

Hochschule Karlsruhe - Technik und Wirtschaft
Fakultät für Maschinenbau und Mechatronik

**Modellbasierter Entwurf und Embedded
Implementierung eines Mehrgrößenreglers für
einen balancierenden Würfel**

Bachelorarbeit (B.Eng.)

von
Michael Meindl
Matrikelnr.: 45171

Betreuer: Prof. Dr.-Ing. Joachim Wietzke

Karlsruhe, 01.02.2017 bis 31.05.2017

Inhaltsverzeichnis

1	Einleitung	5
2	Modellbildung Würfel auf Kante	7
2.1	Untersuchung der Kinematik	8
2.2	Untersuchung der Kinetik	12
3	Regelungstechnik	15
3.1	Zustandsraumdarstellung für kontinuierliche Systeme	16
3.2	Zustandsraumdarstellung für zeitdiskrete Systeme	20
3.3	Entwurf von Zustandsregelungen	22
3.3.1	Reglerentwurf durch Eigenwertvorgabe	22
3.3.2	Linear quadratisch optimale Regelung	24
3.4	Verifizierung des Reglers an der Regelstrecke	26
3.5	Beobachtung der Messabweichungen	29
4	Modellbildung Würfel auf Ecke	31
4.1	Systemparameter	32
4.2	Untersuchung der Kinematik	34
4.3	Untersuchung der Kinetik	38
5	Reglerentwurf für das Balancieren auf einer Ecke	45
5.1	Modell basierter Reglerentwurf	46
5.2	Untersuchung der Stellgrößenbeschränkung	52
6	Erfassung der Zustandsgrößen	55
6.1	Winkelschätzung	56
6.2	Justierung der Sensoren	60
7	Software	61
7.1	Zielplattform, Sensorik und Aktorik	62
7.1.1	Peripherie	62
7.2	Implementierung des Signalflusses	65
7.3	Komponentenarchitektur für mechatronische Anwendungen	70
7.4	Entwurf der Regelungskomponente	73

Kapitel 1

Einleitung

Im Sommersemester 2016 begann im Rahmen eines Entwicklungsprojektes an der Hochschule Karlsruhe die Entwicklung eines balancierend Würfels. Die Idee hierfür stammt von dem Cubli, welcher an der ETH Zürich entwickelt wurde. Die Aufgabe des Projekts besteht darin einen Würfel mit einer Kantenlänge von 15cm zu entwerfen, der in der Lage ist auf seinen Kanten und Ecken zu balancieren. Hierfür sind im Inneren des Würfels drei Motoren montiert an welchen jeweils eine Schwungmasse angebracht ist. Die Drehmomente der Motoren dienen als Stellgrößen eines Reglers um den Würfel in einer aufrechten Position zu balancieren. Des weiteren können die Schwungmassen genutzt werden um den Würfel aus einer beliebigen Position auf seine Kanten bzw. Ecken aufzurichten. Hierfür werden die Schwungmassen zunächst beschleunigt und anschließend mittels einer mechanischen Bremse gestoppt. Dadurch wird deren Drehimpuls auf das Würfelgehäuse übertragen.

In dem vorangehenden Entwicklungsprojekt wurde zunächst eine einzelne Würfelseite mit einem Motor konzipiert. Dieses Modell lässt sich mit dem auf einer Kante balancierenden Würfel vergleichen. Die Bewegungsgleichungen der Würfelseite wurden mit Hilfe des Lagrange-Formalismus hergeleitet und anschließend für den Entwurf eines Zustandsreglers verwendet. Der Regler ermöglicht das Balancieren der Würfelseite in der aufrechten Position. Um die Zustandsgrößen zu messen sind an der Würfelseite zwei Sensormodule montiert, welche deren Beschleunigung- und Winkelgeschwindigkeit in Richtung von drei Achsen erfassen. Hierbei werden die Beschleunigungen genutzt um nach [Cubli1] die Ausrichtung der Würfelseite zu bestimmen. Des weiteren sind dem Motor Hall-Sensoren verbaut um die Winkelgeschwindigkeit der Schwungmasse zu ermitteln. Für die Umsetzung des zeitdiskreten Reglers wird ein BeagleBone Black verwendet, auf welchen ein Linux-Betriebssystem ausgeführt wird. Der letzte Aufgabenteil des Entwicklungsprojektes bestand in der Konstruktion und Fertigung des Würfels. Diese Arbeit beschäftigt sich nun mit der Entwicklung und Implementierung eines Reglers um den Würfel auf einer Ecke zu balancieren. Den Ausgangspunkt des Reglerentwurfs stellt ein Modell dar, welches aus den Bewegungsgleichungen des Würfels hervorgeht. Da sich in der Vorarbeit der Lagrange-Formalismus als ineffiziente Vorgehensweise zur Bestimmung der Bewegungsgleichungen erwiesen hat, wird in dieser Arbeit Kanes Methodik [Kane] verwendet.

Des weiteren stellt das Balancieren auf einer Ecke im Vergleich zu einer Kante ein deutlich komplexeres Regelungsproblem dar. Die Lösung dieses Problems erfordert ein Verständnis der Zustandsregelung zugrunde liegenden Systemtheorie. Aus diesen Gründen wird in dieser

Arbeit zunächst das Balancieren auf einer Kante untersucht. An diesem simplen Beispiel wird Kanes Methodik erläutert um die Bewegungsgleichungen zu ermitteln. Anschließend werden die theoretischen Grundlagen der Zustandsregelung diskutiert und ein LQR-basierter Regler entwickelt. Dieser wird danach an der Regelstrecke validiert.

Im nächsten Schritt werden die Bewegungsgleichung für das Balancieren auf einer Ecke ermittelt. Die Bewegungsgleichungen werden dann in eine Zustandsraumdarstellung überführt, welche für den Entwurf des Reglers genutzt wird. Hier wird zunächst die Beobachtbar- und Steuerbarkeit des Würfels diskutiert und an Hand einer Simulation untersucht. Des weiteren wird die Direktionalitätsproblematik erläutert, welche durch die Stelgrößenbeschränkung von Mehrgrößensystemen entsteht. Unter Beachtung dieser Probleme wird mit Hilfe des LQR-Verfahren ein Zustandsregler entworfen, der an der Regelstrecke erprobt wird. Hier zeigt sich, dass der Regler die Forderung an den geschlossenen Kreis nicht vollständig erfüllt. Dshalb wird zuletzt eine empirische Optimierung an Hand der Versuchsergebnisse durchgeführt.

Der nächste Teil der Arbeit beschäftigt sich mit der Erfassung der Zustandsgrößen. Hierfür sind an dem Würfel sechs Sensormodule angebracht, welche jeweils einen Beschleunigungs- und Drehratensensor besitzen. Die Messung der Winkelgeschwindigkeiten des Würfels und der Schwungmassen wird dabei nur kurz behandelt, da das Konzept aus der Vorarbeit übernommen werden kann. Allerdings muss das Verfahren zur Bestimmung der Winkel, welche die Ausrichtung des Würfels beschreiben, auf den Fallen der räumlichen Bewegung erweitert werden. Ebenso wird das Komplementärfilter erweitert um die Störgrößen, welche auf die Sensoren wirken, zu eliminieren. Des weiteren wird für das Balancieren auf einer Kante ein Beobachter entworfen um die Ausrichtung des Würfels zu schätzen. Dadurch ist es möglich die Anzahl der nötigen Sensoren zu reduzieren.

Der letzte Teil der Arbeit behandelt die softwareseitige Implementierung der Regler. Hier wird zunächst die Kombination des BeagleBone Black mit einem Linux-Betriebssystem als Zielplattform diskutiert. Anschließend wird die Ansteuerung und Auswertung der Periphereinheiten vorgestellt. Im nächsten Schritt wird eine Komponenten-Architektur vorgestellt, welche sich auf weitere mechatronische Anwendungen übertragen lässt. Ziel hierbei ist es eine effiziente Infrastruktur für die Durchführung von Versuchen zu schaffen. Hierfür werden Ansätze der Template-Meta-Programmierung verwendet. Diese ermöglichen es Algorithmen der Signalverarbeitung und Regelungstechnik unter minimalem Aufwand zu beliebigen Signalflüssen zusammenzusetzen. Des weiteren wird eine Verbindung zu einem Entwicklungsrechner implementiert, welche auf dem TCP/IP-Protokoll basiert. Dadurch können Messdaten und Signale aufgezeichnet und auf einer Benutzeroberfläche visualisiert werden.

Kapitel 2

Modellbildung Würfel auf Kante

Dieses Kapitel widmet sich der Herleitung der Bewegungsgleichungen für den Fall, dass der Würfel auf einer seiner Kanten steht. Hierfür wird Kanes Methodik [Kane] angewandt, welche sich in die Untersuchung der kinematischen und kinetischen Gegebenheiten unterteilen lässt. Bei der Kinematikanalyse werden zunächst generalisierte Koordinaten eingeführt um die Konfiguration des Systems zu beschreiben. Des weiteren werden Bezugssysteme verwendet um einzelnen Körper des Gesamtsystems im dem mathematischen Modell darzustellen. Die kinematische Untersuchung endet mit der Definition der generalisierten Geschwindigkeiten und Bestimmung der partiellen Geschwindigkeiten, welche als Betrag und Richtung der Systembewegung interpretiert werden können. In dem zweiten Teil des Kapitels wird die Kinematik betrachtet. Zu Beginn werden die resultierenden Drehmomente formuliert, welche auf die verschiedenen Körper wirken. Diese werden verwendet um mit Hilfe der partiellen Geschwindigkeiten die generalisierten aktiven Kräfte F_i zu ermitteln. Analog werden die Trägheitsmomente und daraus die generalisierten Trägheitskräfte F_i^* berechnet, welche mittels Kanes Gleichung $F_i + F_i^* = 0$ auf die gesuchten Bewegungsgleichungen führen. Zuletzt wird die Auswirkung der generalisierten Geschwindigkeiten auf die resultierenden Bewegungsgleichungen an Hand eines einfachen Beispiels diskutiert.

2.1 Untersuchung der Kinematik

Der erste Schritt besteht darin die kinematischen Zusammenhänge des Systems zu analysieren. Im Fall, dass der Würfel auf einer Kante balanciert, verfügt dieser über zwei rotatorische Freiheitsgrade. Der erste wird von der generalisierten Koordinate φ beschrieben und gibt die Orientierung des Würfelgehäuses wieder. Die zweite generalisierte Koordinate ψ erfasst die Rotation der Schwungmasse relativ zu dem Würfelgehäuse.

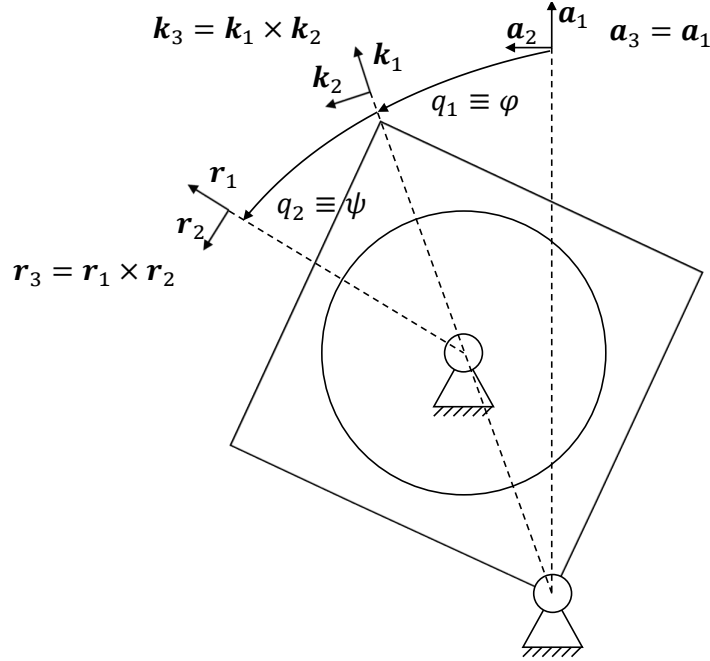


Abbildung 2.1: Kinematikplan des Würfels auf einer Kante, Quelle: eigene Darstellung

Um die Körper des Systems zu modellieren werden im nächsten Schritt die Bezugssysteme A , K und R eingeführt, wobei ein beliebiges Bezugssystem B durch drei Vektoren \mathbf{b}_i $i \in \{1, 2, 3\}$ definiert wird. Die Vektoren \mathbf{b}_i werden als Vektorbasis bezeichnet.¹ In diesem Fall ist A ein Inertialsystem, das heißt die Ausrichtung der Vektoren \mathbf{a}_i ist konstant. Deshalb wird A bzw. dessen Vektorbasis auch als raumfest bezeichnet. Bei K handelt es sich um ein körperfestes Bezugssystem, die Vektoren \mathbf{k}_i sind an dem Würfelgehäuse fixiert und ändern ihre Ausrichtung in Abhängigkeit von φ . Das Bezugssystem R ist ebenfalls körperfest, wobei die Vektorbasis an der Schwungmasse fixiert ist. Folglich beschreiben die Koordinaten φ und ψ die Rotation von K relativ zu A bzw. von R gegenüber K . Mittels eines Bezugssystems kann ein beliebiger Vektor als Linearkombination der Vektorbasis dargestellt werden. Als Beispiel sei der Ortsvektor \mathbf{c} des Würfelschwerpunktes genannt, wobei das verwendete Bezugssystem durch den vorangehenden Superskript gekennzeichnet wird.

$$\mathbf{c} = l_C \cdot \mathbf{k}_1 + 0 \cdot \mathbf{k}_2 + 0 \cdot \mathbf{k}_3 = \begin{pmatrix} l_C \\ 0 \\ 0 \end{pmatrix}^K \quad (2.1)$$

¹Allgemein genügen drei linear unabhängige Vektoren als Vektorbasis, allerdings werden in dieser Arbeit ausschließlich paarweise orthogonale Einheitsvektoren verwendet, da diese eine einfachere Handhabung ermöglichen. Deshalb wird im weiteren Verlauf nicht explizit erwähnt, dass es sich bei den Vektorbasen um paarweise orthogonale Einheitsvektoren handelt.

Um nun die Komponenten α_i des Vektors \mathbf{c} in dem Bezugssystem A zu erhalten, muss das Skalarprodukt $\langle \mathbf{a}_i, \mathbf{c} \rangle$ berechnet werden. Wird dieser Ansatz auf alle drei Komponenten appliziert ergibt sich der folgende Zusammenhang.²

$$\begin{aligned}
 \mathbf{c} &= \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} = \langle \mathbf{c}, \mathbf{a}_1 \rangle \cdot \mathbf{a}_1 + \langle \mathbf{c}, \mathbf{a}_2 \rangle \cdot \mathbf{a}_2 + \langle \mathbf{c}, \mathbf{a}_3 \rangle \cdot \mathbf{a}_3 = \begin{pmatrix} \langle \mathbf{c}, \mathbf{a}_1 \rangle \\ \langle \mathbf{c}, \mathbf{a}_2 \rangle \\ \langle \mathbf{c}, \mathbf{a}_3 \rangle \end{pmatrix} \\
 &= \begin{pmatrix} \beta_1 \cdot \langle \mathbf{a}_1, \mathbf{k}_1 \rangle + \beta_2 \cdot \langle \mathbf{a}_1, \mathbf{k}_2 \rangle + \beta_3 \cdot \langle \mathbf{a}_1, \mathbf{k}_3 \rangle \\ \beta_1 \cdot \langle \mathbf{a}_2, \mathbf{k}_1 \rangle + \beta_2 \cdot \langle \mathbf{a}_2, \mathbf{k}_2 \rangle + \beta_3 \cdot \langle \mathbf{a}_2, \mathbf{k}_3 \rangle \\ \beta_1 \cdot \langle \mathbf{a}_3, \mathbf{k}_1 \rangle + \beta_2 \cdot \langle \mathbf{a}_3, \mathbf{k}_2 \rangle + \beta_3 \cdot \langle \mathbf{a}_3, \mathbf{k}_3 \rangle \end{pmatrix} = \begin{pmatrix} \begin{bmatrix} \langle \mathbf{a}_1, \mathbf{k}_1 \rangle & \langle \mathbf{a}_1, \mathbf{k}_2 \rangle & \langle \mathbf{a}_1, \mathbf{k}_3 \rangle \\ \langle \mathbf{a}_2, \mathbf{k}_1 \rangle & \langle \mathbf{a}_2, \mathbf{k}_2 \rangle & \langle \mathbf{a}_2, \mathbf{k}_3 \rangle \\ \langle \mathbf{a}_3, \mathbf{k}_1 \rangle & \langle \mathbf{a}_3, \mathbf{k}_2 \rangle & \langle \mathbf{a}_3, \mathbf{k}_3 \rangle \end{bmatrix} \cdot \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} \\ \underbrace{\hspace{10em}}_{\equiv {}^K \mathbf{P}^A} \underbrace{\hspace{10em}}_{\equiv {}^K \mathbf{c}} \end{pmatrix} \\
 &= \begin{pmatrix} \begin{bmatrix} c_\varphi & -s_\varphi & 0 \\ s_\varphi & c_\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot {}^K \mathbf{c} \end{pmatrix}
 \end{aligned} \tag{2.2}$$

Somit handelt es sich bei der Projekt eines Vektors, der in einem Bezugssystem E dargestellt ist, in das Bezugssystem F um eine lineare Abbildung, welche durch die Abbildungsmatrix ${}^E \mathbf{P}^F$ definiert ist. Des weiteren zeigt dieses Beispiel die perspektivische Bedeutung eines Bezugssystem. Wird der Vektor \mathbf{c} in K dargestellt hängt er lediglich von der Länge l_C ab und ist deshalb konstant. Aus der Perspektive von A ist der Vektor eine Funktion von φ und somit variabel. Die Begründung hierfür ist, dass das Inertialsystem A die Bewegung des Würfels im Raum wahrnimmt, wodurch der Ortsvektor des Schwerpunktes variable erscheint. Im Gegensatz dazu bewegt sich das körperfeste Bezugssystem K mit dem Würfel und erfasst dessen Bewegung somit nicht. Dieser Umstand kann genutzt werden, indem ein Vektor zunächst in einer einfachen Form dargestellt und anschließend in das gewünschte Bezugssystem projiziert wird. Als Beispiel dient die Berechnung des Gravitationsmoments $\mathbf{T}^{K/O}$. Der Erdbeschleunigungsvektor \mathbf{g} ist im Inertialsystem A konstant und wird von dort in das körperfeste System K transformiert.

$$\begin{aligned}
 \mathbf{T}^{K/O} = \mathbf{c} \times m \cdot \mathbf{g} &= \begin{pmatrix} l_C \\ 0 \\ 0 \end{pmatrix} \times \begin{pmatrix} -m \cdot g \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} l_C \\ 0 \\ 0 \end{pmatrix} \times \begin{pmatrix} {}^A \mathbf{P}^K \cdot \begin{pmatrix} -m \cdot g \\ 0 \\ 0 \end{pmatrix} \end{pmatrix} \\
 &= \begin{pmatrix} l_C \\ 0 \\ 0 \end{pmatrix} \times \begin{pmatrix} -c_\varphi \cdot m \cdot g \\ s_\varphi \cdot m \cdot g \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ s_\varphi \cdot m \cdot g \cdot l_C \end{pmatrix}
 \end{aligned} \tag{2.3}$$

Dieses Beispiel zeigt, dass die Einführung von Bezugssystemen eine eindeutige Notation und Vorgehensweise zur Berechnung von vektorwertigen Funktionen mit sich bringt. Man möge bei diesem Anwendungsfall zwar argumentieren, dass die trigonometrischen Zusammenhänge

²Um die allgemeine Gültigkeit der linearen Abbildung zu verdeutlichen wird in dieser Gleichung die Definition $\mathbf{c} = \begin{pmatrix} l_C \\ 0 \\ 0 \end{pmatrix} \equiv \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix}$ verwendet.

des Gravitationsmomentes auch aus der Abbildung [?] ablesen lassen. Allerdings ist die hier betrachtete Vorgehensweise allgemeingültig und kann auf beliebig komplexe Systeme angewendet werden. Beispielsweise kann das Gravitationsmoment in dem Fall, dass der Würfel auf einer Ecke steht, kaum mehr aus einer Skizze bestimmt werden.

Zur vollständigen Beschreibung der Kinematik müssen die Geschwindigkeiten des Systems ermittelt werden. Als Beispiel für die Ableitung eines Vektors dient wieder der Ortsvektor \mathbf{c} des Schwerpunktes. Dieser ist aus der Perspektive des Bezugssystem K konstant, folglich verschwindet auch dessen Ableitung bzw. Geschwindigkeit. Wird der Vektor aber in A abgebildet hängt dieser von der Zeitfunktion $\varphi(t)$ ab. Deshalb muss bei der Ableitung eines Vektors das Bezugssystem angegeben werden, aus dessen Perspektive differenziert wird ([Kane], S.25ff). Wenn \mathbf{c} in A differenziert werden soll, so muss dieser zunächst in A dargestellt und anschließend komponentenweise nach der Zeit abgeleitet werden.

$$\frac{{}^A d\mathbf{c}}{dt} = \frac{d(c_\varphi \cdot l_C)}{dt} \cdot \mathbf{a}_1 + \frac{d(0)}{dt} \cdot \mathbf{a}_3 = \begin{pmatrix} -s_\varphi \cdot \dot{\varphi} \cdot l_C \\ c_\varphi \cdot \dot{\varphi} \cdot l_C \\ 0 \end{pmatrix} \quad (2.4)$$

Analog handelt es sich bei der Geschwindigkeit ${}^B \mathbf{v}^P$ bzw. Beschleunigung ${}^B \mathbf{a}^P$ eines Punktes oder Körpers P um eine Bewegung relativ zu einem Bezugssystem B . Diese werden wie folgt definiert ([Kane], S.28).

$${}^B \mathbf{v}^P = \frac{{}^B d\mathbf{p}}{dt} \quad {}^B \mathbf{a}^P \quad (2.5)$$

Die selbe Notation wird für Winkelgeschwindigkeiten und -beschleunigungen eingeführt, wobei ${}^A \boldsymbol{\omega}^B$ die Winkelgeschwindigkeit bzw. ${}^A \boldsymbol{\alpha}^B$ die Winkelbeschleunigung eines Punktes oder Bezugssystems B relativ zu A ist. Die Geschwindigkeit relativ zu einem Inertialsystem wird auch als absolute Geschwindigkeit bezeichnet. Für die Bezugssysteme K und R lassen sich mit Hilfe der Abbildung [?] die folgenden Winkelgeschwindigkeiten bestimmen, wobei die absolute Winkelgeschwindigkeit eines Bezugssystems die Summe seiner relativen Winkelgeschwindigkeiten ist ([Kane], S.24f).

$${}^A \boldsymbol{\omega}^K = \begin{pmatrix} 0 \\ 0 \\ \dot{\varphi} \end{pmatrix} \quad {}^K \boldsymbol{\omega}^R = \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix} \quad {}^A \boldsymbol{\omega}^R = {}^A \boldsymbol{\omega}^K + {}^K \boldsymbol{\omega}^R = \begin{pmatrix} 0 \\ 0 \\ \dot{\varphi} + \dot{\psi} \end{pmatrix} \quad (2.6)$$

Der letzte Schritt bei der Untersuchung der Kinematik besteht darin die generalisierten Geschwindigkeiten zu definieren und daraus die partiellen Geschwindigkeiten zu berechnen. Bei den generalisierten Geschwindigkeiten u_i handelt es sich um skalare Größen deren Anzahl gleich der Zahl der generalisierten Koordinaten ist. Die generalisierten Geschwindigkeiten sind für gewöhnlich abhängig von den Zeitableitungen der generalisierten Geschwindigkeiten \dot{q}_i , wobei sie so gewählt werden müssen, dass die Definitionen eindeutig nach den Ableitungen \dot{q}_i aufgelöst werden können. Die Einführung der generalisierten Geschwindigkeiten verfolgt das Ziel, möglichst einfache Ausdrücke für die absoluten Geschwindigkeiten ${}^A \boldsymbol{\omega}^K$ und ${}^A \boldsymbol{\omega}^R$ und, wie sich im späteren Verlauf zeigen wird, einfachere Ausdrücke der Bewegungsgleichungen zu

erhalten. Deshalb werden hier die folgenden Definitionen gewählt.

$$\begin{aligned} u_1 &\equiv \dot{\varphi} & u_2 &\equiv \dot{\varphi} + \dot{\psi} \\ {}^A\boldsymbol{\omega}^K &= \begin{pmatrix} 0 \\ 0 \\ u_1 \end{pmatrix}^K & {}^A\boldsymbol{\omega}^R &= \begin{pmatrix} 0 \\ 0 \\ u_2 \end{pmatrix}^K \end{aligned} \quad (2.7)$$

Diese Geschwindigkeitsausdrücke können auch in die Form

$$\begin{aligned} {}^A\boldsymbol{\omega}^K &= u_1 \cdot {}^A\boldsymbol{\omega}_1^K + u_2 \cdot {}^A\boldsymbol{\omega}_2^K & {}^A\boldsymbol{\omega}^R &= u_1 \cdot {}^A\boldsymbol{\omega}_1^R + u_2 \cdot {}^A\boldsymbol{\omega}_2^R \\ {}^A\boldsymbol{\omega}_1^K &= \mathbf{k}_3 & {}^A\boldsymbol{\omega}_2^K &= 0 & {}^A\boldsymbol{\omega}_1^R &= 0 & {}^A\boldsymbol{\omega}_2^R &= \mathbf{k}_3 \end{aligned} \quad (2.8)$$

gebracht werden, wobei die Größen ${}^A\boldsymbol{\omega}_i^K$ und ${}^A\boldsymbol{\omega}_i^R$ partielle Geschwindigkeiten genannt werden. Diese Zerlegung kann so interpretiert werden, dass die generalisierten Geschwindigkeiten u_i als skalare Größen den Betrag der Bewegung und die entsprechende partielle Geschwindigkeit, als vektorielle Größe, deren Richtung wiedergibt. Die Vorteile diese Zerlegung in generalisierte und partielle Geschwindigkeiten wird sich im folgenden Abschnitt zeigen.

2.2 Untersuchung der Kinetik

Im nächsten Schritt wird die Kinematik des Systems untersucht, um die Bewegungsgleichungen herzuleiten. Hierfür werden zunächst die Drehmomente analysiert, welche auf den Würfelgehäuse und die Schwungmasse wirken. Letztere wird einerseits von dem Motormoment $\mathbf{T}_M^{R/M}$ beschleunigt, andererseits wirkt ein verzögerndes Reibmoment $\mathbf{T}_M^{R/M}$. Das Reibmoment wird als proportional zu der Winkelgeschwindigkeit $\dot{\psi}$ modelliert.

$$\mathbf{T}^{R/M} = \mathbf{T}_M^{R/M} + \mathbf{T}_R^{R/M} = [T_M - C_\psi(u_2 - u_1)] \cdot \mathbf{k}_3 \quad (2.9)$$

Das Motor- und Reibmoment wirken an dem Verbindungspunkt zwischen Schwungmasse und Würfelgehäuse. Deshalb beeinflussen die beiden den Würfelkörper in umgekehrter Richtung. Des weiteren wird das Würfelgehäuse von dem Gravitationsmoment $\mathbf{T}_G^{K/O}$ beschleunigt. Die Summe der Komponenten ergibt das resultierende Drehmoment $\mathbf{T}^{K/O}$.

$$\mathbf{T}^{K/O} = \mathbf{T}_G^{K/O} - \mathbf{T}_M^{R/M} - \mathbf{T}_R^{R/M} = [m \cdot g \cdot l_C \cdot s_\varphi - T_M + C_\psi(u_2 - u_1)] \cdot \mathbf{k}_3 \quad (2.10)$$

Um die Bewegungsgleichungen zu ermitteln, werden die generalisierten aktiven Kräfte F_i benötigt. Hierfür werden die Skalarprodukte der Drehmomente, welche auf die Körper wirken, und die partiellen Winkelgeschwindigkeiten, des entsprechenden Körpers, berechnet. Die Summe über alle Körper ergibt die generalisierte aktive Kraft F_i .

$$\begin{aligned} F_1 &= \langle \mathbf{T}^{K/O}, {}^A\boldsymbol{\omega}_1^K \rangle + \langle \mathbf{T}^{R/M}, {}^A\boldsymbol{\omega}_1^R \rangle = m \cdot g \cdot l_C \cdot s_\varphi - T_M + C_\psi(u_2 - u_1) \\ F_2 &= \langle \mathbf{T}^{K/O}, {}^A\boldsymbol{\omega}_2^K \rangle + \langle \mathbf{T}^{R/M}, {}^A\boldsymbol{\omega}_2^R \rangle = T_M - C_\psi(u_2 - u_1) \end{aligned} \quad (2.11)$$

Die partiellen Geschwindigkeiten können als Bewegungsrichtungen der Körper interpretiert werden. Somit stellen die generalisierten Kräfte F_i den Beitrag der Drehmomente in die jeweilige Bewegungsrichtung wieder.

Als Gegenstück zu den generalisierten aktiven Kräften müssen nun die generalisierten Trägheitskräfte F_i^* ermittelt werden. Das resultierende Trägheitsmoment der Schwungmasse ergibt sich aus dem Produkt des Skalars I_R , welcher das Massenträgheitsmoment des Körpers relativ zu M beschreibt, und der Winkelbeschleunigung ${}^A\boldsymbol{\alpha}^R$.³

$$\mathbf{T}_*^{R/M} = -I_R \cdot \begin{pmatrix} 0 \\ 0 \\ \dot{u}_2 \end{pmatrix}^K \quad (2.12)$$

Das Trägheitsmoment des Würfelkörpers berechnet sich aus dessen Trägheitsskalar I_K und der Winkelbeschleunigung ${}^A\boldsymbol{\alpha}^K$. Die Größe I_K ist die Summe des Massenträgheitsmoments des Würfelgehäuses $I^{GH/O}$ und dem Einfluss der Schwungmasse, welche dabei als Punkt mit

³Die hier verwendete Berechnung ist nicht allgemeingültig. Für gewöhnlich wird zur Berechnung der Trägheitsmomente ein Trägheitstensor \mathbf{I} verwendet. Die in einem späteren Kapitel erläutert.

der Masse m_R und dem Abstand l_{MO} betrachtet wird.

$$\mathbf{T}_*^{K/O} = -(I^{GH/O} + m_R \cdot l_{MO}^2) \cdot \begin{pmatrix} 0 \\ 0 \\ \dot{u}_1 \end{pmatrix} = -I_K \cdot \begin{pmatrix} 0 \\ 0 \\ \dot{u}_1 \end{pmatrix} \quad (2.13)$$

Die generalisierten Trägheitskräfte F_i^* werden wieder aus der folgenden Summe von Skalarprodukten berechnet.

$$\begin{aligned} F_1^* &= \langle \mathbf{T}_*^{K/O}, {}^A\boldsymbol{\omega}_1^K \rangle + \langle \mathbf{T}_*^{R/M}, {}^A\boldsymbol{\omega}_1^R \rangle = -I_K \cdot \dot{u}_1 \\ F_2^* &= \langle \mathbf{T}_*^{K/O}, {}^A\boldsymbol{\omega}_2^K \rangle + \langle \mathbf{T}_*^{R/M}, {}^A\boldsymbol{\omega}_2^R \rangle = -I_R \cdot \dot{u}_2 \end{aligned} \quad (2.14)$$

Die Bewegungsgleichungen resultieren nun aus Kanes Gleichung $F_i + F_i^* = 0$.

$$\begin{aligned} F_1 + F_1^* &= 0 \quad \leftrightarrow \quad I_K \cdot \dot{u}_1 = m \cdot g \cdot l_C \cdot s_\varphi - T_M + C_\psi(u_2 - u_1) \\ F_2 + F_2^* &= 0 \quad \leftrightarrow \quad I_R \cdot \dot{u}_2 = T_M - C_\psi(u_2 - u_1) \end{aligned} \quad (2.15)$$

Rückblickend kann Kanes Methodik in die folgenden Schritte unterteilt werden. Zunächst wird die Kinematik des Systems analysiert, wobei die Körper mit Hilfe von Bezugssystemen modelliert werden. Die Definition von generalisierten Geschwindigkeiten ermöglicht die Bestimmung der partiellen Geschwindigkeiten, wodurch die Bewegung des Systems in eine Art Betrag und Richtung unterteilt wird. Anschließend werden die partiellen Geschwindigkeiten genutzt um die wirkenden Kräfte und Trägheitskräfte in die Bewegungsrichtungen zu projizieren. Hieraus resultieren die generalisierten aktiven Kräfte und generalisierten Trägheitskräfte, welche mittels Kanes Gleichung auf die gesuchten Bewegungsgleichungen führen.

Um die Bedeutung der generalisierten Geschwindigkeiten zu verdeutlichen wird der Fall betrachtet, dass die folgende Definition gewählt wird.

$$\tilde{u}_1 \equiv \dot{\varphi} \quad \tilde{u}_2 \equiv \dot{\psi} \quad (2.16)$$

$$\begin{aligned} {}^A\tilde{\boldsymbol{\omega}}_1^K &= \mathbf{k}_3 & {}^A\tilde{\boldsymbol{\omega}}_2^K &= 0 \\ {}^A\tilde{\boldsymbol{\omega}}_1^R &= 0 & {}^A\tilde{\boldsymbol{\omega}}_2^R &= \mathbf{k}_3 \end{aligned} \quad (2.17)$$

$$\begin{aligned} \tilde{F}_1 &= m \cdot g \cdot l_C \cdot s_\varphi & \tilde{F}_2 &= T_M - C_\psi(u_2 - u_1) \\ \tilde{F}_1^* &= -I_R \cdot (\dot{\tilde{u}}_1 + \dot{\tilde{u}}_2) - I_K \cdot \dot{\tilde{u}}_1 & \tilde{F}_2^* &= -I_R \cdot (\dot{\tilde{u}}_1 + \dot{\tilde{u}}_2) \end{aligned} \quad (2.18)$$

$$\begin{aligned} \tilde{F}_1 + \tilde{F}_1^* &= 0 \quad \leftrightarrow \quad I_K \cdot \dot{\tilde{u}}_1 + I_R \cdot (\dot{\tilde{u}}_1 + \dot{\tilde{u}}_2) = m \cdot g \cdot l_C \cdot s_\varphi \\ \tilde{F}_2 + \tilde{F}_2^* &= 0 \quad \leftrightarrow \quad I_R \cdot (\dot{\tilde{u}}_1 + \dot{\tilde{u}}_2) = T_M - C_\psi(\tilde{u}_2 - \tilde{u}_1) \end{aligned} \quad (2.19)$$

Dieses Beispiel zeigt, dass die Wahl der generalisierten Geschwindigkeiten eine direkte Auswirkung auf die Form der resultierenden Bewegungsgleichungen hat. Zwar kann die letztere mittels $u_1 = \tilde{u}_1$, $u_2 = \tilde{u}_1 + \tilde{u}_2$ und dem Einsetzen der zweiten in die erste Gleichung in die ursprüngliche Form überführt werden. Allerdings stellt die Modellierung des Würfels auf einer Ecke einen Anwendungsfall dar, bei dem der Berechnungsaufwand durch eine geschickte Wahl der generalisierten Geschwindigkeiten drastisch reduziert werden kann.

Kapitel 3

Regelungstechnik

Nachdem die Systemdynamik mit Hilfe von Bewegungsgleichungen beschrieben wurde, besteht der nächste Schritt darin ein Regler zu entwerfen, welcher die Stabilisierung des Würfels auf einer Kante ermöglicht. Hierfür wird in diesem Abschnitt zunächst die Beschreibung eines Systems mit Hilfe der Zustandsraumdarstellung diskutiert. Anschließend der Zusammenhang zeitkontinuierlicher Systeme und deren zeitdiskreten Darstellungen ermittelt um den Entwurf von digitalen Reglern zu ermöglichen. Daraufhin werden die Vorteile von Zustandsregler gegenüber klassischen Regelungsalgorithmen verglichen und ein Regler für die Stabilisierung des Würfels auf einer Kante entworfen. Anschließend wird diese Regelungsvorschrift an der realen Regelstrecke erprobt und validiert. Im letzten Teil werden Beobachter eingesetzt um die Auswirkungen von Messabweichungen zu kompensieren und die Anzahl der nötigen Sensoren zu reduzieren.

3.1 Zustandsraumdarstellung für kontinuierliche Systeme

Aus dem vorherigen Kapitel sind die beiden Bewegungsgleichungen

$$I_K \cdot \dot{u}_K = m \cdot g \cdot l \cdot s_\varphi - C_\psi \cdot u_K + C_\psi \cdot u_R - T_M \quad (3.1)$$

$$I_R \cdot \dot{u}_R = C_\psi \cdot u_K - C_\psi \cdot u_R + T_M \quad (3.2)$$

hervorgegangen, welche die Systemdynamik vollständig beschreiben. Mit Hilfe der Definitionen

$$\mathbf{x} = \begin{bmatrix} \varphi \\ u_K \\ u_R \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} \varphi \\ u_K \\ u_R \end{bmatrix} \quad u = T_M \quad (3.3)$$

können die linearisierten Bewegungsgleichungen in die folgende Zustandsraumdarstellung überführt werden.

$$\dot{\mathbf{x}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ \frac{m \cdot g \cdot l}{I_K} & \frac{-C_\psi}{I_K} & \frac{C_\psi}{I_K} \\ 0 & \frac{C_\psi}{I_R} & \frac{-C_\psi}{I_R} \end{bmatrix}}_{\equiv \mathbf{A}} \cdot \mathbf{x} + \underbrace{\begin{bmatrix} 0 \\ \frac{-1}{I_K} \\ \frac{1}{I_R} \end{bmatrix}}_{\equiv \mathbf{b}} \cdot u \quad (3.4)$$

$$\mathbf{y} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\equiv \mathbf{C}} \cdot \mathbf{x} \quad (3.5)$$

Prinzipiell kann jedes lineare zeitinvariante System $\mathfrak{S}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ als Zustandsraumdarstellung beschreiben lassen, wobei folgendes gilt.

$$\mathfrak{S} : \begin{cases} \dot{\mathbf{x}}(t) = \mathbf{A} \cdot \mathbf{x}(t) + \mathbf{B} \cdot \mathbf{u}(t) \\ \mathbf{y}(t) = \mathbf{C} \cdot \mathbf{x}(t) + \mathbf{D} \cdot \mathbf{u}(t) \end{cases} \quad (3.6)$$

Wobei $\mathbf{x} \in \mathbb{R}^n$ Zustandsvektor, $\mathbf{u} \in \mathbb{R}^r$ Eingangsvektor und $\mathbf{y} \in \mathbb{R}^m$ Ausgangsvektor heißt. Im weiteren Verlauf wird die Zeitabhängigkeit dieser drei Vektoren nicht mehr explizit angegeben. Des weiteren werden in dieser Arbeit lediglich nicht sprungfähige Systeme $\mathfrak{S}(\mathbf{A}, \mathbf{B}, \mathbf{C})$ betrachtet, deren Eingangsvektor \mathbf{u} den Ausgangsvektor \mathbf{y} nicht direkt beeinflusst und somit $\mathbf{D} = \mathbf{0}$ gilt.

Ein großer Vorteil dieser Modellierungsform besteht darin, dass für jedes Systems unendlich viele Zustandsraumdarstellungen existieren. Dieser Umstand wird ersichtlich wenn man für die Herleitung der Bewegungsgleichungen alternative generalisierte Geschwindigkeiten wählt und diese anschließend in eine Zustandsraumdarstellung transformiert.

$$\tilde{u}_K \equiv \dot{\phi} \quad \tilde{u}_R \equiv \dot{\psi} \quad (3.7)$$

$$I_K \cdot \dot{\tilde{u}}_K = m \cdot g \cdot l \cdot \sin \varphi + C_\psi \cdot \tilde{u}_R - T_M \quad (3.8)$$

$$I_R \cdot \dot{\tilde{u}}_R = -\frac{I_R \cdot m \cdot g \cdot l \cdot \sin \varphi}{I_K} - \frac{(I_K + I_R) \cdot C_\psi}{I_K} + \frac{I_K + I_R}{I_K} \cdot T_M \quad (3.9)$$

$$\dot{\tilde{\mathbf{x}}} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{m \cdot g \cdot l}{I_K} & 0 & \frac{C_\psi}{I_K} \\ \frac{-I_R \cdot m \cdot g \cdot l}{I_R \cdot I_K} & 0 & \frac{-C_\psi(I_K + I_R)}{I_R \cdot I_K} \end{bmatrix} \cdot \tilde{\mathbf{x}} + \begin{bmatrix} 0 \\ \frac{-1}{I_K} \\ \frac{I_K + I_R}{I_K \cdot I_R} \end{bmatrix} \cdot u \quad (3.10)$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \cdot \tilde{\mathbf{x}} \quad (3.11)$$

Sowohl die verschiedenen Bewegungsgleichungen als auch die daraus resultierenden Zustandsraumdarstellungen sind gültige Beschreibungsformen des Systems. Allgemein kann ein Zustandsraumdarstellung mit Hilfe einer Transformationsmatrix $\mathbf{T} \in \mathbb{R}^{n \times n}$ in eine äquivalente Darstellung überführt werden. Hierfür muss \mathbf{T} lediglich regulär sein. Die neue Darstellung ergibt sich aus den folgenden Transformationen.

$$\tilde{\mathbf{x}} = \mathbf{T}^{-1} \cdot \mathbf{x} \quad \dot{\tilde{\mathbf{x}}} = \mathbf{T}^{-1} \cdot \dot{\mathbf{x}} \quad (3.12)$$

$$\begin{aligned} \dot{\tilde{\mathbf{x}}} &= \mathbf{T}^{-1} \mathbf{A} \mathbf{T} \cdot \tilde{\mathbf{x}} + \mathbf{T}^{-1} \cdot \mathbf{u} \\ &= \tilde{\mathbf{A}} \cdot \tilde{\mathbf{x}} + \tilde{\mathbf{B}} \cdot \mathbf{u} \quad | \quad \tilde{\mathbf{A}} = \mathbf{T}^{-1} \mathbf{A} \mathbf{T}, \tilde{\mathbf{B}} = \mathbf{T}^{-1} \mathbf{B} \end{aligned} \quad (3.13)$$

$$\mathbf{y} = \mathbf{C} \mathbf{T} \cdot \tilde{\mathbf{x}} = \tilde{\mathbf{C}} \cdot \tilde{\mathbf{x}} \quad | \quad \tilde{\mathbf{C}} = \mathbf{C} \mathbf{T} \quad (3.14)$$

Mit Hilfe derartiger Transformationen kann ein beliebiges System in diverse Normalformen überführt werden, welche für die Systemanalyse und den Reglerentwurf besonders geeignet sind. Als erstes Beispiel sei die Transformation in kanonische Normalform genannt. Hierfür sei ein System $\mathfrak{S}(\mathbf{A}, \mathbf{B}, \mathbf{C})$ der Ordnung n gegeben, dessen Systemmatrix \mathbf{A} n einfache Eigenwerte λ_i mit den zugehörigen Eigenvektoren \mathbf{v}_i besitzt. Sind die Eigenvektoren \mathbf{v}_i linear unabhängig dann ist die Matrix

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n \end{bmatrix} \quad (3.15)$$

regulär und kann somit als Transformationsmatrix verwendet werden. Die resultierende Darstellung

$$\begin{aligned} \dot{\tilde{\mathbf{x}}} &= \mathbf{V}^{-1} \mathbf{A} \mathbf{V} \cdot \tilde{\mathbf{x}} + \mathbf{V}^{-1} \mathbf{B} \cdot \mathbf{u} \\ \mathbf{y} &= \mathbf{C} \mathbf{V} \cdot \tilde{\mathbf{x}} \end{aligned} \quad (3.16)$$

heißt kanonische Normalform, wobei die Matrix $\tilde{\mathbf{A}}$ die folgende Form besitzt.

$$\tilde{\mathbf{A}} = \mathbf{V}^{-1} \mathbf{A} \mathbf{V} = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix} \quad (3.17)$$

Folglich sind die Elemente des Zustandvektors $\tilde{\mathbf{x}}$ vollständig voneinander entkoppelt und werden deshalb auch als Eigenvorgänge bzw. Eigenbewegungen des Systems bezeichnet. Die vollständige Entkopplung der Zustandsgrößen ist nicht immer möglich, für eine ausführliche Diskussion Thematik sei auf [RT1, Lunze, S.135ff] verwiesen. Die homogene Lösung der

kanonischen Normalform lässt sich mit Hilfe des Exponentialansatzes ermitteln.

$$\tilde{x}_{i,h}(t) = e^{\lambda_i \cdot t} \cdot \tilde{x}_i(0) \rightarrow \tilde{\mathbf{x}}_h(t) = \begin{bmatrix} e^{\lambda_1 \cdot t} & & & \\ & e^{\lambda_2 \cdot t} & & \\ & & \ddots & \\ & & & e^{\lambda_n \cdot t} \end{bmatrix} \cdot \tilde{\mathbf{x}}(0) \quad (3.18)$$

Die Rücktransformation

$$\mathbf{x}_h = \mathbf{V} \cdot \tilde{\mathbf{x}}_h \quad (3.19)$$

zeigt, dass der Verlauf der ursprünglichen Zustandsgrößen eine Linearkombination der kanonischen Zustandsvariablen ist. Folglich wird die homogene Lösung eines Systems durch die Eigenwerte und -vektoren der Systemmatrix \mathbf{A} vorgegeben. Formal kann dieser Zusammenhang durch die Erweiterung des Exponentialansatzes auf vektorwertige Differentialgleichungen ermittelt werden.

$$\mathbf{x}(t) = e^{\mathbf{A} \cdot t} \cdot \mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)} \mathbf{B} \cdot \mathbf{u}(\tau) d\tau \quad (3.20)$$

Die Matrix-Exponentialfunktion $e^{\mathbf{A} \cdot t}$ heißt Fundamentalmatrix $\Phi(t)$ und kann über die folgende Reihenentwicklung bestimmt werden. Eine Herleitung kann in [RT2, Unbehauen, S. 6. ff] gefunden werden.

$$\Phi(t) = e^{\mathbf{A} \cdot t} = \sum_{k=0}^{\infty} \mathbf{A}^k \frac{t^k}{k!} \quad (3.21)$$

Um einen weiteren Weg zur Ermittlung der Fundamentalmatrix und den Zusammenhang zu der Übertragungsfunktion des Systems herzustellen wird die Systemgleichung in den Bildbereich transformiert.

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A} \cdot \mathbf{x}(t) + \mathbf{B} \cdot \mathbf{u}(t) & \circ \bullet & \quad s \cdot \mathbf{x}(s) - \mathbf{x}(t=0) = \mathbf{A} \cdot \mathbf{x}(s) + \mathbf{B} \cdot \mathbf{u} \\ & \leftrightarrow & & \quad \mathbf{x}(s) = \underbrace{(s \cdot \mathbf{I} - \mathbf{A})^{-1} \mathbf{x}(t=0)}_{\equiv \Phi(s)} + (s \cdot \mathbf{I} - \mathbf{A})^{-1} \mathbf{B} \cdot \mathbf{u}(s) \\ & \leftrightarrow & & \quad \mathbf{x}(s) = \Phi(s) \cdot \mathbf{x}(t=0) + \Phi(s) \mathbf{B} \cdot \mathbf{u}(s) \end{aligned} \quad (3.22)$$

Aus dem Vergleich von ([?]) mit ([?]) wird ersichtlich, dass die Fundamentalmatrix im Bildbereich durch die Invertierung der Matrix $s \cdot \mathbf{I} - \mathbf{A}$ berechnet werden kann. Im nächsten Schritt wird lediglich das Übertragungsverhalten eines Systems betrachtet.

$$\mathbf{x}(s) = \Phi(s) \mathbf{B} \cdot \mathbf{u}(s) \quad | \quad \mathbf{x}(t=0) = 0 \quad (3.23)$$

$$\mathbf{y}(s) = \mathbf{C} \cdot \mathbf{x}(s) = \underbrace{\mathbf{C} \Phi(s) \mathbf{B}}_{\equiv \mathbf{G}(s)} \cdot \mathbf{u}(s) = \mathbf{G}(s) \cdot \mathbf{u}(s) \quad (3.24)$$

Im Falle eines SISO-Systems reduzieren sich die Matrizen \mathbf{C} und \mathbf{B} auf die Vektoren \mathbf{c}^T und \mathbf{b} . Somit handelt es sich bei $\mathbf{G}(s)$ um die Übertragungsfunktion des Eingrößensystems. Bei MIMO-Systemen heißt $\mathbf{G}(s)$ Übertragungsfunktionmatrix, wobei die einzelnen Elemente

$G_{ij}(s)$ Teilübertragungsfunktionen heißen und das E/A-Verhalten der Eingangsgröße u_j auf die Ausgangsgröße y_i beschreiben. Bei der Berechnung der Übertragungsmatrix $\mathbf{G}(s)$ stellt die Fundamentalmatrix $\Phi(s)$ die einzige Größe dar, die von s abhängt, und muss somit auch die Pole des Systems enthalten. Aus der Berechnung

$$\Phi(s) = (s \cdot \mathbf{I} - \mathbf{A})^{-1} = \frac{1}{\det(s \cdot \mathbf{I} - \mathbf{A})} \text{adj}(s \cdot \mathbf{I} - \mathbf{A}) \quad (3.25)$$

folgt, dass das charakteristische Polynom

$$\det(s \cdot \mathbf{I} - \mathbf{A}) \quad (3.26)$$

den gemeinsamen Nenner der Teilübertragungsfunktionen darstellt. Die Nullstellen des Polynoms entsprechen den Eigenwerten der Systemmatrix \mathbf{A} . Hieraus folgt, dass die Eigenwerte λ_i eine Übermenge der Pole s_i des Systems bilden, da ggf. Eigenwerte gegen Zählernullstellen gekürzt werden können. Dies bedeutet, dass die Systemmatrix \mathbf{A} nicht nur die Eigenbewegung des Systems bestimmt sondern auch zur Untersuchung der Stabilität verwendet werden kann. Allgemein ist ein System asymptotisch stabil wenn die Realteile aller Eigenwerte λ_i der Matrix \mathbf{A} negativ sind.

$$\text{Re}\lambda_i < 0 \quad (3.27)$$

Dieses Stabilitätskriterium stellt einen weiteren Vorteil der Systembeschreibung mittels Zustandsraumdarstellung dar. Die Systemanalyse erfolgt durch die Untersuchung numerischer Matrizen, was mit Hilfe von Matlab effizient umgesetzt werden kann. Des weiteren müssen bei MIMO-Systemen mit klassischen Stabilitätskriterien alle Elemente der Übertragungsfunktionsmatrix $\mathbf{G}(s)$ untersucht werden. In der Zustandsraumdarstellung genügt allerdings die Analyse der Systemmatrix \mathbf{A} .

3.2 Zustandsraumdarstellung für zeitdiskrete Systeme

Bisher wurden ausschließlich zeitkontinuierliche Systeme betrachtet. Allerdings wird der Regler mit Hilfe eines Digitalrechners implementiert und stellt somit ein zeitdiskretes System dar. Derartige Systeme werden in Form von Differenzengleichungen beschrieben, die ebenfalls in eine Zustandsraumdarstellung gebracht werden können. Hierbei wird lediglich der Integrierer durch ein vektorielles Totzeitglied ersetzt, welches eine Abtastperiode repräsentiert. Somit gilt für ein diskretes System $\mathfrak{D}(\mathbf{A}_d, \mathbf{B}_d, \mathbf{C}_d)$:

$$\mathfrak{D} : \begin{cases} \mathbf{x}(k+1) = \mathbf{A} \cdot \mathbf{x}(k) + \mathbf{B} \cdot \mathbf{u}(k) \\ \mathbf{y}(k) = \mathbf{C} \cdot \mathbf{x}(k) \end{cases} \quad (3.28)$$

Die Ergebnisse aus dem vorherigen Abschnitt können auf diskrete Systeme übertragen werden. Hierfür wird zunächst die Z-Transformierte der Systemgleichung gebildet.

$$\begin{aligned} \mathbf{x}(k+1) = \mathbf{A}_d \cdot \mathbf{x}(k) + \mathbf{B}_d \cdot \mathbf{u}(k) & \quad \circ \longrightarrow \quad z \cdot \mathbf{x}(z) - z \cdot \mathbf{x}(t=0) = \mathbf{A}_d \cdot \mathbf{x}(z) + \mathbf{B}_d \cdot \mathbf{u}(z) \\ \Leftrightarrow & \quad (z \cdot \mathbf{I} - \mathbf{A}_d) \mathbf{x}(z) = z \cdot \mathbf{x}(t=0) + \mathbf{B}_d \cdot \mathbf{u}(z) \\ \Leftrightarrow & \quad \mathbf{x}(z) = \underbrace{(z \cdot \mathbf{I} - \mathbf{A}_d)^{-1}}_{\equiv \Phi_d(z)} \cdot z \cdot \mathbf{x}(t=0) + (z \cdot \mathbf{I} - \mathbf{A}_d)^{-1} \mathbf{B}_d \cdot \mathbf{u}(z) \end{aligned} \quad (3.29)$$

$$\mathbf{y}(z) = \mathbf{C}_d \cdot \mathbf{x}(z) = \underbrace{\mathbf{C}_d \Phi_d(z) \mathbf{B}_d}_{\equiv \mathbf{G}_d(z)} \cdot \mathbf{u}(z) \quad (3.30)$$

An Hand der obigen Gleichungen ist leicht zu erkennen, dass sowohl die Fundamentalmatrix $\Phi_d(z)$ als auch die Übertragungsfunktionmatrix $\mathbf{G}_d(z)$ analog zu zeitkontinuierlichen Systemen berechnet wird. Deshalb behalten die Eigenwerte und -vektoren der Systemmatrix \mathbf{A}_d ihre Bedeutung bei der Systemanalyse. Lediglich die Interpretation muss durch den Zusammenhang der Laplace- und Z-Transformation angepasst werden.

$$z = e^{T \cdot s} \quad s = \sigma + j \cdot \omega \quad (3.31)$$

$$|z| = e^{T \cdot \sigma} \quad \phi = \omega \cdot T \quad (3.32)$$

Stabilitätsbedingung für kontinuierliche Systeme ist, dass die Realteile aller Eigenwerte λ_i negativ sind. Aus ([?]) folgt, dass der Betrag der diskreten Eigenwerte im Einheitskreis liegen muss, damit diese Forderung erfüllt ist. Des weiteren führen diskrete Eigenwerte, die nahe am Ursprung liegen, zu einer schnelleren Systemdynamik. Somit bringt Zustandsraumdarstellung für diskrete Systeme die gleichen Vorteile mit sich wie bei zeitkontinuierlichen Systemen. Allerdings handelt es sich bei der Regelstrecke nach wie vor um ein kontinuierliches System, lediglich der Regler stellt ein diskretes System dar. Durch die Abtastung werden kontinuierliche Zustandssignale in Zahlenfolgen überführt. Die Stellgröße wird in Form eines Halteglieds nullter Ordnung realisiert. Um den geschlossenen Regelkreis einheitlich zu beschreiben, muss deshalb die kontinuierliche Zustandsraumdarstellung der Regelstrecke in eine diskrete transformiert werden. Hierzu wird zunächst die Lösung eines kontinuierlichen Systems $\mathfrak{S}(\mathbf{A}, \mathbf{B}, \mathbf{C})$

betrachtet. Herleitung aus [RT2, Unbehauen]

$$\mathbf{x}(t) = e^{\mathbf{A} \cdot t} \cdot \mathbf{x}(t_0) + \int_0^t e^{\mathbf{A}(t-\tau)} \mathbf{B} \cdot \mathbf{u}(\tau) d\tau \quad (3.33)$$

Der Verlauf zwischen zwei Abtastzeitpunkten $(k+1)T$ und kT wird berechnet indem $t_0 = kT$ und $t = (k+1)T$ gilt.

$$\begin{aligned} \mathbf{x}((k+1)T) &= e^{\mathbf{A}(k+1)T} \cdot \mathbf{x}(kT) + \int_{kT}^{(k+1)T} e^{\mathbf{A}([k+1]T-\tau)} \mathbf{B} \mathbf{u}(\tau) d\tau \\ &= e^{\mathbf{A}(k+1)T} \cdot \mathbf{x}(kT) + \int_0^T e^{\mathbf{A}(kT-\tau)} \mathbf{B} \mathbf{u}(\tau) d\tau \end{aligned} \quad (3.34)$$

Um das Integral zu lösen muss der Verlauf von \mathbf{u} zwischen den Abtastpunkten bekannt sein. Das Halteglied nullter Ordnung führt die diskrete Zahlenfolge der Stellgröße $\mathbf{u}(k)$ in eine kontinuierliche Treppenfunktion über. Somit folgt, dass \mathbf{u} über den Verlauf einer Abtastperiode konstant ist.

$$\mathbf{u}(kT + t) = \mathbf{u}(kT) \quad | \quad 0 \leq t < T \quad (3.35)$$

Die Lösung des Integrals führt auf:

$$\mathbf{x}([k+1]T) = e^{\mathbf{A}T} \cdot \mathbf{x}(kT) + e^{\mathbf{A}T} (\mathbf{I} - e^{-\mathbf{A}T}) \mathbf{A}^{-1} \mathbf{B} \cdot \mathbf{u}(kT) \quad (3.36)$$

Diese Lösung stellt den Zusammenhang zwischen der kontinuierlichen und diskreten Zustandsraumdarstellung dar. Wobei der Vergleich mit ([?]) folgenden Werte für die Matrizen des diskreten Systems $\mathfrak{D}(\mathbf{A}_d, \mathbf{B}_d, \mathbf{C}_d)$ liefert.

$$\mathbf{A}_d = e^{\mathbf{A}T} \quad \mathbf{B}_d = (e^{\mathbf{A}T} - \mathbf{I}) \mathbf{A}^{-1} \mathbf{B} \quad \mathbf{C}_d = \mathbf{C} \quad (3.37)$$

Mit der Definition

$$\mathbf{S} = T \sum_{v=0}^{\infty} \mathbf{A}^v \frac{T^v}{(v+1)!} \quad (3.38)$$

kann die Definition vereinfacht und die Berechnung der inversen Matrix vermieden werden.

$$\mathbf{A}_d = \mathbf{I} + \mathbf{S} \mathbf{A} \quad \mathbf{B}_d = \mathbf{S} \mathbf{B} \quad (3.39)$$

Die Matrix \mathbf{S} beschreibt einerseits den Einfluss des Halteglieds auf den Regelkreis, andererseits ermöglicht sie die Überführung in eine diskrete Darstellung, welche nötig ist um die Regelstrecke und den Regler in einer einheitlichen Form zu beschreiben.

3.3 Entwurf von Zustandsregelungen

Nachdem die Grundlagen der Zustandsraumdarstellung in den vorherigen Abschnitten diskutiert wurden, dient dieser Teil dem Entwurf des letztendlichen Reglers. Hierfür wird zunächst am Beispiel des Entwurfs durch Eigenwertvorgabe der Grundgedanke und die Vorteile der Zustandsregelung aufgezeigt. Anschließend wird ein Regler nach dem Prinzip der optimalen Regelung entworfen um den Würfel auf einer Kante zu stabilisieren. Diese Ergebnisse werden an der reellen Regelstrecke überprüft.

3.3.1 Reglerentwurf durch Eigenwertvorgabe

Der Grundgedanke der Zustandsregelung besteht darin den vollständigen Zustandsvektor \mathbf{x} zurückzuführen und durch die Multiplikation mit einer Reglermatrix \mathbf{K} den Stellgrößenvektor \mathbf{u} zu ermitteln. In dieser Arbeit werden lediglich Regelkreise ohne Führungsvektor ($\boldsymbol{\omega} = 0$) verwendet, weshalb auf dessen Einfluss und den Entwurf des nötigen Vorfilters \mathbf{V} nicht weiter eingegangen wird.

Aus der Abbildung gehen die Systemgleichungen des geschlossenen Regelkreises $\mathfrak{D}(\mathbf{A}_g, \mathbf{0}, \mathbf{C}_g)$ hervor.

$$\mathbf{u} = -\mathbf{B}_d \mathbf{K} \quad (3.40)$$

$$\mathfrak{D}_g : \begin{cases} \mathbf{x}(k+1) = \mathbf{A} \cdot \mathbf{x}(k) - \mathbf{BK} \cdot \mathbf{x}(k) = \underbrace{(\mathbf{A} - \mathbf{BK})}_{\equiv \mathbf{A}_g} \cdot \mathbf{x}(k) \\ \mathbf{y}(k) = \underbrace{\mathbf{C}}_{\equiv \mathbf{C}_g} \cdot \mathbf{x}(k) \end{cases} \quad (3.41)$$

Die Reglermatrix \mathbf{K} ist nun so zu entwerfen, dass die gewünschten Eigenschaften des geschlossenen Regelkreises erreicht werden. Hierfür wird zunächst das SISO-System $\mathfrak{D}(\mathbf{A}, \mathbf{b}, \mathbf{c}^T)$ betrachtet, wessen Systemmatrix \mathbf{A} und Eingangsvektor \mathbf{b} die folgende Form besitzt.

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-1} \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \quad (3.42)$$

Diese Darstellung heißt Regelungsnormalform und kann mittels einer Zustandstransformation erreicht werden. Diese Transformation wird erst in einem späteren Abschnitt erläutert, da das Verfahren lediglich als Beispiel der Zustandsregelung dienen soll. Der besondere Vorteil der Regelungsnormalform besteht darin, dass die Werte a_i die Koeffizienten des charakteristischen Polynoms

$$\det(\lambda \cdot \mathbf{I} - \mathbf{A}) = \prod_{i=1}^n (\lambda - \lambda_i) = \lambda^n + a_{n-1} \cdot \lambda^{n-1} + \dots + a_1 \cdot \lambda + a_0 \quad (3.43)$$

der Systemmatrix \mathbf{A} sind. Wird der Regelkreis über den Reglervektor

$$\mathbf{k} = \begin{bmatrix} k_1 & k_2 & \dots & k_n \end{bmatrix} \quad (3.44)$$

geschlossen, ergibt sich die folgende Systemmatrix \mathbf{A}_g des geschlossenen Regelkreises.

$$\begin{aligned} \mathbf{A}_g = \mathbf{A} - \mathbf{b}\mathbf{k} &= \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-1} \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ k_1 & k_2 & k_3 & \dots & k_n \end{bmatrix} \\ &= \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\bar{a}_0 & -\bar{a}_1 & -\bar{a}_2 & \dots & -\bar{a}_{n-1} \end{bmatrix} \quad | \quad \bar{a}_i = a_i + k_{i+1} \end{aligned} \quad (3.45)$$

Die Systemmatrix \mathbf{A}_g des geschlossenen Regelkreises liegt nun ebenfalls in Regelungsnormform vor, weshalb wiederum für deren charakteristisches Polynom gilt:

$$\det(\lambda \cdot \mathbf{I} - \mathbf{A}_g) = \prod_{i=1}^n (\lambda - \bar{\lambda}_i) = \lambda^n + \bar{a}_{n-1} \cdot \lambda^{n-1} + \dots + \bar{a}_1 \cdot \lambda + \bar{a}_0 \quad (3.46)$$

Werden nun die Eigenwerte $\bar{\lambda}_i$ des geschlossenen Regelkreises vorgegeben können durch ([?]) die Koeffizienten \bar{a}_i berechnet werden. Die Koeffizienten a_i der Regelstrecke können aus der Matrix \mathbf{A} abgelesen werden. Somit können aus der Beziehung

$$\bar{a}_i = a_i + k_i \quad (3.47)$$

die einzelnen Reglerfaktoren k_i berechnet bzw. die folgende Vektorschreibweise verwendet werden.

$$\underbrace{\begin{bmatrix} \bar{a}_0 & \bar{a}_1 & \dots & \bar{a}_{n-1} \end{bmatrix}}_{\equiv \bar{\mathbf{a}}} = \underbrace{\begin{bmatrix} a_0 & a_1 & \dots & a_{n-1} \end{bmatrix}}_{\equiv \mathbf{a}} + \underbrace{\begin{bmatrix} k_1 & k_2 & \dots & k_n \end{bmatrix}}_{\equiv \mathbf{k}} \quad (3.48)$$

$$\mathbf{k} = \bar{\mathbf{a}} - \mathbf{a} \quad (3.49)$$

An diesem Entwurfsverfahren werden bereits einige interessante Eigenschaften der Zustandsregelung deutlich. Zunächst können die Eigenwerte des geschlossenen Regelkreises beliebig gewählt werden wodurch das dynamische Verhalten maßgeblich bestimmt wird. Des weiteren wird der Regler durch eine Matrix-Vektor-Multiplikation realisiert. Somit werden dem Regelkreis, im Gegensatz zu PID-Reglern, keine weiteren Pole durch die Reglerdynamik hinzugefügt. Zuletzt sei die einfache Berechnung der Reglermatrix erwähnt. Sowohl die Transformation auf Regelungsnormform als auch die Ermittlung der Reglerparameter sind numerische Operationen die mit Hilfe von Matlab durchgeführt werden können.

Nichtsdestotrotz wird das Entwurfsverfahren in dieser Arbeit nicht verwendet. Die Gründe hierfür sind, dass das primäre Ziel darin besteht eine robuste Regelung zur Stabilisierung des Systems zu entwerfen und nicht eine spezielle Dynamik des geschlossenen Regelkreises zu erzielen. Des weiteren können bei diesem Verfahren die Verläufe der physikalischen Größen φ , u_K und u_R nicht direkt beeinflusst werden. Durch die Vorgabe der Eigenwerte werden lediglich die Trajektorien der kanonischen Zustandsvariablen bestimmt, welche jeweils eine Linearkombination der ursprünglichen Zustände sind. Zuletzt fließt die Stellgröße u bei dieser

Vorgehensweise nicht in die Bestimmung der Reglermatrix ein. Ist die Stellgröße, wie in den hier behandelten Anwendungsfällen, begrenzt müssen diese Umstände bei dem Reglerentwurf unter Umständen beachtet werden.

3.3.2 Linear quadratisch optimale Regelung

Im vorherigen Abschnitt wurden die Nachteile der Eigewertvorgabe aufgezeigt, weshalb in hier der Entwurf eines linear quadratisch optimalen Reglers vorgestellt und an der realen Regelstrecke appliziert wird. Der Grundgedanke der optimalen Regelung ist es ein Regelgesetz zu entwerfen, welches ein zu einem geschlossenen Regelkreis führt, der im Sinne eines Gütekriteriums optimal ist. Hier wird das linear quadratische Gütekriterium

$$J = \sum_{k=0}^{\infty} [\mathbf{x}^T(k) \cdot \mathbf{Q} \cdot \mathbf{x}(k) + \mathbf{u}^T(k) \cdot \mathbf{R} \cdot \mathbf{u}(k)] \quad (3.50)$$

verwendet. Die Matrizen \mathbf{Q} und \mathbf{R} dienen der Gewichtung des Zustands- bzw. Stellvektors und müssen positiv definit bzw. semidefinit sein. Durch die Gewichtungsmatrizen ist es möglich Forderung an den ursprünglichen Zustands- und Stellvektor direkt in den Reglerentwurf einfließen zu lassen. Der letztendliche Regler ergibt sich aus der Lösung des Optimierungsproblems, dass J minimal wird. Nach [Ludyk, S.177] gilt für die Reglermatrix

$$\mathbf{K} = (\mathbf{R} + \mathbf{B}^T \mathbf{P} \mathbf{B})^{-1} \mathbf{B}^T \mathbf{P} \mathbf{A}, \quad (3.51)$$

wobei \mathbf{P} durch die Lösung der folgenden Matrix-Ricatti-Gleichung bestimmt wird.

$$\mathbf{P} = \mathbf{Q} + \mathbf{A}^T [\mathbf{P} - \mathbf{P} \mathbf{B} (\mathbf{R} + \mathbf{B}^T \mathbf{P} \mathbf{B})^{-1} \mathbf{B}^T \mathbf{P}] \mathbf{A} \quad (3.52)$$

Ähnlich wie bei dem Entwurf durch Eigenwertvorgabe besteht ein Vorteil der linear quadratisch optimalen Regelung darin, dass das Optimierungsproblem mittels des Matlab-Befehls `dlqr()` gelöst werden kann. Somit fällt der Rechenaufwand für die Ermittlung der Reglermatrix bei der Bewertung des Entwurfverfahrens nicht weiter ins Gewicht. Deshalb werden hier lediglich die Bedeutung der Gewichtsmatrizen und die Eigenschaften des resultierenden Regelkreises diskutiert.

Zunächst lässt sich zeigen [Lunze RT2, S.301], dass der geschlossene Regelkreis asymptotisch stabil ist insofern das System stabilisierbar ist. Bei einem stabilisierbaren System können alle instabilen Eigenvorgänge durch die Stellgröße und somit durch den Regler beeinflusst werden.¹ Hieraus folgt, dass alle Eigenwerte des geschlossenen Regelkreises im Einheitskreis liegen. Des weiteren lässt sich durch

$$|z| = e^{\sigma_i} \quad | \quad s_i = \sigma_i + \omega_i \cdot j \quad (3.53)$$

der Abstand der Eigenwerte zum Ursprung der komplexen Ebene interpretieren. Je näher ein Eigenwert am Ursprung liegt, desto schneller wird die zugehörige Eigenbewegung. Im geschlossenen Regelkreis wird eine schnelle Systemdynamik durch hohe Stellgrößen verursacht, welche durch die Reglermatrix \mathbf{K} bestimmt werden. Daraus kann geschlossen werden, dass

¹Die Systemeigenschaft heißt Steuerbarkeit und wird in dem anschließenden Abschnitt erläutert.

die Erhöhung der Reglerfaktoren k_{ij} die Eigenwerte des geschlossenen Regelkreises näher an den Ursprung rückt. Umgekehrt kann die Bedeutung der Gewichtungsmatrix \mathbf{R} interpretiert werden. Je größer die Elemente von \mathbf{R} werden, desto stärker fließt der Verlauf des Stellvektors in das Gütekriterium ein. Da dieses minimiert werden soll, resultieren kleinere Elemente der Reglermatrix und somit eine langsamere Systemdynamik. Die Matrix \mathbf{Q} wirkt entgegengesetzt. Das das Gütekriterium den zeitlichen Verlauf des Zustandvektors erfasst, führt eine Erhöhung der Matrix \mathbf{Q} auf eine schnellere Eigenbewegung, welche durch erhöhte Reglerwerte erreicht wird. Hier zeigt sich ein Vorteil der linear quadratisch optimalen Regelung. Bei der Eigenwertvorgabe kann der Verlauf einer Zustandsgröße nur indirekt durch die Definition der Eigenbewegung beeinflusst werden. Im Gegensatz dazu erfolgt durch die Diagonalelemente der Matrix \mathbf{Q} eine direkte Gewichtung der Zustandsgrößen.² Da das Zeitverhalten des geschlossenen Regelkreises in dem hier behandelten Anwendungsfall eine untergeordnete Rolle spielt, wird nun die Empfindlichkeit gegenüber Störungen betrachtet. Aus dem Reglergesetz

$$\mathbf{u} = -\mathbf{K} \cdot \mathbf{x} \quad (3.54)$$

geht hervor, dass sich ein beliebiges Messrauschen proportional zur Reglermatrix in der Stellgröße widerspiegelt. Folglich beeinflusst das Entwurfsverfahren nicht direkt die Störempfindlichkeit des Regelkreises. Vielmehr wird diese durch die Lage der Eigenwerte bestimmt. Je näher diese an dem Einheitskreis liegen, desto robuster wird das System gegenüber Messrauschen. Ein weiterer Vorteil der linear quadratisch optimalen Regelung besteht in derer Robustheit gegenüber Parameteränderungen. Diese Eigenschaft beschreibt wie groß die Abweichungen einzelner Parameter sein können um nach wie vor einen stabilen Regelkreis zu erhalten. In [Lunze RT2, S. 303ff] werden diese Eigenschaften mit Hilfe des Phasen- und Amplitudenrandes erläutert. Ludynk ergänzen

²Allerdings sei angemerkt, dass sich durch die Gewichtungsmatrix \mathbf{Q} keine Entkopplung der Zustandsgrößen erzwingen lässt. Für die Entkopplung der Zustandsgrößen müssen nicht nur die Eigenwerte sondern auch die Eigenvektoren durch den Regler beeinflusst werden. Dies ist nur möglich wenn die Reglermatrix mehr Elemente als zu regelnde Zustandsgrößen besitzt [Lunze RT2, S.254 ff]. Des weiteren sind solche System in der Realität oftmals nicht realisierbar, wären beispielsweise alle Eigenwerte und -vektoren des Würfels frei wählbar könnten die Größen φ und $\dot{\varphi}$ voneinander entkoppelt werden.

3.4 Verifizierung des Reglers an der Regelstrecke

Die letztendliche Aufgabe besteht darin einen Regler zu entwerfen, damit der Würfel auf einer Kante balanciert. Hierfür wird ein linear quadratisch optimaler Regler entworfen, wobei die Gewichtsmatrizen

$$\mathbf{Q} = \begin{bmatrix} \varphi_{max}^{-2} & 0 & 0 \\ 0 & u_{K,max}^{-2} & 0 \\ 0 & 0 & u_{R,max}^{-2} \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} T_{M,max}^{-2} \end{bmatrix} \quad (3.55)$$

verwendet werden. Daraus resultiert das Gütekriterium

$$J = \sum_{k=0}^{\infty} \left(\frac{\varphi^2}{\varphi_{max}^2} + \frac{u_K^2}{u_{K,max}^2} + \frac{u_R^2}{u_{R,max}^2} + \frac{T_M^2}{T_{M,max}^2} \right). \quad (3.56)$$

Somit werden der Zustandsvektor und die Stellgröße quadratisch über ihren maximalen Wert minimiert. Der daraus resultierende Regelkreis besitzt die Eigenwerte

$$\lambda_1 = 0,7895 \quad \lambda_{2,3} = 0,8830 \pm 0,0087j \quad (3.57)$$

und ist somit asymptotisch stabil. Im nächsten Schritt wird der Regler an der realen Regelstrecke validiert. Abbildung hier zeigt den Verlauf der Zustands- und Stellgröße, wobei das System bei dem Zeitpunkt yy durch eine äußere Störung erregt wurde.

placeholder plot

Die Abbildungen zeigen, dass der Regler das System nicht in die Ruhelage $\mathbf{x} = \mathbf{0}$ überführt, sondern die Schwungmasse mit konstanter Geschwindigkeit rotiert. Dieser Umstand ist darauf zurückzuführen, dass der Zustandsvektor mit einem systematischen Messfehler erfasst wird. Im Modell wird diese Gegebenheit durch die Einführung der Zustandsgrößen

$$\hat{\mathbf{x}} = \begin{pmatrix} \hat{\varphi} \\ \hat{u}_K \\ \hat{u}_R \end{pmatrix} \quad (3.58)$$

erfasst, welche die jeweiligen Messabweichungen darstellen. Der Ausgangsvektor \mathbf{y} stellt die Messwerte dar und wird durch die Summe des ursprünglichen Zustandvektors \mathbf{x} und der Messabweichungen $\hat{\mathbf{x}}$ berechnet. Aus diesen Überlegungen ergibt sich für den offenen Regelkreis das System

$$\mathfrak{D}_0 \equiv \begin{cases} \mathbf{x}_o(k+1) = \underbrace{\begin{bmatrix} \mathbf{A} & \mathbf{0}^{3 \times 3} \\ \mathbf{0}^{3 \times 3} & \mathbf{I}^{3 \times 3} \end{bmatrix}}_{\equiv \mathbf{A}_o} \cdot \underbrace{\begin{bmatrix} \mathbf{x} \\ \hat{\mathbf{x}} \end{bmatrix}}_{\equiv \mathbf{x}_0}(k) + \underbrace{\begin{bmatrix} \mathbf{b} \\ \mathbf{0}^3 \end{bmatrix}}_{\equiv \mathbf{b}_o} \cdot u(k) \\ \mathbf{y}(k) = \underbrace{\begin{bmatrix} \mathbf{I}^{3 \times 3} & \mathbf{I}^{3 \times 3} \end{bmatrix}}_{\equiv \mathbf{C}_o} \cdot \mathbf{x}_o(k) \end{cases} \quad (3.59)$$

Das Reglergesetz ergibt sich nun aus dem Produkt des Reglervektors \mathbf{k}^T und des Ausgangsvektors \mathbf{y} .

$$u = \mathbf{k}^T \cdot \mathbf{y} = \mathbf{k}^T \cdot (\mathbf{x} + \hat{\mathbf{x}}) = \underbrace{\begin{bmatrix} \mathbf{k}^T & \mathbf{k}^T \end{bmatrix}}_{\equiv \mathbf{k}_o^T} \cdot \mathbf{x}_o \quad (3.60)$$

Hiermit kann der Regelkreis geschlossen werden und es resultiert das System

$$\overline{\mathfrak{D}}_o = \begin{cases} \mathbf{x}_o(k+1) = \begin{bmatrix} \mathbf{A} - \mathbf{b}_o \mathbf{k}_o^T & -\mathbf{b}_o \mathbf{k}_o^T \\ \mathbf{0}^{3 \times 3} & \mathbf{I}^{3 \times 3} \end{bmatrix} \cdot \mathbf{x}_o(k) \\ \mathbf{y}(k) = \mathbf{C}_o \cdot \mathbf{x}_o(k) \end{cases} \quad (3.61)$$

Für den geschlossenen Regelkreis ergeben sich die Eigenwerte

$$\lambda_1 = 0,7895 \quad \lambda_{2,3} = 0,8830 \pm 0,0087j \quad \lambda_{4,5,6} = 1. \quad (3.62)$$

Diese zeigen, dass die ursprünglichen Eigenwerte erhalten bleiben und somit die Eigenbewegung nach wie vor stabilisiert wird. Die zusätzlichen Eigenwerte $\lambda_{4,5,6} = 1$ sind den Messabweichungen zuzuordnen, welche als konstant modelliert wurden. Hieraus lässt sich folgern, dass der geschlossene Regelkreis als lineares System für beliebige Messabweichungen stabil ist, insofern diese nicht von einem instabilen Vorgang verursacht werden. Um den Einfluss der Messabweichungen auf die Trajektorie \mathbf{x} zu ermitteln wird das System $\overline{\mathfrak{D}}_o$ in die kanonische

Normalform transformiert.

$$\tilde{\mathbf{A}}_o = \mathbf{V}_o^{-1} \overline{\mathbf{A}}_o \mathbf{V}_o = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_6 \end{bmatrix} \quad \mathbf{x}_o = \mathbf{V}_o \cdot \tilde{\mathbf{x}}_o \quad (3.63)$$

Interessant ist hierbei die Form der Transformationsmatrix

$$\mathbf{V}_o = \begin{bmatrix} & 0 & 0 & 0 \\ \mathbf{V} & 0 & 0 & 0 \\ & \gamma_1 & \gamma_2 & \gamma_3 \\ \mathbf{0}^{3 \times 3} & & \mathbf{I}^{3 \times 3} & \end{bmatrix}. \quad (3.64)$$

Zunächst geht aus den unteren drei Zeilen hervor, dass es sich bei den Messabweichungen $\hat{\mathbf{x}}$ um kanonische Zustandsvariablen handelt. Des weiteren bleiben die Eigenvektoren \mathbf{V} des ursprünglichen Systems $\overline{\mathfrak{D}}$ und somit auch dessen Eigenbewegung erhalten. Die Messabweichungen wirken lediglich über die Faktoren γ_i auf die Geschwindigkeit u_R der Schwungmasse ein. Unter der Annahme, dass die Messabweichungen konstant sind, führen diese zu einer verbleibenden Bewegung der Schwungmasse. Deshalb muss die vorherige Stabilitätsaussage auf das Gebiet der Zustands- und Stellgrößen begrenzt werden für welche die Stellgröße nicht begrenzt ist. Beispielsweise kann das System nicht mehr stabilisiert werden, wenn der Messfehler so groß ist, dass $\gamma \cdot \hat{\mathbf{x}}$ die maximale Drehzahl des Motors überschreitet.

Hier gehört ein versuch rein der dieses Theorie untermauert. Nach diesem Modell geht phi gegen null folglich ist der Anzeigewert am ende reine Messabweichung, die kann ich händisch korrigieren, dann sollte die Schwungmasse langsamer rotieren.

3.5 Beobachtung der Messabweichungen

Prinzipiell ist es möglich die Messabweichungen von φ und u_K empirisch zu ermitteln. Allerdings ist die Annahme, dass es sich bei den Messabweichungen um Konstanten handelt, kritisch zu betrachten. Die Ursache der Messfehler wird im Modell nicht erfasst. Werden die Größen $\hat{\mathbf{x}}$ von einer Zeitfunktion verursacht so werden diese ebenfalls zeitvariant. Allerdings ist die exakte Modellierung aufwendig und führt zu einer Erhöhung der Modellordnung. Um diese Problematik zu vermeiden wird ein Luenberger-Beobachter verwendet, welcher die Messfehler ermittelt.

Hierfür müssen zunächst die Begriffe der Steuer- und Beobachtbarkeit geklärt werden. Bei den Zustandsgrößen $\hat{\mathbf{x}}$ handelt es sich um so genannte nicht steuerbare Eigenvorgänge. Die Begründung liegt einerseits darin, dass die unteren drei Elemente des Eingangsvektors \mathbf{b} gleich null sind. Folglich wirkt die Stellgröße nicht direkt auf die Zustände ein. Andererseits beeinflusst keiner der Zustände \mathbf{x} die Messabweichungen, weshalb keine indirekte Steuerung über die Stellgröße erfolgen kann. Allgemein gilt:

” Ein System Σ heißt vollständig steuerbar, wenn es in endlicher Zeit t_e von jedem beliebigen Anfangszustand \mathbf{x}_0 durch eine geeignet gewählte Eingangsgröße $\mathbf{u}_{[0,t_e]}$ in einen beliebig vorgegebenen Endzustand $\mathbf{x}(t_e)$ überführt werden kann.”

LunzeRT2, S. 64

Zur Prüfung der Steuerbarkeit kann das Kalmankriterium angewandt werden [Lunze RT2, S.65ff]. Die nicht Steuerbarkeit der Messabweichungen begründet, weshalb kein Regler verwendet werden kann um diese zu eliminieren. Allerdings sind diese beobachtbar. Das heißt sie können aus dem Verlauf des Ausgangs- und Stellvektors rekonstruiert werden.

” Ein System $\Sigma = (\mathbf{A}, \mathbf{B}, \mathbf{C})$ heißt vollständig beobachtbar, wenn der Anfangszustand \mathbf{x}_0 aus dem über einem endlichen Intervall $[0, t_e]$ bekannten Verlauf der Eingangsgröße $\mathbf{u}_{[0,t_e]}$ und der Ausgangsgröße $\mathbf{y}_{[0,t_e]}$ bestimmt werden kann.

”

LunzeRT2, S. 92

Zu der Prüfung der Beobachtbarkeit kann ebenfalls das Kalmankriterium genutzt werden [LunzeRT2, S.93ff]. Da das System $\overline{\mathfrak{D}}_o$ vollständig beobachtbar ist, kann ein Luenberger-Beobachter verwendet werden um den Zustandsvektor \mathbf{x}_o zu schätzen. Das Grundprinzip der Beobachtung besteht darin, parallel zu der Regelstrecke das Modell zu berechnen und daraus den Zustandsvektor zu schätzen. Die Problematik dieses Ansatz besteht darin, dass ein beliebiger Modellfehler zu einem, mit der Zeit zunehmenden, Schätzfehler führt. Deshalb wird das in der Regelungstechnik bewährte Prinzip der Fehlerrückführung übernommen. Da der Zustandsvektor \mathbf{x} der Regelstrecke nicht zur Verfügung steht wird die Differenz

$$\delta \mathbf{y} = \mathbf{y} - \hat{\mathbf{y}} \quad (3.65)$$

der Ausgangsgrößen berechnet. Diese fließt über die Beobachtermatrix \mathbf{L} in den Schätzwert $\hat{\mathbf{x}}$ des Zustandes ein.

BILD

Um das Verhalten des Beobachters zu untersuchen wird der Schätzfehler $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$ betrachtet.

$$\begin{aligned}
 \mathbf{e}(k+1) &= \mathbf{x}(k+1) - \hat{\mathbf{x}}(k+1) \\
 &= [\mathbf{A} \cdot \mathbf{x}(k) + \mathbf{B} \cdot \mathbf{u}(k)] - [\mathbf{A} \cdot \hat{\mathbf{x}}(k) + \mathbf{B} \cdot \mathbf{u}(k) + \mathbf{L}\hat{\mathbf{y}}(k)] \\
 &= [\mathbf{A} \cdot (\mathbf{x}(k) - \hat{\mathbf{x}}(k))] - \mathbf{LC} \cdot [\mathbf{x}(k) - \hat{\mathbf{x}}(k)] \\
 &= \mathbf{A} \cdot \mathbf{e}(k) - \mathbf{LC} \cdot \mathbf{e}(k) = (\mathbf{A} - \mathbf{LC}) \cdot \mathbf{e}(k)
 \end{aligned} \tag{3.66}$$

Kapitel 4

Modellbildung Würfel auf Ecke

Das nächste Ziel besteht darin ein Regelungskonzept zu entwickeln, welches das Balancieren des Würfels auf einer Ecke ermöglicht. Hierfür werden drei Motor verwendet, wodurch das gesamte System über sechs Freiheitsgrade verfügt. Die Vorgehensweise erfolgt analog zu dem Reglerentwurf der Würfelseite. Somit besteht der erste Schritt in dem Entwurf eines mechanischen Modells, welches wiederum zu einer Zustandsraumdarstellung führt, die als Grundlage für den Reglerentwurf verwendet wird. Zu Beginn dieses Kapitels werden die Systemparameter vorgestellt, deren Bestimmung diskutiert und erste Annahmen getroffen um die weitere Modellbildung zu vereinfachen. Im zweiten Abschnitt wird die Kinematik des Systems untersucht. Hierbei werden zunächst die nötigen Bezugssysteme und generalisierten Koordinaten definiert, die anschließend für die Bestimmung der generalisierten und partiellen Geschwindigkeiten benötigt werden. Der nächste Abschnitt widmet sich der Kinematik. Hierunter fallen sowohl die wirkenden Kräfte und die draus resultierenden Drehmomente als auch die Trägheitsmomente der Körper. Daraus werden die generalisierten Kräfte und Trägheitskräfte ermittelt, welche nach Kanes Gleichungen auf die Bewegungsgleichungen führen. Diese werden anschließend in eine Zustandsraumdarstellung überführt.

4.1 Systemparameter

Zunächst soll die Parameter des mechanischen Systems vorgestellt und diskutiert werden. Das System setzt sich aus drei Schwungmassen und dem Würfelförper. Unter dem Würfelförper ist das Würfelgehäuse inklusive der montierten Motoren, Sensoren und Elektrik zu verstehen und wird mit K bezeichnet. Bei der Herleitung der Bewegungsgleichungen wird die Annahme getroffen, dass der Würfelförper nicht translativ bewegt wird, sondern lediglich um Punkt O rotiert. Der Punkt O ist hierbei die Ecke auf welcher der Würfel balanciert. Des weiteren beschreiben alle Ortsvektoren den Vektor von O zu dem jeweiligen Zielpunkt. Die drei Schwungmassen R_i sind mit jeweils einem rotatorischem Freiheitsgrad auf den Motorwellen gelagert. Die Position der Lagerung wird mit M_i bezeichnet und fällt auf Grund des symmetrischen Aufbau der Schwungmassen mit deren Schwerpunkt zusammen. Die Massen m_R und Trägheitstensoren $\mathbf{I}^{Ri/Mi}$ der Schwungmassen werden mit Hilfe der CAD-Anwendung ermittelt, wobei die Trägheitstensoren relativ zu den Punkten M_i berechnet werden.

$$m_R = 0,155[kg] \quad (4.1)$$

Die Trägheitstensoren werden dabei aus der Perspektive des körperfesten Bezugssystem K ermittelt, welches in den folgenden Abschnitt näher erläutert wird. Für den Tensor der Schwungmasse R_1 ergibt sich der folgende Wert.

$$\mathbf{I}^{R1/M1} = \begin{pmatrix} 3,358 \cdot 10^{-4} & 2,641 \cdot 10^{-11} & 0 \\ 2,651 \cdot 10^{-11} & 1,961 \cdot 10^{-4} & 4,527 \cdot 10^{-9} \\ 0 & 4,527 \cdot 10^{-9} & 1,691 \cdot 10^{-4} \end{pmatrix} [kg \cdot m^2] \quad (4.2)$$

Hieran ist zu erkennen, dass die Vektorbasis des Bezugssystem K nahezu den Haupträgheitsachsen der Schwungmasse entspricht, da die Deviationsmomente um die Größenordnung 10^5 kleiner als die Haupträgheitsmomente sind. Deshalb wird bei der folgenden Untersuchung die Annahme getroffen, dass die Deviationsmomente vernachlässigt werden können.

$$\begin{aligned} \mathbf{I}^{R1/M1} &= \begin{pmatrix} I_{11}^{R1} & 0 & 0 \\ 0 & I_{22}^{R1} & 0 \\ 0 & 0 & I_{33}^{R1} \end{pmatrix} = \begin{pmatrix} 3,358 \cdot 10^{-4} & 0 & 0 \\ 0 & 1,961 \cdot 10^{-4} & 0 \\ 0 & 0 & 1,691 \cdot 10^{-4} \end{pmatrix} [kg \cdot m^2] \\ \mathbf{I}^{R2/M2} &= \begin{pmatrix} I_{11}^{R2} & 0 & 0 \\ 0 & I_{22}^{R2} & 0 \\ 0 & 0 & I_{33}^{R2} \end{pmatrix} = \begin{pmatrix} 1,691 \cdot 10^{-4} & 0 & 0 \\ 0 & 3,358 \cdot 10^{-4} & 0 \\ 0 & 0 & 1,961 \cdot 10^{-4} \end{pmatrix} [kg \cdot m^2] \\ \mathbf{I}^{R3/M3} &= \begin{pmatrix} I_{11}^{R3} & 0 & 0 \\ 0 & I_{22}^{R3} & 0 \\ 0 & 0 & I_{33}^{R3} \end{pmatrix} = \begin{pmatrix} 1,961 \cdot 10^{-4} & 0 & 0 \\ 0 & 1,691 \cdot 10^{-4} & 0 \\ 0 & 0 & 3,358 \cdot 10^{-4} \end{pmatrix} [kg \cdot m^2] \end{aligned} \quad (4.3)$$

Für die Masse m_K und den Trägheitstensor $\mathbf{I}^{GH/O}$ des Würfelförpers um den Punkt O aus Perspektive des Bezugssystem K ergeben sich die folgenden Werte. Bei der Berechnung der des Tensors $\mathbf{I}^{GH/O}$ wird der Einfluss der Schwungmassen nicht beachtet, dies erfolgt bei der

Berechnung der Trägheitsmomente in den folgenden Abschnitten.

$$m_K = 1.07[kg] \quad (4.4)$$

$$\mathbf{I}^{GH/O} = \begin{pmatrix} I_{11}^{GH/O} & I_{12}^{GH/O} & I_{13}^{GH/O} \\ I_{21}^{GH/O} & I_{22}^{GH/O} & I_{23}^{GH/O} \\ I_{31}^{GH/O} & I_{32}^{GH/O} & I_{33}^{GH/O} \end{pmatrix} = \begin{pmatrix} 1,520 \cdot 10^{-2} & -5,201 \cdot 10^{-3} & 5,375 \cdot 10^{-3} \\ -5,201 \cdot 10^{-3} & 1,52 \cdot 10^{-2} & 5,225 \cdot 10^{-3} \\ 5,375 \cdot 10^{-3} & 5,225 \cdot 10^{-3} & 1,542 \cdot 10^{-2} \end{pmatrix} [kg \cdot m^2] \quad (4.5)$$

Für die Masse m des Gesamtsystems folgt

$$m = 1,532[kg]. \quad (4.6)$$

Der Ortsvektor \mathbf{c} des Schwerpunkts des Gesamtsystems wird ebenfalls numerisch ermittelt. Da sich die Komponenten des Ortsvektors \mathbf{c} lediglich um $10^{-1}mm$ unterscheiden werden diese als identischen angenommen.

$$\mathbf{c} = \begin{pmatrix} -6,61 \\ -6,60 \\ -6,57 \end{pmatrix} [cm] \approx \begin{pmatrix} l_C \\ l_C \\ l_C \end{pmatrix} \quad | \quad l_C = 6,6[cm] \quad (4.7)$$

Des weiteren entsteht durch die Bewegung der Schwungmassen ein Reibmoment, welches analog zu dem Modell der Würfelseite, als proportional zu den Winkelgeschwindigkeiten der Schwungmassen modelliert wird. Für Proportionalitätsfaktor C_ψ wurde experimentell der folgende Wert ermittelt.

$$C_\psi = 3,1176 \cdot 10^{-5} [kg \cdot m^2 \cdot s^{-1}] \quad (4.8)$$

4.2 Untersuchung der Kinematik

Der erste Schritt in der Modellbildung besteht in der Definition der Bezugssystem, welche zur Beschreibung der Systembewegung dienen. Der Ausgangspunkt ist das Inertialsystem A , welches durch die drei Einheitsvektoren \mathbf{a}_1 , \mathbf{a}_2 und \mathbf{a}_3 definiert wird. Das Würfelgehäuse verfügt über drei rotatorische Freiheitsgrade, welche durch die Winkel φ_1 , φ_2 und φ_3 beschrieben werden.

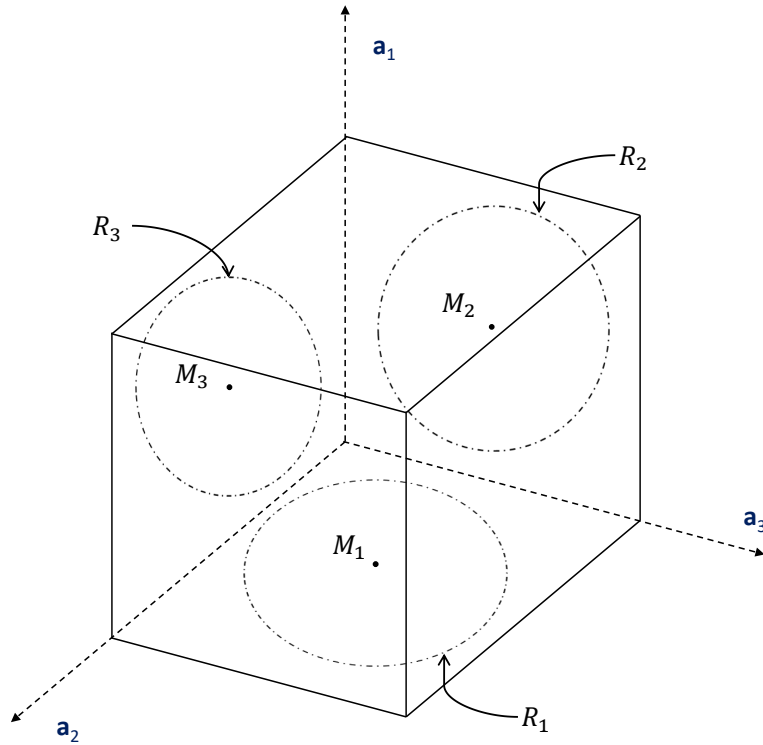


Abbildung 4.1: Mechanischer Aufbau der Würfelhälfte, Quelle: eigene Darstellung

Durch die Rotation des Würfels um den Winkel φ_1 in Richtung des Vektors \mathbf{a}_1 entsteht das Hilfsbezugssystem B , das durch die Einheitsvektoren \mathbf{b}_1 , \mathbf{b}_2 und \mathbf{b}_3 aufgespannt wird.

$${}^A\mathbf{v} = {}^B({}^A\mathbf{P}^B \cdot {}^A\mathbf{v}) = {}^B\mathbf{v} \quad | \quad {}^A\mathbf{P}^B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_{\varphi_1} & s_{\varphi_1} \\ 0 & -s_{\varphi_1} & c_{\varphi_1} \end{pmatrix} \quad (4.9)$$

Die Rotation um den Winkel φ_2 in Richtung des Vektors \mathbf{b}_2 führt zu dem zweiten Hilfsbezugssystem C mit den drei Einheitsvektoren \mathbf{c}_1 , \mathbf{c}_2 und \mathbf{c}_3 .

$${}^B\mathbf{v} = {}^C({}^B\mathbf{P}^C \cdot {}^B\mathbf{v}) = {}^C\mathbf{v} \quad | \quad {}^B\mathbf{P}^C = \begin{pmatrix} c_{\varphi_2} & 0 & -s_{\varphi_2} \\ 0 & 1 & 0 \\ s_{\varphi_2} & 0 & c_{\varphi_2} \end{pmatrix} \quad (4.10)$$

Die letzte Rotation des Würfels in Richtung von \mathbf{c}_3 um den Winkel φ_3 führt zu dem körperfesten

Bezugssystem K , welches durch die drei Vektoren \mathbf{k}_1 , \mathbf{k}_2 und \mathbf{k}_3 definiert ist.

$${}^C\mathbf{v} = {}^K({}^C\mathbf{P}^K \cdot {}^C\mathbf{v}) = {}^K\mathbf{v} \quad | \quad {}^C\mathbf{P}^K = \begin{pmatrix} c_{\varphi_3} & s_{\varphi_3} & 0 \\ -s_{\varphi_3} & c_{\varphi_3} & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (4.11)$$

Hier sei angemerkt, dass es sich bei den Bezugssystem B und C um theoretische Konstrukte handelt, für die kein physisches Gegenstück existiert. Sie werden lediglich als Hilfsmittel zur Beschreibung des Systems verwendet.

Durch die Rotation der Schwungmassen besitzt das System drei weitere Freiheitsgrade, welche von den Winkeln ψ_1 , ψ_2 und ψ_3 beschrieben werden. Somit entstehen drei weitere Bezugssysteme, deren Vektorbasen jeweils an den Schwungmassen fixiert sind. Allerdings spielen diese keine weitere Rolle, da es sich bei den Winkeln ψ_i um zyklische Koordinaten handelt. Das heißt, dass der Impuls des Systems nicht von der Ausrichtung der Schwungmassen abhängt. Lediglich die Winkelgeschwindigkeiten $\dot{\psi}_i$ beeinflussen auf Grund der Reibung das System.

Die Position und Ausrichtung des Systems wird von den sechs Winkeln φ_i und ψ_i vollständig beschrieben. Deshalb werden diese als generalisierte Koordinaten q_i definiert.

$$q_i = \varphi_i \quad q_j = \psi_i \quad (i = 1, 2, 3; j = 4, 5, 6) \quad (4.12)$$

Mit Hilfe der Bezugssysteme und generalisierten Koordinaten können nun die Winkelgeschwindigkeit des Würfels ${}^A\boldsymbol{\omega}^K$ und der Schwungmassen ${}^A\boldsymbol{\omega}^{R_i}$ bestimmt werden. Diese ergeben sich aus der Addition der relativen Rotationsgeschwindigkeiten der Bezugssysteme zueinander.

$$\begin{aligned} {}^A\boldsymbol{\omega}^K &= {}^A\boldsymbol{\omega}^B + {}^B\boldsymbol{\omega}^C + {}^C\boldsymbol{\omega}^K = \begin{pmatrix} \dot{\varphi}_1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \dot{\varphi}_2 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \dot{\varphi}_3 \end{pmatrix} \\ &= \begin{pmatrix} \dot{\varphi}_2 \cdot s_{\varphi_3} + \dot{\varphi}_1 \cdot c_{\varphi_2} \cdot c_{\varphi_3} \\ \dot{\varphi}_2 \cdot c_{\varphi_3} - \dot{\varphi}_1 \cdot c_{\varphi_2} \cdot s_{\varphi_3} \\ \dot{\varphi}_3 + \dot{\varphi}_1 \cdot s_{\varphi_2} \end{pmatrix} \end{aligned} \quad (4.13)$$

Die Winkelgeschwindigkeiten der Schwungmassen ${}^K\boldsymbol{\omega}^{R_i}$ relativ zu dem Würfel entsprechen der ersten Ableitung der Winkel ψ_i . Mit Hilfe des Additionstheorems für Winkelgeschwindigkeiten kann daraus auch die absolute Winkelgeschwindigkeit der Schwungmassen ${}^A\boldsymbol{\omega}^{R_i}$ berechnet werden.

$${}^K\boldsymbol{\omega}^{R_1} = \begin{pmatrix} \dot{\psi}_1 \\ 0 \\ 0 \end{pmatrix} \quad {}^K\boldsymbol{\omega}^{R_2} = \begin{pmatrix} 0 \\ \dot{\psi}_2 \\ 0 \end{pmatrix} \quad {}^K\boldsymbol{\omega}^{R_3} = \begin{pmatrix} 0 \\ 0 \\ \dot{\psi}_3 \end{pmatrix} \quad (4.14)$$

$${}^A\boldsymbol{\omega}^{R_i} = {}^A\boldsymbol{\omega}^K + {}^K\boldsymbol{\omega}^{R_i} \quad (i = 1, 2, 3) \quad (4.15)$$

Im nächsten Schritt werden die absoluten Geschwindigkeiten der Teilsysteme in Komponenten zerlegt, welche sich aus den generalisierten Geschwindigkeiten u_i und partiellen Geschwin-

digkeiten ${}^A\omega_i^j$ zusammensetzen. Hierfür müssen zu nächst die generalisierten Geschwindigkeiten u_i definiert werden. An dieser Stelle sei erwähnt, dass die Definition der generalisierten Geschwindigkeiten die Form der resultierenden Bewegungsgleichungen stark beeinflusst. Das letztendliche Ziel bei der Wahl der generalisierten Geschwindigkeiten ist es möglichst einfache Bewegungsgleichungen zu erhalten. Dies kann, nach einer Heuristik von Kane, erreicht werden, in dem die generalisierten Geschwindigkeiten so gewählt werden, dass die Geschwindigkeiten der Körper im Intertialsystem auf möglichst einfache Terme reduziert werden können. In diesem Anwendungsfall werden nach diesem Ansatz die folgenden generalisierten Geschwindigkeiten gewählt.

$$\begin{aligned}
u_1 &= \dot{\varphi}_2 \cdot s_{\varphi_3} + \dot{\varphi}_1 \cdot c_{\varphi_2} \cdot c_{\varphi_3} \\
u_2 &= \dot{\varphi}_2 \cdot c_{\varphi_3} - \dot{\varphi}_1 \cdot c_{\varphi_2} \cdot s_{\varphi_3} \\
u_3 &= \dot{\varphi}_3 + \dot{\varphi}_1 \cdot s_{\varphi_2} \\
u_4 &= \dot{\varphi}_2 \cdot s_{\varphi_3} + \dot{\varphi}_1 \cdot c_{\varphi_2} \cdot c_{\varphi_3} + \dot{\psi}_1 \\
u_5 &= \dot{\varphi}_2 \cdot c_{\varphi_3} - \dot{\varphi}_1 \cdot c_{\varphi_2} \cdot s_{\varphi_3} + \dot{\psi}_2 \\
u_6 &= \dot{\varphi}_3 + \dot{\varphi}_1 \cdot s_{\varphi_2} + \dot{\psi}_3
\end{aligned} \tag{4.16}$$

Mit diesen Definitionen können die Winkelgeschwindigkeiten der Körper im A in die folgende Form gebracht werden.

$${}^A\omega^K = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}, {}^A\omega^{R1} = \begin{pmatrix} u_4 \\ u_2 \\ u_3 \end{pmatrix}, {}^A\omega^{R2} = \begin{pmatrix} u_1 \\ u_5 \\ u_3 \end{pmatrix}, {}^A\omega^{R3} = \begin{pmatrix} u_1 \\ u_2 \\ u_6 \end{pmatrix} \tag{4.17}$$

Die Einführung der generalisierten Geschwindigkeiten führt einerseits zu einem stark vereinfachten Ausdruck der absoluten Winkelgeschwindigkeiten. Andererseits können dadurch auch die partiellen Geschwindigkeiten ${}^A\omega_i^j$ in einfachen Termen ausgedrückt werden, wie die folgenden Gleichungen zeigen. Dadurch werden die folgenden Schritte der Modellbildung zunehmend erleichtert.

$${}^A\omega^K = u_1 \cdot \mathbf{k}_1 + u_2 \cdot \mathbf{k}_2 + u_3 \cdot \mathbf{k}_3 \rightarrow {}^A\omega_1^K = \mathbf{k}_1, {}^A\omega_2^K = \mathbf{k}_2, {}^A\omega_3^K = \mathbf{k}_3 \tag{4.18}$$

$${}^A\omega_4^K = 0, {}^A\omega_5^K = 0, {}^A\omega_6^K = 0 \tag{4.19}$$

$${}^A\omega^{R1} = u_4 \cdot \mathbf{k}_1 + u_2 \cdot \mathbf{k}_2 + u_3 \cdot \mathbf{k}_3 \rightarrow {}^A\omega_1^K = 0, {}^A\omega_2^K = \mathbf{k}_2, {}^A\omega_3^K = \mathbf{k}_3 \tag{4.20}$$

$${}^A\omega_4^K = \mathbf{k}_1, {}^A\omega_5^K = 0, {}^A\omega_6^K = 0 \tag{4.21}$$

$${}^A\omega^{R2} = u_1 \cdot \mathbf{k}_1 + u_5 \cdot \mathbf{k}_2 + u_3 \cdot \mathbf{k}_3 \rightarrow {}^A\omega_1^K = \mathbf{k}_1, {}^A\omega_2^K = 0, {}^A\omega_3^K = \mathbf{k}_3 \tag{4.22}$$

$${}^A\omega_4^K = 0, {}^A\omega_5^K = \mathbf{k}_2, {}^A\omega_6^K = 0 \tag{4.23}$$

$${}^A\omega^{R3} = u_1 \cdot \mathbf{k}_1 + u_2 \cdot \mathbf{k}_2 + u_6 \cdot \mathbf{k}_3 \rightarrow {}^A\omega_1^K = \mathbf{k}_1, {}^A\omega_2^K = \mathbf{k}_2, {}^A\omega_3^K = 0 \tag{4.24}$$

$${}^A\omega_4^K = 0, {}^A\omega_5^K = 0, {}^A\omega_6^K = \mathbf{k}_3 \tag{4.25}$$

Die Unterteilung der absoluten Winkelgeschwindigkeiten in generalisierte und partielle Geschwindigkeiten ermöglicht den Übergang der vektoriellen Rechnung in die skalare. Die generalisierten Geschwindigkeiten geben als skalare Größen den Betrag der Bewegung in die Richtung der jeweiligen Freiheitsgrade wieder. Wobei die partiellen Geschwindigkeiten diese

Richtung der Freiheitsgrade als vektorielle Größe mit Bezug auf das körperfeste Bezugssystem K wiedergeben. Dadurch können weitere Vektoren wie z.B. Trägheitsmomente durch die Skalarmultiplikation mit den partiellen Geschwindigkeiten in die Richtung der Freiheitsgrade abgebildet werden.

4.3 Untersuchung der Kinetik

Der nächste Schritt besteht darin die Kräfte zu modellieren, die auf den Würfelförper und die drei Schwungmassen wirken. Aus diesen können im Anschluss mit Hilfe der partiellen Geschwindigkeiten die generalisierten aktiven Kräfte F_i ermittelt werden. Zunächst sollen die resultierenden Drehmomente $\mathbf{T}^{Ri/Mi}$ der Schwungmassen R_i um ihren jeweiligen Drehpunkt M_i ermittelt werden. Hierfür muss einerseits das Drehmoment $\mathbf{T}_M^{Ri/Mi}$ des antreibenden Motors und das verzögernde Reibmoment $\mathbf{T}_R^{Ri/Mi}$ beachtet werden.

$$\mathbf{T}^{R1/M1} = \mathbf{T}_M^{R1/M1} + \mathbf{T}_R^{R1/M1} = \begin{pmatrix} T_{M1} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} -C_\psi \cdot (u_4 - u_1) \\ 0 \\ 0 \end{pmatrix} \quad (4.26)$$

$$\mathbf{T}^{R2/M2} = \mathbf{T}_M^{R2/M2} + \mathbf{T}_R^{R2/M2} = \begin{pmatrix} 0 \\ T_{M2} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ -C_\psi \cdot (u_5 - u_2) \\ 0 \end{pmatrix} \quad (4.27)$$

$$\mathbf{T}^{R3/M3} = \mathbf{T}_M^{R3/M3} + \mathbf{T}_R^{R3/M3} = \begin{pmatrix} 0 \\ 0 \\ T_{M3} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -C_\psi \cdot (u_6 - u_3) \end{pmatrix} \quad (4.28)$$

Der Würfelförper wird einerseits durch das Gravitationsmoment \mathbf{T}^{K/O_G} und die resultierenden Momente der Schwungmassen $\mathbf{T}^{Ri/Mi}$, welche in umgekehrter Richtung auf den Würfel wirken, beeinflusst. Das Gravitationsmoment hängt von der Gewichtskraft \mathbf{G} und der Position des Schwerpunktes \mathbf{c} ab.

$$\begin{aligned} \mathbf{T}_G^{K/O} = \mathbf{c} \times \mathbf{G} &= \begin{pmatrix} l_C \\ l_C \\ l_C \end{pmatrix} \times \begin{pmatrix} -m \cdot g \cdot c_{\varphi_2} \cdot c_{\varphi_3} \\ -m \cdot g \cdot c_{\varphi_2} \cdot s_{\varphi_3} \\ -m \cdot g \cdot s_{\varphi_2} \end{pmatrix} \\ &= -m \cdot l_C \cdot g \begin{pmatrix} s_{\varphi_2} + c_{\varphi_2} s_{\varphi_3} \\ -s_{\varphi_2} + c_{\varphi_2} c_{\varphi_3} \\ -c_{\varphi_2} c_{\varphi_3} - c_{\varphi_2} \cdot s_{\varphi_3} \end{pmatrix} \end{aligned} \quad (4.29)$$

$$\begin{aligned} \mathbf{T}^{K/O} &= \mathbf{T}_G^{K/O} - \mathbf{T}^{R1/M1} - \mathbf{T}^{R2/M2} - \mathbf{T}^{R3/M3} \\ &= -m \cdot l_C \cdot g \begin{pmatrix} s_{\varphi_2} + c_{\varphi_2} s_{\varphi_3} \\ -s_{\varphi_2} + c_{\varphi_2} c_{\varphi_3} \\ -c_{\varphi_2} c_{\varphi_3} - c_{\varphi_2} \cdot s_{\varphi_3} \end{pmatrix} - \begin{pmatrix} T_{M1} - C_\psi(u_4 - u_1) \\ T_{M2} - C_\psi(u_5 - u_2) \\ T_{M3} - C_\psi(u_6 - u_3) \end{pmatrix} \end{aligned} \quad (4.30)$$

Im nächsten Schritt können die generalisierten aktiven Kräfte berechnet werden. Dies geschieht durch die Summe der inneren Produkte der resultierenden Drehmomente und der entsprechenden partiellen Geschwindigkeit der Körper. Wenn die partiellen Geschwindigkeiten als die möglichen Bewegungsrichtungen der Körper betrachtet werden, so stellt die Skalarmultiplikation der partiellen Geschwindigkeit und dem resultierenden Drehmoments dessen Abbildung in die Bewegungsrichtung dar. Folglich handelt es sich bei den generalisierten aktiven Kräften um skalare Größen, welche den Einfluss der wirkenden Kräfte und

Momente in Richtung der Freiheitsgrade wiedergeben.

$$F_1 = \langle {}^A\boldsymbol{\omega}_1^K, \mathbf{T}^{K/O} \rangle + \sum_{i=1}^3 \langle {}^A\boldsymbol{\omega}_1^{Ri}, \mathbf{T}^{Ri/Mi} \rangle \quad (4.31)$$

$$= m \cdot l_C \cdot g(-s_{\varphi_2} - c_{\varphi_2}s_{\varphi_3}) - T_{M1} + C_\psi(u_4 - u_1)$$

$$F_2 = \langle {}^A\boldsymbol{\omega}_2^K, \mathbf{T}^{K/O} \rangle + \sum_{i=1}^3 \langle {}^A\boldsymbol{\omega}_2^{Ri}, \mathbf{T}^{Ri/Mi} \rangle \quad (4.32)$$

$$= m \cdot l_C \cdot g(s_{\varphi_2} - c_{\varphi_2}c_{\varphi_3}) - T_{M2} + C_\psi(u_5 - u_2)$$

$$F_3 = \langle {}^A\boldsymbol{\omega}_3^K, \mathbf{T}^{K/O} \rangle + \sum_{i=1}^3 \langle {}^A\boldsymbol{\omega}_3^{Ri}, \mathbf{T}^{Ri/Mi} \rangle \quad (4.33)$$

$$= m \cdot l_C \cdot g(c_{\varphi_2}c_{\varphi_3} + c_{\varphi_2}s_{\varphi_3}) - T_{M3} + C_\psi(u_6 - u_3)$$

$$F_4 = \langle {}^A\boldsymbol{\omega}_4^K, \mathbf{T}^{K/O} \rangle + \sum_{i=1}^3 \langle {}^A\boldsymbol{\omega}_4^{Ri}, \mathbf{T}^{Ri/Mi} \rangle = T_{M1} - C_\psi(u_4 - u_1) \quad (4.34)$$

$$F_5 = \langle {}^A\boldsymbol{\omega}_5^K, \mathbf{T}^{K/O} \rangle + \sum_{i=1}^3 \langle {}^A\boldsymbol{\omega}_5^{Ri}, \mathbf{T}^{Ri/Mi} \rangle = T_{M2} - C_\psi(u_5 - u_2) \quad (4.35)$$

$$F_6 = \langle {}^A\boldsymbol{\omega}_6^K, \mathbf{T}^{K/O} \rangle + \sum_{i=1}^3 \langle {}^A\boldsymbol{\omega}_6^{Ri}, \mathbf{T}^{Ri/Mi} \rangle = T_{M3} - C_\psi(u_6 - u_3) \quad (4.36)$$

Neben den aktiven Kräften müssen auch die generalisierten Trägheitskräfte F_i^* ermittelt werden um die Bewegungsgleichungen zu bestimmen. Hierfür werden zunächst die Trägheitsmomente \mathbf{T}_* der Körper ermittelt werden. Nach Kane gilt für diese Trägheitsmomente der folgende Zusammenhang.

$$\mathbf{T}_* = -\boldsymbol{\alpha} \cdot \mathbf{I} - \boldsymbol{\omega} \times (\mathbf{I} \cdot \boldsymbol{\omega}) \quad (4.37)$$

Wobei $\boldsymbol{\alpha}$ und $\boldsymbol{\omega}$ die Winkelbeschleunigung- bzw. geschwindigkeit des Körpers und \mathbf{I} dessen Trägheitstensor bezeichnen. Die Winkelgeschwindigkeiten der Körper sind bereits bekannt, die Winkelbeschleunigung ergeben sich durch die Ableitung der Winkelgeschwindigkeiten mit Bezug auf die Vektorbasis von A .

$${}^A\boldsymbol{\alpha}^K = \frac{d}{dt} {}^A\boldsymbol{\omega}^K, {}^A\boldsymbol{\alpha}^{R1} = \frac{d}{dt} {}^A\boldsymbol{\omega}^{R1}, {}^A\boldsymbol{\alpha}^{R2} = \frac{d}{dt} {}^A\boldsymbol{\omega}^{R2}, {}^A\boldsymbol{\alpha}^{R3} = \frac{d}{dt} {}^A\boldsymbol{\omega}^{R3} \quad (4.38)$$

Zunächst sollen die Trägheitsmomente $\mathbf{T}_*^{Ri/Mi}$ der Schwungmassen bestimmt werden. Deren Trägheitstensoren $\mathbf{I}^{Ri/Mi}$ im Bezug auf die Drehpunkte M_i wurde, wie in Abschnitt 1 erläutert, mit Hilfe der CAD-Programme ermittelt.

$$\mathbf{I}^{Ri/Mi} = \begin{pmatrix} I_{11}^{Ri} & 0 & 0 \\ 0 & I_{22}^{Ri} & 0 \\ 0 & 0 & I_{33}^{Ri} \end{pmatrix} \quad (4.39)$$

$$\mathbf{T}_*^{Ri/Mi} = -{}^A\boldsymbol{\alpha}^{Ri} - {}^A\boldsymbol{\omega}^{Ri} \times (\mathbf{I}^{Ri/Mi} \cdot {}^A\boldsymbol{\omega}^{Ri}) \quad (4.40)$$

Der Trägheitstensor $\mathbf{I}^{K/O}$ des Würfelkörpers um den Punkt O setzt sich aus mehreren Komponenten zusammen. Einerseits wird er durch den Trägheitstensor $\mathbf{I}^{GH/O}$ des Gehäuses um

den Punkt O beeinflusst, dieser Tensor beschreibt die Trägheitseigenschaften des Würfels ohne die Schwungmassen und kann mit Hilfer CAD-Anwendung ermittelt werden, andererseits beeinflussen die Schwungmassen das Trägheitsmoment des Würfelskörpers. Um diesen Einfluss nachzuvollziehen wird die Bewegung der Schwungmassen in zwei Komponenten zerlegt. Einerseits rotieren die Schwungmassen R_i um die Punkte M_i , welche die Schwerpunkte der Schwungmassen sind. Andererseits bewegen sich die Schwerpunkte M_i um den Punkt O und sind bei der Betrachtung der Trägheitseigenschaften dem Würfelskörper zuzuordnen, da sie auf diesem fixiert sind. Hierbei sind sie als Punktmassen mit der Masse m_R zu behandeln. Diese Interpretation entspricht der Aussage des Steiner'schen Satzes. Somit ist der Trägheitstensor $\mathbf{I}^{K/O}$ gleich der Summe von $\mathbf{I}^{GH/O}$ und den Trägheitstensoren der drei Massepunkte M_i mit der Masse m_R um O , wobei \mathbf{r}_i den Ortsvektor des Punktes M_i beschreibt.

$$\mathbf{I}^{K/O} = \mathbf{I}^{GH/O} + \sum_{i=1}^3 m_R (\langle \mathbf{r}_i, \mathbf{r}_i \rangle - \mathbf{r}_i \otimes \mathbf{r}_i) \quad (4.41)$$

$$\mathbf{T}_*^{K/O} = - {}^A\boldsymbol{\alpha}^K \cdot \mathbf{I}^{K/O} - {}^A\boldsymbol{\omega}^K \times (\mathbf{I}^{K/O} \cdot {}^A\boldsymbol{\omega}^K) \quad (4.42)$$

Prinzipiell können nun analog zu den generalisierten aktiven Kräfte durch die Skalarmultiplikation mit den partiellen Geschwindigkeiten die generalisierten Trägheitskräfte berechnet werden. Allerdings handelt es sich bei Trägheitsmomenten um recht komplexe und insbesondere stark nichtlinearen Termen. Diese führen wiederum auf schwer nachzuvollziehende Bewegungsgleichungen. Außerdem werden für die Transformation in eine Zustandsraumdarstellung lineare Differentialgleichungen vorliegen. Für diesen Fall schlägt Kane eine vorzeitige Linearisierung vor. Das heißt an Stelle die vollständigen, nicht linearen Bewegungsgleichungen zu bestimmen und anschließend zu linearisieren, werden bereits die generalisierten Geschwindigkeiten $\hat{\boldsymbol{\omega}}$, sowie die resultierenden Dreh- und Trägheitsmomente \hat{F}_i und \hat{F}_i^* linearisiert und daraus die linearen Bewegungsgleichungen bestimmt. Hierfür muss zunächst der Arbeitspunkt des Systems bestimmt werden, welcher der balancieren Position entspricht. Die Winkel φ_i müssen folglich so gewählt werden, dass der Ortsvektor des Schwerpunktes \mathbf{c} aus Perspektive des Inertialsystems lediglich eine Komponente in Richtung von \mathbf{a}_1 besitzt.

$${}^A \begin{pmatrix} |\mathbf{c}| \\ 0 \\ 0 \end{pmatrix} \stackrel{!}{=} {}^K \mathbf{P}^A \cdot \begin{pmatrix} l_C \\ l_C \\ l_C \end{pmatrix} \rightarrow \varphi_{10} = 0, \quad \varphi_{20} = -2 \cdot \text{atan}(\sqrt{2} - \sqrt{3}), \quad \varphi_{30} = \frac{-\pi}{4} \quad (4.43)$$

$$\hat{\varphi}_i = \varphi_{i0} + \bar{\varphi}_i \quad (4.44)$$

In dem Gleichgewichtspunkt verschwinden die Winkelgeschwindigkeiten des Systems, folglich gilt für die generalisierten Geschwindigkeiten im Arbeitspunkt $u_{i0} = 0$.

$$\hat{u}_1 = \dot{\varphi}_2 \cdot s_{\varphi_{30}} + \dot{\varphi}_1 \cdot c_{\varphi_{20}} c_{\varphi_{30}} \quad (4.45)$$

$$\hat{u}_2 = \dot{\varphi}_2 \cdot c_{\varphi_{30}} - \dot{\varphi}_1 \cdot c_{\varphi_{20}} s_{\varphi_{30}} \quad (4.46)$$

$$\hat{u}_3 = \dot{\varphi}_3 + \dot{\varphi}_1 \cdot s_{\varphi_{20}} \quad (4.47)$$

$$\hat{u}_4 = \dot{\varphi}_2 \cdot s_{\varphi_{30}} + \dot{\varphi}_1 \cdot c_{\varphi_{20}} c_{\varphi_{30}} + \dot{\psi}_1 \quad (4.48)$$

$$\hat{u}_5 = \dot{\varphi}_2 \cdot c_{\varphi_{30}} - \dot{\varphi}_1 \cdot c_{\varphi_{20}} s_{\varphi_{30}} + \dot{\psi}_2 \quad (4.49)$$

$$\hat{u}_6 = \dot{\varphi}_3 + \dot{\varphi}_1 \cdot s_{\varphi_{20}} + \dot{\psi}_3 \quad (4.50)$$

$${}^A\hat{\omega}^K = \begin{pmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_3 \end{pmatrix}, \quad {}^A\hat{\omega}^{R1} = \begin{pmatrix} \hat{u}_4 \\ \hat{u}_2 \\ \hat{u}_3 \end{pmatrix}, \quad {}^A\hat{\omega}^{R2} = \begin{pmatrix} \hat{u}_1 \\ \hat{u}_5 \\ \hat{u}_3 \end{pmatrix}, \quad {}^A\hat{\omega}^{R3} = \begin{pmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_6 \end{pmatrix} \quad (4.51)$$

Insbesondere die Winkelbeschleunigungen werden durch die linearisierten Winkelgeschwindigkeiten stark vereinfacht.

$${}^A\hat{\alpha}^K = \frac{A_d}{dt} {}^A\hat{\omega}^K = \begin{pmatrix} \ddot{\varphi}_2 \cdot s_{\varphi_{30}} + \ddot{\varphi}_1 \cdot c_{\varphi_{20}} c_{\varphi_{30}} \\ \ddot{\varphi}_2 \cdot c_{\varphi_{30}} - \ddot{\varphi}_1 \cdot c_{\varphi_{20}} s_{\varphi_{30}} \\ \ddot{\varphi}_3 + \ddot{\varphi}_1 \cdot s_{\varphi_{20}} \end{pmatrix} = \begin{pmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_3 \end{pmatrix} \quad (4.52)$$

$${}^A\hat{\alpha}^{R1} = \frac{A_d}{dt} {}^A\hat{\omega}^{R1} = \begin{pmatrix} \ddot{\varphi}_2 \cdot s_{\varphi_{30}} + \ddot{\varphi}_1 \cdot c_{\varphi_{20}} c_{\varphi_{30}} + \ddot{\psi}_1 \\ \ddot{\varphi}_2 \cdot c_{\varphi_{30}} - \ddot{\varphi}_1 \cdot c_{\varphi_{20}} s_{\varphi_{30}} \\ \ddot{\varphi}_3 + \ddot{\varphi}_1 \cdot s_{\varphi_{20}} \end{pmatrix} = \begin{pmatrix} \hat{u}_4 \\ \hat{u}_2 \\ \hat{u}_3 \end{pmatrix} \quad (4.53)$$

$${}^A\hat{\alpha}^{R2} = \frac{A_d}{dt} {}^A\hat{\omega}^{R2} = \begin{pmatrix} \ddot{\varphi}_2 \cdot s_{\varphi_{30}} + \ddot{\varphi}_1 \cdot c_{\varphi_{20}} c_{\varphi_{30}} \\ \ddot{\varphi}_2 \cdot c_{\varphi_{30}} - \ddot{\varphi}_1 \cdot c_{\varphi_{20}} s_{\varphi_{30}} + \ddot{\psi}_2 \\ \ddot{\varphi}_3 + \ddot{\varphi}_1 \cdot s_{\varphi_{20}} \end{pmatrix} = \begin{pmatrix} \hat{u}_1 \\ \hat{u}_5 \\ \hat{u}_3 \end{pmatrix} \quad (4.54)$$

$${}^A\hat{\alpha}^K = \frac{A_d}{dt} {}^A\hat{\omega}^K = \begin{pmatrix} \ddot{\varphi}_2 \cdot s_{\varphi_{30}} + \ddot{\varphi}_1 \cdot c_{\varphi_{20}} c_{\varphi_{30}} \\ \ddot{\varphi}_2 \cdot c_{\varphi_{30}} - \ddot{\varphi}_1 \cdot c_{\varphi_{20}} s_{\varphi_{30}} \\ \ddot{\varphi}_3 + \ddot{\varphi}_1 \cdot s_{\varphi_{20}} + \ddot{\psi}_3 \end{pmatrix} = \begin{pmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_6 \end{pmatrix} \quad (4.55)$$

Somit können nun die linearisierten Trägheitsmomente berechnet werden, wobei der zweite Term auf Grund der Linearisierung entfällt.

$$\hat{T}_*^{R1/M1} = - {}^A\hat{\alpha}^{R1} \cdot I^{R1/M1} = \begin{pmatrix} -\hat{u}_4 \cdot I_{11}^{R1} \\ -\hat{u}_2 \cdot I_{22}^{R1} \\ -\hat{u}_3 \cdot I_{33}^{R3} \end{pmatrix} \quad (4.56)$$

$$\hat{T}_*^{R2/M2} = - {}^A\hat{\alpha}^{R2} \cdot I^{R2/M2} = \begin{pmatrix} -\hat{u}_1 \cdot I_{11}^{R2} \\ -\hat{u}_5 \cdot I_{22}^{R2} \\ -\hat{u}_3 \cdot I_{33}^{R2} \end{pmatrix} \quad (4.57)$$

$$\hat{\mathbf{T}}_*^{R3/M3} = - {}^A\hat{\boldsymbol{\alpha}}^{R3} \cdot \mathbf{I}^{R3/M3} = \begin{pmatrix} -\hat{u}_1 \cdot I_{11}^{R3} \\ -\hat{u}_2 \cdot I_{22}^{R3} \\ -\hat{u}_6 \cdot I_{33}^{R3} \end{pmatrix}^K \quad (4.58)$$

Bei der Berechnung des Trägheitsmomentes $\hat{\mathbf{T}}^{K/O}$ muss beachtet werden, dass die Deviationsmomente des Tensors $\mathbf{I}^{K/O}$ nicht verschwinden und deshalb ein etwas komplexerer Ausdruck für das Trägheitsmoment resultiert.

$$\hat{\mathbf{T}}_*^{K/O} = - {}^A\hat{\boldsymbol{\alpha}}^K \cdot \mathbf{I}^{K/O} = - \begin{pmatrix} \hat{u}_1 \cdot I_{11}^K + \hat{u}_2 \cdot I_{21}^K + \hat{u}_3 \cdot I_{31}^K \\ \hat{u}_1 \cdot I_{12}^K + \hat{u}_2 \cdot I_{22}^K + \hat{u}_3 \cdot I_{32}^K \\ \hat{u}_1 \cdot I_{13}^K + \hat{u}_2 \cdot I_{23}^K + \hat{u}_3 \cdot I_{33}^K \end{pmatrix}^K \quad (4.59)$$

Nun können nun mit Hilfe der partiellen Geschwindigkeiten, welche durch die Linearisierung nicht beeinflusst werden, die generalisierten Trägheitskräfte \hat{F}_i^* berechnet werden.

$$\begin{aligned} \hat{F}_1^* &= \langle \hat{\mathbf{T}}_*^{K/O}, {}^A\boldsymbol{\omega}_1^K \rangle + \sum_{i=1}^3 \langle \hat{\mathbf{T}}_*^{Ri/Mi}, {}^A\boldsymbol{\omega}_1^{Ri} \rangle \\ &= -\hat{u}_1(I_{11}^K + I_{11}^{R2} + I_{11}^{R3}) - \hat{u}_2 \cdot I_{21}^K - \hat{u}_3 \cdot I_{31}^K \end{aligned} \quad (4.60)$$

$$\begin{aligned} \hat{F}_2^* &= \langle \hat{\mathbf{T}}_*^{K/O}, {}^A\boldsymbol{\omega}_2^K \rangle + \sum_{i=1}^3 \langle \hat{\mathbf{T}}_*^{Ri/Mi}, {}^A\boldsymbol{\omega}_2^{Ri} \rangle \\ &= -\hat{u}_1 \cdot I_{12}^K - \hat{u}_2(I_{22}^K + I_{22}^{R1} + I_{22}^{R3}) - \hat{u}_3 \cdot I_{32}^K \end{aligned} \quad (4.61)$$

$$\begin{aligned} \hat{F}_3^* &= \langle \hat{\mathbf{T}}_*^{K/O}, {}^A\boldsymbol{\omega}_3^K \rangle + \sum_{i=1}^3 \langle \hat{\mathbf{T}}_*^{Ri/Mi}, {}^A\boldsymbol{\omega}_3^{Ri} \rangle \\ &= -\hat{u}_1 \cdot I_{13}^K - \hat{u}_2 \cdot I_{23}^K - \hat{u}_3(I_{33}^K + I_{33}^{R1} + I_{33}^{R2}) \end{aligned} \quad (4.62)$$

$$\hat{F}_4^* = \langle \hat{\mathbf{T}}_*^{K/O}, {}^A\boldsymbol{\omega}_4^K \rangle + \sum_{i=1}^3 \langle \hat{\mathbf{T}}_*^{Ri/Mi}, {}^A\boldsymbol{\omega}_4^{Ri} \rangle - \hat{u}_4 \cdot I_{11}^{R1} \quad (4.63)$$

$$\hat{F}_5^* = \langle \hat{\mathbf{T}}_*^{K/O}, {}^A\boldsymbol{\omega}_5^K \rangle + \sum_{i=1}^3 \langle \hat{\mathbf{T}}_*^{Ri/Mi}, {}^A\boldsymbol{\omega}_5^{Ri} \rangle = -\hat{u}_5 \cdot I_{22}^{R2} \quad (4.64)$$

$$\hat{F}_6^* = \langle \hat{\mathbf{T}}_*^{K/O}, {}^A\boldsymbol{\omega}_6^K \rangle + \sum_{i=1}^3 \langle \hat{\mathbf{T}}_*^{Ri/Mi}, {}^A\boldsymbol{\omega}_6^{Ri} \rangle = -\hat{u}_6 \cdot I_{33}^{R3} \quad (4.65)$$

Zuletzt müssen nur noch die generalisierten aktiven Kräfte linearisiert werden um mit Hilfe von Kanes Gleichungen die Bewegungsgleichungen zu ermitteln. Hierbei muss lediglich das Gravitationsmoment $\mathbf{T}_G^{K/O}$ linearisiert werden, da die restlichen Momente bereits in linearer Form vorliegen.

$$\hat{\mathbf{T}}_G^{K/O} = \Delta \mathbf{G} \cdot \bar{\boldsymbol{\varphi}} \quad (4.66)$$

$$\Delta \mathbf{G} = -m \cdot g \cdot l_C \cdot \begin{bmatrix} 0 & (c_{\varphi 20} - s_{\varphi 20} c_{\varphi 30}) & c_{\varphi 20} c_{\varphi 30} \\ 0 & -c_{\varphi 20} - s_{\varphi 20} c_{\varphi 20} & -c_{\varphi 20} s_{\varphi 30} \\ 0 & s_{\varphi 20} s_{\varphi 30} + s_{\varphi 20} c_{\varphi 30} & c_{\varphi 20} s_{\varphi 30} - c_{\varphi 20} c_{\varphi 30} \end{bmatrix}, \quad \bar{\boldsymbol{\varphi}} = \begin{bmatrix} \bar{\varphi}_1 \\ \bar{\varphi}_2 \\ \bar{\varphi}_3 \end{bmatrix}$$

$$\hat{F}_1 = \langle \overline{\mathbf{T}}_G, \mathbf{k}_1 \rangle \quad \hat{F}_4 = T_{M1} - C_\psi(\hat{u}_4 - \hat{u}_1) \quad (4.67)$$

$$\hat{F}_2 = \langle \overline{\mathbf{T}}_G, \mathbf{k}_2 \rangle \quad \hat{F}_5 = T_{M2} - C_\psi(\hat{u}_5 - \hat{u}_2) \quad (4.68)$$

$$\hat{F}_3 = \langle \overline{\mathbf{T}}_G, \mathbf{k}_3 \rangle \quad \hat{F}_6 = T_{M3} - C_\psi(\hat{u}_6 - \hat{u}_3) \quad (4.69)$$

An dieser Stelle können nun prinzipiell nach Kanes Gleichung

$$F_i + F_i^* = 0 \quad (4.70)$$

die Bewegungsgleichungen bestimmt werden. Allerdings wird in diesem Fall eine vektorielle Vorgehensweise gewählt, welche einen eleganten Weg zur gewünschten Zustandsraumdarstellung darstellt. Zunächst werden die folgenden Definition getroffen.

$$\mathbf{u}_K \equiv \begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_3 \end{bmatrix} \quad \mathbf{u}_R \equiv \begin{bmatrix} \hat{u}_4 \\ \hat{u}_5 \\ \hat{u}_6 \end{bmatrix} \quad (4.71)$$

$$\begin{aligned} \mathbf{F}_K^* &\equiv \begin{bmatrix} -F_1^* \\ -F_2^* \\ -F_3^* \end{bmatrix} = \begin{bmatrix} \hat{u}_1(I_{11}^K + I_{11}^{R2} + I_{11}^{R3}) + \hat{u}_2 \cdot I_{21}^K + \hat{u}_3 \cdot I_{31}^K \\ \hat{u}_1 \cdot I_{12}^K + \hat{u}_2(I_{22}^K + I_{22}^{R1} + I_{22}^{R3}) + \hat{u}_3 \cdot I_{13}^K \\ \hat{u}_1 \cdot I_{13}^K + \hat{u}_2 \cdot I_{23}^K + \hat{u}_3(I_{33}^K + I_{33}^{R1} + I_{33}^{R2}) \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} I_{11}^K + I_{11}^{R1} + I_{11}^{R2} & I_{21}^K & I_{31}^K \\ I_{12}^K & I_{22}^K + I_{22}^{R1} + I_{22}^{R3} & I_{32}^K \\ I_{13}^K & I_{23}^K & I_{33}^K + I_{33}^{R1} + I_{33}^{R2} \end{bmatrix}}_{\equiv \mathbf{I}_K} \cdot \underbrace{\begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_3 \end{bmatrix}}_{\equiv \dot{\mathbf{u}}_K} \end{aligned} \quad (4.72)$$

$$\mathbf{F}_R^* \equiv \begin{bmatrix} -F_4^* \\ -F_5^* \\ -F_6^* \end{bmatrix} = \begin{bmatrix} \hat{u}_4 \cdot I_{11}^{R1} \\ \hat{u}_5 \cdot I_{22}^{R2} \\ \hat{u}_6 \cdot I_{33}^{R3} \end{bmatrix} = \underbrace{\begin{bmatrix} I_{11}^{R1} & 0 & 0 \\ 0 & I_{22}^{R2} & 0 \\ 0 & 0 & I_{33}^{R3} \end{bmatrix}}_{\equiv \mathbf{I}_R} \cdot \underbrace{\begin{bmatrix} \hat{u}_4 \\ \hat{u}_5 \\ \hat{u}_6 \end{bmatrix}}_{\equiv \dot{\mathbf{u}}_R} \quad (4.73)$$

$$\begin{aligned} \mathbf{F}_K &\equiv \Delta \mathbf{G} \cdot \overline{\boldsymbol{\varphi}} + C_\psi \cdot \begin{bmatrix} \hat{u}_4 - \hat{u}_1 \\ \hat{u}_5 - \hat{u}_2 \\ \hat{u}_6 - \hat{u}_3 \end{bmatrix} - \underbrace{\begin{bmatrix} T_{M1} \\ T_{M2} \\ T_{M3} \end{bmatrix}}_{\equiv \mathbf{T}_M} \\ &= \Delta \mathbf{G} \cdot \overline{\boldsymbol{\varphi}} + C_\psi \cdot \mathbf{u}_R - C_\psi \cdot \mathbf{u}_K - \mathbf{T}_M \end{aligned} \quad (4.74)$$

$$\mathbf{F}_R \equiv \begin{bmatrix} F_4 \\ F_5 \\ F_6 \end{bmatrix} = \sum_{i=1}^3 {}^K \mathbf{T}^{Ri/Mi} = \begin{bmatrix} T_{M1} \\ T_{M2} \\ T_{M3} \end{bmatrix} - C_\psi \cdot \begin{bmatrix} \hat{u}_4 - \hat{u}_1 \\ \hat{u}_5 - \hat{u}_2 \\ \hat{u}_6 - \hat{u}_3 \end{bmatrix} \quad (4.75)$$

Nach Kanes Gleichung gilt $F_i = -F_i^*$, woraus sowohl $\mathbf{F}_K = \mathbf{F}_K^*$ als auch $\mathbf{F}_R = \mathbf{F}_R^*$ folgt. Daraus resultieren die Gleichungen

$$\dot{\mathbf{u}}_K = \mathbf{I}_K^{-1} \cdot \mathbf{F}_K^* = \mathbf{I}_K^{-1} \cdot \mathbf{F}_K \quad (4.76)$$

$$\begin{aligned} &= \mathbf{I}_K^{-1} \cdot \Delta \mathbf{G} \cdot \overline{\boldsymbol{\varphi}} - C_\psi \cdot \mathbf{I}_K^{-1} \cdot \mathbf{u}_K + C_\psi \cdot \mathbf{I}_K^{-1} \cdot \mathbf{u}_R - \mathbf{I}_K^{-1} \cdot \mathbf{T}_M \\ \dot{\mathbf{u}}_R &= \mathbf{I}_R^{-1} \cdot \mathbf{F}_R^* = \mathbf{I}_R^{-1} \cdot \mathbf{F}_K = C_\psi \cdot \mathbf{I}_R^{-1} \cdot \mathbf{u}_K - C_\psi \cdot \mathbf{I}_R^{-1} \cdot \mathbf{u}_R + \mathbf{I}_R^{-1} \cdot \mathbf{T}_M. \end{aligned} \quad (4.77)$$

Wenn nun ein Zustandsvektor $\mathbf{x} \equiv [\bar{\varphi} \quad \mathbf{u}_K \quad \mathbf{u}_R]^T$ und ein Eingangsvektor $\mathbf{u} \equiv \mathbf{T}_M$ definiert werden, geben die obigen Gleichungen bereits zwei Drittel der gesuchten Zustandsraumdarstellung wieder. Der letzte Teil ergibt sich durch die Ableitung des Winkelvektors $\bar{\varphi}$. Hierfür wird die Definition der generalisierten Geschwindigkeiten \mathbf{u}_K nach den Winkeln $\bar{\varphi}$ aufgelöst.

$$\dot{\bar{\varphi}} = \underbrace{\begin{bmatrix} \frac{c\varphi_{30}}{c\varphi_{20}} & \frac{-s\varphi_{30}}{c\varphi_{20}} & 0 \\ s\varphi_{30} & c\varphi_{30} & 0 \\ \frac{-s\varphi_{20}c\varphi_{30}}{c\varphi_{20}} & \frac{s\varphi_{20}s\varphi_{30}}{c\varphi_{20}} & 1 \end{bmatrix}}_{\equiv \Delta\Phi} \cdot \mathbf{u}_K \quad (4.78)$$

Nun können die Gleichungen (4.78), (4.76) und (4.77) zu einer Zustandsraumdarstellung zusammengeführt werden.

$$\underbrace{\begin{bmatrix} \dot{\bar{\varphi}} \\ \dot{\mathbf{u}}_K \\ \dot{\mathbf{u}}_R \end{bmatrix}}_{\equiv \dot{\mathbf{x}}} = \underbrace{\begin{bmatrix} 0^{3 \times 3} & \Delta\Phi & 0^{3 \times 3} \\ \mathbf{I}_K^{-1} \cdot \Delta\mathbf{G} & -C_\psi \cdot \mathbf{I}_K^{-1} & C_\psi \cdot \mathbf{I}_K^{-1} \\ 0^{3 \times 3} & C_\psi \cdot \mathbf{I}_R^{-1} & -C_\psi \cdot \mathbf{I}_R^{-1} \end{bmatrix}}_{\equiv \mathbf{A}} \cdot \underbrace{\begin{bmatrix} \bar{\varphi} \\ \mathbf{u}_K \\ \mathbf{u}_R \end{bmatrix}}_{\equiv \mathbf{x}} + \underbrace{\begin{bmatrix} 0^{3 \times 3} \\ -\mathbf{I}_K^{-1} \\ \mathbf{I}_R^{-1} \end{bmatrix}}_{\equiv \mathbf{B}} \cdot \underbrace{\begin{bmatrix} T_{M1} \\ T_{M2} \\ T_{M3} \end{bmatrix}}_{\equiv \mathbf{u}} \quad (4.79)$$

Kapitel 5

Reglerentwurf für das Balancieren auf einer Ecke

Nachdem im vorherigen Kapitel ein Modell des Systems entwickelt wurde muss nun für dieses ein Regler entworfen werden. Hierfür wird zunächst die kontinuierliche Zustandsraumdarstellung in eine diskrete überführt. Anschließend wird diese in eine Kalman-Dekomposition transformiert, um das System auf eine Minimalrealisierung zu reduzieren. Diese besteht lediglich aus steuer- und beobachtbaren Zuständen und kann somit als Ausgangspunkt für den Reglerentwurf genutzt werden. Für die Bestimmung des Reglers wird ebenfalls das LQR-Verfahren verwendet. Der Regler wird im nächsten Schritt an Hand einer Simulation überprüft, wobei der Verlauf der nicht steuerbar bzw. beobachtbaren Zustände untersucht wird. Des weiteren wird mittels der Simulation die Direktionalitätsproblematik des Mehrgrößensystems analysiert, welche durch die Stellgrößenbeschränkungen verursacht wird. Der resultierende Regler wird im Anschluss auf die reelle Strecke übertragen und dort empirisch optimiert.

5.1 Modell basierter Reglerentwurf

Aus dem letzten Kapitel ist die Zustandsgleichung

$$\dot{\mathbf{x}} = \underbrace{\begin{bmatrix} \mathbf{0}^{3 \times 3} & \Delta\Phi & \mathbf{0}^{3 \times 3} \\ \mathbf{I}_K^{-1} \cdot \Delta\mathbf{T}_G & -C_\psi \cdot \mathbf{I}_K^{-1} & C_\psi \cdot \mathbf{I}_K^{-1} \\ \mathbf{0}^{3 \times 3} & C_\psi \cdot \mathbf{I}_R^{-1} & -C_\psi \cdot \mathbf{I}_R^{-1} \end{bmatrix}}_{=\mathbf{A}_c} \cdot \underbrace{\begin{bmatrix} \bar{\varphi} \\ \mathbf{u}_K \\ \mathbf{u}_R \end{bmatrix}}_{=\mathbf{x}} + \underbrace{\begin{bmatrix} \mathbf{0}^{3 \times 3} \\ -\mathbf{I}_K^{-1} \\ \mathbf{I}_R^{-1} \end{bmatrix}}_{=\mathbf{B}_c} \cdot \underbrace{\begin{bmatrix} T_{M1} \\ T_{M2} \\ T_{M3} \end{bmatrix}}_{=\mathbf{u}} \quad (5.1)$$

hervorgegangen. Der Ausgangsvektor $\mathbf{y} \in \mathcal{R}^8$ enthält alle Zustandsgrößen bis auf den Winkel φ_1 , da dieser nicht aus den Sensordaten ermittelt werden kann.

$$\mathbf{y} = \underbrace{\begin{bmatrix} \mathbf{0}^8 & \mathbf{I}^{8 \times 8} \end{bmatrix}}_{\equiv \mathbf{C}_c} \cdot \mathbf{x} \quad (5.2)$$

Im nächsten Schritt wird das kontinuierliche System $\mathfrak{S}(\mathbf{A}_c, \mathbf{B}_c, \mathbf{C}_c)$ in das diskrete System $\mathfrak{D}(\mathbf{A}, \mathbf{B}, \mathbf{C})$ überführt. Hierfür wird die Abtastrate $f_a = 50\text{Hz}$ verwendet. Anschließend wird das System mittels der Kalmankriterien auf Steuer- und Beobachtbarkeit geprüft. Hieraus folgt, dass das System sowohl einen nicht steuerbaren als auch einen nicht beobachtbaren Zustand besitzt. Eine Voraussetzung für den Entwurf eines Zustandsreglers besteht darin, dass das System vollständig steuer- und beobachtbar ist. Deshalb kann das System $\mathfrak{D}(\mathbf{A}, \mathbf{B}, \mathbf{C})$ nicht direkt für den Reglerentwurf verwendet werden. Zunächst muss eine Minimalrealisierung $\mathfrak{D}(\mathbf{A}_M, \mathbf{B}_M, \mathbf{C}_M)$ gefunden werden, welche nur den steuer- und beobachtbaren Unterraum des Systems enthält. Um eine Minimalrealisierung des Systems zu erhalten wird es in eine so genannte Kalman-Dekomposition $\mathfrak{D}(\mathbf{A}_K, \mathbf{B}_K, \mathbf{C}_K)$ überführt. Die hierfür nötige Transformationsmatrix \mathbf{T}_K wird mittels der Matlab-Funktion *minreal()* bestimmt.

$$\mathbf{x}_K = \mathbf{T}_K^{-1} \cdot \mathbf{x} \quad \mathbf{A}_K = \mathbf{T}_K^{-1} \cdot \mathbf{A} \cdot \mathbf{T}_K \quad \mathbf{B}_K = \mathbf{T}_K^{-1} \cdot \mathbf{B} \quad \mathbf{C}_K = \mathbf{C} \cdot \mathbf{T}_K \quad (5.3)$$

$$\underbrace{\begin{bmatrix} \mathbf{x}_M \\ x_{\bar{S}} \\ x_{\bar{B}} \end{bmatrix}}_{=\mathbf{x}_K}(k+1) = \underbrace{\begin{bmatrix} \mathbf{A}_M & \mathbf{a}_{M/\bar{S}} & 0 \\ \mathbf{0}^{1 \times 7} & a_{\bar{S}} & 0 \\ \mathbf{a}_{\bar{B}/M} & \mathbf{a}_{\bar{B}/\bar{S}} & a_{\bar{B}} \end{bmatrix}}_{=\mathbf{A}_K} \cdot \underbrace{\begin{bmatrix} \mathbf{x}_M \\ x_{\bar{S}} \\ x_{\bar{B}} \end{bmatrix}}_{=\mathbf{x}_K}(k) + \underbrace{\begin{bmatrix} \mathbf{B}_M \\ \mathbf{0}^{1 \times 3} \\ \mathbf{b}_{\bar{B}}^T \end{bmatrix}}_{=\mathbf{B}_K} \cdot \mathbf{u}(k) \quad (5.4)$$

$$\mathbf{y}(k) = \underbrace{\begin{bmatrix} \mathbf{C}_M & c_{\bar{S}} & \mathbf{0}^8 \end{bmatrix}}_{=\mathbf{C}_K} \cdot \mathbf{x}_K(k) \quad (5.5)$$

Die Kalman-Dekomposition $\mathfrak{D}_K(\mathbf{A}_K, \mathbf{B}_K, \mathbf{C}_K)$ setzt sich aus der Minimalrealisierung $\mathfrak{D}_M(\mathbf{A}_M, \mathbf{B}_M, \mathbf{C}_M)$, dem nicht steuerbaren Teilsystem $\mathfrak{D}_{\bar{S}}(a_{\bar{S}}, \mathbf{0}^{3 \times 1}, c_{\bar{S}})$ und dem nicht beobachtbaren Teilsystem $\mathfrak{D}_{\bar{B}}(a_{\bar{B}}, \mathbf{b}_{\bar{B}}^T, \mathbf{0}^{8 \times 1})$ zusammen.¹ Prinzipiell kann nun für das System \mathfrak{D}_M ein Zustandsregler entwickelt werden, der zu einem asymptotischen System führt. Allerdings ist dabei zu beachten, dass die Teilsysteme \mathfrak{D}_M , $\mathfrak{D}_{\bar{S}}$ und $\mathfrak{D}_{\bar{B}}$ nicht voneinander entkoppelt sind. Die Abhängigkeiten zwischen den Teilsystemen werden von den Größen $\mathbf{a}_{M/\bar{S}}$, $\mathbf{a}_{\bar{B}/M}$ und $a_{\bar{B}/\bar{S}}$ beschrieben. Da der Zustandsregler lediglich die Eigenwerte des Teilsystem \mathfrak{D}_M beeinflussen kann, muss

¹Die hier gezeigte Dekomposition stellt nicht den allgemein gültigen Fall dar, der ein weiteres Teilsystem weder steuer- noch beobachtbaren Zustände enthält [Lunze RT2, S.108ff]

zusätzlich die Stabilität der Teilsysteme $\mathfrak{D}_{\bar{S}}$ und $\mathfrak{D}_{\bar{B}}$ geprüft werden. In diesem Kontext wird der Begriff der Stabilisierbarkeit verwendet, wobei ein System stabilisierbar heißt wenn all seine Eigenvorgänge, die nicht Teil seines vollständig steuer- und beobachtbaren Unterraums sind, asymptotisch stabil sind. In diesem Anwendungsfall handelt es sich bei $\mathfrak{D}_{\bar{S}}$ und $\mathfrak{D}_{\bar{B}}$ um Systeme erster Ordnung, deren Eigenwert $\lambda_{\bar{S}}$ bzw. $\lambda_{\bar{B}}$ auf dem Einheitskreis liegt.

$$\lambda_{\bar{S}} = \lambda_{\bar{B}} = 1 \quad (5.6)$$

Somit sind die beiden Teilsysteme grenzstabil. Um den Verlauf und Einfluss der Zustände $x_{\bar{S}}$ und $x_{\bar{B}}$ nachzuvollziehen wird zunächst ein Regler für die Minimalrealisierung \mathfrak{D}_M entworfen und in einer Simulation erprobt. Für die Bestimmung der Reglermatrix \mathbf{K}_M wird das LQR-Verfahren angewandt, wobei die Gewichtungsmatrizen \mathbf{Q} und \mathbf{R} in Diagonalf orm gewählt werden. Als Diagonalelemente werden die Kehrwerte der maximal zulässigen Zustandsgrößen im Quadrat verwendet. Hierbei ist zu beachten, dass der Reglerentwurf für das System \mathfrak{D}_M durchgeführt wird. Deshalb muss der maximale Zustandsvektor \mathbf{x}_{max} in dessen Darstellung transformiert werden. Hierfür wird mit der Transformationsmatrix \mathbf{T}_K^{-1} und einer nicht regulären Reduktionsmatrix multipliziert.

$$\mathbf{x}_{M,max} = \begin{bmatrix} \mathbf{I}^{7 \times 7} & \mathbf{0}^{7 \times 2} \end{bmatrix} \cdot \mathbf{T}_K^{-1} \cdot \mathbf{x}_{max} \quad (5.7)$$

$$\mathbf{Q} = \text{diag}(\mathbf{x}_{M,max})^{-2} \quad \mathbf{R} = \text{diag}(\mathbf{u}_{max})^{-2} \quad (5.8)$$

In der Simulation wird für die Regelstrecke das kontinuierliche System $\mathfrak{S}_C(\mathbf{A}_C, \mathbf{B}_C, \mathbf{C}_C)$ verwendet. Für die Implementierung des Reglers ist ebenfalls in ([?]) beschriebene Transformation nötig.

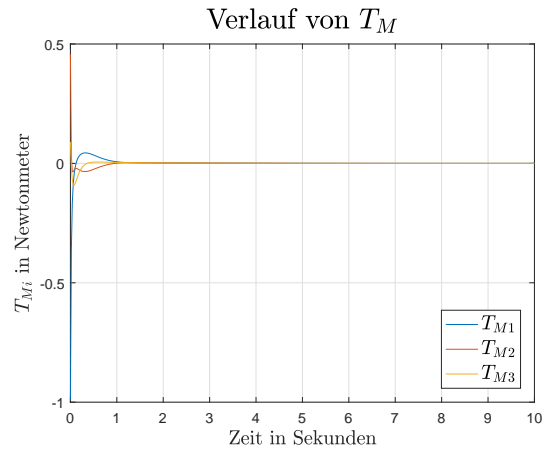
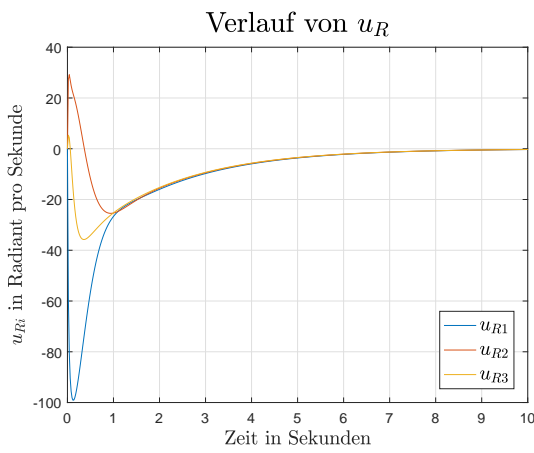
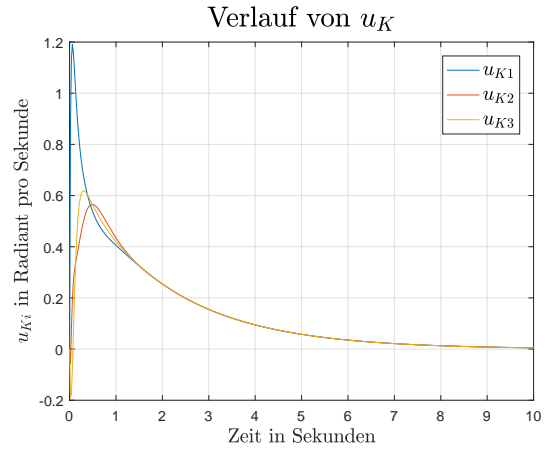
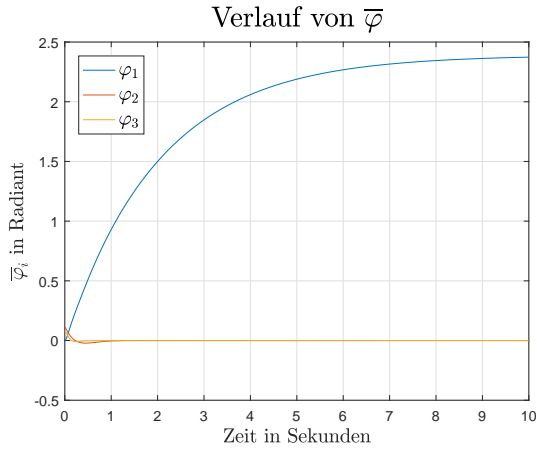
$$\mathbf{u}(k) = - \underbrace{\mathbf{K}_M \cdot \begin{bmatrix} \mathbf{I}^{7 \times 7} & \mathbf{0}^{7 \times 2} \end{bmatrix} \cdot \mathbf{T}_K^{-1}}_{\equiv \mathbf{K}} \cdot \mathbf{x}(k) \quad (5.9)$$

Die folgenden Abbildungen zeigen den Verlauf der Zustandsgrößen $\bar{\varphi}$, \mathbf{u}_K und \mathbf{u}_R sowie der Stellgrößen \mathbf{T}_M bei dem Anfangszustand

$$\mathbf{x}(0) = \begin{bmatrix} 0 \\ 0.11 \\ 0.08 \\ \mathbf{0}^{6 \times 1} \end{bmatrix}. \quad (5.10)$$

Die Simulation zeigt, dass alle Zustandsgrößen bis auf den Winkel φ_1 gegen null konvergieren. Folglich ist anzunehmen, dass der Zustand φ_1 nicht teil der Minimalrealisierung \mathfrak{D}_M ist. Diese Vermutung lässt sich mittels der Matrizen \mathbf{C}_C und \mathbf{A}_C bestätigen. Die erste Spalte der Matrix \mathbf{C}_C ist ein Nullvektor und somit beeinflusst φ_1 den Ausgangsvektor \mathbf{y} nicht. Eine solche Zustandsvariable wird auch als nicht ausgangsverbunden bezeichnet. Des weiteren hat der Zustand φ_1 keine Auswirkung auf die Systemdynamik. Dies ist auf die erste Spalte der Systemmatrix \mathbf{A}_C

$$\mathbf{A}_C(:, 1) = \begin{bmatrix} \mathbf{0}^{3 \times 1} \\ [\mathbf{I}_K^{-1} \cdot \Delta \mathbf{T}_G](:, 1) \\ \mathbf{0}^{3 \times 1} \end{bmatrix} = \mathbf{0}^{9 \times 1} \quad (5.11)$$



zurückzuführen, welche ebenfalls ein Nullvektor ist. Hieraus lässt sich schließen, dass es sich bei dem Winkel φ_1 um den nicht beobachtbaren Zustand $x_{\bar{B}}$ handelt. Dies wird auch von der Struktur der inversen Transformationsmatrix

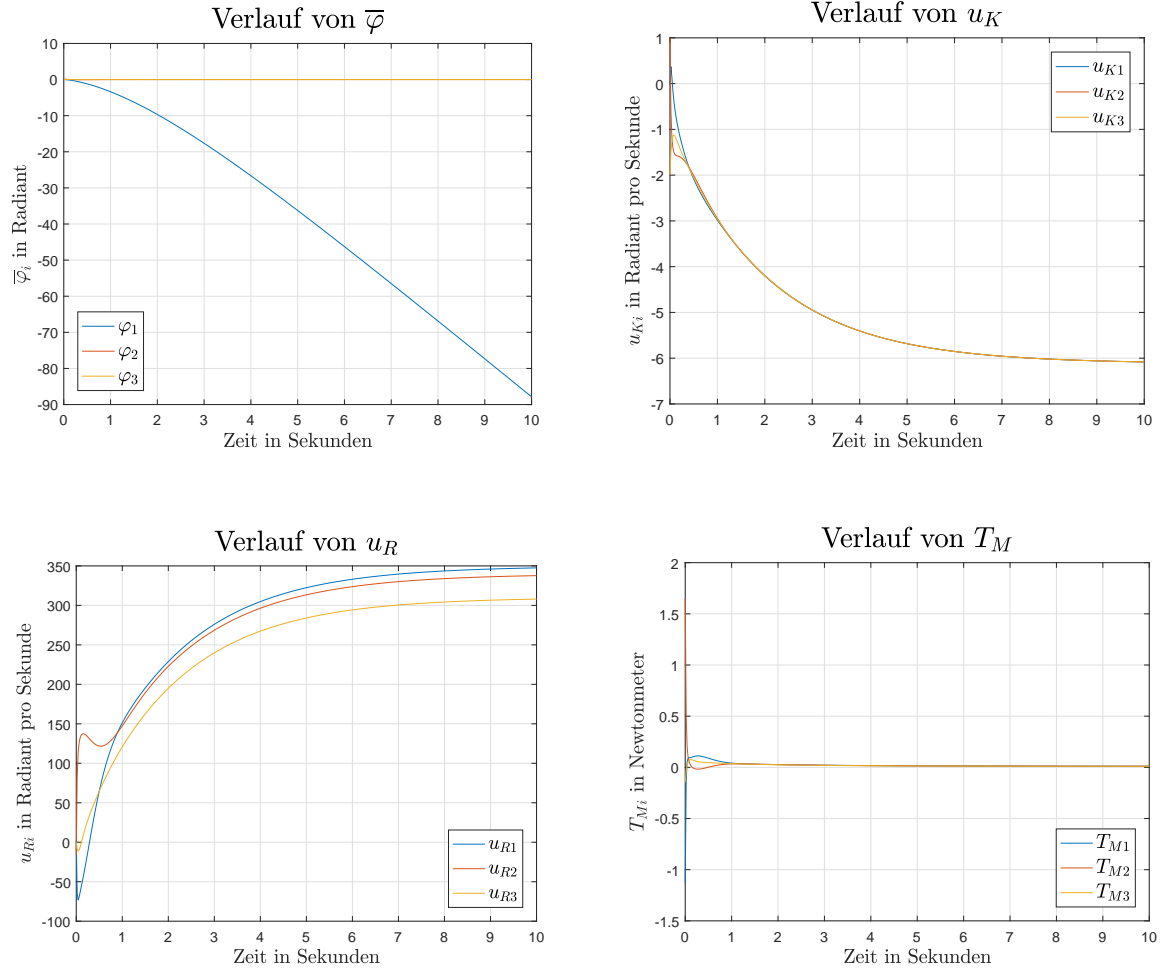
$$\mathbf{T}_K^{-1} = \begin{bmatrix} \mathbf{0}^{8 \times 1} & \mathbf{T}_K^{-1}(2:9, 2:9) \\ 1 & \mathbf{0}^{1 \times 8} \end{bmatrix} \quad (5.12)$$

bestätigt, welche φ_1 in $x_{\bar{B}}$ abbildet.

Mit Hilfe der Matrix \mathbf{T}_K^{-1} kann auch der nicht steuerbare Zustand $x_{\bar{S}}$ interpretiert werden. Die zugehörige Zeile

$$\mathbf{T}_K^{-1}(8, :) \approx \begin{bmatrix} 0 & 0 & 0 & 0,5 & 0,5 & 0,7 & 0,01 & 0,01 & 0,01 \end{bmatrix} \quad (5.13)$$

bildet den Zustandsvektor \mathbf{x} in die Variable $x_{\bar{S}}$ ab. Diese ist somit eine Linearkombination der Geschwindigkeiten \mathbf{u}_K und \mathbf{u}_R . Dieser Umstand erklärt auch das asymptotische Verhalten des Zustandsvektors in der vorherigen Simulation. Da $\mathbf{u}_K(0) = \mathbf{u}_R(0) = \mathbf{0}$ und somit $x_{\bar{S}}(0) = 0$ galt, kam die Grenzstabilität des nicht steuerbaren Zustandes nicht zum Tragen. Wird der Anfangszustand $\mathbf{x}(0)$ so angepasst, dass $x_{\bar{S}}(0) \neq \mathbf{0}$ gilt, ergibt sich das folgende Simulationsergebnis.



Die Simulation zeigt, dass die Winkel φ_2 und φ_3 nach wie vor asymptotisch stabil sind. Allerdings konvergieren die Winkelgeschwindigkeiten \mathbf{u}_K gegen den gleichen Endwert $u_{Ke} \neq 0$. Mittels der Winkelgeschwindigkeit

$${}^A\boldsymbol{\omega}^K = \begin{pmatrix} u_{K1} \\ u_{K2} \\ u_{K3} \end{pmatrix} \quad (5.14)$$

bedeutet der Endwert u_{Ke} , dass der Würfel mit einer konstanten Winkelgeschwindigkeit um seine Hauptdiagonale rotiert. Ebenso konvergieren die Geschwindigkeiten der Schwungmasse \mathbf{u}_R und die Stellgrößen gegen einen von Null verschiedenen Endwert. Dieser Umstand kann an Hand des Systems \mathfrak{D}_K erklärt werden. Der Unterraum \mathfrak{D}_M ist im geschlossenen Regelkreis asymptotisch stabil. Allerdings wirkt der nicht steuerbare Zustand $x_{\bar{S}}$ proportional zu dem Vektor $\mathbf{a}_{M/\bar{S}}$ auf \mathbf{x}_M ein.

$$\mathbf{x}_M(k+1) = (\mathbf{A}_M - \mathbf{B}_M \cdot \mathbf{K}_M) \cdot \mathbf{x}_M(k) + \mathbf{a}_{M/\bar{S}} \cdot \sigma(t) \cdot x_{\bar{S}}(0) \quad (5.15)$$

Mit Hilfe des Endwertsatzes kann das Konvergenzverhalten bestimmt werden.

$$\begin{aligned} z \cdot \mathbf{X}_M(z) &= (\mathbf{A}_M - \mathbf{B}_M \cdot \mathbf{K}_M) \cdot \mathbf{X}_M(z) + \mathbf{a}_{M/\bar{S}} \cdot X_{\bar{S}}(z) \\ \Leftrightarrow \mathbf{X}_M(z) &= [z \cdot \mathbf{I} - (\mathbf{A}_M - \mathbf{B}_M \cdot \mathbf{K}_M)]^{-1} \cdot \mathbf{a}_{M/\bar{S}} \cdot \underbrace{X_{\bar{S}}(z)}_{=\frac{z}{z-1} \cdot x_{\bar{S}}(t=0)} \end{aligned} \quad (5.16)$$

$$\lim_{k \rightarrow \infty} \mathbf{x}_M(k) = \lim_{z \rightarrow 1} (z - 1) \mathbf{X}_M(z) \equiv \mathbf{x}_{M,end} \quad (5.17)$$

Wird der Zustandsvektor $\mathbf{x}_{K,end}$ aus den Endwerten der Teilsysteme zusammengesetzt kann dieser in den ursprünglichen Zustandsraum transformiert werden.

$$\begin{aligned} \mathbf{x}_{end} &= \mathbf{T}_K \cdot \begin{bmatrix} \mathbf{x}_{M,end} \\ x_{\bar{S}} \\ x_{\bar{B}} \end{bmatrix} \\ &= \begin{bmatrix} \varphi_1 & 0 & 0 & 5,4x_{\bar{S}} & 5,4x_{\bar{S}} & 5,4x_{\bar{S}} & -306,9x_{\bar{S}} & -298,2x_{\bar{S}} & -272,1x_{\bar{S}} \end{bmatrix}^T \end{aligned} \quad (5.18)$$

Dieses Ergebnis besagt, dass das System des Würfels ein Ruhelagenkontinuum besitzt. Die Winkel $\bar{\varphi}_2$ und $\bar{\varphi}_3$ konvergieren stets gegen null, was der aufrechten Position des Würfels auf einer Ecke entspricht. Allerdings rotiert der Würfel dabei mit einer konstanten, von $x_{\bar{S}}$ abhängigen Geschwindigkeit um seine Raumdiagonale. Des weiteren rotieren die drei Schwungmassen mit einer konstanten Endgeschwindigkeit, welche ebenfalls von $x_{\bar{S}}$ abhängt. In diesem Zusammenhang kann auch die nicht Steuerbarkeit des Zustandes $x_{\bar{S}}$ erklärt werden. Angenommen der Würfel rotiert in einer Ruhelage des Kontinuums mit einer konstanten Winkelgeschwindigkeit

$${}^A\boldsymbol{\omega}^K = \begin{pmatrix} u_{end} \\ u_{end} \\ u_{end} \end{pmatrix}^K. \quad (5.19)$$

Nun ist es zwar möglich diese Rotation durch ein parallel gerichtetes Motormoment

$$\mathbf{T}_M = \begin{pmatrix} T \\ T \\ T \end{pmatrix}^K \quad (5.20)$$

zu beeinflussen, allerdings werden dadurch zwangsläufig die Winkelgeschwindigkeiten \mathbf{u}_R der Schwungmassen beeinflusst. Somit ist es nicht möglich die Rotation des Würfels um seine Raumdiagonale unabhängig von der Bewegung der Schwungmassen zu beeinflussen, was wiederum gegen die Forderung der vollständigen Steuerbarkeit eines Systems verstößt. Dieses Ergebnis ist kritisch zu betrachten, da hier lediglich ein linearisiertes System untersucht wurde. Um eine finale Aussage über die Steuerbarkeit des Systems zu treffen, müssen dessen nicht lineare Bewegungsgleichung auf geprüft werden. Da es sich bei dem Würfel um ein eingangslineares System handelt ist dessen Prüfung auf Steuerbarkeit analytisch möglich [Adamy, S.155ff]. Diese Untersuchung wird in dieser Arbeit allerdings nicht durchgeführt, da an der realen Regelstrecke lediglich Anfangszustände mit

$$\bar{\varphi} \neq \mathbf{0} \quad \mathbf{u}_K \approx \mathbf{0} \quad \mathbf{u}_R \approx \mathbf{0} \quad (5.21)$$

untersucht werden und der nicht steuerbare Zustand $x_{\bar{S}}$ somit nicht besonders ins Gewicht fällt.

5.2 Untersuchung der Stellgrößenbeschränkung

Bisher wurde das System unter der Annahme betrachtet, dass die Stellgrößen \mathbf{u} unbegrenzt sind. Allerdings können die Motoren nur ein maximales Drehmoment $u_{max} = 0,11Nm$ aufbringen. Formal kann diese Begrenzung durch die Einführung der begrenzten Stellgröße

$$u_{i,sat} = \begin{cases} u_{max} & u_i > u_{max} \\ u & -u_{max} < u_i < u_{max} \\ -u_{max} & u_i < -u_{max} \end{cases} \quad (5.22)$$

ausgedrückt werden. Wird zunächst der Fall des auf einer Kante balancierenden Würfels betrachtet, so besitzt dieser lediglich eine Stellgröße. Tritt nun der Fall der Sättigung und die Stellgröße wird auf ihr Maximum beschränkt, so wird lediglich die Norm des Drehmomentvektors

$$\mathbf{T}_M = \begin{pmatrix} 0 \\ 0 \\ u_{i,sat} \end{pmatrix}^K \quad (5.23)$$

beeinflusst. Die Richtung des Drehmoments bleibt hingegen erhalten. Dies hat hinsichtlich der Stabilität die Folge, dass lediglich ein begrenztes Gebiet von Anfangszuständen in die Ruhelage überführt werden kann. Da in den restlichen Anfangszuständen z.B. das Gravitationsmoment so groß ist, dass es von dem begrenzten Motormoment nicht überwunden werden kann.

Der auf der Ecke balancierende Würfel besitzt allerdings mehrere Eingangsgrößen. Dies hat zur Folge, dass die Sättigung einer oder mehrerer Stellgrößen dazu führt, dass der Drehmomentvektor

$$\mathbf{T}_M = \begin{pmatrix} u_{1,sat} \\ u_{2,sat} \\ u_{3,sat} \end{pmatrix}^K \quad (5.24)$$

nicht nur in seiner Norm sondern auch der Richtung verändert wird. Die Richtungsänderung des Stellvektors durch Beschränkungen wird als Direktionalitätsproblem bezeichnet und kann unerwünschte Auswirkungen auf die Stabilität des Systems mit sich bringen [Ortseifen S.33]. Diese Problematik zeigt sich wenn die Simulation um eine Stellgrößenbeschränkung erweitert wird. Nun wird ist das Stabil bereits bei einem Anfangswinkel

$$\bar{\varphi}(0) \approx \begin{bmatrix} 0 \\ 2^\circ \\ 2^\circ \end{bmatrix} \quad (5.25)$$

instabil. Um das Einzugsgebiet der Ruhelage zu erweitern wird die Wichtungsmatrix \mathbf{Q} modifiziert. Da diese den Verlauf des Zustandsvektors gewichtet führt eine Reduktion ihrer Elemente zu niedrigeren Reglerwerten. Dadurch wird das Gebiet der Zustandsvektoren, welche eine Beschränkung der Stellgrößen verursacht, erweitert. Die Gewichtsmatrix \mathbf{Q} wird iterativ an Hand der Simulation variiert bis ein zufriedenstellendes Ergebnis erreicht ist. Hierbei

können Anfangswinkel von bis zu

$$\overline{\varphi}(0) \approx \begin{bmatrix} 0 & 5^\circ & 5^\circ \end{bmatrix} \quad (5.26)$$

erreicht werden, bei denen sich das System stabil verhält. Jedoch ist es mit dieser Vorgehensweise nicht möglich das Einzugsgebiet der Zustände, welche in die Ruhelage überführt werden, in einem geschlossenen Ausdruck zu formulieren. Die Simulation ermöglicht lediglich eine Abschätzung des Stabilitätsverhaltens. Somit kann die Stabilität erst empirisch an der reellen Regelstrecke bewiesen werden. Um diese Problematik zu beseitigen müssen alternative Ansätze verfolgt werden. Mögliche Lösungen stellen der Entwurf eines Sättigungsreglers [Adamy, S.264 ff] oder die Implementierung eines modellbasierten Anti-Windups [Ortseifen] dar. Diese Methoden kommen im Rahmen dieser Arbeit jedoch nicht zum Einsatz.

Kapitel 6

Erfassung der Zustandsgrößen

6.1 Winkelschätzung

Ein relevantes Problem stellt die Bestimmung der Winkel φ dar welche nur indirekt über die Beschleunigungssensoren ermittelt werden können. Die Messwerte \mathbf{s} der Beschleunigungssensoren setzen sich aus der resultierenden Beschleunigung ${}^A\mathbf{a}^{S_i}$ und dem überlagerten Erdbeschleunigungsvektor \mathbf{g} zusammen. Zunächst wird der Fall betrachtet, dass die Messachsen der Sensoren mit dem körperfesten Bezugssystem K zusammenfallen.

$$\mathbf{s}_i = {}^A\mathbf{a}^{S_i} + \mathbf{g} \quad (6.1)$$

Unter der Annahme, dass der Würfel nicht bewegt wird verschwindet der Einfluss der Beschleunigung ${}^A\mathbf{a}^{S_i}$.

$$\mathbf{s}_i = \mathbf{g} = \|\mathbf{g}\| \cdot \begin{pmatrix} -c_{\varphi_2} \cdot c_{\varphi_3} \\ c_{\varphi_2} \cdot s_{\varphi_3} \\ -s_{\varphi_2} \end{pmatrix} \quad (6.2)$$

Nun können die Winkel φ_2 und φ_3 aus den Komponenten des Messvektor \mathbf{s} ermittelt werden.

$$\varphi_2 = -\arcsin\left(\frac{\langle \mathbf{s}_i, \mathbf{k}_3 \rangle}{\|\mathbf{g}\|}\right) \quad \varphi_3 = -\arctan\left(\frac{\langle \mathbf{s}_i, \mathbf{k}_2 \rangle}{\langle \mathbf{s}_i, \mathbf{k}_3 \rangle}\right) \quad (6.3)$$

Der Winkel φ_1 kann nicht aus dem Erdbeschleunigungsvektor ermittelt werden. Jedoch schränkt dieser Umstand das Gesamtsystem nicht ein, da die Größe φ_1 keinen Einfluss auf die Systemdynamik hat. Um die Winkelschätzung auf den Fall des bewegten Würfels zu erweitern wird im nächsten Schritt die Beschleunigung ${}^A\mathbf{a}^{S_i}$ betrachtet. Da der Würfel eine rein rotatorische Bewegung durchführt genügt die Untersuchung der Winkelbeschleunigung und -geschwindigkeit [Kane S. 30].

$$\begin{aligned} {}^A\mathbf{a}^{S_i} &= {}^A\boldsymbol{\alpha}^K \times \mathbf{r}_{S_i} + {}^A\boldsymbol{\omega}^K \times ({}^A\boldsymbol{\omega}^K \times \mathbf{r}_{S_i}) \\ &= [{}^A\boldsymbol{\alpha}^K]_{\times} \cdot \mathbf{r}_{S_i} + [{}^A\boldsymbol{\omega}^K]_{\times} \cdot ([{}^A\boldsymbol{\omega}^K]_{\times} \cdot \mathbf{r}_{S_i}) \\ &= ([{}^A\boldsymbol{\alpha}^K]_{\times} + [{}^A\boldsymbol{\omega}^K]_{\times}^2) \cdot \mathbf{r}_{S_i} \end{aligned} \quad (6.4)$$

Wird nun die Summe der Beschleunigungswerte \mathbf{s}_i berechnet, welche mit dem frei wählbaren Faktor $b_i \in \mathbb{R}$ gewichtet werden, ergibt sich

$$\begin{aligned} \sum_{i=1}^6 b_i \cdot \mathbf{s}_i &= \sum_{i=1}^6 \left[b_i \cdot ([{}^A\boldsymbol{\alpha}^K]_{\times} + [{}^A\boldsymbol{\omega}^K]_{\times}^2) \cdot \mathbf{r}_{S_i} + b_i \cdot \mathbf{g} \right] \\ &= ([{}^A\boldsymbol{\alpha}^K]_{\times} + [{}^A\boldsymbol{\omega}^K]_{\times}^2) \cdot \sum_{i=1}^6 b_i \cdot \mathbf{r}_{S_i} + \mathbf{g} \cdot \sum_{i=1}^6 b_i. \end{aligned} \quad (6.5)$$

Werden die Faktoren b_i so gewählt, dass

$$\sum_{i=1}^6 b_i \cdot \mathbf{r}_{S_i} \quad | \quad \sum_{i=1}^6 b_i \neq 0 \quad (6.6)$$

gilt, folgt

$$\sum_{i=1}^6 b_i \cdot \mathbf{s}_i = \mathbf{g} \cdot \sum_{i=1}^6 \quad \leftrightarrow \quad \mathbf{g} = \frac{\sum_{i=1}^6 b_i \cdot \mathbf{s}_i}{\sum_{i=1}^6 b_i}. \quad (6.7)$$

Somit kann der Einfluss der resultierenden Beschleunigung ${}^A\mathbf{a}^{S_i}$ mittels der Faktoren b_i eliminiert werden. Werden n Sensoren verwendet, so muss für die Bestimmung der Faktoren b_i das Gleichungssystem

$$\sum_{i=1}^n b_i \cdot \mathbf{r}_{S_i} = 0 \quad (6.8)$$

gelöst werden, wobei die Nebenbedingung

$$\sum_{i=1}^n b_i \neq 0 \quad (6.9)$$

zu beachten ist. Aus dieser Vorgehensweise können Rückschlüsse auf den Entwurf des Würfels gezogen werden. Sind die Ortsvektoren \mathbf{r}_{S_i} linear abhängig genügen bereits zwei Sensoren um zwischen der resultierenden Beschleunigung ${}^A\mathbf{a}^{S_i}$ und der Erdbeschleunigung \mathbf{g} zu unterscheiden. Allerdings schränkt die Forderung nach linearer Abhängigkeiten die konstruktiven Möglichkeiten ein. Werden mehr als zwei Sensoren verwendet entfällt die Notwendigkeit der linearen Abhängigkeiten. Prinzipiell genügen drei Sensoren um die Einflüsse der Beschleunigung ${}^A\mathbf{a}^{S_i}$ zu eliminieren.

Die hier verwendeten Beschleunigungssensoren sind von zwei weiteren Einschränkungen betroffen. Zunächst ist die Empfindlichkeit der Messung in z-Richtung gegenüber den x- und y-Achsen geringer, weshalb lediglich die letzteren verwendet. Des weiteren stimmen die Messachsen der Sensoren nicht mit dem körperfesten Bezugssystem K überein. Um diese Umstände im Modell auszudrücken werden die drei Messachsen des Sensors als Bezugssystem S_i interpretiert. Unter der Annahme, dass die Messachsen und Vektoren \mathbf{k}_i paarweise orthogonal zueinander stehen kann die Projektionsmatrix ${}^{S_i}\mathbf{P}^K$ aus dem Aufbau bestimmt werden. Zusätzlich wird die dritte Spalte ${}^{S_i}\mathbf{P}^K$ durch den Nullvektor ersetzt um die Vernachlässigung der z-Messwerte darzustellen.

$$\mathbf{s}_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot {}^{S_1} \begin{pmatrix} s_{1x} \\ s_{1y} \\ s_{1z} \end{pmatrix} = {}^K \begin{pmatrix} \langle \mathbf{s}_1, \mathbf{k}_1 \rangle \\ \langle \mathbf{s}_1, \mathbf{k}_2 \rangle \\ 0 \end{pmatrix} \quad (6.10)$$

$$\mathbf{s}_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot {}^{S_2} \begin{pmatrix} s_{2x} \\ s_{2y} \\ s_{2z} \end{pmatrix} = {}^K \begin{pmatrix} \langle \mathbf{s}_2, \mathbf{k}_1 \rangle \\ \langle \mathbf{s}_2, \mathbf{k}_2 \rangle \\ 0 \end{pmatrix} \quad (6.11)$$

$$\mathbf{s}_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \cdot {}^{S_3} \begin{pmatrix} s_{3x} \\ s_{3y} \\ s_{3z} \end{pmatrix} = {}^K \begin{pmatrix} 0 \\ \langle \mathbf{s}_3, \mathbf{k}_2 \rangle \\ \langle \mathbf{s}_3, \mathbf{k}_3 \rangle \end{pmatrix} \quad (6.12)$$

$$\mathbf{s}_4 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \cdot \begin{matrix} S_4 \\ \begin{pmatrix} s_{4x} \\ s_{4y} \\ s_{4z} \end{pmatrix} \end{matrix} = \begin{matrix} K \\ \begin{pmatrix} 0 \\ \langle \mathbf{s}_4, \mathbf{k}_2 \rangle \\ \langle \mathbf{s}_4, \mathbf{k}_3 \rangle \end{pmatrix} \end{matrix} \quad (6.13)$$

$$\mathbf{s}_5 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{matrix} S_5 \\ \begin{pmatrix} s_{5x} \\ s_{5y} \\ s_{5z} \end{pmatrix} \end{matrix} = \begin{matrix} K \\ \begin{pmatrix} \langle \mathbf{s}_5, \mathbf{k}_1 \rangle \\ 0 \\ \langle \mathbf{s}_5, \mathbf{k}_3 \rangle \end{pmatrix} \end{matrix} \quad (6.14)$$

$$\mathbf{s}_6 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{matrix} S_6 \\ \begin{pmatrix} s_{6x} \\ s_{6y} \\ s_{6z} \end{pmatrix} \end{matrix} = \begin{matrix} K \\ \begin{pmatrix} \langle \mathbf{s}_6, \mathbf{k}_1 \rangle \\ 0 \\ \langle \mathbf{s}_6, \mathbf{k}_3 \rangle \end{pmatrix} \end{matrix} \quad (6.15)$$

Da die z-Messwerte nicht verwendet werden gibt jeder Sensor nur die Beschleunigung in Richtung zweier Vektoren \mathbf{k}_i wieder. Um dieses Problem zu beheben werden die Messwerte jeweils zweier Sensoren zu einem abstrakten Messvektor $\tilde{\mathbf{s}}_i$ zusammengefasst. Um hierbei die Auswirkung der Beschleunigungen ${}^A\mathbf{a}^{S_i}$ darzustellen wird die Definition

$$[{}^A\boldsymbol{\alpha}^K]_{\times} + [{}^A\boldsymbol{\omega}^K]_{\times}^2 \equiv \mathbf{M} = \begin{bmatrix} \mathbf{m}_1^T \\ \mathbf{m}_2^T \\ \mathbf{m}_3^T \end{bmatrix} \quad (6.16)$$

verwendet. Die Vektoren \mathbf{m}_i^T werden mit dem Ortsvektor des zugehörigen Sensors multipliziert.

$$\begin{aligned} \tilde{\mathbf{s}}_1 &\equiv \begin{bmatrix} s_{1y} \\ s_{1x} \\ s_{3x} \end{bmatrix} = \begin{bmatrix} \mathbf{m}_1^T \cdot \mathbf{r}_{S_1} \\ \mathbf{m}_2^T \cdot \mathbf{r}_{S_1} \\ \mathbf{m}_3^T \cdot \mathbf{r}_{S_3} \end{bmatrix} + \mathbf{g} & \quad \tilde{\mathbf{s}}_2 &\equiv \begin{bmatrix} s_{2y} \\ s_{2x} \\ s_{4x} \end{bmatrix} = \begin{bmatrix} \mathbf{m}_1^T \cdot \mathbf{r}_{S_2} \\ \mathbf{m}_2^T \cdot \mathbf{r}_{S_2} \\ \mathbf{m}_3^T \cdot \mathbf{r}_{S_4} \end{bmatrix} + \mathbf{g} \\ \tilde{\mathbf{s}}_3 &\equiv \begin{bmatrix} s_{5x} \\ s_{3y} \\ s_{5y} \end{bmatrix} = \begin{bmatrix} \mathbf{m}_1^T \cdot \mathbf{r}_{S_5} \\ \mathbf{m}_2^T \cdot \mathbf{r}_{S_3} \\ \mathbf{m}_3^T \cdot \mathbf{r}_{S_5} \end{bmatrix} + \mathbf{g} & \quad \tilde{\mathbf{s}}_4 &\equiv \begin{bmatrix} s_{6x} \\ s_{4y} \\ s_{6y} \end{bmatrix} = \begin{bmatrix} \mathbf{m}_1^T \cdot \mathbf{r}_{S_6} \\ \mathbf{m}_2^T \cdot \mathbf{r}_{S_4} \\ \mathbf{m}_3^T \cdot \mathbf{r}_{S_6} \end{bmatrix} + \mathbf{g} \end{aligned} \quad (6.17)$$

In dieser Darstellung werden die Vektoren $\tilde{\mathbf{s}}_i$ mit den Diagonalmatrizen

$$\mathbf{B}_i = \begin{bmatrix} b_{ix} & 0 & 0 \\ 0 & b_{iy} & 0 \\ 0 & 0 & b_{iz} \end{bmatrix} \quad (6.18)$$

multipliziert um die Einflüsse der Beschleunigungen zu eliminieren. Für die Summe der gewichteten Vektoren gilt

$$\sum_{i=1}^4 \mathbf{B}_i \cdot \tilde{\mathbf{s}}_i = \begin{bmatrix} \mathbf{m}_1^T \cdot \sum_{i=1}^4 b_{ix} \cdot \mathbf{r}_{\tilde{S}_{xi}} \\ \mathbf{m}_2^T \cdot \sum_{i=1}^4 b_{iy} \cdot \mathbf{r}_{\tilde{S}_{yi}} \\ \mathbf{m}_3^T \cdot \sum_{i=1}^4 b_{iz} \cdot \mathbf{r}_{\tilde{S}_{zi}} \end{bmatrix} + \sum_{i=1}^4 \mathbf{B}_i \cdot \mathbf{g} \quad (6.19)$$

Wenn nun die Matrizen \mathbf{B}_i so gewählt werden, dass einerseits

$$\sum_{i=1}^4 b_{ix} \cdot \mathbf{r}_{\tilde{S}_{xi}} = 0 \quad \sum_{i=1}^4 b_{iy} \cdot \mathbf{r}_{\tilde{S}_{yi}} = 0 \quad \sum_{i=1}^4 b_{iz} \cdot \mathbf{r}_{\tilde{S}_{zi}} = 0 \quad (6.20)$$

und andererseits

$$\det \left(\sum_{i=1}^4 \mathbf{B}_i \right) \neq 0 \quad (6.21)$$

gelten, ergibt sich für die Summe der Messvektoren $\tilde{\mathbf{s}}_i$

$$\sum_{i=1}^4 \mathbf{B}_i \cdot \tilde{\mathbf{s}}_i = \sum_{i=1}^4 \mathbf{B}_i \cdot \mathbf{g} \quad \Leftrightarrow \quad \mathbf{g} = \left(\sum_{i=1}^4 \mathbf{B}_i \right)^{-1} \cdot \sum_{i=1}^4 \mathbf{B}_i \cdot \tilde{\mathbf{s}}_i. \quad (6.22)$$

Somit können die Einflüsse der Beschleunigungen ${}^A\mathbf{a}^{S_i}$ auf die Messwerte auch bei den Messvektoren $\tilde{\mathbf{s}}_i$ eliminiert werden.

6.2 Justierung der Sensoren

Bisher wurden die Anzeigewerte der Sensoren als Beschleunigungswerte betrachtet. Allerdings geben die Sensormodule ihre Messwerte in digitaler 2K-Darstellung aus, weshalb eine Ausgleichsfunktion bestimmt werden muss um die Anzeigewerte in die entsprechende SI-Einheit zu überführen. Im ersten Schritt wird die Annahme getroffen, dass die Anzeigewerte proportional zu den Beschleunigungswerten und von einer systematischen Messabweichung überlagert sind. Um diese Fehler auszugleichen wird ein Polynom erster Ordnung verwendet, welches den digitalen Anzeigewert x als Argument entgegennimmt und den entsprechenden Beschleunigungswert y in m/s^2 zurückgibt.

$$y = p_1 \cdot x + p_2 \tag{6.23}$$

Mit diesem Ansatz muss für jede Messachse der Sensoren

Kapitel 7

Software

Wie bereits in den letzten Abschnitten erläutert wird der Regler als diskretes System implementiert. Deshalb spielt die verwendete Hardware und darauf ausgeführte Software eine zentrale Rolle. Dieser Umstand wird dadurch verstärkt, dass sämtliche Versuche, wie z.B. die Justierung der Sensoren, Systemidentifikation und Erprobung verschiedener Reglerkonzepte, in Form eines Programms durchgeführt werden. Aus diesem Grund widmet sich dieser Abschnitt dem Aufbau einer Software-Infrastruktur, welche die effiziente Entwicklung von mechatronischen Anwendungen ermöglicht.

7.1 Zielplattform, Sensorik und Aktorik

In diesem Projekt wird ein BeagleBone Black in Kombination mit einer Linux-Distribution verwendet um die digitale Regelung zu realisieren. Die Plattform basiert auf einem AM335x Sitara ARM Cortex-A8 Prozessor, der mit einer Taktrate von 1GHz betrieben wird. Des weiteren steht eine single precision NEON FPU, für die Berechnung von Gleitkommaoperationen, zur Verfügung. Diese Rechnerarchitektur reduziert die nötige Rechenezeit für gewöhnliche Filter- und Regelungsalgorithmen auf wenige Mikrosekunden. Somit kann die, durch die Rechnung resultierende Totzeit, des digitalen Systems bei dem Reglerentwurf vernachlässigt werden. Das Linux-Betriebssystem bringt weitere Vorteile für die Entwicklung des Gesamtsystems mit sich. Zunächst existiert eine Vielzahl von Werkzeugen für die Entwicklung von Embedded-Linux-Anwendungen. Dadurch wird der nötige Zeitaufwand für die Implementierung des Reglerprogramms reduziert werden. Des weiteren kann bei der Entwicklung auf Pakete und Bibliotheken der Linux-Gemeinde zurückgegriffen werden. Somit können auch komplexe Subsysteme, wie z.B. der hier verwendete WebSocketServer, in das Gesamtsystem eingebettet werden. Zuletzt kann das Dateisystem genutzt werden um Konfigurationen für Filter- und Regleralgorithmen auszutauschen, wodurch die Erprobung von verschiedenen Reglerkonzepten vereinfacht wird. Allerdings muss der Einfluss des Betriebssystems auf das Zeitverhalten der Regelung kritisch betrachtet werden. Einerseits kann die Abtastung an äquidistanten Stützstellen nicht mehr garantiert werden. Andererseits entsteht durch die Verwendung von Linux-Treibern Verzögerungen, die zu weiteren Totzeiten führen. Deshalb beschäftigt sich der nächste Abschnitt mit der verwendeten Peripherie und deren softwareseitige Auswertung.

7.1.1 Peripherie

Auf dem Würfelgehäuse sind insgesamt sechs MPU9250-Module[Datenblatt MPU] montiert. Die Module besitzen jeweils einen Beschleunigungs- und Drehratensensor, welche genutzt werden um einen Teil des Zustandvektors zu bestimmen. Die Kommunikation zwischen den Sensormodulen und dem BeagleBone Black erfolgt über einen SPI-Bus. Das das SPI-Modul des BeagleBone Black nur einen ChipSelectPin besitzt wird dieser über einen AnalogSwitch[DatenBlatt] mit den Sensoren verbunden. Drei digitale Ausgänge des BeagleBone Black

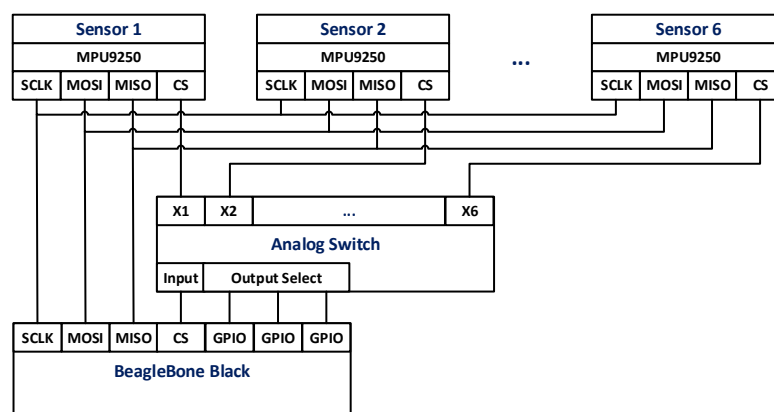


Abbildung 7.1: Blockschaltbild Sensorkommunikation, Quelle: eigene Darstellung

sind mit den Steuereingängen des Schalters verbunden um die ChipSelect-Leitung auf den Ausgang des gewünschten Sensors zu legen. Im Quellcode werden die Peripheriegeräte durch Klassen repräsentiert. Für die Steuerung der Digitalausgänge wird die Klasse *CGPIO* implementiert, welche als Konstruktorargument die Nummer des zu konfigurierenden Pins entgegennimmt. Des weiteren bietet sie Methoden zum Setzen bzw. Rücksetzen des Ausgangs. Hierfür wird die Schnittstelle des Treibers im Dateisystem verwendet. Zur Steuerung des Switches wird die Klasse *CSwitch* aus drei Instanzen des Typs *CGPIO* komponiert und implementiert die Methoden *selectXi()* um die Schalterausgänge auszuwählen. Analog wird für die Konfiguration und Auswertung der Sensoren die Klasse *CMPU9250* implementiert. Zur Interaktion mit dem SPI-Treiber wird die Posix-Funktion *ioctl()* verwendet, welche zu einer kürzeren Ausführungszeit der Treiberaufrufe führt und im Vergleich zu der Dateisystemschnittstelle detaillierte Konfigurationsoptionen bietet. Die Klasse nutzt die SPI-Schnittstelle um die Sensoren in der *init()*-Methode zu konfigurieren. Anschließend können über die Methode *fetchData()* die aktuellen Beschleunigungs- und Winkelgeschwindigkeitswerte ausgelesen werden.

Die letztendliche Anwenderschnittstelle bietet die Klasse *CSensorSystem*, welche aus einer Instanz des Typ *CSwitch* und *CMPU9250* komponiert wird. Im Konstruktor werden die sechs Sensoren initialisiert und anschließend über die Methode *fetchSensorData()* ausgelesen. Die Daten werden in der Struktur *SSensorData* gespeichert, welche Membervariablen für die Messwerte besitzt.

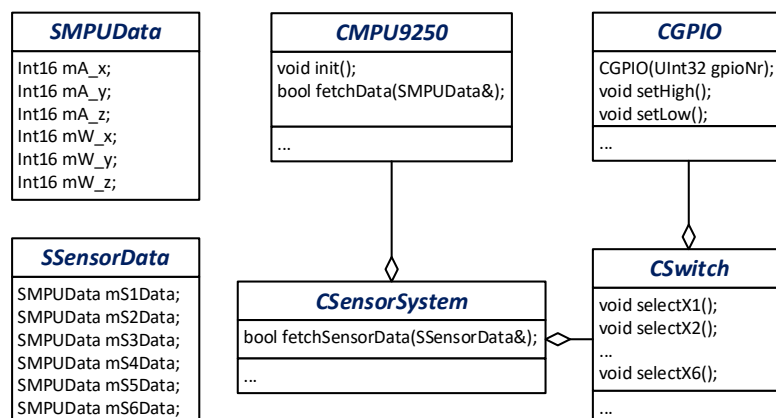


Abbildung 7.2: Klassendiagramm der Sensorschnittstelle, Quelle: eigene Darstellung

Die Stellgrößen der Regelung werden durch drei Motoren generiert, die jeweils über einen Treiberbaustein kontrolliert werden. Jeder Motortreiber ist mit zwei digitalen Ausgängen verbunden. Diese steuern die Freigabe und Drehrichtung des Motors. Die Vorgabe des Drehmoments erfolgt über ein PWM-Signal. Zusätzlich erfassen die Motortreiber die Drehzahl und geben diese in Form eines Analogsignals an das BeagleBone Black zurück. Die Klasse *CPWM* ermöglicht das Setzen eines Duty-Cycles, wobei die Treiberschnittstelle im Dateisystem verwendet wird. Eine Instanz dieser Klasse wird in *CMotor* genutzt um das gewünschte Drehmoment einzustellen. Des weiteren werden zwei Instanz von *CGPIO* genutzt um die Freigabe und Drehrichtung des Motors einzustellen. Für das Auslesen der ADC-Werte wird ebenfalls eine Klasse *CADC* angelegt. Da der Linux-Treiber für die Nutzung der AD-Wandler teilweise fehlerhaft ist wird ein alternative Vorgehensweise zur Nutzung der Peripherie genutzt.

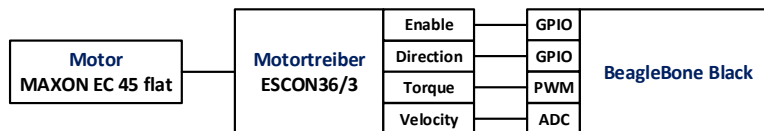


Abbildung 7.3: Blockschaltbild der Motoransteuerung, Quelle: eigene Darstellung

Hierbei wird mittels der Posix-Funktion *mmap()* der Adressbereich der ADC-Peripherie in den Userspace gelegt. Dadurch kann in der Anwendung direkt auf die ADC-Register zugegriffen werden. Durch diesen Ansatz ergeben sich zwei Vorteile. Einerseits werden dem Nutzer keine Einschränkungen durch die Treiberschnittstelle aufgezwungen. Andererseits wird die nötige Zeit der AD-Wandlung durch den direkten Zugriff auf die Register reduziert. Allerdings ist diese Vorgehensweise mit einem größeren Implementierungsaufwand verbunden und wird deshalb nur genutzt wenn die Einschränkungen des Treibers nicht annehmbar sind. die Klasse *CSensorSystem* wird um eine Instanz von *CADC* erweitert und umfasst somit die vollständige Sensorik des Systems. Um die Peripherie vollständig zu kapseln wird die Klasse *CHardware* aus einer Instanz von *CMotor* und *CSensorSystem* komponiert.

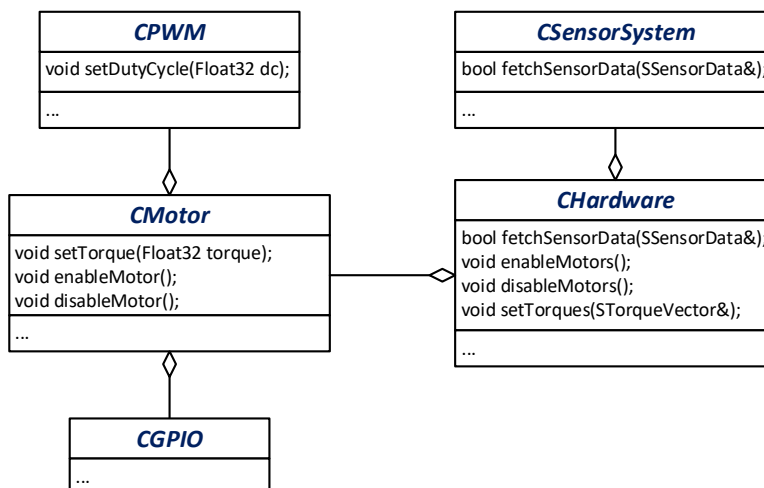


Abbildung 7.4: Klassendiagramm der Hardwareansteuerung, Quelle: eigene Darstellung

7.2 Implementierung des Signalflusses

Im nächsten Schritt muss der Signalfluss implementiert werden. Hierfür wird ein Ansatz der Template-Metaprogrammierung verwendet, um ein einheitliches Konzept für die Umsetzung von Singalverarbeitungsalgorithmen zu schaffen. Ebenso soll das Konzept Änderung im Singalfluss ermöglichen, ohne dabei größere Eingriffe im Quellcode vornehmen zu müssen. Als Beispiel wird das folgende Blockschaltbild verwendet. Zunächst wird eine, auf den Sensorwerten basierende, Zustandsschätzung durchgeführt. Dieser Vektor wird anschließend gefiltert und zur Berechnung des Reglers genutzt.

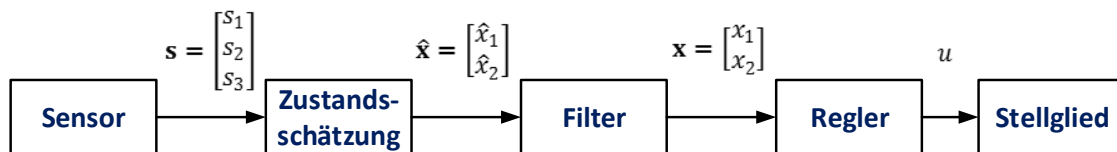


Abbildung 7.5: Blockschaltbild des Signalflusses, Quelle: eigene Darstellung

Für die Implementierung werden die Signale als Datenobjekte implementiert. Das heißt es werden Strukturen oder Klassen entworfen, welche die Daten enthalten und ggf. Methoden für den Zugriff oder Bearbeitung bieten. Für dieses Beispiel repräsentieren die Strukturen *SSensorData* und *SStateVector* den Sensor- bzw. Zustandsvektor. Für die Modellierung der Stellgröße genügt eine *float*-Variable.

```

1 struct SSensorData
2 {
3     Float32 mS1;
4     Float32 mS2;
5     Float32 mS3;
6 };
7 struct SStateVector
8 {
9     Float32 mX1;
10    Float32 mX2;
11 };
12 using UType = Float32;
  
```

Die Systeme im Signalfluss werden als Klassen implementiert, die als Aktionsobjekte bezeichnet werden. Die Klassen werden nach dem folgenden Schema entworfen, wobei als Beispiel ein Objekt zur Filter Zustandsschätzung aufgezeigt wird.

```

1 class CStateEstimate
2 {
3 public:
4     using InputType = SSensorData;
5     using OutputType = SStateVector;
6 public:
7     const OutputType& calcOutput(const InputType& input);
8     const OutputType& getValue() const;
9     ...
10 private:
11     OutputType mOutput;
12 };
  
```

Die Aktionsobjekte definieren zunächst ihren Ein- und Ausgangstyp. Die Berechnung des Systems erfolgt über die Methode *calcOutput()*. Nun müssen die Aktionsobjekte zu dem vor-

gegebenen Signalfluss zusammengefasst werden. Um diesen Schritt im Entwicklungsprozess zu erleichtern, soll ein Konzept implementiert werden, dem eine Typenliste der Aktionsobjekte übergeben wird und daraus das Blockschaltbild erzeugt. Für die Implementierung der Typenliste wird der Ansatz nach [ModernCppDes] verwendet. Zunächst wird die leere Struktur *CNullType* definiert, welche als Terminierungssymbol in der Typenliste fungiert. Die Liste wird mit Hilfe der Templatestruktur *TTypeList* implementiert, welche als Parameter einen Anfangs- und Endtypen entgegennimmt.

```

1 struct CNullType{};

3 template<class HeadType, class TailType>
4 struct TTypeList
5 {
6     using Head = HeadType;
7     using Tail = TailType;
8 };

```

Die obige Implementierung unterstützt lediglich Listen der Länge zwei. Deshalb werden Makros verwendet, die rekursive Instanzierungen des Templates nutzen um die Listen beliebiger Länge zu erhalten. Mittels dieser Makros kann dann auch die Typenliste der Aktionsobjekte erzeugt werden.

```

1 #define TYPELIST_1(T1)          TTypeList<T1, CNullType>
2 #define TYPELIST_2(T1, T2)     TTypeList<T1, TYPELIST_1(T2)>
3 #define TYPELIST_3(T1, T2, T3) TTypeList<T1, TYPELIST_2(T2, T3)>
4 ...
5 using ActionObjList = TYPELIST_3(CStateEstimate, CFilter, CController);

```

Um nun die Aktionstypen zu einem Signalfluss-Objekt zusammenzufügen, wird das Muster der linearen Typenhierarchie nach [ModernCppDesign] angewandt. Dessen Aufbau erinnert an eine verkettete Liste, wobei allerdings Typen durch Vererbung verknüpft werden. Die Templateklasse *TActionHolder* wird als Träger für die Aktionsobjekte entworfen.

```

1 template<class ActionObj, class Base>
2 class TActionHolder : public Base, public ActionObj
3 {
4 public:
5     void calcOutput(const ActionObj::InputType& input)
6     {
7         ActionObj::calcOutput(input);
8         Base::calcOutput(ActionObj::getValue());
9     }
10 };
11 template<class ActionObj>
12 class TActionHolder<ActionObj, CNullType> :
13     public CNullType, public ActionObj
14 {
15 public:
16     void calcOutput(const ActionObj::InputType& input)
17     {
18         ActionObj::calcOutput(input);
19     }
20 };

```

Im allgemeinen Fall wird dem Template der Typ eines Aktionsobjektes *ActionObj* und eine beliebige Elternklasse *Base* übergeben, die beide an *TActionHolder* vererben. In der *calcOutput()*-Methode wird zunächst die Berechnung des Aktionsobjektes durchgeführt und anschließend *calcOutput()* der zweiten Elternklasse aufgerufen.

Des weiteren besteht eine Templatespezialisierung für den Fall, dass ein Aktionstyp und *CNullType*, der das Ende der Typenliste signalisiert, übergeben werden. Nun wird in der *calcOutput()*-Methode lediglich das Aktionsobjekt berechnet, da das Ende der Typenliste und somit des Signalflusses erreicht ist.

Die zweite Templateklasse ist *TLinHierarchy*, mit dem folgenden Prototyp.

```

1  template<class TList,
2      template<AtomicType, class Base> class Unit,
3      class Root = CNullType>
4  class TLinHierarchy;
```

Der erste Templateparameter ist die Typenliste, welche die zu generierende Typenhierarchie vorgibt. Der zweite Parameter ist eine Templateklasse, die als Träger der Objekte aus der Typenliste agiert. In diesem Anwendungsfall wird die zuvor definierte Templateklasse *TActionHolder* verwendet. Der letzte Parameter ist der Terminierungstyp der Hierarchie, welcher als Standardargument *CNullType* übergeben wird. Analog zu *TActionHolder* werden durch Spezialisierungen zwei Fälle unterschieden. Zunächst sei der Fall betrachtet, dass der Parameter *TList* eine Typenliste aus zwei beliebigen Typen ist.

```

1  template<class T1, class T2, template<class, class> class Unit, class Root>
2  class TLinHierarchy<TTypeList<T1, T2>, Unit, Root>
3  : public Unit<T1, TLinHierarchy<T2, Unit, Root> >
4  {};
```

Diese Instanziierung wird solange genutzt bis das Ende der ursprünglichen Typenliste erreicht ist. Die instantiierte Klasse von *TLinHierarchy*, ist eine leere Klasse die von *Unit* erbt. *Unit* wird wiederum mit *TLinHierarchy* instantiiert, wobei lediglich der zweite Parameter *T2* übergeben wird. Dadurch wird die ursprüngliche Typenliste schrittweise abgearbeitet. Die zweite Templatespezialisierung wird genutzt um die Generation am Ende der Typenliste zu terminieren. Dies ist der Fall wenn *TLinHierarchy* mit einer Typenliste der Länge eins instantiiert wird.

```

1  template<class T, template<class, class> class Unit, class Root>
2  class TLinHierarchy<TYPELIST_1(T), Unit, Root>
3  : public Unit<T, Root>
4  {};
```

Für das hier aufgeführte Beispiel ergibt sich die folgende Vererbungshierarchie. Der Vorteil dieses Konzept, welches mit einem nicht zu vernachlässigenden Programmieraufwand verbunden ist, zeigt sich bei der letztendlichen Nutzung. Sind die Aktionsobjekte definiert können sie in wenigen Zeilen zu dem Signalfluss zusammengesetzt werden.

```

1  using ActionList = TYPELIST_3(CStateEstimate, CFilter, CController);
2  using SignalFlow = TLinHierarchy<ActionList, TActionHandler>;
3  SignalFlow mySF;
```

Sollen nun im Projektverlauf einzelne Elemente ausgetauscht oder erweitert werden muss lediglich die Typendefinition von *ActionList* geändert werden. Da die Hierarchie mittels Vererbung realisiert wird, können durch die Angabe des Namensraum die Methoden aller Elternklassen verwendet werden. Beispielsweise zeigt der folgende Ausschnitt die Abfrage aller berechneten Signale.

```

1  SStateVector x_estimate = mySF.CStateEstimate::getValue();
2  SStateVector x_filtered = mySF.CFilter::getValue();
3  UType        u          = mySF.CController::getValue();
```

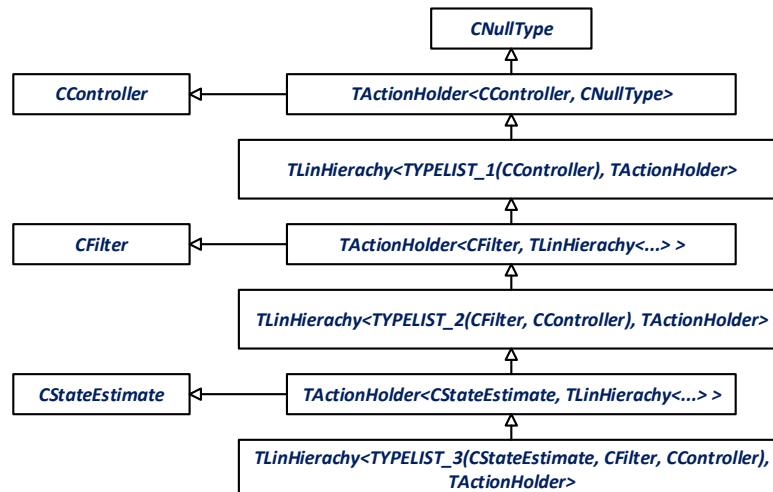


Abbildung 7.6: Klassendiagramm der generierten Hierarchie, Quelle: eigene Darstellung

Ebenso können zur Laufzeit konfigurierbare Elemente oder Verzweigungen implementiert werden. Angenommen zur Filterung sollen entweder ein Komplementär-Filter (*CCompFilter*) oder ein Tiefpass erster Ordnung (*CPT1*) genutzt werden. Dann kann eine Klasse *CFilterSystem* aus diesen beiden komponiert werden, die zusätzliche Methoden zur Filterauswahl bietet.

```

1  class CFilterSystem
2  {
3  public:
4      using InputType  = SStateVector;
5      using OutputType = SStateVector;
6  public:
7      const OutputType& calcOutput(const InputType& input)
8      {
9          mCompFilter.calcOutput(input);
10         mPT1Filter.calcOutput(input);
11
12         return this->getValue();
13     }
14     const OutputType& getValue()
15     {
16         switch(mActiveFilter)
17         {
18             case EFilter::CompFilter:
19                 return mCompFilter.getValue();
20             case EFilter::PT1Filter:
21                 return mPT1Filter.getValue();
22             default:
23                 return input;
24         }
25     }
26     void setFilter(EFilter filter)
27     {
28         mActiveFilter = filter;
29     }
30 private:
31     CCompFilter mCompFilter;
32     CPT1        mPT1Filter;
33     EFilter     mActiveFilter;
34 };

```

Analog können auch komplexere Verzweigungen realisiert werden, wobei *TLinHierachy* zur Generation von Teilzweigen des Blockschaltbildes verwendet werden kann.

7.3 Komponentenarchitektur für mechatronische Anwendungen

In dem letzten Abschnitt wurden die elementaren Funktionen des Regelkreises, wie die Peripherieinteraktion und der Signalfluss, implementiert. Um eine effiziente Versuchsdurchführung zu ermöglichen müssen allerdings weitere Funktionalitäten bereitstehen. Einerseits muss einzelne Elemente des Regelkreises während der Ausführung konfigurierbar sein. Beispielsweise soll zwischen unterschiedlichen Regler umgeschaltet werden und einzelne Parameter geändert werden können. Des weiteren müssen die gemessenen und berechneten Werte des Signalflusses an einen Entwicklungsrechner übertragen werden. Dort werden die Daten visualisiert und für eine spätere Analyse gespeichert. Folglich muss ein Kommunikationskonzept implementiert werden um Daten zwischen der Ziel- und Entwicklungsplattform auszutauschen. Um die Berechnung des Regelkreises und die Kommunikationsaufgaben voneinander zu trennen wird eine Komponentenarchitektur eingeführt. Die erste Komponente, welche als Regelungskomponente bezeichnet wird, führt die Berechnung des Signalflusses und die Interaktion mit den Peripheriegeräten durch. Des weiteren übernimmt sie die logische Steuerung der Versuchsabläufe. Die Kommunikationskomponente ist für die Verbindung mit dem Entwicklungsrechner verantwortlich und ermöglicht den Datenaustausch zwischen den beiden Plattformen. Hierfür wird ein WebSocketServer verwendet (siehe Abschnitt...). Durch die Verwendung der

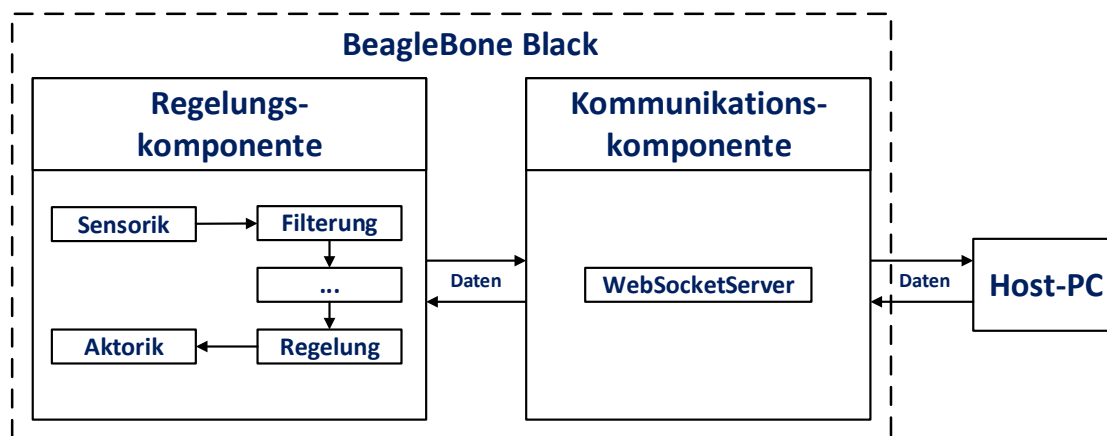


Abbildung 7.7: Blockschaltbild Gesamtsystem, Quelle: eigene Darstellung

Komponentenarchitektur entstehen einige Vorteile. Zunächst können die beiden Komponenten in unterschiedlichen Threads ausgeführt werden. Dadurch wird die Bearbeitung der beiden Aufgabenbereiche zum Großteil entkoppelt. Der Datenaustausch zwischen den Komponenten reduziert sich auf eine kontrollierte Schnittstelle, wodurch die Fehleranfälligkeit des Gesamtsystems minimiert wird. Des weiteren können die Aufgaben priorisiert werden, da der Scheduler eine preemptive Round-Robin-Strategie verfolgt. Somit kann die Ausführung der Regelungskomponente, welche für die Bearbeitung der zeit- und sicherheitsrelevanten Aufgaben zuständig ist, gegenüber der Kommunikationseinheit priorisiert werden. Hieraus resultiert, dass das Zeitverhalten der Regelung als deterministisch angenommen werden kann.

Für die Implementierung wird das Interface *IRunnable* definiert, welches virtuelle Me-

thoden zur Initialisierung und Ausführung der Komponenten vorschreibt. Diese Schnittstelle wird von der abstrakten Klasse *AComponentBase* geerbt, welche Membervariablen zum Datenaustausch besitzt. Die letztendlichen Komponenten werden in Form der beiden Klassen *CControlComp* und *CCommComp* realisiert. Die Erzeugung der Threads erfolgt mit Hilfe der Klasse *CThread*, deren Instanz als Trägerobjekte der Threads agieren.

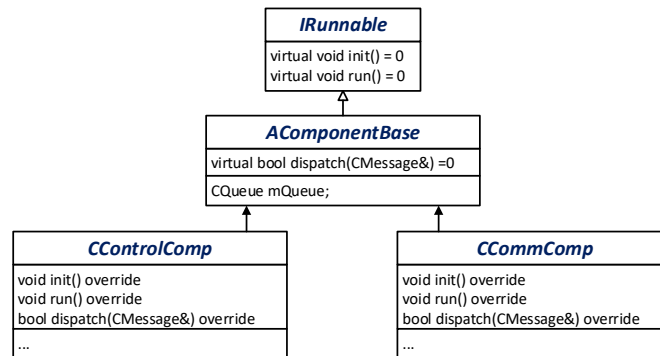


Abbildung 7.8: Klassendiagramm der Komponenten, Quelle: eigene Darstellung

Im nächsten Schritt muss ein Weg zum Datenaustausch zwischen den Komponenten etabliert werden. Einerseits müssen Messdaten und Signale aus dem Regelkreis von der Regelungs- und die Kommunikationseinheit gesendet werden, welche diese anschließend an den Host-PC weiterleitet. Andererseits werden die Steuerbefehle des Host-PCs von der Kommunikationskomponente empfangen und an die Regelungskomponente gereicht. Da in diesem Anwendungsfall lediglich Datenpakete versendet werden, die kleiner als 32 Byte sind, werden Nachrichten für deren Austausch verwendet. Die Nachrichten werden in Form der Klasse *CMessage* implementiert, welche aus einem Datenfeld, einem Ereignis und einem Zeitstempel komponiert werden. Das Ereignis wird als Enumeration realisiert und gibt den Inhalt der Daten bzw. den jeweiligen Befehl wieder. Der Zeitstempel gibt den Abtastzeitpunkt der Signale aus dem Regelkreis wieder.

```

1  enum class EEvent
2  {
3      DEFAULT_IGNORE = 0,
4      ASDF = 1
5  }
6  class CMessage
7  {
8  public:
9      EEvent getEvent() const;
10     UInt8* getDataPtr();
11     ...
12  public:

14  private:
15     static constexpr UInt32 sDataSize = 32U;

17     EEvent mEvent;
18     Float32 mTime;
19     UInt8 mData[sDataSize];
20 };
  
```

Des weiteren besitzt die Klasse Methoden und Konstruktoren um die Datenfelder entsprechend zu füllen und auszulesen. Um die Nachrichten zu empfangen besitzen die Komponenten

Eingangspuffer die als Queues implementiert werden. Die Erzeugung der Nachrichten wird von einem Proxy übernommen, der Methoden für die unterschiedlichen Ereignisse bereitstellt. Des weiteren kennt der Proxy die Queues der Komponenten und legt neue Nachrichten, in Abhängigkeit von dem jeweiligen Event, in die Eingangspuffer des zugehörigen Empfängers.

```
1 class CProxy
2 {
3     public:
4         bool transmitStateVector(const StateVector& x);
5         ...
6 };
```

Mit Hilfe der Nachrichtenkommunikation kann auch das Ablaufschema der Komponenten verallgemeinert werden. Beim Starten der Threads werden die Komponenten zunächst über den Aufruf ihrer *init()*-Methoden initialisiert, anschließend wird die *run()*-Methode ausgeführt, welche in diesem Fall eine Endlosschleife darstellt. In einem Durchlauf wird geprüft ob neue Nachrichten vorhanden sind, ist dies der Fall wird die Nachricht über die *dispatch()*-Methode verarbeitet. Andernfalls legt sich der Thread schlafen bis neue Nachrichten zur Verfügung stehen. Hieraus ergeben sich zwei Vorteile. Einerseits werden die Komponenten nach einem einheitlichen Konzept entworfen, lediglich die Implementierung der *init()*- und *dispatch()*-Methode unterscheiden sich, andererseits wird durch die Synchronisation der Komponenten deren Laufzeit reduziert.

7.4 Entwurf der Regelungskomponente

Die Aufgabe der Regelungskomponente besteht darin den Kontroll- und Signalfluss der verschiedenen Versuche zu steuern. Diese Umsetzung erfolgt mit Hilfe eines Zustandsautomats, welcher die Trennung der Kontrolllogik und der auszuführenden Aktionen ermöglicht. Des weiteren kann die Applikation bei diesem Ansatz problemlos durch weitere Versuche und Anwendungsfälle erweitert werden. Prinzipiell lässt sich der logische Ablauf der Komponente mit einem einfachen Zustandsdiagramm modellieren. Auf der obersten Ebene existiert ein *Standby*-Zustand, der die Inaktivität der Komponente widerspiegelt. Des weiteren enthält diese Ebene Zustände für die verschiedenen Versuche. Diese werden betreten, wenn die Komponente ein Event mit dem entsprechenden Befehl zur Ausführung des Versuchs erhält.

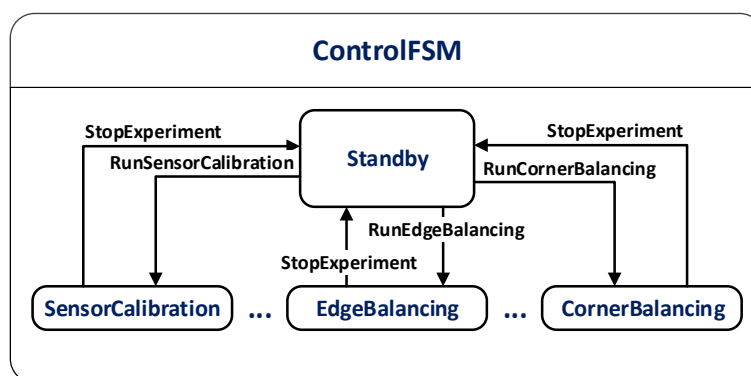


Abbildung 7.9: Zustandsdiagramm der Kontrol-Komponente, Quelle: eigene Darstellung

Ein solches Zustandsdiagramm kann z.B. mit einer objektorientierten Adaption der Methode von Samek [Wie Kapitel14] implementiert werden. Hierbei werden Zustände als Methoden der FSM realisiert, wobei für Oberzustände eigene Klassen entworfen werden, die wiederum Methoden für die jeweiligen Unterzustände besitzen. Die Referenzierung des aktuellen Zustandes erfolgt über einen Funktionszeiger, der auf die Methode des entsprechenden Zustandes gerichtet ist.

Als Basis der Typenhierarchie dient die abstrakte Klasse *AState*, welche den Zustandstypen als Methodenzeiger definiert, eine Standardimplementierung der *dispatch()*-Methode vorgibt und statische Membervariablen deklariert um die Verarbeitung interner Events zu ermöglichen.

```

1  class AState
2  {
3  protected:
4      using StatePtr = bool (AState::*) (CMessage&);
5  public:
6      virtual bool onInitial(CMessage& msg) = 0;
7      virtual bool dispatch(CMessage& msg);
8      ...
9  protected:
10     StatePtr mStatePtr;
11     static constexpr StatePtr sInitial =
12         static_cast<StatePtr>(AState::onInitial);
13     static CMessage sInternalQueue;
14     static UInt32   sQueueSize;
15 };
  
```

Die Methode *onInitial()* wird verwendet um die FSM und Oberzustände über ein *Init*-Event zu initialisieren. Die *dispatch()*-Methode beschränkt sich auf den Aufruf des aktuellen Zustands.

```

1  bool AState::dispatch(CMessage& msg)
2  {
3      return *(this->mStatePtr) (msg);
4  }

```

Die auszuführenden Aktionen, wie z.B. die Berechnung des Regelkreises, werden in der Klasse *CActionHandler* gekapselt. Dadurch erfolgt eine klare Trennung des Kontroll- und Signalflusses. Prinzipiell besitzt *CActionHandler* Methoden, die jeweils beim Betreten und Verlassen der Zustände aufgerufen wird. Zusätzlich kann er um nötige Hilfsmethoden erweitert werden.

```

1  class CActionHandler
2  {
3  public:
4      void enterStandby();
5      void exitStandby();
6      void enterSensorCalibration();
7      void exitSensorCalibration();
8      ...
9      void sampleSensorCalibration();
10     void sampleEdgeBalancing();
11     ...
12 private:
13     CThread      mTimerThread;
14     CTimerTask   mTimerTask;

16     CHardware      mHardware;
17     EdgeBalancingSF mEdgeBalancingSF;
18     ConrnerBalacingSF mCornerBalacingSF;
19 };

```

Für die Zeitgebung wird einer separater Timer-Task *CTimerTask* verwendet, der von *CActionHandler* verwaltet wird. Die Timerklasse kann mittels der Methoden *pause()* und *resume()* pausiert bzw. gestartet werden. Während der Ausführung schläft der Timer für eine konfigurierbare Abtastzeit und erzeugt anschließend über den Proxy ein *TIMERTICK*-Event, welches an die Regelungskomponente weitergeleitet wird.

```

1  class CTimerTask : public IRunnable
2  {
3  public:
4      void run() override
5      {
6          while(true)
7          {
8              mRunningSem.take(true);
9              mRunningSem.give();
10             usleep(mPeriod);
11             mProxyPtr->timerTick();
12         }
13     }
14     bool pause(bool waitForever){return mRunningSem.take(waitForever);}
15     bool resume(){mRunningSem.give();}
16     void setPeriod(Int32 period){mPeriod = period;};
17     ...
18 private:
19     CBinarySemaphore mRunningSem;
20     Int32            mPeriod;

```

```

21  CProxy*          mProxyPtr;
22  };

```

Der Ansatz nach Samek bringt bei diesen Anwendungsfall einen Nachteil mit sich. Für die meisten Versuche genügt eine simple Kontrolllogik, weshalb diese meisten als einfacher Unterzustand realisiert werden. Das hat eine flache Zustandshierarchie zur Folge, die der Anzahl von Versuchen entsprechend breit ist. In der Implementierung resultiert hieraus eine umfangreiche Klasse *CFSM*, da diese für jeden Unterzustand um eine Methode erweitert wird. Ebenso nimmt der Umfang der Klasse *CActionHandler* mit der Anzahl der Versuche kontinuierlich zu. Dieses Problem kann zwar durch die Aufteilung in mehrere Actionhandler vermieden werden, allerdings wird dadurch die Komplexität von *CFSM* weiter erhöht. Diese Problematik wird dadurch verschärft, dass es sich bei der Zustandsmaschine um den Kern der Anwendung und somit kritischsten Abschnitt der Anwendung handelt. Die zu Grunde liegende Komponentenarchitektur wird zu Projektbeginn erstellt und danach kaum manipuliert. Die dagegen wird während des Projektverlauf ständig manipuliert, weshalb eine unübersichtliche Implementierung besonders negativ auffällt. Deshalb wird im nächsten Schritte eine alternative Vorgehensweise vorgestellt, die sich die spezielle Struktur des Zustandsdiagrammes zu Nutze macht um eine übersichtliche Implementierung zu schaffen.

Zunächst sei angemerkt, dass es nicht möglich ist direkt zwischen zwei Versuchszuständen zu wechseln. Ein Versuchszustand kann nur aus dem Zustand *Standby* betreten werden. Des weiteren wird nach dem Verlassen eines Versuchszustandes immer in *Standby* gewechselt. Dadurch kann die Kontrolllogik der Zustandsmaschine verallgemeinert werden. Ist der momentane Unterzustand nicht *Standby* und es trifft ein *StopExperiment*-Event ein, so wird der Zustand verlassen und in *Standby* gewechselt. Befindet sich die FSM in *Standby* wird bei Eintreffen eines Events geprüft ob ein Zustandswechsel erfolgen muss. Diese Prüfung kann an die Zustände abgegeben werden. Somit stellt in diesem Fall die Zustandsmaschine lediglich eine Anfrage an alle Versuchszustände ob diese betreten werden möchten.

Die Implementierung der Zustandsmaschine setzt sich folglich aus einem *Standby*-Zustand und einer Liste von Versuchszuständen zusammen. Um eine übersichtliche Codestruktur zu erhalten werden diese als Klassen entworfen, die von der abstrakten Basisklasse *AState* erben.

```

1  class AState
2  {
3  public:
4      virtual bool dispatch(CMessage&) = 0;
5      virtual bool tryEntry(CMessage&, AState*&) = 0;
6      virtual void onEntry() = 0;
7      virtual void onExit() = 0;
8  private:
9      static CMessage sInternalQueue;
10     static UInt32 sQueueSize;
11 };

```

Die Methode *dispatch()* dient zur Verteilung von eintreffenden Nachrichten. Mit Hilfe von *tryEntry()* kann die Zustandsmaschine prüfen ob der Zustand, in Abhängigkeit des Events, betreten werden möchte. Der folgende Ausschnitt zeigt eine mögliche Implementierung für den Zustand *SensorCalibration*.

```

1  bool CSensorCalibration::tryEntry(CMessage& msg, AState*& statePtr)
2  {
3      EEvent event = msg.getEvent();

```

```

4   if(EEvent::RUN_SENSORCALIBRATION == event)
5   {
6       statePtr = this;
7       return true;
8   }
9   return false;
10 }
```

Das zweite Argument ist eine Zeigerreferenz auf den Zustandsreiger der FSM. Falls ein Zustand betreten werden soll überschreibt er die Referenz mit seinem `this`-Zeiger und gibt `true` zurück um den Konsum des Events zu signalisieren. Die Methoden `onEntry()` und `onExit()` werden zum Betreten bzw. Verlassen des Zustandes verwendet. Um auch die auszuführenden Aktionen zu trennen, wird für jede Zustandsklasse ein Actionhandler implementiert. Diese erben von der Klasse `CActionBase`, die gemeinsame Ressourcen als statische Membervariablen deklariert. Ein Beispiel wären hierfür Instanzen der Klassen `CHardware` oder `CTimerTask`.

Um die Zustandsklassen zu einer Liste zusammenzufassen wird wieder eine lineare Templatehierarchie verwendet. Hierfür muss zunächst ein Trägerobjekt `TStateHolder` entworfen werden.

```

1  template<class State, class Base>
2  class TStateHolder : public Base
3  {
4      bool tryEntry(CMessage& msg, AState*& statePtr)
5      {
6          bool consumed = mState.tryEntry(msg, statePtr);
7          if(consumed == false)
8          {
9              return Base::tryEntry(msg, statePtr);
10         }
11         return consumed;
12     }
13 private:
14     State mState;
15 };
16 template<class State>
17 class TStateHolder<State, CNullType> : public CNullType
18 {
19 public:
20     bool tryEntry(CMessage& msg, AState*& statePtr)
21     {
22         return mState.tryEntry(msg, statePtr);
23     }
24 private:
25     State mState;
26 };
```

Analog zu `TActionHolder` wird mittels einer Templatespezialisierung unterschieden ob das Ende der Typenliste erreicht ist. Ist dies nicht der Fall wird zunächst geprüft ob der getragene Zustand betreten werden soll. Trifft dies nicht zu wird die `tryEntry()`-Methode des nächsten Elements in der Hierarchie aufgerufen. Ein Unterschied zu `TActionHolder` ist, dass eine Komposition aus dem Zustandsobjekt verwendet wird. Dadurch werden die Methoden des Zustandes geschützt. Die FSM kann lediglich auf den, über den Zustandszeiger referenzierten, Zustand zugreifen. Die Implementierung der Zustandsmaschine basiert ebenfalls auf einem Template, welchem die Typenliste der Versuchszustände übergeben wird. Des weiteren erbt die Templateklasse von `AState` um zugriff auf die interne Queue zu erhalten.

```

1  template<class StateList>
```

```

2  class TFSM : public AState
3  {
4  public:
5      bool dispatch(CMessage& msg) override;
6      bool tryEntry(CMessage& msg, AState*& statePtr) override;
7      void onEntry() override;
8      void onExit() override;
9      bool onStandby(CMessage& msg);
10     void handleUnconsumedEvent(CMessage& msg);
11 private:
12     AState*          mStatePtr;
13     TLinHierach<StateList> mStateList;
14     CAction          mAction;
15 };

```

Neben dem Interface von *AState* besitzt die Klasse zwei weitere Methoden. Wobei die erste den Zustand *Standby* repräsentiert. Die zweite Methode wird genutzt um nicht konsumierte Events, was für gewöhnlich einem Fehlverhalten der FSM entspricht, abfängt. Zunächst wird die *dispatch()*-Methode betrachtet. Hier wird zunächst der aktuelle Unterzustand aufgerufen. Falls das Ereignis nicht konsumiert und es sich um *StopExperiment* handelt, wird der aktuelle Unterzustand verlassen und in *Standby* gewechselt. Zuletzt wird die interne Queue abgearbeitet.

```

1  template<class StateList>
2  bool TFSM<StateList>::dispatch(CMessage& msg)
3  {
4      bool consumed = false;
5      if(mStatePtr == nullptr)
6      {
7          consumed = this->onStandby(msg);
8      }
9      else
10     {
11         consumed = mStatePtr->dispatch(msg);
12     }

14     if(consumed == false)
15     {
16         EEvent event = msg.getEvent();
17         if(EEvent::StopExperiment == event)
18         {
19             mStatePtr->onExit();
20             mStatePtr = nullptr;
21             mAction.entryStandby();
22         }
23     }

25     while(squeueSize > 0U)
26     {
27         CMessage internalMsg(sInternalQueue);
28         sQueueSize = 0U;
29         consumed = mStatePtr->dispatch(internalMsg);
30     }
31     return consumed;
32 }

```

Im Zustand *Standby* wird das Ereignis an alle Unterzustände übergeben um zu prüfen, ob diese betreten werden sollen.

```

1  template<class StateList>
2  bool TFSM<StateList>::onStandby(CMessage& msg)
3  {
4      bool consumed = this->tryEntry(msg);

```

```

5   if(consumed == true)
6   {
7       mAction.exitStandby();
8       mStatePtr->onEntry();
9   }
10  return consumed;
11 }

```

Die Methode *tryEntry()* stößt lediglich die Abfrage der Unterzustände an.

```

1  template<class StateList>
2  bool TFSM<StateList>::tryEntry(CMessage& msg, AState*& statePtr)
3  {
4      return mStateList.tryEntry(msg, statePtr);
5  }

```

Die Vorteile dieses Konzept verdeutlichen sich wieder bei der Anwendung. Für jeden Versuch wird eine Zustandsklasse und Actionhandler entworfen, wodurch eine übersichtliche Projektstruktur entsteht. Um die letztendliche Zustandsmaschine zu erhalten wird lediglich *TFSM* mit der gewünschten Liste von Zustandstypen instantiiert.

```

1  using StateList  = TYPELIST_4(CSensorCalib, CADCCalib,
2                                CEdgeBalance, CCornerBalance);
3  using ControlFSM = TFSM<StateList>;
4  ControlFSM myFSM;

```

Sollen weitere Zustände hinzugefügt oder entfernt werden muss lediglich die Typdefinition von *StateList* angepasst werden. Um Coderedundanzen zu vermeiden kann für die Unterzustände auch eine Templateklasse entworfen werden, die das Eintrittsereignis und den Actionhandler als Parameter entgegennimmt. Die Unterzustände spezialisieren dann lediglich die Methoden *dispatch()*, *onEntry()* und *onExit()*.

```

1  /* TSubState.h */
2  template<const EEvent entryEvent, class Action>
3  class TSubState : public AState
4  {
5  public:
6      bool tryEntry(CMessage& msg, AState*& statePtr) override
7      {
8          if(entryEvent == msg.getEvent())
9          {
10             statePtr = this;
11             return true;
12         }
13         return false;
14     };
15     bool dispatch(CMessage& msg) override;
16     void onEntry() override;
17     void onExit() override;
18 private:
19     Action mAction;
20 };

```

```

1  /* CADCCalib.cpp */
2  using CADCCalib = TSubState<EEvent::RunADCCalib, CADCCalibAction>;
3  template<>
4  bool CADCCalib::dispatch(CMessage& msg)
5  {
6      EEvent event = msg.getEvent();
7      if(EEvent::TIMERTICK == event)
8      {
9          mAction.sampleADCCalib();

```

```
10     return true;
11 }
12 ...
13 return false;
14 }
15 template<>
16 void CADCCalib::onEntry()
17 {
18     cout << "Entering ADC-Calibration . . . " << endl;
19     mAction.resumeTimer();
20 }
21 template<>
22 void CADCCalib::onExit()
23 {
24     cout << "Exiting ADC-Calibration . . . " << endl;
25     mAction.pauseTimer();
26 }
```

7.5 Entwurf der Kommunikationskomponente

Die zweite Komponente des Systems ist für die Kommunikation mit einem Desktop-PC verantwortlich. Unter die Kommunikationsaufgaben fällt einerseits die Übermittlung relevanter Daten des Regelkreises, wie z.B. der aktuelle Wert des Zustandvektors. Andererseits nimmt die Kommunikationskomponente Steuerbefehle entgegen und leitet diese an die Regelungskomponente weiter. Dadurch können während der Ausführung der Anwendung Konfiguration der Versuche durchgeführt werden. Beispiele hierfür sind die Auswahl des zu verwendeten Regler- oder Filterkonzeptes. Für die Verbindung zwischen der Ziel- und Entwicklungsplattform wird das TCP/IP-Protokoll verwendet, wobei die BeagleBone Black-Anwendung als Server fungiert.

Der Kontrollfluss der Kommunikationskomponente wird ebenfalls als Zustandsmaschine modelliert. Diese besitzt die beiden Zustände *Standby* und *Running*. Im Zustand *Standby* wartet der TCP/IP-Server auf die Verbindungsanfrage eines Clients. Ein Verbindungsaufbau wird durch ein entsprechendes Event signalisiert, woraufhin die Zustandsmaschine in *Running* wechselt. In diesem Zustand werden Nachrichten der Regelungskomponente an den Client und umgekehrt von dem Client an die Regelungskomponente weitergeleitet. Der Verbindungsabbruch durch den Client wird wiederum durch ein Event signalisiert. Daraufhin kehrt die Zustandsmaschine in *Standby* zurück und wartet auf eine erneute Verbindungsanfrage.

Für die Implementierung der Zustandsmaschine wird die objektorientierte Methode nach Samek verwendet (Abschnitt...). An dieser Stelle werden keine Templates verwendet, da die Maschine lediglich zwei Unterstände besitzt. Des weiteren ist nicht zu erwarten, dass sich das Zustandsdiagramm im Entwicklungsprozess verändert.

Der Actionhandler der Zustandsmaschine besitzt eine Instanz der Klasse *CServer*, welche die nötigen Serverfunktionalitäten bereitstellt.

```

1  class CServer
2  {
3  public:
4      void init();
5      bool waitForClient();
6      bool transmitMessage(CMessage& msg);
7      bool receiveMessage(CMessage& msg);
8      ...
9  };

```

In der *waitForClient()*-Methode wird der aufrufende Thread blockiert bis ein Client eine Verbindung aufbaut. Anschließend können mittels der Methoden *transmitMessage()* und *receiveMessage()* Nachrichten versendet bzw. empfangen werden. Da es sich bei einer TCP/IP-Verbindung um einen Bytestream handelt wird der Typ *CMessage* als Datenpaket in der Übertragung verwendet. Das heißt das lediglich Instanzen von *CMessage* übermittelt werden, wobei die Objekte byteweise versendet und empfangen werden. Im Anschluss werden die einzelnen Bytes wieder zu einer Instanz von *CMessage* zusammengesetzt.

Da das Versenden und Empfangen von Nachrichten parallel erfolgen soll, wird die Kommunikationskomponente um einen zweiten Thread erweitert. Dieser wird in der Klasse *CReceiveTask* implementiert, welche von *IRunnable* erbt. Dieser Thread ist sowohl für das Empfangen von Nachrichten als auch die Annahme von Verbindungsanfragen zuständig.

```
1 void CReceiveTask::run()
2 {
3     while(true)
4     {
5         if(mServer.waitForClient() == true)
6         {
7             sProxy.clientConnected();
8         }
9
10        CMessage receivedMsg;
11        while(mServer.receiveMessage(receivedMsg)
12        {
13            sProxy.routeClientMessage(receivedMsg);
14        }
15        sProxy.clientDisconnect();
16    }
17 }
```

Der Thread ruft zunächst die *waitForClient()*-Methode der *CServer*-Instanz auf, welche den Thread solange blockiert bis eine Verbindung aufgebaut wurde. Daraufhin erzeugt der Task mittels des Proxy ein Ereignis um die Komponente zu benachrichtigen. Im nächsten Schritt ruft der Task die *receiveMessage()*-Methode des Servers in einer Schleife auf. Diese blockiert den Thread bis entweder eine Nachricht eintrifft oder die Verbindung abgebrochen wird. Die Fallunterscheidung erfolgt mit Hilfe des Rückgabewertes von *receiveMessage()*, wobei im Fall einer empfangenen Nachricht diese über den Proxy an die Regelungskomponente weitergeleitet wird. Wird die Verbindung von dem Client abgebrochen verlässt der Task die Empfangsschleife. Anschließend wird der Verbindungsabbruch über den Proxy signalisiert und wieder auf eine erneute Anfrage gewartet.

Der Hauptthread der Komponente wird über deren Eventqueue synchronisiert. Hierbei sind lediglich drei Fälle zu unterscheiden. Die Events *ClientConnect* und *ClientDisconnect* führen jeweils zu einem Zustandswechsel der Zustandsmaschine. Befindet diese sich im Zustand *Running* wird zusätzlich geprüft ob es sich um eine Nachricht handelt, welche über den Server an den Desktop-Client weitergeleitet werden muss.