusion of measurement data und system data)
- State determination (position / speed / orientation of objects, charge state of lithium batteries, etc.)
- Without a system model, the Kalman Filter is a „***sliding (weighted) mean filter***“
- Difficulty in implementation: determining the filter- and system parameters

Representation of the state space



Representation of the state space (background knowledge)

- Serves to describe the system behavior (system theory)
- Prerequisite / approach of the Kalman Filter
- State vector $\vec{x}_t$ describes the system at the point in time t , $\vec{x}_t$ new state
- Input vector $\vec{u}_t$: Inputs to the system (interventions)
- Output vector $\vec{y}_t$: Output of the system (measurements)
- Transition matrix A : Dependence of the new state on the previous state
- Input matrix B : Dependence of the new state on the input vector
- Output matrix C : Dependence of the output on the system state
- Pass-through matrix D : Pass-through of the input to the output
- Process noise $\vec{w}_t$ and measurement noise $\vec{v}_t$ with the covariances Q_t and R_t

System equation [Eq. 1a]

$$\vec{x}_t = A \cdot \vec{x}_t + B \cdot \vec{u}_t + \vec{w}_t$$

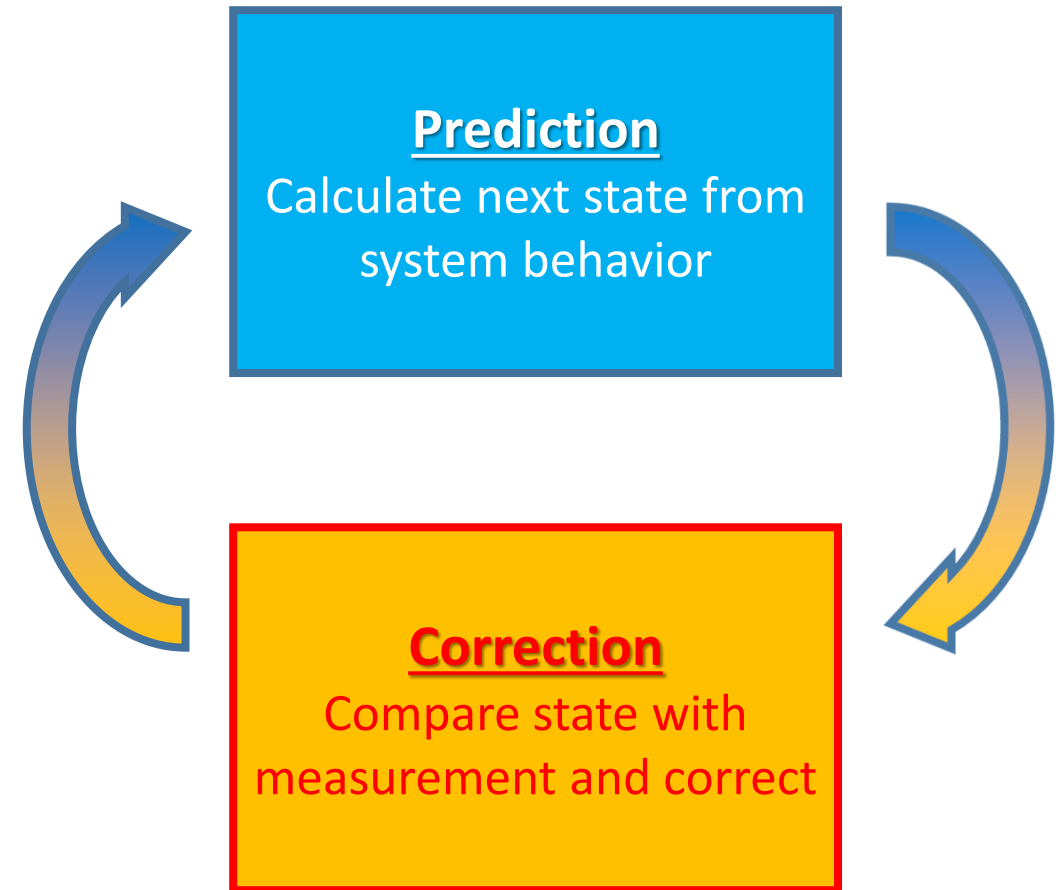
Measurement equation [Eq. 1b]

$$\vec{y}_t = C \cdot \vec{x}_t + D \cdot \vec{u}_t + \vec{v}_t$$



Filter concept

- Prediction
Prediction by system
- Correction
Comparison of prediction with measurement



Filter equation



Time Update (Predict)

(1) Project the state ahead

$$\vec{\dot{x}}_k = \mathbf{A} \cdot \vec{x}_{k-1} + \mathbf{B} \cdot \vec{u}_{k-1}$$

(2) Project error covariance

$$\mathbf{P}'_k = \mathbf{A} \cdot \mathbf{P}_{k-1} \cdot \mathbf{A}^T + \mathbf{Q}_k$$

Time index k



Measurement Update (Correct)

(1) Compute Kalman gain

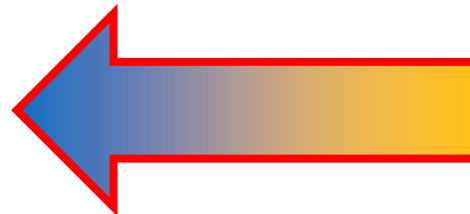
$$\mathbf{K}_k = \mathbf{P}'_k \cdot \mathbf{C}^T \cdot (\mathbf{C} \cdot \mathbf{P}'_k \cdot \mathbf{C}^T + \mathbf{R}_k)^{-1}$$

(2) Update estimate with measurement

$$\vec{\bar{x}}_k = \vec{\dot{x}}_k + \mathbf{K}_k \cdot (\vec{y}_k - \mathbf{C} \cdot \vec{\dot{x}}_k)$$

(3) Update error covariance

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \cdot \mathbf{C}) \cdot \mathbf{P}'_k$$



Filter equation



$$\begin{aligned}\vec{\dot{x}}_k &= \mathbf{A} \cdot \vec{x}_{k-1} + \mathbf{B} \cdot \vec{u}_{k-1} \\ \mathbf{P}'_k &= \mathbf{A} \cdot \mathbf{P}_{k-1} \cdot \mathbf{A}^T + \mathbf{Q}_k\end{aligned}$$

new state, usually $\mathbf{B} = \mathbf{0}$
covariance matrix of the error
(reliability of the state)

Prediction

$\mathbf{Q}$ is the covariance matrix of the system noise, time-varying

$$\mathbf{K}_k = \mathbf{P}'_k \cdot \mathbf{C}^T \cdot \underbrace{(\mathbf{C} \cdot \mathbf{P}'_k \cdot \mathbf{C}^T + \mathbf{R})^{-1}}_{\text{Residual covariance}} \quad \text{Kalman gain, weighting factor of the measurement}$$

Correction

$$\vec{x}_k = \underbrace{\vec{\dot{x}}_k}_{\text{Prediction}} + \mathbf{K}_k \cdot \underbrace{(\vec{y}_k - \mathbf{C} \cdot \vec{\dot{x}}_k)}_{\text{Innovation}} \quad \text{corrected state}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \cdot \mathbf{C}) \cdot \mathbf{P}'_k \quad \text{corrected covariance}$$

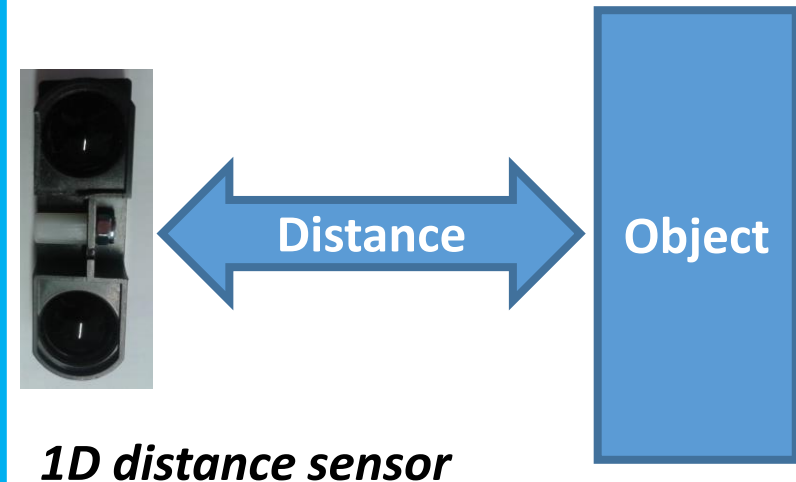
Application example



Example distance measurement (1D, extremely simple):

- Distance measurement of an object with a distance sensor
- One-dimensional case
→ All vectors / matrices simplify to scalars
- Assumptions:
 - Distance measurements are noisy with variance R (constant)
 - Process noise Q is constant
- Kalman Filter should estimate distance optimally:
State of the filter is the distance
→ set up the state space model:
 $A = 1$ (old distance = new distance)
 $C = 1$ (distance (state) is measured directly)

Illustrative explanation according to Welch & Bishop (1/4)



Setting up the filter equations:

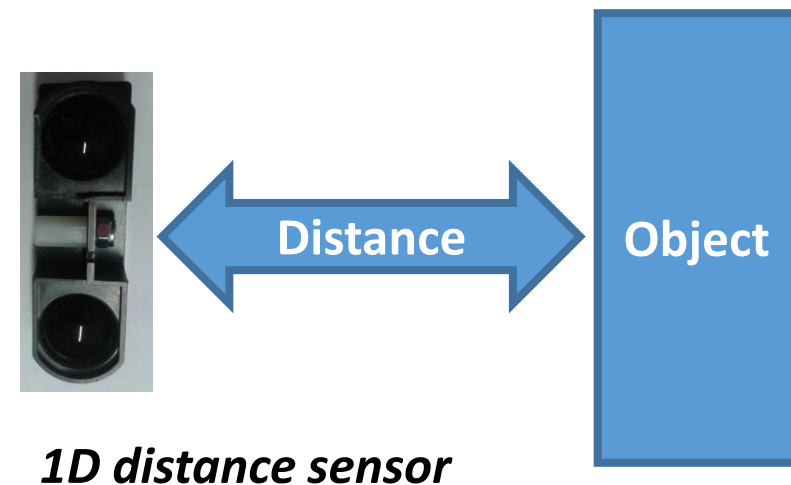
$$(1) P'_k = P_{k-1} + Q \quad (2) K_k = P'_k \cdot (P'_k + R)^{-1}$$

$$(3) \vec{x}_k = \vec{x}_{k-1} + K_k \cdot (\vec{y}_k - \vec{x}_{k-1}) \quad (4) P_k = (I - K_k) \cdot P'_k$$

■ Discussion:

- a. Q indicates how well our **system** models reality.
Here: Model reflects reality exactly $\rightarrow Q = 0$
But it will be shown that a value $Q > 0$ is preferable.
- b. P indicates the reliability of the system state.
If the start state P_0 is known, then $P_0 = 0$ would be expected.
But: If $Q = 0$ and $P_0 = 0 \rightarrow K = 0$ (1), (2)
Then: If the initial state and system are known **for certain**, so the best estimate is always this initial. Measurements would not be considered.
However, since we want to continuously record/update the state via the measurements, Q and $P > 0$ must be selected.

Illustrative explanation according to Welch & Bishop (2/4)



1D distance sensor

Application example



Setting up the filter equations:

$$(1) P'_k = P_{k-1} + Q \quad (2) K_k = P'_k \cdot (P'_k + R)^{-1}$$

$$(3) \vec{x}_k = \vec{x}_{k-1} + K_k \cdot (\vec{y}_k - \vec{x}_{k-1}) \quad (4) P_k = (I - K_k) \cdot P'_k$$

■ Discussion:

c. Let $P_0 = 0$ and $Q > 0$

$$\rightarrow K_1 = Q \cdot (Q + R)^{-1}$$

K_1 is a weighting, namely the ratio of process noise to the sum of process- and measurement noise

d. R indicates the measurement noise:

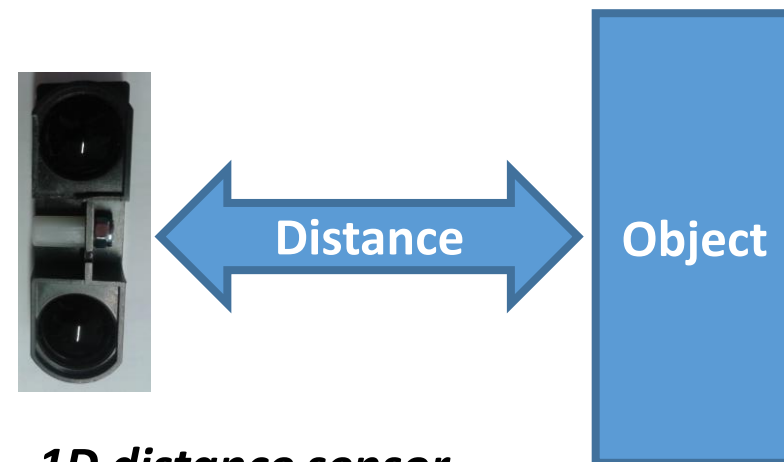
Assuming there is no measurement noise ($R = 0$) (2)

$$\rightarrow K_k = 1$$

$$\rightarrow \vec{x}_k = \vec{y}_k \quad (3)$$

The estimate would always be the measurement.

Illustrative explanation according to Welch & Bishop (3/4)



1D distance sensor

Application example



Setting up the filter equations:

$$(1) \mathbf{P}'_k = \mathbf{P}_{k-1} + \mathbf{Q} \quad (2) \mathbf{K}_k = \mathbf{P}'_k \cdot (\mathbf{P}'_k + \mathbf{R})^{-1}$$

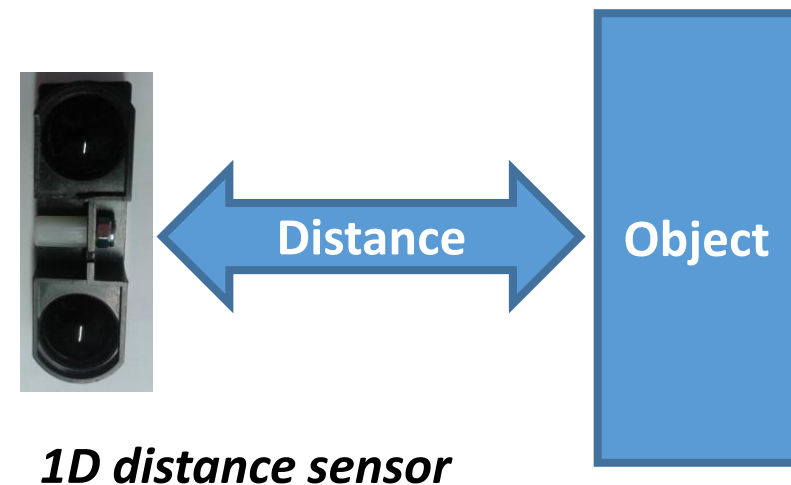
$$(3) \vec{x}_k = \vec{x}_{k-1} + \mathbf{K}_k \cdot (\vec{y}_k - \vec{x}_{k-1}) \quad (4) \mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k) \cdot \mathbf{P}'_k$$

■ Discussion & Conclusion:

The Kalman Filter therefore always takes a weighting $\mathbf{K}$ on the innovation $(\vec{y}_k - \vec{x}_{k-1})$ based on measurement- and process noise, thereby updates the system state.

Here, measurement- and process noise can be time-varying. This allows the different sources of information to be weighted dynamically depending on the situation.

Illustrative explanation according to Welch & Bishop (4/4)0





Derivation of the filter structure (1/3):

Here only filter structure, multi-page derivation of the filter equations can be found e.g. in [„Kalman Filtering“, Chui,1987, S.20-27, Springer] and [„Systemtheorie für stochastische Prozesse“, Schlitt, 1992, S.226ff, Springer]

1. Definition:

Error $\vec{\varepsilon} = \vec{x} - \vec{z}$, with $\vec{x}$ **estimate** and $\vec{z}$ **true value** [Eq. 2a]

2. Requirements:

Input process, measurement- and process noise free of mean value

3. Target:

$$\vec{\hat{x}} = \mathbf{L} \cdot \vec{x} + \mathbf{K} \cdot \vec{y} \quad \text{[Eq. 2b]}$$

with $\mathbf{L}$ and $\mathbf{K}$ sought weightings for estimated value and measurement vector, so that the following two criteria for an optimal filter are fulfilled:

1st criteria: mean value = 0 (true to expectation) $\rightarrow E(\varepsilon) = 0$

2nd criteria: minimum variance $\rightarrow E(\dot{\varepsilon}) = 0$

Derivation of the filter structure (2/3):

3. Setting up the error differential equation: $\vec{\epsilon} = \vec{\hat{x}} - \vec{\hat{z}}$

Inserting [Eq. 2b] and [Eq. 1a+b] into [Eq. 2a] results in:

$$\vec{\epsilon} = \mathbf{L} \cdot \vec{x} + \mathbf{K} \cdot (\mathbf{C} \cdot \vec{z} + \mathbf{D} \cdot \vec{u} + \vec{v}) - \mathbf{A} \cdot \vec{z} - \mathbf{B} \cdot \vec{u} - \vec{w} = \mathbf{L} \cdot \vec{x} - (\mathbf{A} - \mathbf{K} \cdot \mathbf{C}) \cdot \vec{z} + \mathbf{K}(\mathbf{D} \cdot \vec{u} + \vec{v}) - \mathbf{B} \cdot \vec{u} - \vec{w}$$

Again inserting [Eq. 2b resolved to $\vec{z}$] results (transformation):

$$\begin{aligned} \vec{\epsilon} &= \mathbf{L} \cdot \vec{x} - (\mathbf{A} - \mathbf{K} \cdot \mathbf{C}) \cdot (\vec{x} - \vec{\epsilon}) + \mathbf{K}(\mathbf{D} \cdot \vec{u} + \vec{v}) - \mathbf{B} \cdot \vec{u} - \vec{w} \\ &= \{\mathbf{L} - (\mathbf{A} - \mathbf{K} \cdot \mathbf{C})\} \cdot \vec{x} + (\mathbf{A} - \mathbf{K} \cdot \mathbf{C}) \cdot \vec{\epsilon} + (\mathbf{K} \cdot \mathbf{D} - \mathbf{B}) \cdot \vec{u} + \mathbf{K} \cdot \vec{v} - \vec{w} \end{aligned}$$

4. Forming the expected value on both sides results:

$$E[\vec{\epsilon}] = \mathbf{0} \text{ (1st criteria)} = E[\{\mathbf{L} - (\mathbf{A} - \mathbf{K} \cdot \mathbf{C})\} \cdot \vec{x}] + E[(\mathbf{A} - \mathbf{K} \cdot \mathbf{C}) \cdot \vec{\epsilon}] + E[(\mathbf{K} \cdot \mathbf{D} - \mathbf{B}) \cdot \vec{u} + \mathbf{K} \cdot \vec{v} - \vec{w}]$$

because $E[(\mathbf{K} \cdot \mathbf{D} - \mathbf{B}) \cdot \vec{u} + \mathbf{K} \cdot \vec{v} - \vec{w}] = \mathbf{0}$, $E[(\mathbf{A} - \mathbf{K} \cdot \mathbf{C}) \cdot \vec{\epsilon}] = \mathbf{0}$ and $E[\vec{x}]$ not necessarily $= \mathbf{0}$ follows:

$$\mathbf{L} - (\mathbf{A} - \mathbf{K} \cdot \mathbf{C}) = \mathbf{0} \rightarrow \mathbf{L} = \mathbf{A} - \mathbf{K} \cdot \mathbf{C}$$

Derivation of the filter structure

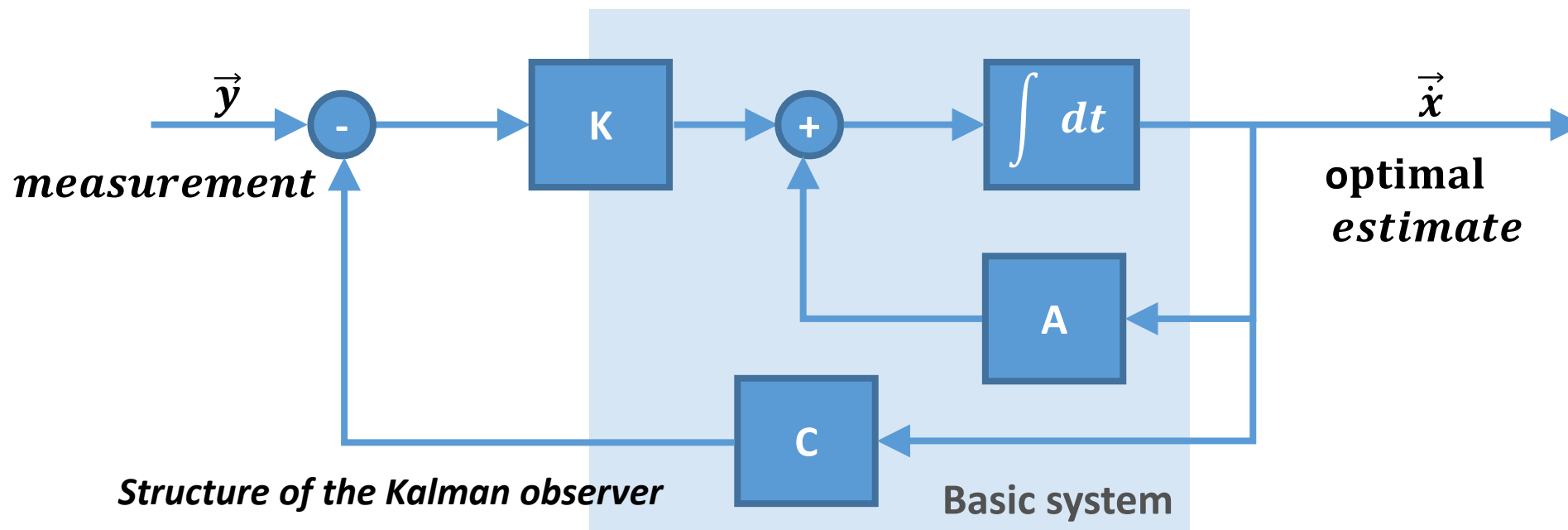


Derivation of the filter structure (3/3):

5. Set up the estimation equation:

$$\begin{aligned}\dot{\vec{x}} &= (A - K \cdot C) \cdot \vec{x} + K \cdot \vec{y} \\ &= A \cdot \vec{x} + K \cdot (\vec{y} - C \cdot \vec{x})\end{aligned}$$

Differential equation of the Kalman Filter



Application in QCS



Idea:

Compensation of the bias drift of the **IMU** (basic problem: gyroscope accumulation error) by adding roll and pitch angle measurement (roll, pitch) based on gravity measurement (Earth's gravity field as reference).

Challenge (additional problem):

With accelerated movements and vibrations such as those caused by motors, the determination of angles with help of the acceleration sensor is very inaccurate.

Solution:

By fusing data of the gyroscope (dynamic, short-term) and accelerometer (static, long-term) with help of a Kalman Filter, both problems can be overcome.

Application in QCS



Setting up the system:

$$A = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$Q = \begin{bmatrix} P^\varphi & 0 \\ 0 & P^\omega \end{bmatrix}$$

$$R = \begin{bmatrix} M_{Acc} & 0 \\ 0 & M_{Gyro} \end{bmatrix}$$

Sampletime $T_s = 0,01$

$$X = [\phi, \omega]'$$

$$Y = [\phi(Acc), \omega(Gyro)]'$$

$$B = D = 0$$

P_φ, P_ω

Process noise of angle & angular velocity

M_{Acc}, M_{Gyro}

Measurement noise of accelerometer & gyroscope



Parameterization:

- Four parameters P_ϕ , P_ω , M_{Acc} , M_{Gyro} must be set
 - Most effort, as trade-off problem: speed versus error susceptibility
 - Higher values mean lower confidence
 - Tip: initial values **1**
 - Tip: test range: **0.01 to 10000**
 - Tip (rule of thumb): Accelerometer **100- to 10000-times** higher variance values (less confidence due to low-pass characteristics)
 - Parameter constant (simple first approach)
 - For a very good, stable control on the **2D/3D-DOF-setup**, this is completely sufficient
 - For free flight, this simple approach is only suitable to a very limited extent, due to:
 - Non-linearities in the system
 - Non-commutativity of rotations
 - Disturbance of the acceleration measurement due to free flight (movements)
- Therefore, free flight with this Kalman Filter is not recommended

Application in QCS



Results:

- Drift error can be compensated → Quadrocopter finds its way back to a horizontal position
- Dynamics remains preserved (no instability due to accelerometer influence on dynamic)

Advantages of this approach:

- **2D** system requires only simple **2D** matrix operations
→ Saving of arithmetic operations and possibly implementation effort
- System stays understandable, low complexity
- Standard formula for matrix inversion (very simple)
- Roll- and pitch independent and identical

Outlook:

- With help of a magnetic sensor, a Kalman Filter can compensate for error in the heading



General notes on the integration:

- The „**Kalman Filter**“ module consists of the Kalman functions (KalmanFilter.h, KalmanFilter.c) and the necessary basic mathematical operations (KalmanMath.h, KalmanMath.c). The latter are already integrated in the library.
- In the **EMQ_QCSF** project, the Kalman Filter is already fully functional integrated and tested for the standard **IMU** in the **QCSF API PRO**. The other **IMU** sensors are in principle integrated in the same way. During integration, the signs and scaling factors of the respective sensor configuration should be correctly considered. Here in particular, the correct axis convention must be observed! (roll, pitch, yaw; right-hand-rule)
- For the further projects (EMQ_Framework, EMQ_Basic), the basic functions of the Kalman Filter are to be inserted and completed manually. To do this, it is recommended to insert the function ***my_Kalman_Init()*** in ***QCSF_Init()*** and ***my_Kalman_Run()*** in ***QCSF_imu()***.



General notes on the integration:

- The module consists of the Kalman Filter folder with the files (*KalmanFilter.c*, *KalmanFilter.h*, *KalmanMath.h*, *KalmanMath.c*)
- To integrate, copy this folder into the source folder (src) as well as the headers in the *QCSF_Main.h* file:

```
#include "KalmanFilter\KalmanMath.h"  
#include "KalmanFilter\KalmanFilter.h"
```

- The Kalman Filter operates now with a sampletime of 10ms.



Notes and tips:

- Remember that at the starting values ($\mathbf{== 1.0}$) for the variances, the accelerometer dominates. Therefore, the values are to be changed as soon as the Kalman Filter outputs meaningful values.
- Remember angular- and radians measure as well as the definition range of the trigonometric functions.
- Check the inputs (measurements) in the filter and always compare with the result.
- With help of the ***asin()*** funktion, an angle can be determined from the accelerometer very easily and sufficient precision.
- The gyroscope measures the angular velocity around an axis (right-hand-rule). The accelerometer measures the acceleration, i.e. gravity, in the direction of the axis. This is a very popular error and it is important to realise this.



Exercise 1:

Based on the slides, put together on the paper (or digitally on a PC) the necessary mathematical operations and formulas to implement a Kalman Filter. It is the system from „***Application in QCS***“ to be implemented.

Write pseudo-code that clarifies the execution order of operations as well as inputs, parameters and outputs.

Answering the following questions will help:

- 1.) Which values are inputs? What is output?
- 2.) Which parameters are given? Are they constant or variable?
- 3.) Which formulas are used in which order?



Exercise 2:

Implement the Kalman Filter by completing the files *KalmanFilter.c* and *KalmanFilter.h*. Tack a quick look at the both files first. Start with a Kalman Filter for the roll axis. The finished „***KalmanMath***“ module will help you to implement the filter equations.

- a) First use the given parameters, i.e. value set to **1**. Here you should generally recognize a change in orientation.
- b) Then set all parameters to **1**, except $P_0 = 1.000.000.000$ and $M_{Acc} = 1.000.000.000$. Now it should still work in principle, but the accelerometer should no longer be included in the calculation, only the gyroscope. The filter will probably drift slowly.

Note: You can thus ($a = Acc$, $b = Gyro$) specifically get to the bottom of errors.



Exercise 3:

Output the Kalman-Filtered roll angle as well as the corresponding angle of the accelerometer and the angular velocity and the summed angle of the gyroscope on the console.

Exercise 4:

Parameterize the Kalman Filter. The parameters should be optimized so that the system does not drift way, but still processes vibrations and translational movements correctly, i.e. it filters them out. As the system later finds application on the QCS, it is completely sufficient if it can process maximum rotations of 90 degrees.

Exercise 5:

Perform multiple rotations around several axis. Hold the sensor horizontally, tilt, yaw and tilt back. What do you notice? Explain the problem and think about improvements.

Basic mathematical operations



Everything in **2D**: It applies $\mathbf{M} = \begin{bmatrix} \mathbf{M}[0][0] & \mathbf{M}[0][1] \\ \mathbf{M}[1][0] & \mathbf{M}[1][1] \end{bmatrix}$ and $\mathbf{V} = \begin{bmatrix} \mathbf{V}[0] \\ \mathbf{V}[1] \end{bmatrix} = [\mathbf{V}[0], \mathbf{V}[1]]'$

`void Matrix_mal_Vektor_2D (double M[2][2], double* V, double * Vr)`

$$\mathbf{V_r} = \mathbf{M} * \mathbf{V}$$

`void Matrixmultiplikation2D (double M1[2][2], double M2[2][2], double R[2][2])`

$$\mathbf{R} = \mathbf{M1} * \mathbf{M2}$$

`void Matrixaddition2D(double M1[2][2], double M2[2][2], double R[2][2])`

$$\mathbf{R} = \mathbf{M1} + \mathbf{M2}$$

`void Matrix_Inverse (double M[2][2], double R[2][2])`

$$\mathbf{R} = \mathbf{M}^{-1}$$

`void Skalar_mal_Vektor(double s, double* R)`

$$\mathbf{R} = \mathbf{R} * s$$

`void Vektoraddition2D(double* V1, double* V2, double* V)`

$$\mathbf{V} = \mathbf{V1} + \mathbf{V2}$$