

# Übungsaufgabe in Luft- und Raumfahrtlabor

## Linearer Kalman Filter

Stefan Lindörfer

22. Januar 2021

Einführende Übung zum Kalmanfilter mit einem eindimensionalen Beispiel. Ausführlich aufbereitet um auch als Anleitung/Nachschlagewerk verwendet werden zu können.

### 1) Aufgabe:

#### a) System Design

Es gilt, die Zustandsmatrix  $F$  zu finden, sodass die Gleichung

$$\vec{x}(k+1) = F \cdot \vec{x}(k) \quad (1)$$

den Zustandsraum  $\vec{x}$ , bestehend aus dem Gleichungssystem

$$h(k+1) = h(k) + \Delta t \cdot v(k) + \frac{1}{2} \Delta t^2 \cdot a(k) \quad (2)$$

$$v(k+1) = v(k) + \Delta t \cdot a(k) \quad (3)$$

$$a(k+1) = a(k) \quad (4)$$

korrekt abbildet. Für den Zustandsraum  $\vec{x}$  gilt somit:

$$\vec{x}(k) = \begin{pmatrix} h(k) \\ v(k) \\ a(k) \end{pmatrix} \quad (5)$$

Die Zustandsmatrix  $F$  muss also so gewählt werden, dass durch eine Matrix-Vektor-Multiplikation (Gl. 1) die oben genannte Gleichungen 2, 3 und 4 der darauffolgenden Iteration entstehen. Damit ergibt sich für  $F$ :

$$F = \begin{pmatrix} 1 & \Delta t & \frac{1}{2} \Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix} \quad (6)$$

#### b) Kalman Filter mit Beschleunigungssensor

Ein lineares Kalman Filter soll den Systemzustand eines Multikopters schätzen. Dabei wird lediglich eine Dimension (Höhe) betrachtet und der Zustandsraum (Gl. 5) zugrunde gelegt.

Zunächst soll ein geeigneter Startzustand  $\vec{x}(0)$  gewählt werden. Sind keine Informationen über den Startzustand vorhanden, kann dies dem Filter ebenfalls mitgeteilt werden, in dem in das entsprechende Zustandsfeld 0 eingetragen wird.

Vermutlich startet ein UAV auf festem Untergrund, der als 0 angenommen werden kann, aber auch jeder andere Wert würde hier grundsätzlich Sinn machen (Anfangshöhe), etwa falls von einem Tisch aus gestartet werden soll. Hier wird von einem Startpunkt am Boden ausgegangen. Damit wird  $h(0) = 0$  festgelegt.

Abhängig vom Startpunkt ist auch die Anfangsgeschwindigkeit  $v(0)$  einzutragen. Da grundsätzlich die Bestimmung einer Momentangeschwindigkeit zu einem exakten Zeitpunkt problematisch ist, wird hier ebenfalls 0 gewählt, zumal von einem festen Punkt gestartet wird und dort ebenfalls keine Anfangsgeschwindigkeit angenommen wird.

Zuletzt ist die Beschleunigung  $a(0)$  festzulegen. Es wird davon ausgegangen, dass zum Zeitpunkt  $t_0$  bereits eine entsprechende Auftriebskraft besteht, sodass eine Beschleunigung von  $0 \frac{m}{s^2}$  wirkt, das UAV also seine Lage nicht nach oben oder unten verändert würde.

Damit wird der Startzeitpunkt  $\vec{x}(k=0) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$  gesetzt.

Als nächstes soll aus den Messdaten des Beschleunigungssensors der ersten 10 Sekunden, die Messrauschkovarianzmatrix  $R_{accel}$  berechnet werden. Diese gibt Auskunft über die Streuung der Messwerte. Da die Frequenz des Sensors  $f_{accel} = 200 \text{ Hz}$  beträgt, müssen so 2.000 Datensätze berücksichtigt werden. Die Problemstellung, wie  $P$ ,  $Q$  und  $R_x$  zu bestimmen sind, wird auch in den Folien zur Übung aufgegriffen. Dort wird  $R = \sigma_{accel}^2 = var_{accel}$  gesetzt. Damit folgt:

$$R_{accel} = 0,1296 \left(\frac{m}{s^2}\right)^2 \quad (7)$$

Da nur ein Sensor im Filterprozess zur Anwendung kommt, ist  $R_{accel}$  eine  $1 \times 1$ -Matrix und kann als Skalar betrachtet werden.

Außerdem ist die Vorhersagematrix  $H_{accel}$  zu bestimmen. Diese gibt Auskunft darüber, was gemessen wird und in welchem Verhältnis es zum Zustandsvektor  $\vec{x}(k)$  steht.

Gemessen wird die auf das UAV wirkende Beschleunigung in bekannten SI-Einheiten. Grundsätzlich muss berücksichtigt werden, dass der Sensor auch die Erdbeschleunigung misst, was später im Prozess (siehe Quellcode) noch berücksichtigt werden muss. Mit diesem Vorgehen lässt sich die Matrix aufstellen:

$$H_{accel} = (0 \ 0 \ 1) \quad (8)$$

Die Messwerte wirken sich zuerst ausschließlich auf die Beschleunigung des Zustandsvektors aus (1 im Feld für Beschleunigung) und werden durch die Multiplikation des Zustandsraumes mit der Zustandsmatrix auch auf die

anderen Eigenschaften Höhe und Geschwindigkeit übertragen. Diese Matrix besitzt eine Größe von  $3 \times 1$ , da drei Systemzustandswerte und ein Sensorinput in diesem Filterprozess existieren.

Abschließend müssen noch die restliche Gleichungen des Kalman-Filters implementiert werden. Nachfolgend befindet sich der gesamte Vorgang ohne etwaige Matrizendeklarationen. Insgesamt ergibt sich damit folgender MATLAB-Quellcode:

```

for ia = 1:length(y_accel) - 1 % Laenge der Messwerte
    %% Vorhersage betreffend des Systemstatus:
    % Vorhersage System-Status:
    x(:, ia + 1) = F * x(:, ia);
    % Vorhersage Systemstatuskovarianz:
    P = F * P * transpose(F) + Q;

    % Vorhersage Messungen:
    z_accel = H_accel * x(:, ia + 1);
    % Einfließen der Messdaten:
    v = (y_accel(ia) - 9.81) - z_accel; % Berücksichtigung von g als
                                         % Offset-Fehler

    % Innovationkovarianz:
    S = (H_accel * P * transpose(H_accel)) + R_accel;
    % Filter Gain (Kalman-Gain):
    W = P * transpose(H_accel) * inv(S);

    % Systemzustand updaten:
    x(:, ia + 1) = x(:, ia + 1) + (W * v);
    % Kovarianzmatrix updaten:
    P = (eye(3) - (W * H_accel)) * P;
end

```

Wird das Filter nun ausgeführt um den Systemzustand zu schätzen, ergeben sich für 100 Sekunden Laufzeit die folgenden graphischen Aussagen:

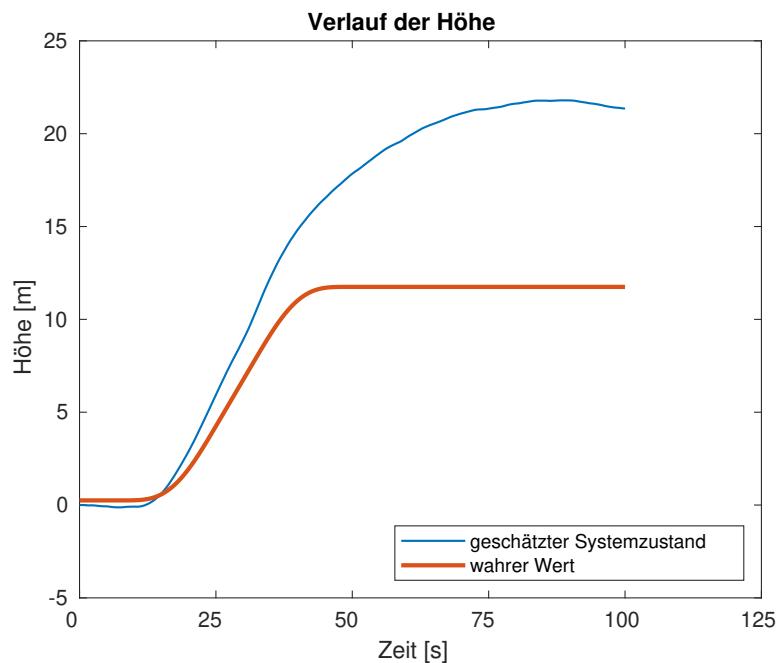


Abbildung 1: Darstellung des zeitlichen Verlaufes der vertikalen Höhe des Multikopters

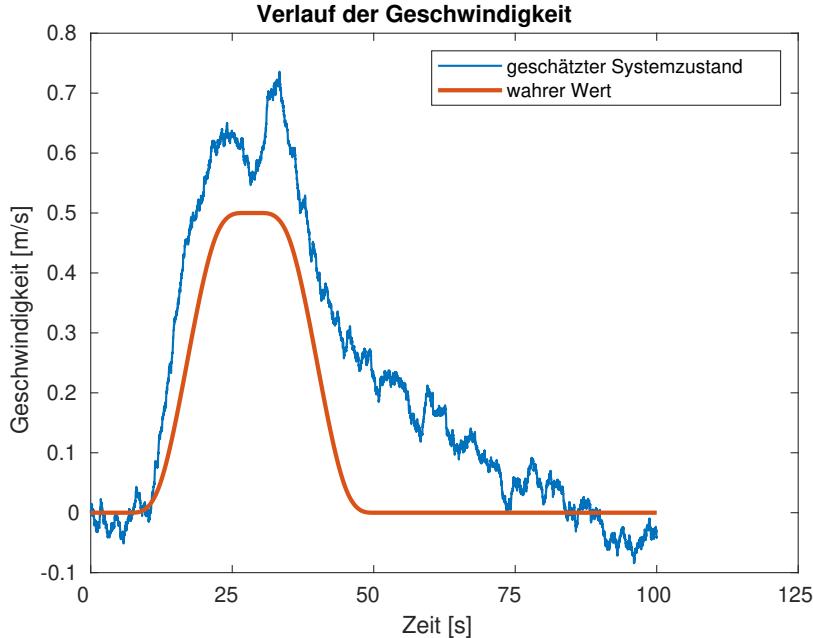


Abbildung 2: Darstellung des zeitlichen Verlaufes der vertikalen Geschwindigkeit des Multikopters

Auffällig ist hierbei in Abbildung 1 etwa, dass das Filter seine Höhe ab der Hälfte der Zeit um fast das Doppelte überschätzt. Auch der Geschwindigkeitsverlauf in Abbildung 2 weist größere Differenzen mit dem wahren Wert auf. Dies könnte daran liegen, dass etwaige Fehler ebenfalls aufsummiert werden (und so immer größer werden) und das dem Filter nur eine Sensoreingabequelle in Form des Accelerometers zur Verfügung steht. Diese gibt ihm aber keine Auskunft über die tatsächlich erreichte Höhe. Hier muss sich bisher ausschließlich auf das mathematische Modell (Gl. 2-4) verlassen werden.

Um die Schätzung durch den Filter exakter zu gestalten, wird im folgenden ein vertikaler Abstandslaser (Lidar) als zusätzliche Messdateneingabe dem Filter zur Verfügung gestellt. Dieser gibt den vertikalen Abstand des Multikopters vom Nullpunkt an.

Dafür ist zunächst – wie auch beim Accelerometer – die Vorhersagematrix  $H_{lidar}$  zu bestimmen. Die Werte des Sensor werden in Zentimetern gespeichert, das Filter selbst rechnet jedoch bereits mit der Einheit Meter. Da die Vorhersagematrix auch das Verhältnis der Messwerte zum Zustandsvektor angibt, kann die Matrix wie folgt aufgestellt werden:

$$H_{accel} = \begin{pmatrix} 100 & 0 & 0 \end{pmatrix} \quad (9)$$

Die Dimension der Matrix gleicht der von  $H_{accel}$  – es gibt drei Zustandsvariablen und nur eine Sensoreingabe. Die vom Lidar gemessene Höhe hat keine Auswirkungen auf die Beschleunigung oder die Geschwindigkeit, weshalb diese Felder eine 0 enthalten. Die hinzukommenden Daten können als Verfeinerung des eigentlichen Ergebnisses angesehen werden, da bisher aufgetretene Fehler durch den Beschleunigungssensors so minimiert werden.

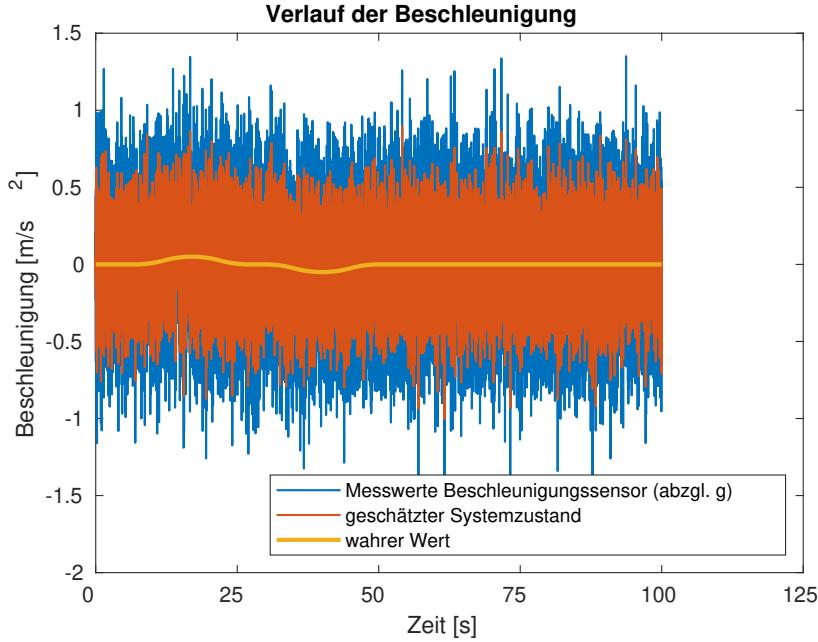


Abbildung 3: Darstellung des zeitlichen Verlaufes der vertikalen Beschleunigung des Multikopters

Da die rohen Messdaten des Lidars um das 100-fache größer ( $cm$  in  $m$ ) in das Filter gegeben werden, wird das Verhältnis des Zustandsvektors mit dem Faktor 100 entsprechend angepasst.

Die Messrauschkovarianzmatrix des Abstandslasers  $R_{lidar}$  wird mit den Messdaten des Lidars genauso bestimmt wie bereits zu Beginn mit dem Beschleunigungssensor, lediglich die verringerte Messfrequenz von  $20\ Hz$  muss berücksichtigt werden. Damit ergibt sich ein Wert von

$$R_{lidar} = 0, 2612 \text{ } cm^2 \quad (10)$$

Die weiteren zu implementierenden Gleichungen werden nach dem obigen Schema (MATLAB-Quellcode) vervollständigt. Dabei muss ebenfalls auf die verringerte Frequenz des Abstandslasers geachtet werden.

Wird nun erneut eine Filterung durchgeführt, welche zusätzlich zum Beschleunigungssensor noch den Abstandslaser als Eingabequelle beinhaltet, werden folgende Grafiken gewonnen:

### c) Auswertung

Sofort auffällig ist die Tatsache, dass das Filter wie in Abbildung 4 ersichtlich, mit dem Abstandslaser als zusätzliche Orientierungseingabe, einen fast exakten Verlauf mit dem wahren Wert aufweist. Dies erreicht das Filter in Abbildung 1 (wie bereits in Aufg. b) festgestellt) bei weitem nicht: Fehler

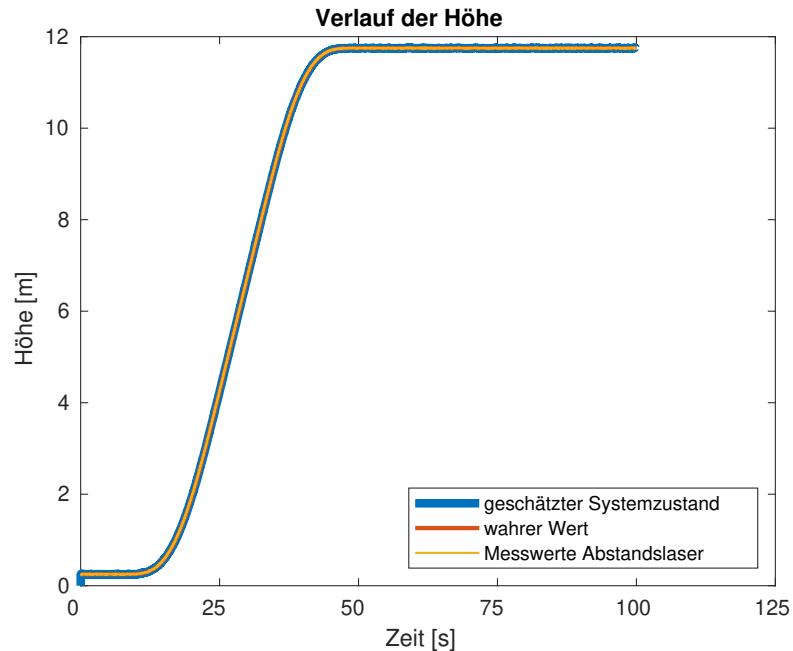


Abbildung 4: Darstellung des zeitlichen Verlaufes der vertikalen Höhe des Multikopters mit Hinzunahme des Abstandslasers

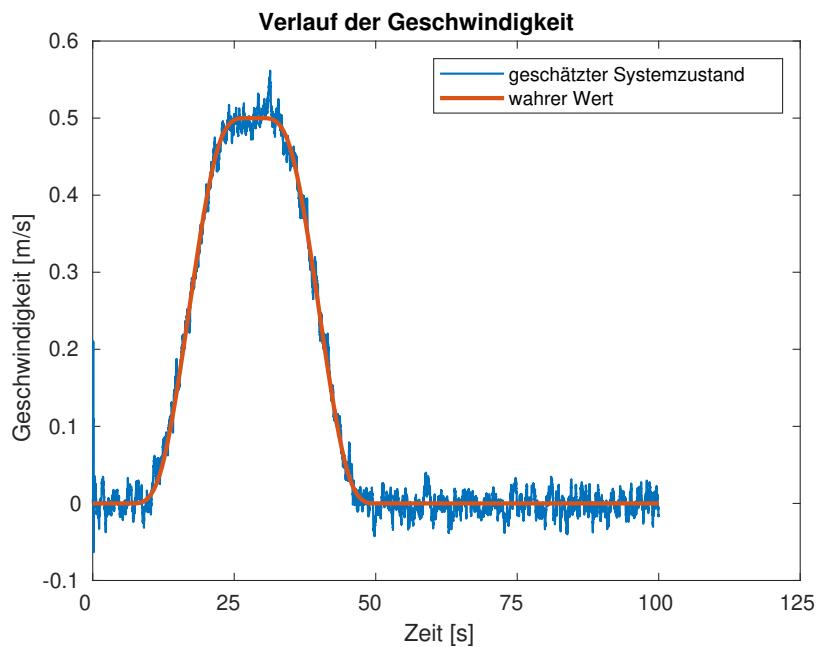


Abbildung 5: Darstellung des zeitlichen Verlaufes der vertikalen Geschwindigkeit des Multikopters mit Hinzunahme des Abstandslasers

werden dort aufsummiert und ergeben letztendlich fast die doppelte Höhe.

Auch bei den beiden Abbildungen zur Geschwindigkeit (Abb. 2 und 5) lässt sich – wenn auch in geringerem Maße – eine solche Beobachtung feststellen. Hier weicht das Filter ohne Lidar, als zweite Sensoreingabequelle, vom wahren Wert in vergleichbarer Größenordnung ab. Auch eine langsamere Konvergenz zu diesem lässt sich ab der Hälfte der Zeit deutlich feststellen,

währenddessen die zweite Version des Filters (siehe Abb. 5), ebenfalls – mit leichtem Prozessrauschen zwar – einen sehr konvergenten und dynamischen Verlauf um die Soll-Geschwindigkeit nimmt.

Auf die Beschleunigung hat das Hinzunehmen des Abstandslasers jedoch keine erkennbaren Auswirkungen, die entsprechende Grafik ist augenscheinlich identisch mit Abbildung 3, was auch dadurch erklärt werden könnte, dass die Beschleunigung als erstes durch die Zustandsgleichung (Gl. 4) Einfluss auf das Filter bzw. dem zugrundeliegendem Zustandsraum (Gl. 5) erhält und kein unmittelbarer Zusammenhang mit dem Abstandslaser besteht.

Im folgenden werden nun die Werte der drei Matrizen  $R_{lidar}$ ,  $R_{accel}$  und  $Q$  variiert, um Auswirkungen davon sichtbar zu machen, zu untersuchen und miteinander zu vergleichen sowie ein grobes Einflussverhalten der Matrizen abzuleiten. Dazu werden die Werte einmal sehr hoch ( $> 10^6$ ) und einmal sehr niedrig ( $< 10^{-6}$ ) gewählt – die anderen Matrizen jeweils in ihren ursprünglichen Werten beibehalten –, um so eindeutige Unterschiede im Verhalten des Filterprozesses sichtbar zu machen. Tabelle 1 fasst grob die Unterschiede zusammen. Die nachfolgenden Abbildung sollen die Verände-

	<b>Höhe</b>	<b>Geschwindigkeit</b>	<b>Beschleunigung</b>
$R_{lidar} \rightarrow 10^6$	leichtes Übersteuern (s Abb. 6a)	Dynamisch (s. Abb. 6b)	Unverändert (s. Abb. 3)
$R_{lidar} \rightarrow 10^{-6}$	Unverändert (s. Abb. 1)	Rauschen (s. Abb. 7a)	Rauschen (s. Abb. 7b)
$R_{accel} \rightarrow 10^6$	Unverändert (s. Abb. 1)	Rauschen (s. Abb. 8a)	Rauschen (s. Abb. 8b)
$R_{accel} \rightarrow 10^{-6}$	leichtes Übersteuern (s. Abb. 9a)	Dynamisch (s. Abb. 9b)	Unverändert (s. Abb. 3)
$Q \rightarrow 10^6$	Unverändert (s. Abb. 1)	Dynamisch (s. Abb. 10a)	Rauschen (s. Abb. 10b)
$Q \rightarrow 10^{-6}$	Unverändert (s. Abb. 1)	Glättung (s. Abb. 11a)	Glättung (s. Abb. 11b)

Tabelle 1: Zusammenfassung d. Ergebnisse der Variation der Matrizenwerte

rungen grafisch deutlich machen. Hierbei werden Grafiken nur angegeben, wenn durch die erfolgte Modifizierung des Filters ein auffälliger Unterschied zu den vorherigen Verläufen aus den Abbildungen 1-3 bzw. 4 & 5 bemerkbar ist.

Wird der Wert der Messrauschkovarianzmatrix des Abstandslasers  $R_{lidar}$  stark erhöht, wirkt sich dies neben geringfügig schlechteren Schätzenwerten der Höhe (Abb. 6a) und mehr Prozessrauschen auf Seiten der Geschwindigkeit (Abb. 6b) (worauf im nächsten Abschnitt näher eingegangen wird) scheinbar auch auf eine größere Trägheit des Filters aus: In geringem Ausmaß treten außerdem Überschwingungen auf und es erfordert deutlich mehr Zeit, wieder auf die wahren Werte zurückzukommen: Das Filter ist nicht mehr so dynamisch wie noch im Ausgangszustand. Dazu gesagt werden muss allerdings, dass die Werte ins unnormal hohe geändert wurden um so eine Reaktion zu provozieren und Änderungen in normalen Intervallen in diesem Fall keine solchen Auswirkungen zeigen dürften.

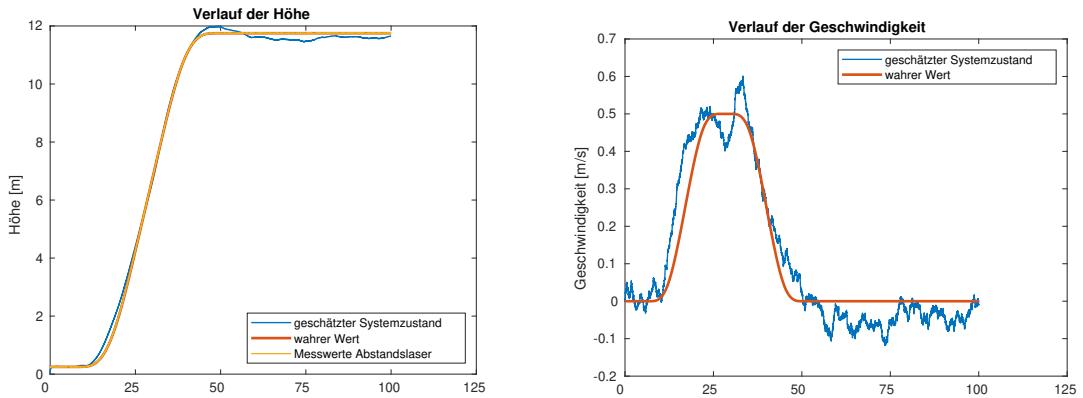


Abbildung 6: Filterverhalten bei hohem  $R_{lidar}$

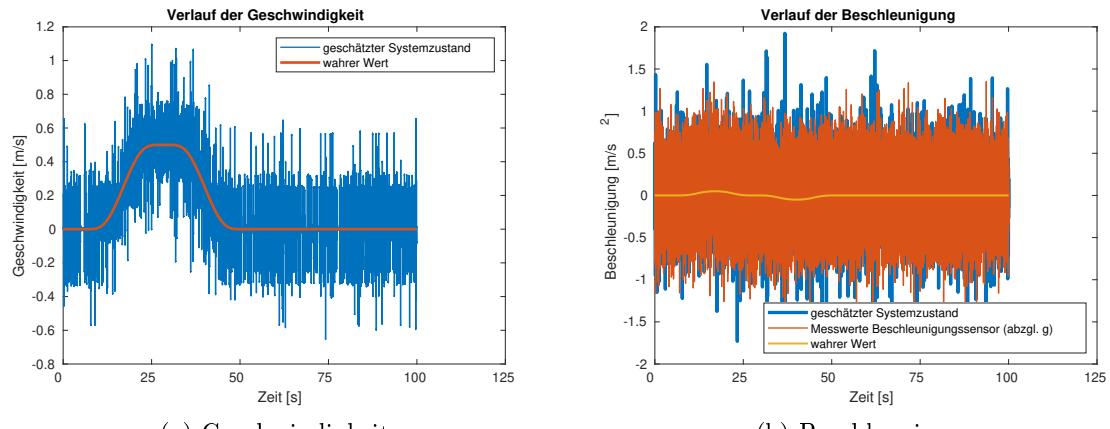


Abbildung 7: Filterverhalten bei niedrigem  $R_{\text{filter}}$

Das bereits festgestellte Rauschen und leichte Überschwingen in Abbildung 6, wird auch im nächsten Beispiel sichtbar: Die Abbildungen 7 und 8 sind sich sehr ähnlich, obwohl zwei zunächst voneinander unabhängige Werte nacheinander in unterschiedliche Richtungen geändert werden: Die Kovarianz bzw. Varianz trifft – wie bereits umrissen – eine Aussage über die Streuung der Messwerte. Daraus folgt, dass sich das Filter bei einer Messrauschkovarianzmatrix  $R_x$  mit kleineren Werten eher auf die Zuverlässigkeit der Messdaten des betreffenden Sensors verlassen kann als dies bei größeren Werten in der Matrix der Fall wäre, wo eine höhere Kovarianz/Varianz auch gleichzeitig einen höheren Rauschanteil bedeuten kann. In beiden Fällen ist der Verlauf der Höhe jedoch fast identisch zu Abbildung 4, da der jeweils andere Sensor eine entsprechend niedrigere Varianz aufweist (siehe Gl. 7 und Gl. 10), womit dem Filter wenigstens eine zuverlässige Datenquelle zur Verfügung steht.

Die Darstellung des Filterverhaltens mit niedriger Varianz für die Messrauschkovarianzmatrix des Beschleunigungssensor (Abb. 9) weist deutliche

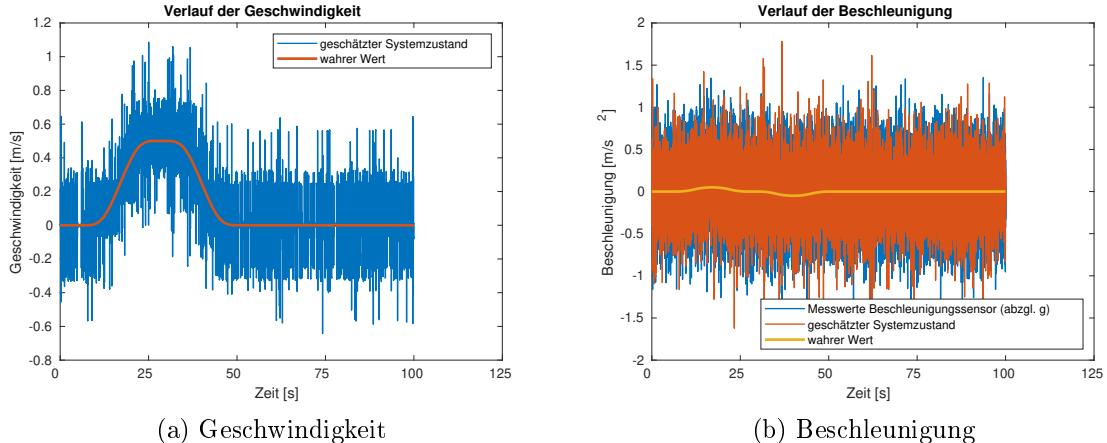


Abbildung 8: Filterverhalten bei hohem  $R_{accel}$

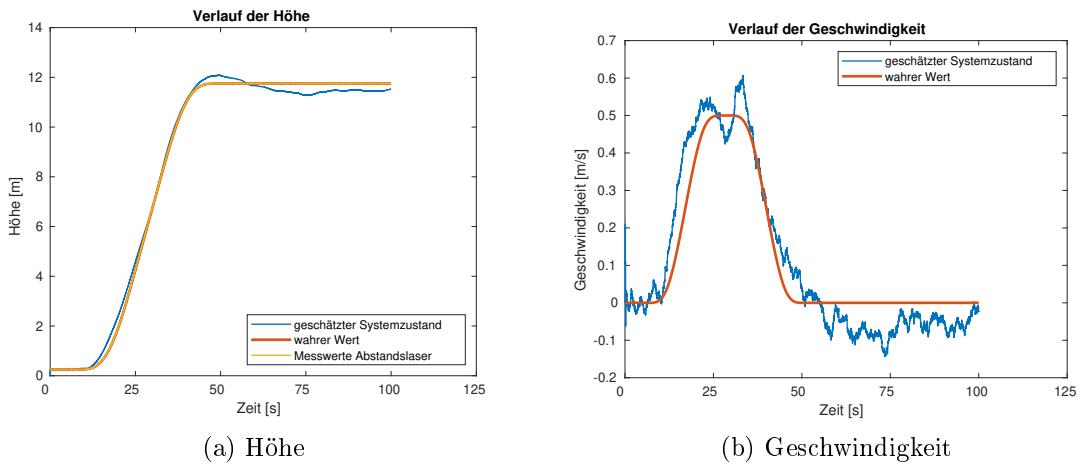


Abbildung 9: Filterverhalten bei niedrigem  $R_{accel}$

Ähnlichkeiten mit Abbildung 6 auf: Die Höhe wird zu Beginn und Ende zwar gut aber nicht exakt geschätzt, ebenso ist ein größerer Rauschanteil in der Abbildung zur Geschwindigkeit (Abb. 9b) bemerkbar. Obwohl ein niedrigerer Wert in der Matrix die Zuverlässigkeit der Messdaten erhöht, wirkt sich dies hier scheinbar kontraproduktiv aus. Eine mögliche Erklärung dafür ist, dass - wie in Abbildung 3 sichtbar - die Messdaten des Beschleunigungssensors einen großen Rauschanteil besitzen (zusammen mit der hohen Messfrequenz von 200 Hz) und deswegen dem Filter den Trugschluss vermitteln, diese Daten besäßen eine große Zuverlässigkeit und Genauigkeit. Durch den parallel wirkenden Einfluss des Abstandslasers, wird dennoch die Höhe relativ genau geschätzt.

Wird die Prozessrauschkovarianzmatrix  $Q$  geändert, die in dieser Filterimplementierung nur im Zusammenspiel mit der Varianz des Beschleunigungssensors (übersetzt:  $R_{accel}$ ) wirkt, lassen sich zwei Verhaltensfälle beobachten: Abbildung 10 zeigt, dass dem Verlauf der Geschwindigkeit (Abb. 10a) durch die Zustandsschätzungen sehr gut gefolgt wird, also auf Änderungen sehr dynamisch reagiert wird. Der geschätzte Zustand der Beschleunigung

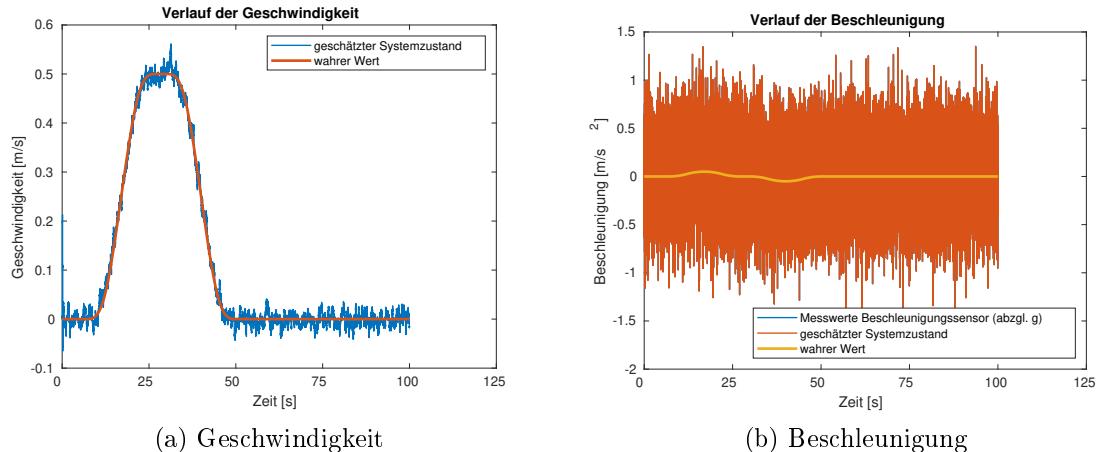


Abbildung 10: Filterverhalten bei hohem  $Q$

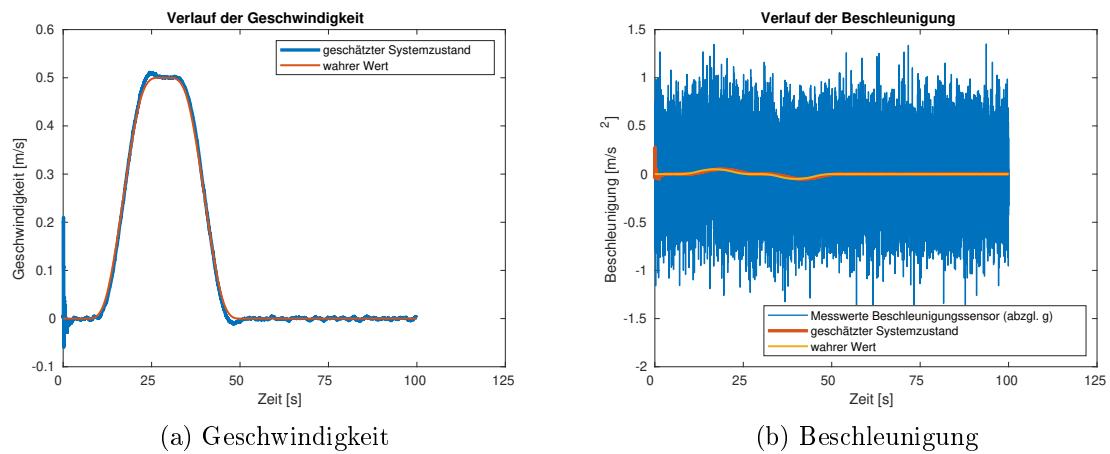


Abbildung 11: Filterverhalten bei niedrigem  $Q$

allerdings weist stark erhöhtes Rauschen auf. Auf die Schätzung der Höhe hat dies jedoch scheinbar keine Auswirkungen, denn diese wird nach wie vor unverändert gut geschätzt.

Das umgekehrte Verhalten – bei niedrigem  $Q$  – zeigt, dass Rauschen minimal und die Schätzung der Geschwindigkeit nahezu optimal durchgeführt wurden. Kleine bemerkbare Schwingungen in Abbildung 11a lassen vermuten, dass dieses Verhalten (geringes Rauschen) auf Kosten der Dynamik geht. Abbildung 11b demonstriert eindrucksvoll, wie der geschätzte Wert der Beschleunigung ebenfalls nahezu mit dem wahren Wert verschmilzt und noch vorhandenes Rauschen (sichtbar in Abb. 3) vollständig geglättet wurde.

## 2) Aufgabe:

Jetzt wird ein verkleinerter Systemzustand betrachtet, mit dem Ziel einer geringeren Rechenzeit. Dieser Trick wird in eingebetteten Systemen verwendet, da in diesen meist weniger Ressourcen für Berechnungen zur Verfügung stehen. Der Geschwindigkeitsvorteil wird später noch gezeigt. Für den verkleinerten System-

zustand gilt:

$$\vec{x}(k) = \begin{pmatrix} h(k) \\ v(k) \end{pmatrix} \quad (11)$$

Außerdem sollen die Messungen des Beschleunigungssensors nicht als Messung, sondern als Systemeingabe  $u(k)$  verwendet werden. Damit entfällt der Update-schritt für die Daten des Beschleunigungssensors aus Aufgabe 1). Das Zustandsraummodell für diese Version des Filters lässt sich damit wie folgt beschreiben:

$$x(k+1) = F \cdot \vec{x}(k) + \vec{G} \cdot u(k); \quad u(k) = y_{accel}(k) \quad (12)$$

Die Prozessrauschkovarianzmatrix  $Q$  ist gegeben mit:

$$Q = \vec{G} \cdot R_{accel} \cdot \vec{G}^T \quad (13)$$

### a) System Design

Zunächst muss für den Zustandsraum  $\vec{x}(k)$  (Gl. 5) wieder die Zustandsma-trix  $F$  bestimmt werden, sowie die Eingangsmatrix  $\vec{G}$  (die nicht zwingend ein Vektor sein muss).

Da der Updateschritt für die Beschleunigungsdaten – wie eingangs erwähnt – entfällt, verkleinert sich der Systemzustand. Die Gleichungen werden nun nach Schema Gl. 12 zusammengesetzt. Damit ergibt sich folgende Zustands-matrix:

$$F = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \quad (14)$$

Damit als Resultat die Gleichungen 2-4 entstehen, folgt für die Eingangs-matrix:

$$\vec{G} = \begin{pmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{pmatrix} \quad (15)$$

Eingesetzt in Gl. 12 ergebt sich damit die drei zugrundeliegenden Zustands-gleichungen.

### b) Kalman Filter

Bevor der Filter starten kann, sind weitere Gleichungen zu implementie-ren. Der verkleinerte Systemzustand resultiert teilweise in weniger Rechen-schritten, so entfällt der eingangs erwähnte Updateschritt der Beschleu-nigungssensormessdaten, womit sich auch ein verkürzter Gesamtquellcode (MATLAB) ergibt. Erneut werden Matrizendeklarationen nicht mit aufge-führt. Im Unterschied zum Quellcode aus Aufgabe 1) wird hier auch die Integration des Abstandslaser in den Filterprozess demonstriert, die sich in beiden Filter-Versionen nicht unterscheidet. Der Filterprozess muss gene-rell die Messfrequenzen der Sensoreingabeketten korrekt berücksichtigen. Der Beschleunigungssensor misst mit einer Frequenz von 200 Hz und der Abstandslaser mit 20 Hz.

```

for ia = 1:length(y_accel)-1 % Laenge der Messwerte
    %% Vorhersage betreffend des Systemstatus,
    % Vorhersage System-Status mit neuem Zustandsraummodell:
    x(:, ia+1) = F * x(:, ia) + G * (y_accel(ia)-9.81); % u(k) = y_accel
    % Offset g
    %% Vorhersage Systemstatuskovarianz:
    P = F * P * transpose(F) + Q;

    %% Update der Lidar Messungen:
    if(time_accel(ia) >= time_lidar(il))
        % Vorhersage Messungen:
        z = H * x(:, ia+1);
        % Einfließen der Messdaten
        v = y_lidar(il) - z;

        % Innovationskovarianz:
        S = (H * P * transpose(H)) + R_lidar;
        % Filter Gain (Kalman-Gain):
        W = P * transpose(H) * inv(S);

        % Systemzustand updaten:
        x(:, ia+1) = x(:, ia+1) + (W * v);
        % Prozesskovarianzmatrix updaten:
        P = (eye(2) - (W * H)) * P;

        % Increment lidar index
        il = il + 1;
    end
end

```

Wird der Filter mit diesen Einstellungen und dem verkleinerten Systemzustand bzw. den leicht veränderten Gleichungen ausgeführt, ergeben sich nachfolgende graphischen Ausgaben: Die Darstellung der Höhe (Abb. 12a) sowie der Geschwindigkeit (Abb. 12b) stimmen augenscheinlich mit den Abbildung 4 und 5 überein. Lediglich die Abbildung der Beschleunigung (Abb. 12c) weist im Mittelteil einen Ausschlag nach oben auf, der in der vorherigen Filerversion (Abb. 3) nicht existiert.

### c) Auswertung

Als letztes soll eine Auswertung der mittleren Laufzeit der beiden Versionen des Kalman Filters aus Aufgabe 1) und dieser Aufgabe durchgeführt werden. Dafür werden jeweils die Zeiten von je 100 Iterationen der Filter gemittelt und miteinander verglichen. Damit ergeben sich die folgenden mittleren Laufzeiten:

$$t_{Filter_1} = (72 \pm 29) \mu s \quad (16)$$

$$t_{Filter_2} = (47 \pm 21) \mu s \quad (17)$$

Somit ist der Filter in der zweiten Version ( $t_{Filter_2}$ ) im Mittel um rund ein Drittel schneller als Version eins: Durch den verkleinerten Systemzustand entfallen eine ganze Reihe an Rechnungen, beispielsweise die Bestimmung des Filter-Gains aus den Daten des Beschleunigungssensors inklusive einer benötigten Invertierung der Matrix  $S$  – ein relativ aufwendiges Rechenverfahren –, welches in dieser Implementierung eingespart werden kann.

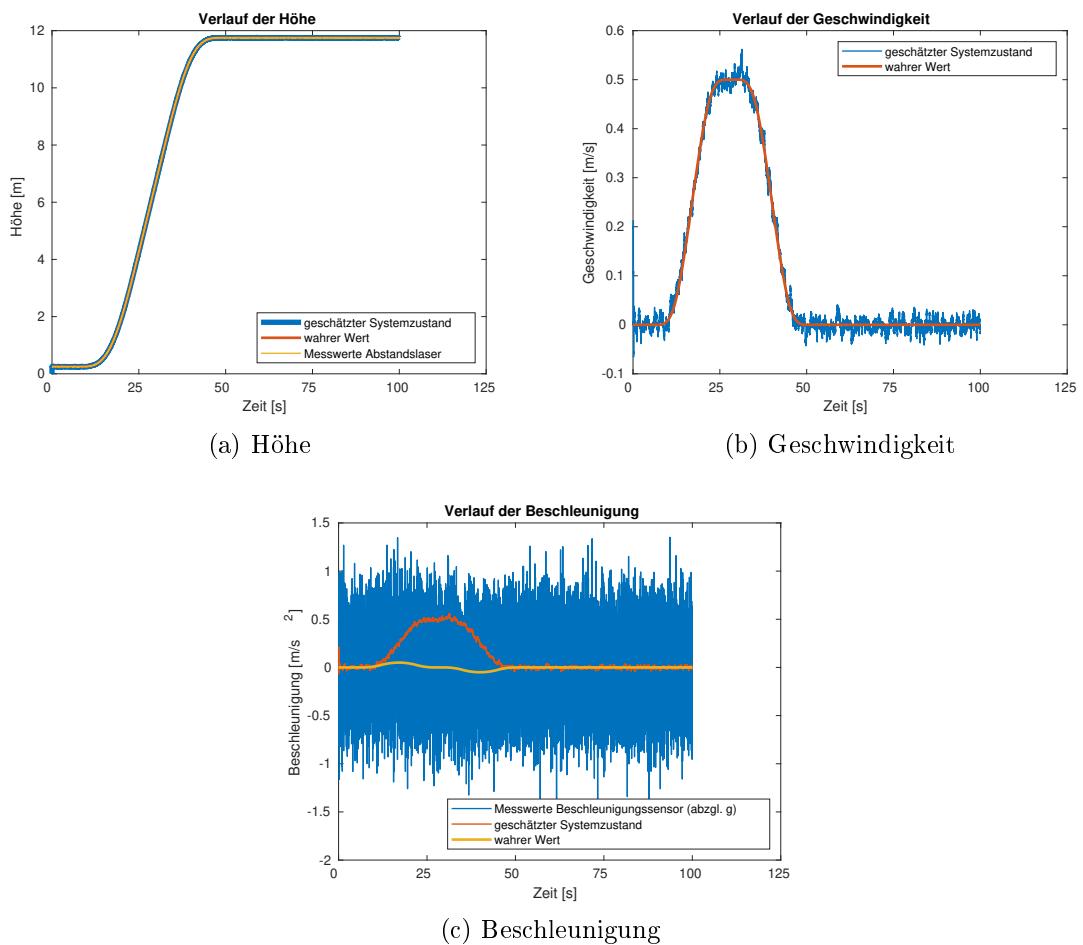


Abbildung 12: Filterverhalten mit verkleinertem Systemzustand