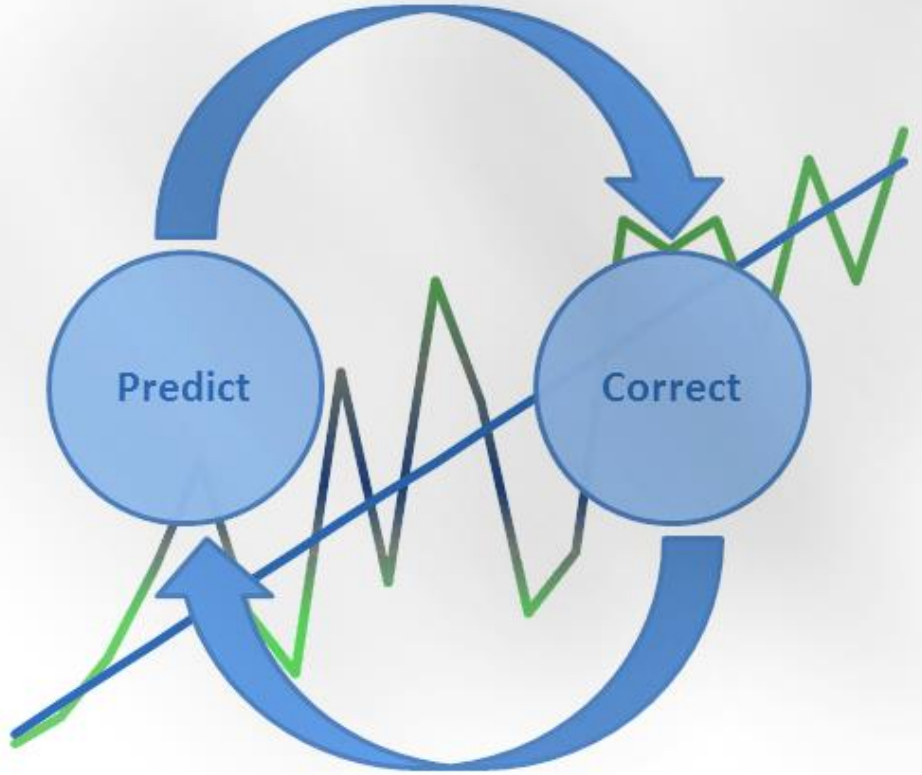
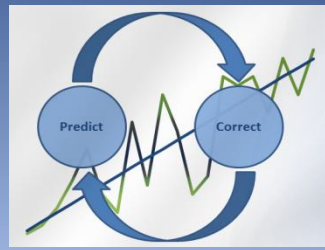
Kalman Filter

Einführung am Beispiel Lagebestimmung

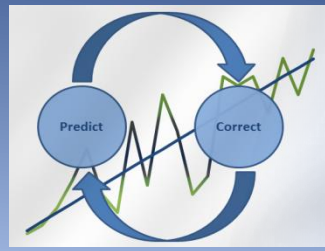




Überblick

Umfang: ca. 2-4 Zeitstunden

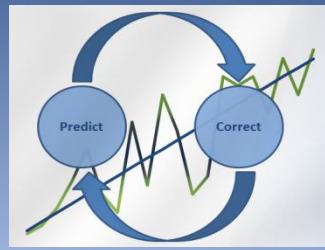
- Motivation (Definition, Eigenschaften, Anwendung)
- Zustandsraumdarstellung
- Filtergleichungen
- Anschauliche Erklärung
- Herleitung der Filterstruktur
- Anwendung im QCS
- EMQ Library / Framework
- Aufgaben und Hinweise
- Mathematische Grundoperationen



Motivation

Definition Kalman-Filter:

- *Das Kalman-Filter ist ein Zustandsschätzer.*
- Problem: Messungen sind fehlerbehaftet (verrauscht).
- Ansatz: Diese Fehler sollen durch eine intelligentes Verfahren überbestimmter Informationen reduziert werden.
- Idee: Mit Hilfe des Wissens über den aktuellen Zustand, dem (teilweise) vorhersagbaren Verhalten des Systems und den neuen Messungen wird der aktuelle Systemzustand optimal geschätzt.
- Vorgehen: Das sagen die Kalman-Filter-Gleichungen.



Motivation

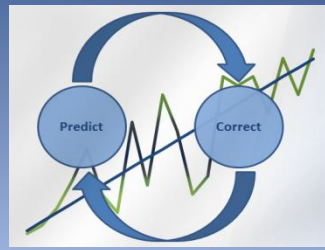
Eigenschaften des Kalman-Filters:

- Rekursiver Algorithmus (real-time geeignet)
- Basiert auf dem Zustandsraummodell
- Kein klassisches Frequenz-Filter, sondern ein Schätzer (Name nur aus historischen Gründen)
- Optimal-Filter bzgl. Mittelwert u. Varianz
  -> folgt aus LSM-Kriterium (siehe Herleitung)

Motivation

Verwendung/Anwendung des Kalman-Filters:

- Datenfusion (Fusion von Messdaten und Systemdaten)
- Zustandsbestimmung (Position / Geschwindigkeit / Orientierung von Objekten, Ladezustand von Lithium-Akkus, etc.)
- Ohne Systemmodell ist das Kalman-Filter ein „gleitender (gewichteter) Mittelwertfilter.“
- Schwierigkeit bei der Implementierung:
  Bestimmung der Filter- und Systemparameter



Zustandsraumdarstellung

Zustandsraumdarstellung (Hintergrundwissen)

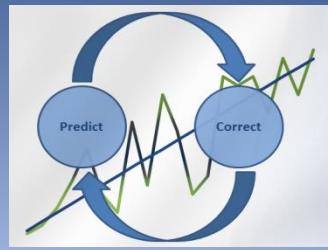
- Dient der Beschreibung des Systemverhaltens (Systemtheorie)
- Voraussetzung / Ansatz des Kalman-Filters
- Zustandsvektor $\overrightarrow{x_t}$ beschreibt das System zum Zeitpunkt t, $\overrightarrow{\dot{x}_t}$ neuer Zustand
- Eingangsvektor $\overrightarrow{u_t}$: Eingänge in das System (Eingriffe)
- Ausgangsvektor $\overrightarrow{y_t}$: Ausgänge des Systems (Messungen)
- Übergangsmatrix **A** : Abhängigkeit des neuen Zustandes vom bisherigen
- Eingangsmatrix **B**: Abhängigkeit des neuen Zustands vom Eingangsvektor
- Ausgangsmatrix **C**: Abhängigkeit des Ausgangs vom Systemzustand
- Durchgangsmatrix **D**: Durchgriff des Eingangs auf den Ausgang
- Prozessrauschen $\overrightarrow{w_t}$ und Messrauschen $\overrightarrow{v_t}$ mit den Kovarianzen $\mathbf{Q_t}$ und $\mathbf{R_t}$

Systemgleichung [Gl. 1a]

$$\overrightarrow{\dot{x}_t} = \boldsymbol{A} \cdot \overrightarrow{x_t} + \boldsymbol{B} \cdot \overrightarrow{u_t} + \overrightarrow{w_t}$$

$$\overrightarrow{y_t} = \boldsymbol{C} \cdot \overrightarrow{x_t} + \boldsymbol{D} \cdot \overrightarrow{u_t} + \overrightarrow{v_t}$$

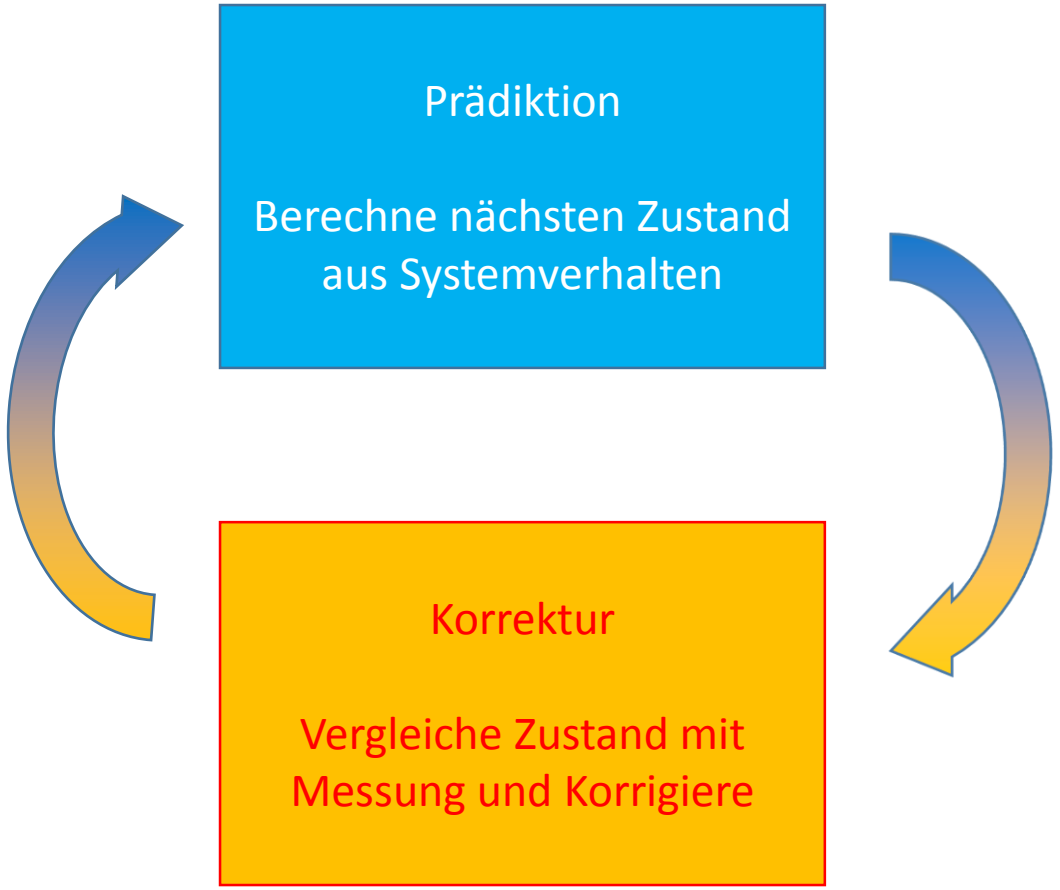
Messgleichung [Gl. 1b]



Filtergleichung

Filterkonzept

- Prädiktion
  Vorhersage durch System
- Korrektur
  Vergleich von Vorhersage mit Messung



Filtergleichung

Time Update (Predict)

(1) Project the state ahead

$$\overrightarrow{x'_k} = \boldsymbol{A} \cdot \overrightarrow{x_{k-1}} + \boldsymbol{B} \cdot \overrightarrow{u_{k-1}}$$

(2) Project error covariance

$$\boldsymbol{P'_k} = \boldsymbol{A} \cdot \boldsymbol{P_{k-1}} \cdot \boldsymbol{A}^T + \boldsymbol{Q_k}$$

Measurement Update (Correct)

(1) Compute Kalman gain

$$\boldsymbol{K_k} = \boldsymbol{P'_k} \cdot \boldsymbol{C}^{\mathrm{T}} \cdot \left(\boldsymbol{C} \cdot \boldsymbol{P'_k} \cdot \boldsymbol{C}^{\mathrm{T}} + \boldsymbol{R_k}\right)^{-1}$$

(2) Update estimate with measurement

$$\overrightarrow{x_k} = \overrightarrow{x'_k} + \boldsymbol{K_k} \cdot (\overrightarrow{y_k} - \boldsymbol{C} \cdot \overrightarrow{x'_k})$$

(3) Update error covariance

$$\boldsymbol{P_k} = (\boldsymbol{I} - \boldsymbol{K_k} \cdot \boldsymbol{C}) \cdot \boldsymbol{P'_k}$$

Zeitindex k

Filtergleichung

$$\overrightarrow{x'_k} = \boldsymbol{A} \cdot \overrightarrow{x_{k-1}} + \boldsymbol{B} \cdot \overrightarrow{u_{k-1}}$$ Neuer Zustand, meist B = 0

$$\boldsymbol{P'}_k = \boldsymbol{A} \cdot \boldsymbol{P_{k-1}} \cdot \boldsymbol{A}^T + \boldsymbol{Q}_k$$ Kovarianzmatrix des Fehlers (Zuverlässigkeit des Zustandes)

Q ist die Kovarianzmatrix des Systemrauschens, zeitveränderlich

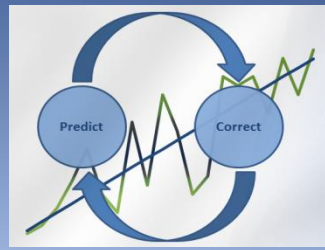
Prädiktion

$$\boldsymbol{K_k} = \boldsymbol{P'}_k \cdot \boldsymbol{C}^{\mathrm{T}} \cdot \underbrace{\left(\boldsymbol{C} \cdot \boldsymbol{P'}_k \cdot \boldsymbol{C}^{\mathrm{T}} + \boldsymbol{R}\right)^{-1}}_{\text{Residualkovarianz}}$$ Kalman-Gain, Gewichtungsfaktor der Messung

$$\overrightarrow{x_k} = \underbrace{\overrightarrow{x'_k}}_{\text{Vorhersage}} + \boldsymbol{K_k} \cdot \underbrace{(\overrightarrow{y_k} - \boldsymbol{C} \cdot \overrightarrow{x'_k})}_{\text{Innovation}}$$ Korrigierter Zustand

$$\boldsymbol{P_k} = (\boldsymbol{I} - \boldsymbol{K_k} \cdot \boldsymbol{C}) \cdot \boldsymbol{P'}_k$$ Korrigierte Kovarianz

Korrektur



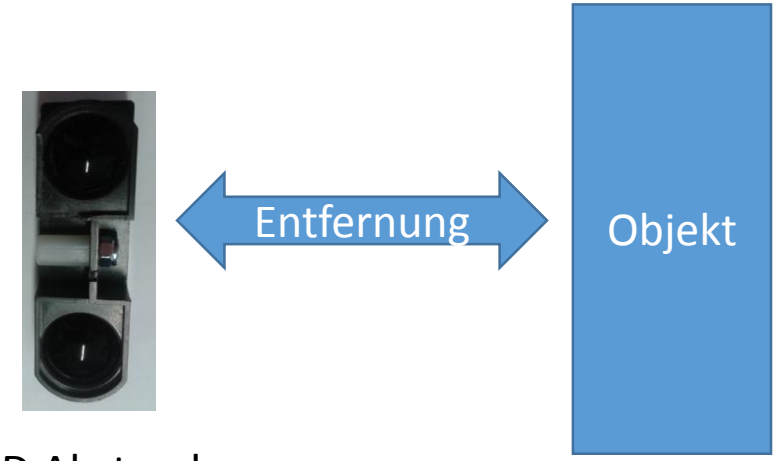
Filtergleichung

Beispiel Entfernungsmessung (1D, extrem einfach):

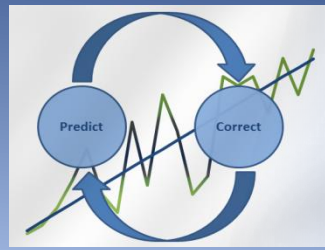
- Entfernungsmessung eines Objekts mit einer Kamera
- Eindimensionaler Fall
  -> Alle Vektoren / Matrizen vereinfachen sich zu Skalaren
- Annahmen:
  - Entfernungsmessungen sind verrauscht mit Varianz **R** (konstant)
  - Prozessrauschen **Q** ist konstant
- Kalman-Filter soll Entfernung optimal schätzen:
  Zustand des Filters ist die Entfernung

-> Aufstellen des Zustandsraummodells:
A = 1 (Alte Entfernung = Neue Entfernung)
C = 1 (Entfernung (Zustand) wird direkt gemessen)

Anschauliche Erklärung nach Welch & Bishop (1/4)



1D Abstandssensor



Filtergleichung

Aufstellen der Filtergleichungen:

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

(3) $\overrightarrow{x_k} = \overrightarrow{x_{k-1}} + \boldsymbol{K_k} \cdot (\overrightarrow{y_k} - \overrightarrow{x_{k-1}})$ (4) $\boldsymbol{P_k = (I - K_k) \cdot P'_k}$

Diskussion:

a) Q gibt an, wie gut unser **System** die Realität modelliert.
   Hier: Modell spiegelt Realität exakt wieder -> Q = 0
   Aber es wird sich zeigen, dass ein Wert **Q > 0** vorzuziehen ist.

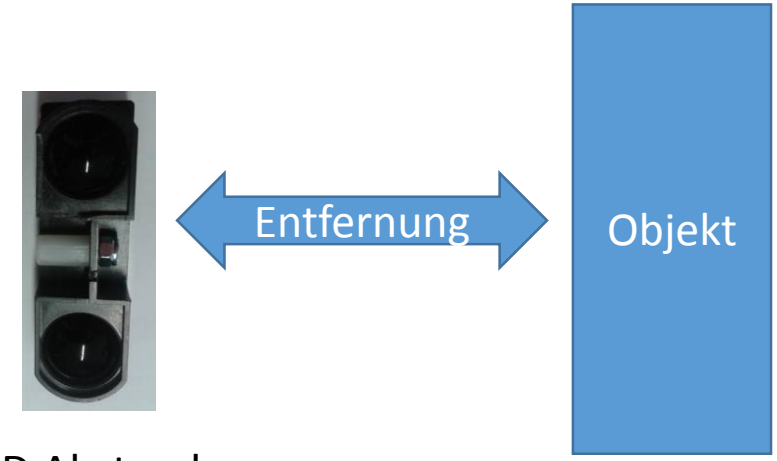
b) P gibt die Zuverlässigkeit des **Systemzustandes** an.
   Ist der Startzustand P_0 bekannt, so wäre $P_0 = 0$ zu erwarten.

Aber: Wenn **Q = 0 und P_0 = 0 -> K = 0** **[2]**

Dann: Sind Anfangszustand und System **sicher** bekannt, so ist die beste Schätzung stets dieser Anfangszustand. Messungen würden nicht berücksichtigt.

Da wir aber den Zustand über die Messungen kontinuierlich erfassen/aktualisieren wollen, sind Q und P >0 zu wählen.

Anschauliche Erklärung nach Welch & Bishop (2/4)



1D Abstandssensor

Filtergleichung

Aufstellen der Filtergleichungen:

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

(3) $\overrightarrow{x_k} = \overrightarrow{x_{k-1}} + \boldsymbol{K_k} \cdot (\overrightarrow{y_k} - \overrightarrow{x_{k-1}})$ (4) $\boldsymbol{P_k = (I - K_k) \cdot P'_k}$

Diskussion:

c) Es sei **P_0 = 0 und Q > 0**

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

$\boldsymbol{K_1}$ ist eine Gewichtung und zwar das Verhältnis aus Prozessrauschen zu der Summe aus Prozess- und Messrauschen

d) R gibt das Messrauschen an:

Angenommen es existiere kein Messrauschen (**R = 0**) [2]

-> $\boldsymbol{K_k}$ = 1

-> $\overrightarrow{x_k} = \overrightarrow{y_k}$ [3]

Die Schätzung wäre stets die Messung.

Anschauliche Erklärung nach Welch & Bishop (3/4)

Entfernung

Objekt

1D Abstandssensor

Filtergleichung

Aufstellen der Filtergleichungen:

(1) $\boldsymbol{P'_k = P_{k-1} + Q}$

(2) $\boldsymbol{K_k = P'_k \cdot \left(P'_k + R\right)^{-1}}$

(3) $\overrightarrow{x_k} = \overrightarrow{x_{k-1}} + \boldsymbol{K_k} \cdot (\overrightarrow{y_k} - \overrightarrow{x_{k-1}})$

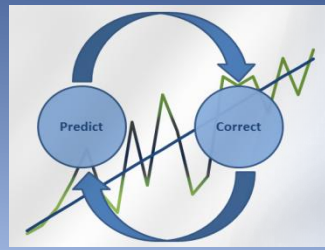
(4) $\boldsymbol{P_k = (I - K_k) \cdot P'_k}$

Diskussion & Fazit:
Das Kalman-Filter nimm also stets ausgehend von Mess- und Prozessrauschen eine Gewichtung (K) der Innovation ($\overrightarrow{y_k} - \overrightarrow{x_{k-1}}$) vor und aktualisiert damit den Systemzustand.

Dabei können Mess- und Prozessrauschen zeitveränderlich sein. Damit lassen sich die verschiedenen Informationsquellen je nach Situation dynamisch gewichten.

Anschauliche Erklärung nach Welch & Bishop (4/4)

1D Abstandssensor



Herleitung der Filterstruktur

Herleitung der Filterstruktur (1/3):

Hier nur Filterstruktur, mehrseitige Herleitung der Filtergleichungen findet sich z.B. bei [Chui,1987, Kalman Filtering, S.20-27, Springer] und [Schlitt, 1992, S.226ff, Springer]

1.) Definition:

Fehler $\vec{\varepsilon} = \vec{x} - \vec{z}$, *mit* $\vec{x}$ ***Schätzung*** *und* $\vec{z}$ ***wahrer Wert*** *[Gl. 2a]*

2.) Voraussetzungen:

Eingangsprozess, Mess- und Prozessrauschen mittelwertfrei

3.) Ziel:

$\dot{\vec{x}} = \mathbf{L} \cdot \vec{x} + \mathbf{K} \cdot \vec{y}$ [Gl. 2c]

mit **L** und **K** gesuchte Gewichtungen für Schätzwert und Messvektor, so dass folgende zwei Kriterien für ein optimales Filter erfüllt werden:

Mittelwert = 0 (erwartungsfreu) -> $E(\varepsilon) = 0$

Minimale Varianz -> $E(\dot{\varepsilon}) = 0$

Herleitung der Filterstruktur

Herleitung der Filterstruktur (2/3):

3.) Aufstellen der Fehlerdifferentialgleichung

$\dot{\vec{\varepsilon}} = \dot{\vec{x}} - \dot{\vec{z}}$

Einsetzen von [Gl. 2b] und [Gl. 1a+b] in [Gl. 2a] ergibt:

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

Erneut einsetzen von [Gl. 2b nach $\vec{z}$ aufgelöst] ergibt (Umformung):

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

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

4.) Bilden des Erwartungswertes auf beiden Seiten ergibt:

$E[\dot{\vec{\varepsilon}}] = 0$ (Kriterium 1) $= E[\{ \mathbf{L} - (\mathbf{A} - \mathbf{K} \cdot \mathbf{C}) \} \vec{x}] + E[(\mathbf{A} - \mathbf{K} \cdot \mathbf{C}) \cdot \vec{\varepsilon}] + E[(\mathbf{K} \cdot \boldsymbol{D} - \boldsymbol{B}) \cdot \vec{\boldsymbol{u}} + \mathbf{K} \cdot \vec{v} - \vec{w}]$

Wegen $E[(\mathbf{K} \cdot \boldsymbol{D} - \boldsymbol{B}) \cdot \vec{\boldsymbol{u}} + \mathbf{K} \cdot \vec{v} - \vec{w}] = 0$, $E[(\mathbf{A} - \mathbf{K} \cdot \mathbf{C}) \cdot \vec{\varepsilon}] = 0$ und

$E[\vec{x}]$ nicht notwendigerweise = 0 folgt: $\mathbf{L} - (\mathbf{A} - \mathbf{K} \cdot \mathbf{C}) = 0$

-> $\mathbf{L} = \mathbf{A} - \mathbf{K} \cdot \mathbf{C}$

Herleitung der Filterstruktur

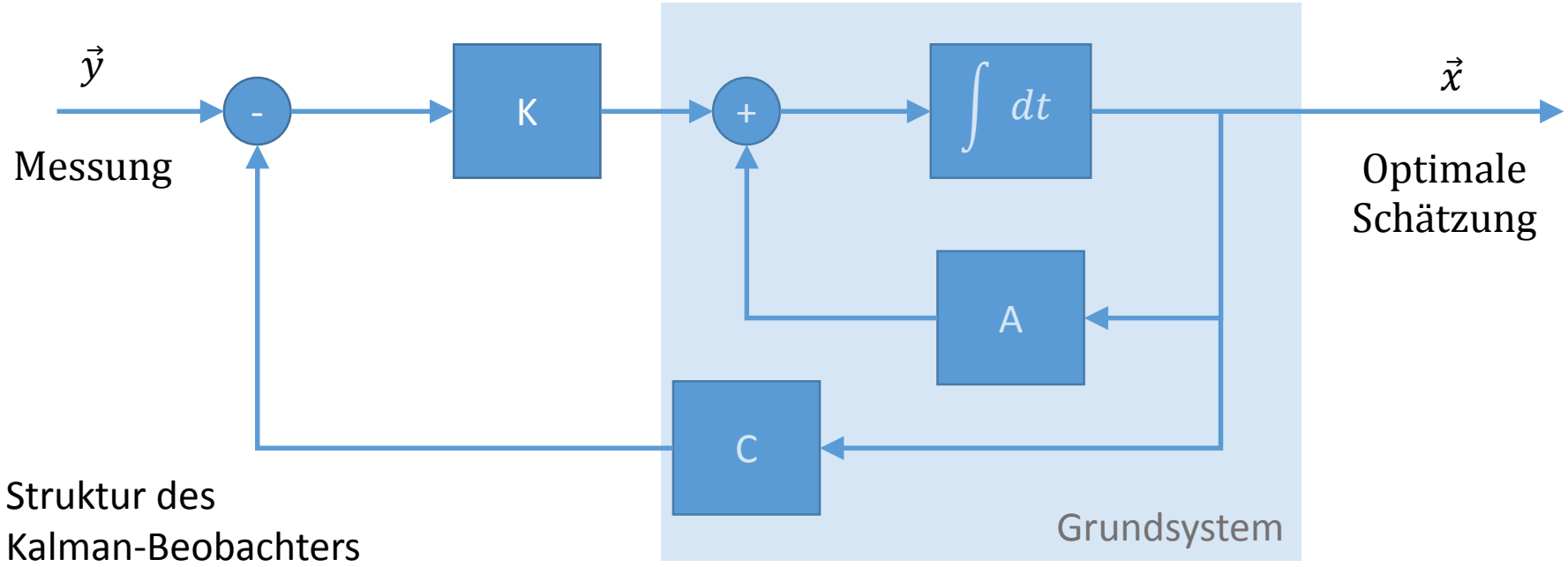
Herleitung der Filterstruktur (3/3):

5.) Aufstellen der Schätzgleichung:

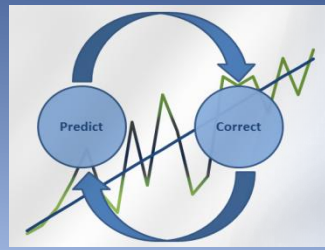
$\dot{\vec{x}} = (\mathbf{A} - \mathbf{K} \cdot \mathbf{C}) \cdot \vec{x} + \mathbf{K} \cdot \vec{y}$

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

Differentialgleichung des Kalman-Filters



Struktur des Kalman-Beobachters



Anwendung im QCS

Idee:
Kompensation des Biasdrifts (Grundproblem Gyroskop-Akkumulationsfehler) durch Hinzuziehen von Roll- und Nickwinkelmessung (Roll, Pitch) basierend auf Gravitationsmessung (Schwerefeld der Erde als Referenz)

Herausforderung (Zusatzproblem):
Bei beschleunigten Bewegungen und Vibrationen wie z.B. durch Motoren ist die Bestimmung der Winkel mit Hilfe des Beschleunigungssensors sehr inakkurat.

Lösung:
Durch Datenfusion von Gyroskop (dynamisch, kurzfristig) und Accelerometer (statisch, langfristig) mit Hilfe eines Kalman-Filters können beide Probleme bewältigt werden.

Anwendung im QCS

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

$$Q = \begin{bmatrix} P_{\varphi} & 0 \\ 0 & P_{\omega} \end{bmatrix} \qquad R = \begin{bmatrix} M_{Acc} & 0 \\ 0 & M_{Gyro} \end{bmatrix}$$

<u>Aufstellen des Systems</u>

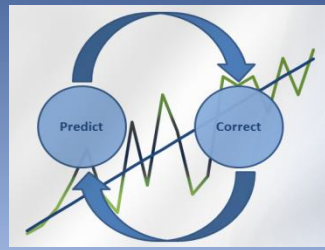
Sampletime T_s = 0,01

X = [ϕ, ω]‘

Y = [ϕ(Acc), ω(Gyro)]‘ B = D = 0

P_{φ}, P_{ω} Prozessrauschen: Zeile 1 (Winkel), Zeile 2 (Winkelgeschwindigkeit)

M_{Acc}, M_{Gyro} Messrauschen von Accelerometer & Gyroskop



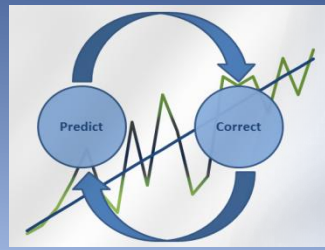
Anwendung im QCS

Parametrisierung:

- Es müssen die 4 Parameter P_{φ}, P_{ω}, M_{Acc}, M_{Gyro} eingestellt werden
- Größter Aufwand, da Trade-Off Problem: Geschwindigkeit und Fehleranfälligkeit
- Höhere Werte bedeuten niedrigeres Vertrauen
- Tipp: Anfangswerte 1
- Tipp: Versuchsbereich: 0.01 bis 10000
- Tipp (Faustformel): Accelerometer 100- bis 10000-fach
  höhere Varianzwerte (weniger Vertrauen wegen Low-Pass-Charakteristik)

- Parameter konstant (einfacher erster Ansatz)
- Für eine sehr gute, stabile Regelung am 2D/3D-DOF-Aufbau reicht das völlig aus
- Für den freien Flug eignet sich dieser einfache Ansatz nur sehr eingeschränkt, wegen:
  - Nicht-Linearitäten im System
  - Nicht-Kommutativität von Drehungen
  - Störung der Beschleunigungsmessung durch freien Flug (Bewegungen)

  -> Daher wird von einem freien Flug mit diesem Kalman Filter abgeraten



Anwendung im QCS

Ergebnisse:

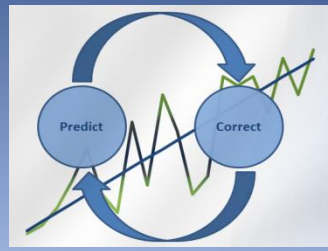
- Driftfehler können kompensiert werden -> Quadrocopter findet in die Horizontale zurück
- Dynamik bleibt erhalten (keine Instabilität durch Accelerometer-Einfluss bei Dynamik)

Vorteile dieses Ansatzes:

- 2D System erfordert nur einfache 2D Matrixoperationen
  -> Ersparnis von Rechenoperationen und ggf. Implementierungsaufwand
- System bleibt verständlich, geringe Komplexität
- Standardformel für Matrix-Invertierung (sehr einfach)
- Roll- und Nick unabhängig und identisch

Ausblick:

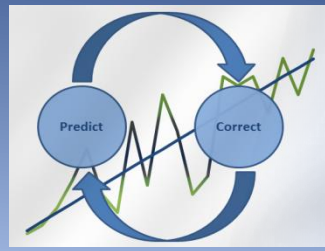
- Mit Hilfe eines Magnetsensors kann ein Kalman-Filter Fehler im Heading kompensieren



EMQ Library / Framework

Generelle Hinweise zur Integration:

- Das Modul Kalman Filter besteht aus den Kalman Funktionen (KalmanFilter.h, (KalmanFilter.c) und den dazu nötigen, mathematischen Grundoperationen (KalmanMath.h, KalmanMath.c). Letztere sind in der Library bereits integriert.
- Im Projekt EMQ QCSF ist der Kalman Filter bereits für die Standard IMU voll funktionsfähig in der QCSF API PRO integriert und getestet. Die Integration anderer IMU Sensoren erfolgt prinzipiell analog. Bei der Integration sind die Vorzeichen und die Skalierungsfaktoren der jeweiligen Sensorkonfiguration entsprechend korrekt zu berücksichtigen. Hierbei ist insbesondere auf die korrekte Achsenkonvention zu achten!
- Für die weiteren Projekte (EMQ_Framework, EMQ_Basic) sind die Grundfunktionen des KalmanFilters händisch einzufügen und auszufüllen. Dazu wird empfohlen die Funktion my_Kalman_Init() in QCSF_Init() und my_Kalman_Run() in QCSF_imu() einzufügen.



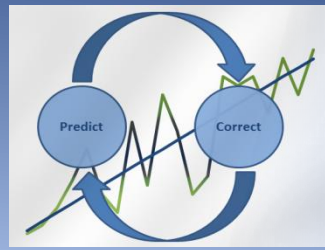
EMQ Library / Framework

Generelle Hinweise zur Integration:

- Die Module bestehend aus dem KalmanFilter-Ordner mit den Dateien (KalmanFilter.c, KalmanFitlter.h, KalmanMath.h, KalmanMath.c) sind darüber hinaus in den Source-Ordner (src) zu kopieren sowie in der Datei QCSF_Main.h hinzuzufügen:

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

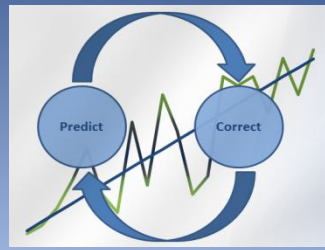
- Das Kalman Filter operiert nun mit einer Sampletime von 10ms.



Hinweise

Hinweise und Tipps:

- Bedenke, dass bei den Startwerten (== 1.0) für die Varianzen der Accelerometer dominiert. Daher sind die Werte zu ändern, sobald der Kalman Filter sinnvolle Werte ausgiebt.
- Denke an Winkel- und Bogenmaß sowie an den Definitionsbereich der trigonometrischen Funktionen.
- Überprüfe die Eingaben (Messungen) ins Filter und vergleiche stets mit dem Ergebnis.
- Mit Hilfe der asin()-Funktion lässt sich aus dem Accelerometer sehr einfach und hinreichend präzise ein Winkel bestimmen.
- Das Gyroskop misst die Winkelgeschwindigkeit um eine Achse (Rechte-Hand-Regel). Der Accelerometer misst die Beschleunigung, d.h. die Gravitation, in Achsrichtung. Dies ist ein sehr beliebter Fehler und es ist wichtig, sich dies klar zu machen.



Aufgaben

<u>Aufgabe 1:</u>
Ausgehend von den Folien stelle auf dem Papier (oder digital am PC) die benötigten mathematischen Operationen und Formeln zur Umsetzung eines Kalman Filters zusammen. Es ist das System aus „Anwendung im QCS“ umzusetzen.

Schreibe Pseudo-Code, der die Ausführreihenfolge der Operationen sowie Eingaben, Parameter und Ausgaben verdeutlicht.

Die Beantwortung der folgenden Fragen wird dabei helfen:
1.) Welche Werte sind Inputs? Was ist Output?
2.) Welche Parameter sind gegeben? Sind diese konstant oder variabel?
3.) Welche Formeln werden in welcher Reihenfolge gebraucht

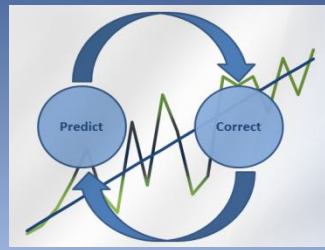
Aufgaben

Aufgabe 2:

Implementiere den Kalman Filter durch Ausfüllen der Dateien KalmanFilter.c und KalmanFilter.h. Guck dir die beiden Dateien zunächst kurz an. Beginne mit einem Kalman Filter für die roll-Achse. Das fertige KalmanMath-Modul wird dir dabei helfen, die Filtergleichungen zu implementieren.

a) Verwende zunächst die gegebenen Parameter, d.h. alles auf 1. Dabei sollte grundsätzlich eine Orientierungsänderung zu erkennen sein.

b) Setzte dann alles auf 1, außer P0 = 1.000.000.000 und M_Acc = 1.000.000.000. Nun sollte es grundsätzlich weiterhin funktionieren, jedoch sollte Accelerometer quasi nicht mehr eingehen, sondern nur das Gyroskop. Das Filter wird vermutlich langsam driften. Beachte: Fehlern kannst du somit (a = Acc, b = Gyro) gezielt auf den Grund gehen.

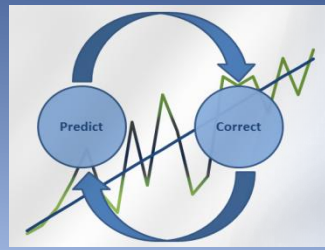


Aufgaben

Aufgabe 3:
Gib dir den Kalman-gefilterten Rollwinkel sowie den entsprechenden Winkel des Accelerometers und die Winkelgeschwindigkeit und den aufaddierten Winkel des Gyroskops auf der Konsole aus.

Aufgabe 4:
Parametrisiere das Kalman-Filter. Die Parameter sollten so optimiert werden, dass das System nicht wegdriftet, Erschütterungen und translatorische Bewegungen aber nach wie vor korrekt verarbeitet, d.h. ausfiltert. Da das System später auf dem QCS Anwendung findet, genügt es völlig, wenn es maximale Drehungen von 90 Grad verarbeiten kann.

Aufgabe 5:
Führe multiple Drehungen um mehrere Achsen durch. Halte den Sensor horizontal, kippe, giere und kippe zurück. Was fällt dir auf? Erkläre das Problem und mach dir Gedanken über Verbesserungen.



Mathematische Grundoperationen

Alles in 2D: Es gilt M = $\begin{bmatrix} M[0][0] & M[0][1] \\ M[1][0] & M[1][1] \end{bmatrix}$ und V = $\begin{bmatrix} V[0] \\ V[1] \end{bmatrix}$ = [V[0], V[1]]‘

void Matrix_mal_Vektor_2D (double M[2][2], double* V, double * Vr)	Vr = M * V
void Matrixmultiplikation2D (double M1[2][2], double M2[2][2], double R[2][2])	R = M1 * M2
void Matrixaddition2D(double M1[2][2], double M2[2][2], double R[2][2])	R = M1 + M2
void Matrix_Inverse (double M[2][2], double R[2][2])	R = M^{-1}
void Skalar_mal_Vektor(double s, double* R)	R = R*s
void Vektoraddition2D(double* V1, double* V2, double* V)	V = V1 + V2