1 Szenario

In diesem Tutorial soll der Kalmanfilter präsentiert werden, es wird dabei auf die grundlegenden Gleichungen eingegangen und anhand eines Beispiels die Anwendung des Kalmanfilters gezeigt. Zum Schluss folgt eine beispielhafte Implementierung des Kalmanfilters für dieses Beispiel.

2 Das Kalmanfilter allgemein

Ein Kalmanfilter ist ein Berechnungsalgorithmus zur Schätzung der internen Zustände eines zeitdiskreten Systems mit Hilfe von Sensordaten. Es basiert auf einem zeitdiskreten linearen Systemmodel und einem ebenfalls zeitdiskreten linearen Messmodell. Es muss also ein physikalisches Modell des zu untersuchenden Vorgangs zugrunde liegen, sei es in Form einer Differentialgleichung oder eines Zustandsraummodells. Ein Kalmanfitler linearisiert das System, also die Ergebnisse sind umso besser, je mehr der wahre physikalische Charackter lineare Eigenschaften aufweist. Eine direkte Anwendung ist aus einem gegeben Zustandsraummodel der Form:

$$x_{k+1} = A_k x_k + B u_k + w_k \tag{1}$$

$$z_k = H_k x_k + v_k \tag{2}$$

Dabei beschreibt x_k einen Zustandsvektor des Systems $\in \mathbb{R}^n$, A eine Überführungsmatrix $\in \mathbb{R}^{n \times n}$, u ein Vektor mit den Eingängen des Systems. B eine Matrix, die beschreibt wie stark der Einfluss der Eingänge u auf den Systemzustand ist und w_k repräsentiert einen Vektor $\in \mathbb{R}^n$, der das Messrauschen des Systems charackterisiert, er besitzt den Erwartungswert 0 und die Kovarianz Q. Die Kovarianz Q repräsentiert eine Kovarianzmatrix $\in \mathbb{R}^{n \times n}$ und sollte empirisch bestimmt werden. In der zweiten Gleichung stellt z_k die Messwerte der Sensoren dar. Da nicht alle Zustände direkt gemessen werden können wird eine Matrix H benötigt, die eben diesen Zusammenhang repräsentiert. Schließlich sind alle Messwerte noch durch einen Fehler behaftet, der durch das Messrauschen v_k, ein Vektor ähnlich wie w_k mit Erwartungswert 0 und Kovarianz R. In realen Systemen sollte auch diese Matrix R empirisch bestimmt werden. Liegt kein Zustandsraummodell vor, sondern eine Differentialgleichung, so kann diese in ein Zustandsraummodell überführt werden. Die Anwendung im noch folgenden Beispiel sollte hier mehr Klarheit schaffen. Ziel des Kalmanfilters ist es, eine Schätzung der internen Zustände $\hat{x}$ so zu berechnen, dass die Kovarianz P des Schätzfehlers e minimiert wird. Mit Hilfe dieses Zustandsraummodels kann nun der Filterprozess

durchgeführt werden. Dazu wird zunächst eine *a priori Schätzung* $\hat{x}^-$ aus dem Zustandsraummodell errechnet:

$$x_k^- = A_k x_{k-1} + B u_{k-1} w_{k-1} \tag{3}$$

Zur Verbesserung der Schätzung wird der Kalmangain K_k mit dessen Hilfe die *a posteriori Schätzung* $\hat{x}$ in der Art errechnet wird, das die Kovarianz des Schätzfehlers minimiert wird.

$$K_k = \frac{P_{k-1} H^T}{H_k P_k^- + R_k} \tag{4}$$

Damit ergibt sich die *a posteriori Schätzung* $\hat{x}$ wie Folgt:

$$\hat{x_k} = x_k^- + K_k(z_k - H_k x_k^-) \tag{5}$$

dabei wird die Kovarianzmatrix P_k^- benötigit, die noch vor dem Kalmangain aus folgender Gleichung ermittelt werden kann:

$$P_{k-1} = A P_{k-1} A^T + Q_{k-1} \tag{6}$$

Zum Schluss wird die Kovarianzmatrix P für den nächsten Schritt korrigiert:

$$P_k = (I - K_k H_k) P_k^- \tag{7}$$

Für den nächsten Schritt gilt $P_{k-1} = P_k$ und $x_{k-1} = \hat{x_k}$. Am einfachsten initialisiert man P zu Beginn mit der 0 Matrix und den Wert x_{k-1} mit einem Messwert.

3 Anwendungsbeispiel

Für ein simples Anwendungsbeispiel denken wir uns ein Flugzeug, dass mit $\ddot{x}(t) = 6(t - t_0) + 2$ beschleunigt. Durch Integration können wir diese Differentialgleichung lösen:

$$\ddot{x}(t) = 6(t - t_0) + \ddot{x}(t_0) \tag{8}$$

$$\dot{x}(t) = 3(t - t_0)^2 + \ddot{x}(t_0)(t - t_0) + \dot{x}(t_0) \tag{9}$$

$$x(t) = (t - t_0)^3 + \frac{1}{2}\ddot{x}(t_0)(t - t_0)^2 + (t - t_0)\dot{x}(t_0) + x(t_0) \tag{10}$$

Setzt man nun $(t - t_0) = 1$ als diskreten Zeitschritt, und $t_0 = k$ so ergibt sich:

$$\ddot{x}(k+1) = 6 + \ddot{x}(k) \tag{11}$$

$$\dot{x}(k+1) = 3 + \ddot{x}(k) + \dot{x}(k) \tag{12}$$

$$x(k+1) = 1 + \frac{1}{2}\ddot{x}(k) + \dot{x}(k) + x(k) \tag{13}$$

Dies kann auch als Matrixgleichung geschrieben werden:

$$x(k+1) = \begin{pmatrix} 1 & 1 & 0.5 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x(k) \\ \dot{x}(k) \\ \ddot{x}(k) \end{pmatrix} + \begin{pmatrix} \frac{1}{6} \\ \frac{1}{2} \\ 1 \end{pmatrix} 6 \tag{14}$$

Vergleicht man diese Form nun mit dem für das Kalmanfilter gefordete Zustandsraummodell, so fällt auf, das bis auf der Störvektor w bereits die erste Gleichung erfüllt wurde. Da dieser im allgemeinen empirisch ermittelt werden muss, kann in diesem Tutorial ein Wert frei gewählt werden, implementiert wurde der Vektor $w = [0.022, 0.022, 0.022]^T$. Als nächstes legen wir fest, wir möchten alle 3 Größen (Ort,Geschwindigkeit, Beschleunigung) direkt messen, die Matrix H zerfällt hiermit zur Identitätsmatrix im $\mathbb{R}^3$ Für die 2 Gleichung benötigen wir nun auch nur noch den Störvektor v, der analog zu w hier wieder willkürlich mit $v = [0.617, 0.617, 0.617]^T$ gewählt wurde. Zusammenfassend ergibt sich für das Zustandsraummodell dann:

$$x(k+1) = \begin{pmatrix} 1 & 1 & 0.5 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x(k) \\ \dot{x}(k) \\ \ddot{x}(k) \end{pmatrix} + \begin{pmatrix} \frac{1}{6} \\ \frac{1}{2} \\ 1 \end{pmatrix} 6 + \begin{pmatrix} 0.022 \\ 0.022 \\ 0.022 \end{pmatrix} \tag{15}$$

$$z_k = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x(k) \\ \dot{x}(k) \\ \ddot{x}(k) \end{pmatrix} + \begin{pmatrix} 0.617 \\ 0.617 \\ 0.617 \end{pmatrix} \tag{16}$$

Wer noch einmal einen Blick in die benötigten Größen der Kalmangleichungen wirft sieht, dass nur noch die Matrizen Q und R fehlen um den Filter anzuwenden. In den meisten fällen genügt eine Diagonalmatrix mit den Einträgen des Vektors w in Q und eine Diagonalmatrix des Vektors v in R, also :

$$Q = \begin{pmatrix} 0.022 & 0 & 0 \\ 0 & 0.022 & 0 \\ 0 & 0 & 0.022 \end{pmatrix} \quad R = \begin{pmatrix} 0.617 & 0 & 0 \\ 0 & 0.617 & 0 \\ 0 & 0 & 0.617 \end{pmatrix} \tag{17}$$

In der nun folgenden Implementierung wurde eine Funktion geschrieben, die die korrekten Werte der Zustandsgrößen $(x(t), \dot{x}(t), \ddot{x}(t))$ berechnet und mit einer Zufallszahl verfälscht, um die Störgrößen zu simulieren. Als initiale Matrix für P wurde die Nullmatrix und als Startwert für den Zustandsvektor eine Messung für $t_0 = 0$ Sekunden. Es folgt nun eine Beispielhafte Implementierung des obigen Beispiels

4 Beispielhafte Implementierung

```
//Funktion fuer einen Stoervektor
Vector3D frand() {
          double r = 2*((rand()/(double)RAND_MAX)) - 0.5;
          Vector3D d(900*r,130*r,300*r);
    return d;
}
//_____________________________________

//Funktion um die korrekten Werte zu berechnen
Vector3D getReal(double t){
  double z = 6*t+2;
  double y = 3*t*t+2*t;
  double x = t*t*t +t*t ;
  Vector3D real(x,y,z);
  return real;

}
//Initialisierung
    Vector3D x_est_last (0,0,0);
//Initialmatrix P, Nullmatrix
    Matrix3D P_last(x_est_last,x_est_last,x_est_last);

//Stoermatrix Q
    Vector3D q(0.022,0.022,0.022);
    Matrix3D Q(q);

 //Stoermatrix R
    Vector3D r (4.617,4.617,4.617);
    Matrix3D R(r);

//Kalmangain , wird im ersten Schritt berechnet
    Matrix3D K;
// Matrix P , wird im ersten Schritt berechnet
    Matrix3D P;
//Matrix H, die Identitaet
    Matrix3D H;
//Systemmatrix A, wie oben angegeben
    double arr[]={1,1,0.5,0,1,1,0,0,1};
    Matrix3D A (arr);
// wird in den apriori Gleichungen errechnet
    Matrix3D P_temp;
```

```
// wird in den apriori Gleichungen errechnet
    Vector3D x_temp_est;
//wird in den aposteriori Gleichungen errechnet
    Vector3D x_est;
// Zusammenfassung des Vektors 6*u + w
    Vector3D w(1.022,3.022,6.022);
// eigentlicher Messwert , wird von frand() errechnet
    Vector3D x_measured;
// Initialisiert als Messwert
        x_est_last = getReal(0.0).vecAdd(frand());
//_____________________________________

// 30 Schritte
for(int i = 0;i<30;i++){
//Prediction , a priori Gleichungen
  x_temp_est =  x_est_last.matVecMult(A).vecAdd(w);
  P_temp = A.mMult(P_last.mMult(A.transpose())).mAdd(Q);

//Kalmangain
  K = P_temp.mMult(P_temp.mAdd(R).invert());
//measure
  x_measured = getReal(i).vecAdd(frand());

//correct , a posteriori
  Vector3D temp;
  temp = (x_measured.vecSub(x_temp_est).matVecMult(K));
  x_est = x_temp_est.vecAdd(temp);
  P = H.mSub(K).mMult(P_temp);


//Update System
  P_last = P;
  x_est_last = x_est;

}
```