Inhaltsverzeichnis

Kapitel 1

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.

1.1 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.

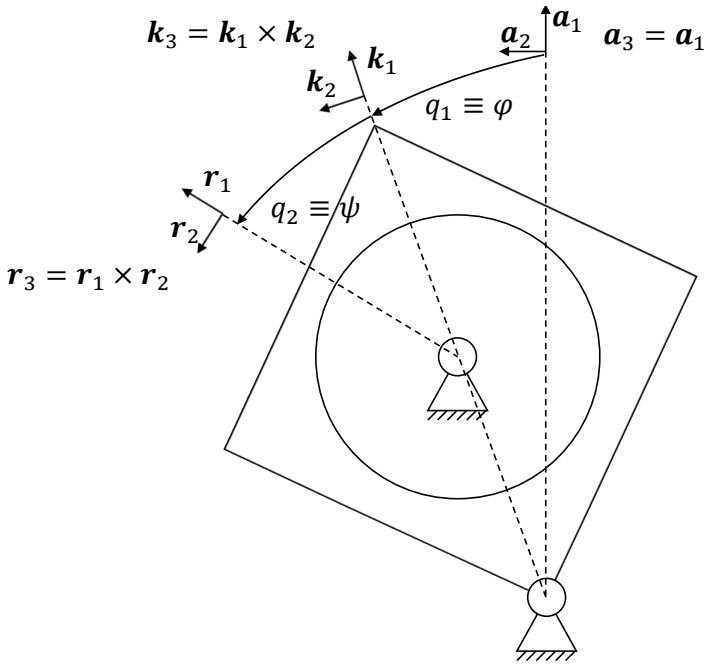

Abbildung 1.1: Kinematikplan des Würfels auf einer Kante, Quelle: eigene Darstellung

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 $\boldsymbol{b_i}$ $i \in \{1, 2, 3\}$ definiert wird. Die Vektoren $\boldsymbol{b_i}$ werden als Vektorbasis bezeichnet.[1] In diesem Fall ist A ein Intertialsystem, das heißt die Ausrichtung der Vektoren $\boldsymbol{a_i}$ ist konstant. Deshalb wird A bzw. dessen Vektorbasis auch als raumfest bezeichnet. Bei K handelt es sich um ein körperfestes Bezugssystem, die Vektoren $\boldsymbol{k_i}$ 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. Foglich beschreiben die Koordiaten φ und ψ die Rotation von K relativ zu A bzw. von R gegenüber K. Mittels eines Bezugssystems kann ein bliebiger Vektor als Linearkombination der Vektorbasis dargestellt werden. Als Beispiel sei der Ortsvektor $\boldsymbol{c}$ des Würfelschwerpunktes geannt, wobei das verwendete Bezugssystem durch den vorangehenden Superskript gekennzeichnet wird.

$$\boldsymbol{c} = l_C \cdot \boldsymbol{k}_1 + 0 \cdot \boldsymbol{k}_2 + 0 \cdot \boldsymbol{k}_3 = {}^{K}\begin{pmatrix} l_C \\ 0 \\ 0 \end{pmatrix} \tag{1.1}$$

[1] Allgemein 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 $\boldsymbol{c}$ in dem Bezugssystem A zu erhalten, muss das Skalarprodukt $\langle \boldsymbol{a}_i, \boldsymbol{c} \rangle$ berechnet werden. Wird dieser Ansatz auf alle drei Komponenten appliziert ergibt sich der folgende Zusammenhang.[2]

$$\begin{aligned}
\boldsymbol{c} = {}^{A}\begin{pmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\end{pmatrix} &= \langle \boldsymbol{c}, \boldsymbol{a}_1\rangle \cdot \boldsymbol{a}_1 + \langle \boldsymbol{c}, \boldsymbol{a}_2\rangle \cdot \boldsymbol{a}_2 + \langle \boldsymbol{c}, \boldsymbol{a}_3\rangle \cdot \boldsymbol{a}_3 = {}^{A}\begin{pmatrix}\langle \boldsymbol{c}, \boldsymbol{a}_1\rangle\\ \langle \boldsymbol{c}, \boldsymbol{a}_2\rangle\\ \langle \boldsymbol{c}, \boldsymbol{a}_3\rangle\end{pmatrix} \\
&= {}^{A}\begin{pmatrix}\beta_1 \cdot \langle \boldsymbol{a}_1, \boldsymbol{k}_1\rangle + \beta_2 \cdot \langle \boldsymbol{a}_1, \boldsymbol{k}_2\rangle + \beta_3 \cdot \langle \boldsymbol{a}_1, \boldsymbol{k}_3\rangle\\ \beta_1 \cdot \langle \boldsymbol{a}_2, \boldsymbol{k}_1\rangle + \beta_2 \cdot \langle \boldsymbol{a}_2, \boldsymbol{k}_2\rangle + \beta_3 \cdot \langle \boldsymbol{a}_2, \boldsymbol{k}_3\rangle\\ \beta_1 \cdot \langle \boldsymbol{a}_3, \boldsymbol{k}_1\rangle + \beta_2 \cdot \langle \boldsymbol{a}_3, \boldsymbol{k}_2\rangle + \beta_3 \cdot \langle \boldsymbol{a}_3, \boldsymbol{k}_3\rangle\end{pmatrix} = {}^{K}\left(\underbrace{\begin{bmatrix}\langle \boldsymbol{a}_1, \boldsymbol{k}_1\rangle & \langle \boldsymbol{a}_1, \boldsymbol{k}_2\rangle & \langle \boldsymbol{a}_1, \boldsymbol{k}_3\rangle\\ \langle \boldsymbol{a}_2, \boldsymbol{k}_1\rangle & \langle \boldsymbol{a}_2, \boldsymbol{k}_2\rangle & \langle \boldsymbol{a}_2, \boldsymbol{k}_3\rangle\\ \langle \boldsymbol{a}_3, \boldsymbol{k}_1\rangle & \langle \boldsymbol{a}_3, \boldsymbol{k}_2\rangle & \langle \boldsymbol{a}_3, \boldsymbol{k}_3\rangle\end{bmatrix}}_{\equiv {}^{K}\boldsymbol{P}^{A}} \cdot \underbrace{\begin{bmatrix}\beta_1\\ \beta_2\\ \beta_3\end{bmatrix}}_{= {}^{K}\boldsymbol{c}}\right) \\
&= {}^{A}\left(\begin{bmatrix}c_\varphi & -s_\varphi & 0\\ s_\varphi & c_\varphi & 0\\ 0 & 0 & 1\end{bmatrix} \cdot {}^{K}\boldsymbol{c}\right)
\end{aligned} \tag{1.2}$$

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 ${}^{E}\boldsymbol{P}^{F}$ definiert ist. Des weiteren zeigt dieses Beispiel die perspektivische Bedeutung eines Bezugssystem. Wird der Vektor $\boldsymbol{c}$ 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 Intertialsystem 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 finfachen Form dargestellt und anschließend in das gewünschte Bezugssystem projiziert wird. Als Beispiel dient die Berechnung des Gravitationsmoments $\boldsymbol{T}^{K/O}$. Der Erdbeschleunigungsvektor $\boldsymbol{g}$ ist im Inertialsystem A konstant und wird von dort in das körperfeste System K transformiert.

$$\begin{aligned}
\boldsymbol{T}^{K/O} = \boldsymbol{c} \times m \cdot \boldsymbol{g} &= {}^{K}\begin{pmatrix}l_C\\ 0\\ 0\end{pmatrix} \times {}^{A}\begin{pmatrix}-m \cdot g\\ 0\\ 0\end{pmatrix} = {}^{K}\begin{pmatrix}l_C\\ 0\\ 0\end{pmatrix} \times {}^{K}\left({}^{A}\boldsymbol{P}^{K} \cdot {}^{A}\begin{pmatrix}-m \cdot g\\ 0\\ 0\end{pmatrix}\right) \\
&= {}^{K}\begin{pmatrix}l_C\\ 0\\ 0\end{pmatrix} \times {}^{K}\begin{pmatrix}-c_\varphi \cdot m \cdot g\\ s_\varphi \cdot m \cdot g\\ 0\end{pmatrix} = {}^{K}\begin{pmatrix}0\\ 0\\ s_\varphi \cdot m \cdot g \cdot l_C\end{pmatrix}
\end{aligned} \tag{1.3}$$

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

[2] Um die allgemeine Gültigkeit der linearen Abbildung zu verdeutlichen wird in dieser Gleichung die Definition $\boldsymbol{c} = {}^{K}\begin{pmatrix}l_C\\ 0\\ 0\end{pmatrix} \equiv {}^{K}\begin{pmatrix}\beta_1\\ \beta_2\\ \beta_3\end{pmatrix}$ 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 Vketors dient wieder der Ortsvektor $\boldsymbol{c}$ 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 $\boldsymbol{c}$ in A differenziert werden soll, so muss dieser zunächst in A dargestellt und anschließend komponentenweise nach der Zeit abgeleitet werden.

$$\frac{^A d\boldsymbol{c}}{dt} = \frac{d(c_\varphi \cdot l_C}{dt} \cdot \boldsymbol{a}_1 + \frac{d(0)}{dt} \cdot \boldsymbol{a}_3 = {}^A\begin{pmatrix} -s_\varphi \cdot \dot{\varphi} \cdot l_C \\ c_\varphi \cdot \dot{\varphi} \cdot l_C \\ 0 \end{pmatrix} \tag{1.4}$$

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

$${}^B\boldsymbol{v}^P = \frac{^B d\boldsymbol{p}}{dt} \qquad {}^B\boldsymbol{a}^P \tag{1.5}$$

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).

$${}^A\boldsymbol{\omega}^K = {}^K\begin{pmatrix} 0 \\ 0 \\ \dot{\varphi} \end{pmatrix} \quad {}^K\boldsymbol{\omega}^R = {}^R\begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix} \quad {}^A\boldsymbol{\omega}^R = {}^A\boldsymbol{\omega}^K + {}^K\boldsymbol{\omega}^R = {}^A\begin{pmatrix} 0 \\ 0 \\ \dot{\varphi} + \dot{\psi} \end{pmatrix} \tag{1.6}$$

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 generalsierten 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.

$$\begin{array}{cc} u_1 \equiv \dot{\varphi} & u_2 \equiv \dot{\varphi} + \dot{\psi} \\ {}^{A}\boldsymbol{\omega}^{K} = {}^{K}\begin{pmatrix} 0 \\ 0 \\ u_1 \end{pmatrix} & {}^{A}\boldsymbol{\omega}^{R} = {}^{K}\begin{pmatrix} 0 \\ 0 \\ u_2 \end{pmatrix} \end{array} \tag{1.7}$$

Diese Geschwindigkeitsausdrücke können auch in die Form

$$\begin{array}{cccc} {}^{A}\boldsymbol{\omega}^{K} = u_1 \cdot {}^{A}\boldsymbol{\omega}_1^{K} + u_2 \cdot {}^{A}\boldsymbol{\omega}_2^{K} & & {}^{A}\boldsymbol{\omega}^{R} = u_1 \cdot {}^{A}\boldsymbol{\omega}_1^{R} + u_2 \cdot {}^{A}\boldsymbol{\omega}_2^{R} & \\ {}^{A}\boldsymbol{\omega}_1^{K} = \boldsymbol{k}_3 & {}^{A}\boldsymbol{\omega}_2^{K} = 0 & {}^{A}\boldsymbol{\omega}_1^{R} = 0 & {}^{A}\boldsymbol{\omega}_2^{R} = \boldsymbol{k}_3 \end{array} \tag{1.8}$$

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.

1.2 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 $\boldsymbol{T}_M^{R/M}$ beschleunigt, andererseits wirkt ein verzögerndes Reibmoment $\boldsymbol{T}_M^{R/M}$. Das Reibmoment wird als proportional zu der Winkelgeschwindigkeit $\dot{\psi}$ modelliert.

$$\boldsymbol{T}^{R/M} = \boldsymbol{T}_M^{R/M} + \boldsymbol{T}_R^{R/M} = [T_M - C_\psi(u_2 - u_1)] \cdot \boldsymbol{k}_3 \tag{1.9}$$

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 $\boldsymbol{T}_G^{K/O}$ beschleunigt. Die Summe der Komponenten ergibt das resultierende Drehmoment $\boldsymbol{T}^{K/O}$.

$$\boldsymbol{T}^{K/O} = T_G^{K/O} - \boldsymbol{T}_M^{R/M} - \boldsymbol{t}_R^{R/M} = [m \cdot g \cdot l_C \cdot s_\varphi - T_M + C_\psi(u_2 - u_1)] \cdot \boldsymbol{k}_3 \tag{1.10}$$

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.

$$\begin{aligned} F_1 &= \langle \boldsymbol{T}^{K/O}, {}^A\boldsymbol{\omega}_1^K \rangle + \langle \boldsymbol{T}^{R/M}, {}^A\boldsymbol{\omega}_1^R \rangle = m \cdot g \cdot l_C \cdot s_\varphi - T_M + C_\psi(u_2 - u_1) \\ F_2 &= \langle \boldsymbol{T}^{K/O}, {}^A\boldsymbol{\omega}_2^K \rangle + \langle \boldsymbol{T}^{R/M}, {}^A\boldsymbol{\omega}_2^R \rangle = T_M - C_\psi(u_2 - u_1) \end{aligned} \tag{1.11}$$

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]

$$\boldsymbol{T}_*^{R/M} = -I_R \cdot {}^K \begin{pmatrix} 0 \\ 0 \\ \dot{u}_2 \end{pmatrix} \tag{1.12}$$

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, wleche dabei als Punkt mit

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

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

$$\boldsymbol{T}_*^{K/O} = -(I^{GH/O} + m_R \cdot l_{MO}^2) \cdot {}^K\begin{pmatrix} 0 \\ 0 \\ \dot{u}_1 \end{pmatrix} = -I_K \cdot {}^K\begin{pmatrix} 0 \\ 0 \\ \dot{u}_1 \end{pmatrix} \tag{1.13}$$

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

$$\begin{aligned} F_1^* &= \langle \boldsymbol{T}_*^{K/O}, {}^A\boldsymbol{\omega}_1^K \rangle + \langle \boldsymbol{T}_*^{R/M}, {}^A\boldsymbol{\omega}_1^R \rangle = -I_K \cdot \dot{u}_1 \\ F_2^* &= \langle \boldsymbol{T}_*^{K/O}, {}^A\boldsymbol{\omega}_2^K \rangle + \langle \boldsymbol{T}_*^{R/M}, {}^A\boldsymbol{\omega}_2^R \rangle = -I_R \cdot \dot{u}_2 \end{aligned} \tag{1.14}$$

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

$$\begin{aligned} F_1 + F_1^* = 0 \quad &\leftrightarrow \quad I_K \cdot \dot{u}_1 = m \cdot g \cdot l_C \cdot s_\varphi - T_M + C_\psi(u_2 - u_2) \\ F_2 + F_2^* = 0 \quad &\leftrightarrow \quad I_R \cdot \dot{u}_2 = T_M - C_\psi(u_2 - u_1) \end{aligned} \tag{1.15}$$

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.

$$\tilde{u}_1 \equiv \dot{\varphi} \qquad \tilde{u}_2 \equiv \dot{\psi} \tag{1.16}$$

$$\begin{aligned} {}^A\tilde{\boldsymbol{\omega}}_1^K &= \boldsymbol{k}_3 & {}^A\tilde{\boldsymbol{\omega}}_2^K &= 0 \\ {}^A\tilde{\boldsymbol{\omega}}_1^R &= 0 & {}^A\tilde{\boldsymbol{\omega}}_2^R &= \boldsymbol{k}_3 \end{aligned} \tag{1.17}$$

$$\begin{aligned} \tilde{F}_1 &= m \cdot g \cdot l_C \cdot s_\varphi & \tilde{F}_2 &= T_M - C_\psi(u_2 - u_1) \\ \tilde{F}_1^* &= -I_R \cdot (\dot{\tilde{u}}_1 + \dot{\tilde{u}}_2) - I_K \cdot \dot{\tilde{u}}_1 & \tilde{F}_2^* &= -I_R \cdot (\dot{\tilde{u}}_1 + \dot{\tilde{u}}_2) \end{aligned} \tag{1.18}$$

$$\begin{aligned} \tilde{F}_1 + \tilde{F}_1^* = 0 \quad &\leftrightarrow \quad I_k \cdot \dot{\tilde{u}}_1 + I_R \cdot (\dot{\tilde{u}}_1 + \dot{\tilde{u}}_2) = m \cdot g \cdot l_C \cdot s_\varphi \\ \tilde{F}_2 + \tilde{F}_2^* = 0 \quad &\leftrightarrow \quad I_R \cdot (\dot{\tilde{u}}_1 + \dot{\tilde{u}}_2) = T_M - C_\psi(\dot{\tilde{u}}_2 - \dot{\tilde{u}}_1) \end{aligned} \tag{1.19}$$

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 2

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 reellen 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.

2.1 Zustandsraumdarstellung für kontinuierliche Systeme

Aus dem vorherigen Kapitel sind die beiden Bewegungsgleichungen

$$I_K \cdot \dot{u}_K = m \cdot g \cdot l \cdot s_\varphi - C_\psi \cdot u_K + C_\psi \cdot u_R - T_M \tag{2.1}$$

$$I_R \cdot \dot{u}_R = C_\psi \cdot u_K - C_\psi \cdot u_R + T_M \tag{2.2}$$

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

$$\boldsymbol{x} = \begin{bmatrix} \varphi \\ u_K \\ u_R \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} \varphi \\ u_K \\ u_R \end{bmatrix} \qquad u = T_M \tag{2.3}$$

können die linearisierten Bewegungsgleichungen in die folgende Zustandsraumdarstellung überführt werden.

$$\dot{\boldsymbol{x}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ \frac{m \cdot g \cdot l}{I_K} & \frac{-C_\psi}{I_K} & \frac{C_\psi}{I_K} \\ 0 & \frac{C_\psi}{I_R} & \frac{-C_\psi}{I_R} \end{bmatrix}}_{\equiv \boldsymbol{A}} \cdot \boldsymbol{x} + \underbrace{\begin{bmatrix} 0 \\ \frac{-1}{I_K} \\ \frac{1}{I_R} \end{bmatrix}}_{\equiv \boldsymbol{b}} \cdot \boldsymbol{u} \tag{2.4}$$

$$\boldsymbol{y} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\equiv \boldsymbol{C}} \cdot \boldsymbol{x} \tag{2.5}$$

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

$$\mathfrak{S} : \begin{cases} \dot{\boldsymbol{x}}(t) = \boldsymbol{A} \cdot \boldsymbol{x}(t) + \boldsymbol{B} \cdot \boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \boldsymbol{C} \cdot \boldsymbol{x}(t) + \boldsymbol{D} \cdot \boldsymbol{u}(t) \end{cases} \tag{2.6}$$

Wobei $\boldsymbol{x} \in \mathbb{R}^n$ Zustandsvektor, $\boldsymbol{u} \in \mathbb{R}^r$ Eingangsvektor und $\boldsymbol{y} \in \mathbb{R}^m$ 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 $\mathfrak{S}(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C})$ betrachtet, deren Eingangsvektor $\boldsymbol{u}$ den Ausgangsvektor $\boldsymbol{y}$ nicht direkt beeinflusst und somit $\boldsymbol{D} = \boldsymbol{0}$ 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.

$$\tilde{u}_K \equiv \dot{\phi} \qquad \tilde{u}_R \equiv \dot{\psi} \tag{2.7}$$

$$I_K \cdot \dot{\tilde{u}}_K = m \cdot g \cdot l \cdot sin_\varphi + C_\psi \cdot \tilde{u}_R - T_M \tag{2.8}$$

$$I_R \cdot \dot{\tilde{u}}_R = -\frac{I_R \cdot m \cdot g \cdot l \cdot sin_\varphi}{I_K} - \frac{(I_K + I_R) \cdot C_\psi}{I_K} + \frac{I_K + I_R}{I_K} \cdot T_M \tag{2.9}$$

$$\dot{\tilde{\boldsymbol{x}}} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{m\cdot g\cdot l}{I_K} & 0 & \frac{C_\psi}{I_K} \\ \frac{-I_R\cdot m\cdot g\cdot l}{I_R\cdot I_K} & 0 & \frac{-C_\psi(I_K+I_R)}{I_R\cdot I_K} \end{bmatrix} \cdot \tilde{\boldsymbol{x}} + \begin{bmatrix} 0 \\ \frac{-1}{I_K} \\ \frac{I_K+I_R}{I_K\cdot I_R} \end{bmatrix} \cdot u \tag{2.10}$$

$$\boldsymbol{y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \cdot \tilde{\boldsymbol{x}} \tag{2.11}$$

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

$$\tilde{\boldsymbol{x}} = \boldsymbol{T}^{-1}\cdot\boldsymbol{x} \qquad \dot{\tilde{\boldsymbol{x}}} = \boldsymbol{T}^{-1}\cdot\dot{\boldsymbol{x}} \tag{2.12}$$

$$\begin{aligned} \dot{\tilde{\boldsymbol{x}}} &= \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T}\cdot\tilde{\boldsymbol{x}} + \boldsymbol{T}^{-1}\cdot\boldsymbol{u} \\ &= \tilde{\boldsymbol{A}}\cdot\tilde{\boldsymbol{x}} + \tilde{\boldsymbol{B}}\cdot\boldsymbol{u} \qquad | \quad \tilde{\boldsymbol{A}} = \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T}, \tilde{\boldsymbol{B}} = \boldsymbol{T}^{-1}\boldsymbol{B} \end{aligned} \tag{2.13}$$

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{T}\cdot\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{C}}\cdot\tilde{\boldsymbol{x}} \qquad | \quad \tilde{\boldsymbol{C}} = \boldsymbol{C}\boldsymbol{T} \tag{2.14}$$

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 $\mathfrak{S}(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C})$ der Ordnung n gegeben, dessen Systemmatrix $\boldsymbol{A}$ n einfache Eigenwerte λ_i mit den zugehörigen Eigenvektoren $\boldsymbol{v}_i$ besitzt. Sind die Eigenvektoren $\boldsymbol{v}_i$ linear unabhängig dann ist die Matrix

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_i & \boldsymbol{v}_2 & \dots & \boldsymbol{v}_n \end{bmatrix} \tag{2.15}$$

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

$$\begin{aligned} \dot{\tilde{\boldsymbol{x}}} &= \boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V}\cdot\tilde{\boldsymbol{x}} + \boldsymbol{V}^{-1}\boldsymbol{B}\cdot\boldsymbol{u} \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{V}\cdot\tilde{\boldsymbol{x}} \end{aligned} \tag{2.16}$$

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

$$\tilde{\boldsymbol{A}} = \boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V} = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix} \tag{2.17}$$

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

kanonischen Normalform lässt sich mit Hilfe des Exponentialansatzes ermitteln.

$$\tilde{x}_{i,h}(t) = e^{\lambda_i \cdot t} \cdot \tilde{x}_i(0) \;\rightarrow\; \tilde{\boldsymbol{x}}_h(t) = \begin{bmatrix} e^{\lambda_1 \cdot t} & & & \\ & e^{\lambda_2 \cdot t} & & \\ & & \ddots & \\ & & & e^{\lambda_n \cdot t} \end{bmatrix} \cdot \tilde{\boldsymbol{x}}(0) \tag{2.18}$$

Die Rücktransformation

$$\boldsymbol{x}_h = \boldsymbol{V} \cdot \tilde{\boldsymbol{x}}_h \tag{2.19}$$

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

$$\boldsymbol{x}(t) = e^{\boldsymbol{A} \cdot t} \cdot \boldsymbol{x}(0) + \int_0^t e^{\boldsymbol{A}(t-\tau)} \boldsymbol{B} \cdot \boldsymbol{u}(\tau) d\tau \tag{2.20}$$

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

$$\boldsymbol{\Phi}(t) = e^{\boldsymbol{A} \cdot t} = \sum_{k=0}^{\infty} \boldsymbol{A}^k \frac{t^k}{k!} \tag{2.21}$$

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

$$\begin{aligned} \dot{\boldsymbol{x}}(t) = \boldsymbol{A} \cdot \boldsymbol{x}(t) + \boldsymbol{B} \cdot \boldsymbol{u}(t) \quad &\circ\!\!-\!\!\bullet \quad s \cdot \boldsymbol{x}(s) - \boldsymbol{x}(t=0) = \boldsymbol{A} \cdot \boldsymbol{x}(s) + \boldsymbol{B} \cdot \boldsymbol{u} \\ &\leftrightarrow \quad \boldsymbol{x}(s) = \underbrace{(s \cdot \boldsymbol{I} - \boldsymbol{A})^{-1}}_{\equiv \boldsymbol{\Phi}(s)} \boldsymbol{x}(t=0) + (s \cdot \boldsymbol{I} - \boldsymbol{A})^{-1} \boldsymbol{B} \cdot \boldsymbol{u}(s) \\ &\leftrightarrow \quad \boldsymbol{x}(s) = \boldsymbol{\Phi}(s) \cdot \boldsymbol{x}(t=0) + \boldsymbol{\Phi}(s) \boldsymbol{B} \cdot \boldsymbol{u}(s) \end{aligned} \tag{2.22}$$

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

$$\boldsymbol{x}(s) = \boldsymbol{\Phi}(s) \boldsymbol{B} \cdot \boldsymbol{u}(s) \qquad | \quad \boldsymbol{x}(t=0) = 0 \tag{2.23}$$

$$\boldsymbol{y}(s) = \boldsymbol{C} \cdot \boldsymbol{x}(s) = \underbrace{\boldsymbol{C} \boldsymbol{\Phi}(s) \boldsymbol{B}}_{\equiv \boldsymbol{G}(s)} \cdot \boldsymbol{u}(s) = \boldsymbol{G}(s) \cdot \boldsymbol{u}(s) \tag{2.24}$$

Im Falle eines SISO-Systems reduzieren sich die Matrizen $\boldsymbol{C}$ und $\boldsymbol{B}$ auf die Vektoren $\boldsymbol{c^T}$ und $\boldsymbol{b}$. Somit handelt es sich bei $G(s)$ um die Übertragungsfunktion des Eingrößensystems. Bei MIMO-Systemen heißt $\boldsymbol{G}(s)$ 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 $\boldsymbol{G}(s)$ stellt die Fundamentalmatrix $\boldsymbol{\Phi}(s)$ die einzige Größe dar, die von s abhängt, und muss somit auch die Pole des Systems enthalten. Aus der Berechnung

$$\boldsymbol{\Phi}(s) = (s \cdot \boldsymbol{I} - \boldsymbol{A})^{-1} = \frac{1}{det(s \cdot \boldsymbol{I} - \boldsymbol{A}}adj(s \cdot \boldsymbol{I} - \boldsymbol{A}) \tag{2.25}$$

folgt, dass das charakteristische Polynom

$$det(s \cdot \boldsymbol{I} - \boldsymbol{A}) \tag{2.26}$$

den gemeinsamen Nenner der Teilübertragungsfunktionen darstellt. Die Nullstellen des Polynoms entsprechen den Eigenwerten der Systemmatrix $\boldsymbol{A}$. 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 $\boldsymbol{A}$ 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 $\boldsymbol{A}$ negativ sind.

$$Re\lambda_i < 0 \tag{2.27}$$

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

2.2 Zustandsraumdarstellung für zeitdiskrete Systeme

Bisher wurden ausschließlich zeitkontinuierliche Systeme betrachtet. Allerdings wird der Regler mit Hilfe eines Digitalrechners implementiert und stellt somit ein zeitdiskretes Systems 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 $\mathfrak{D}(\boldsymbol{A_d},\boldsymbol{B_d},\boldsymbol{C_d})$:

$$\mathfrak{D}: \begin{cases} \boldsymbol{x}(k+1) = \boldsymbol{A} \cdot \boldsymbol{x}(k) + \boldsymbol{B} \cdot \boldsymbol{u}(k) \\ \boldsymbol{y}(k) = \boldsymbol{C} \cdot \boldsymbol{x}(k) \end{cases} \tag{2.28}$$

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

$$\begin{aligned} \boldsymbol{x}(k+1) = \boldsymbol{A_d} \cdot \boldsymbol{x}(k) + \boldsymbol{B_d} \cdot \boldsymbol{u}(k) \quad &\circ\!\!-\!\!\bullet \quad z \cdot \boldsymbol{x}(z) - z \cdot \boldsymbol{x}(t=0) = \boldsymbol{A_d} \cdot \boldsymbol{x}(z) + \boldsymbol{B_d} \cdot \boldsymbol{u}(z) \\ &\leftrightarrow \quad (z \cdot \boldsymbol{I} - \boldsymbol{A_d})\boldsymbol{x}(z) = z \cdot \boldsymbol{x}(t=0) + \boldsymbol{B_d} \cdot \boldsymbol{u}(z) \\ &\leftrightarrow \quad \boldsymbol{x}(z) = \underbrace{(z \cdot \boldsymbol{I} - \boldsymbol{A_d})^{-1}}_{\equiv \boldsymbol{\Phi_d}(z)} \cdot z \cdot \boldsymbol{x}(t=0) + (z \cdot \boldsymbol{I} - \boldsymbol{A_d})^{-1} \boldsymbol{B_d} \cdot \boldsymbol{u}(z) \end{aligned} \tag{2.29}$$

$$\boldsymbol{y}(z) = \boldsymbol{C_d} \cdot \boldsymbol{x}(z) = \underbrace{\boldsymbol{C_d} \boldsymbol{\Phi_d}(z) \boldsymbol{B_d}}_{\equiv \boldsymbol{G_d}(z)} \cdot \boldsymbol{u}(z) \tag{2.30}$$

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

$$z = e^{T \cdot s} \qquad s = \sigma + j \cdot \omega \tag{2.31}$$

$$|z| = e^{T \cdot \sigma} \qquad \phi = \omega \cdot T \tag{2.32}$$

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 kontinuerliches 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 Zustandsarumdarstellung der Regelstrecke in eine diskrete transformiert werden. Hierzu wird zunächst die Lösung eines kontinuierlichen Systems $\mathfrak{S}(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C})$

betrachtet. Herleitung aus [RT2, Unbehauen]

$$\boldsymbol{x}(t) = e^{\boldsymbol{A}\cdot t} \cdot \boldsymbol{x}(t_0) + \int_0^t e^{\boldsymbol{A}(t-\tau)}\boldsymbol{B} \cdot \boldsymbol{u}(\tau)d\tau \tag{2.33}$$

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

$$\begin{aligned}\boldsymbol{x}((k+1)T) &= e^{\boldsymbol{A}(k+1)T} \cdot \boldsymbol{x}(kT) + \int_{kT}^{(k+1)T} e^{\boldsymbol{A}([k+1]T-\tau}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau \\ &= e^{\boldsymbol{A}(k+1)T} \cdot \boldsymbol{x}(kT) + \int_0^T e^{\boldsymbol{A}(kT-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau\end{aligned} \tag{2.34}$$

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

$$\boldsymbol{u}(kT + t) = \boldsymbol{u}(kT) \qquad | \quad 0 \leqslant t < T \tag{2.35}$$

Die Lösung des Integrals führt auf:

$$\boldsymbol{x}([k+1]T) = e^{\boldsymbol{A}T} \cdot \boldsymbol{x}(kT) + e^{\boldsymbol{A}T}(\boldsymbol{I} - e^{-\boldsymbol{A}T})\boldsymbol{A}^{-1}\boldsymbol{B} \cdot \boldsymbol{u}(kT) \tag{2.36}$$

Diese Lösung stellt den Zusammenhang zwischen der kontinuierlichen und diskreten Zustandsraumdarstellung dar. Wobei der Vergleich mit ([**?**]) folgenden Werte für die Matrizen des diskreten Systems $\mathfrak{D}(\boldsymbol{A_d},\boldsymbol{B_d},\boldsymbol{C_d})$ liefert.

$$\boldsymbol{A_d} = e^{\boldsymbol{A}T} \qquad \boldsymbol{B_d} = (e^{\boldsymbol{A}T} - \boldsymbol{I})\boldsymbol{A}^{-1}\boldsymbol{B} \qquad \boldsymbol{C_d} = \boldsymbol{C} \tag{2.37}$$

Mit der Definition

$$\boldsymbol{S} = T\sum_{v=0}^{\infty} \boldsymbol{A}^v \frac{T^v}{(v+1)!} \tag{2.38}$$

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

$$\boldsymbol{A_d} = \boldsymbol{I} + \boldsymbol{SA} \qquad \boldsymbol{B_d} = \boldsymbol{SB} \tag{2.39}$$

Die Matrix $\boldsymbol{S}$ 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.

2.3 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.

2.3.1 Reglerentwurf durch Eigenwertvorgabe

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

Aus der Abbildung gehen die Systemgleichungen des geschlossenen Regelkreises $\mathfrak{D}(\boldsymbol{A_g},\boldsymbol{0},\boldsymbol{C_g})$ hervor.

$$\boldsymbol{u} = -\boldsymbol{B_d}\boldsymbol{K} \tag{2.40}$$

$$\mathfrak{D}_g : \begin{cases} \boldsymbol{x}(k+1) = \boldsymbol{A}\cdot\boldsymbol{x}(k) - \boldsymbol{B}\boldsymbol{K}\cdot\boldsymbol{x}(k) = \underbrace{(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{K})}_{\equiv \boldsymbol{A_g}}\cdot\boldsymbol{x}(k) \\ \boldsymbol{y}(k) = \underbrace{\boldsymbol{C}}_{\equiv \boldsymbol{C_g}}\cdot\boldsymbol{x}(k) \end{cases} \tag{2.41}$$

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

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-1} \end{bmatrix} \qquad \boldsymbol{b} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \tag{2.42}$$

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 deinen soll. Der besondere Vorteil der Regelungsnormalform besteht darin, dass die Werte a_i die Koeffizienten des charakteristischen Polynoms

$$det(\lambda\cdot\boldsymbol{I}-\boldsymbol{A}) = \prod_{i=1}^{n}(\lambda-\lambda_i) = \lambda^n + a_{n-1}\cdot\lambda^{n-1} + \ldots + a_1\cdot\lambda + a_0 \tag{2.43}$$

der Systemmatrix $\boldsymbol{A}$ sind. Wird der Regelkreis über den Reglervekor

$$\boldsymbol{k} = \begin{bmatrix} k_1 & k_2 & \dots & k_n \end{bmatrix} \tag{2.44}$$

geschlossen, ergibt sich sich die folgende Systemmatrix $\boldsymbol{A_g}$ des geschlossenen Regelkreises.

$$\begin{aligned}\boldsymbol{A_g} = \boldsymbol{A} - \boldsymbol{bk} &= \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-1} \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ k_1 & k_2 & k_3 & \dots & k_n \end{bmatrix} \\ &= \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\overline{a}_0 & -\overline{a}_1 & -\overline{a}_2 & \dots & -\overline{a}_{n-1} \end{bmatrix} \quad | \quad \overline{a}_i = a_i + k_{i+1}\end{aligned} \tag{2.45}$$

Die Systemmatrix $\boldsymbol{A_g}$ des geschlossenen Regelkreises liegt nun ebenfalls in Regelungsnormalform vor, weshalb wiederum für deren charakteristisches Polynom gilt:

$$det(\lambda \cdot \boldsymbol{I} - \boldsymbol{A_g}) = \prod_{i=1}^{n}(\lambda - \overline{\lambda}_i) = \lambda^n + \overline{a}_{n-1} \cdot \lambda^{n-1} + \ldots + \overline{a}_1 \cdot \lambda + \overline{a}_0 \tag{2.46}$$

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

$$\overline{a}_i = a_i + k_i \tag{2.47}$$

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

$$\underbrace{\begin{bmatrix} \overline{a}_0 & \overline{a}_1 & \dots & \overline{a}_{n-1} \end{bmatrix}}_{\equiv \overline{\boldsymbol{a}}} = \underbrace{\begin{bmatrix} a_0 & a_1 & \dots & a_{n-1} \end{bmatrix}}_{\equiv \boldsymbol{a}} + \underbrace{\begin{bmatrix} k_1 & k_2 & \dots & k_n \end{bmatrix}}_{\equiv \boldsymbol{k}} \tag{2.48}$$

$$\boldsymbol{k} = \overline{\boldsymbol{a}} - \boldsymbol{a} \tag{2.49}$$

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 Regelungsnormalform als auch die Ermittlung der Regerparameter 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.

2.3.2 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

$$J = \sum_{k=0}^{\infty} [\boldsymbol{x}^T(k) \cdot \boldsymbol{Q} \cdot \boldsymbol{x}(k) + \boldsymbol{u}^T(k) \cdot \boldsymbol{R} \cdot \boldsymbol{u}(k)] \tag{2.50}$$

verwendet. Die Matrizen $\boldsymbol{Q}$ und $\boldsymbol{R}$ 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

$$\boldsymbol{K} = (\boldsymbol{R} + \boldsymbol{B}^T \boldsymbol{P} \boldsymbol{B})^{-1} \boldsymbol{B}^T \boldsymbol{P} \boldsymbol{A}\,, \tag{2.51}$$

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

$$\boldsymbol{P} = \boldsymbol{Q} + \boldsymbol{A}^T [\boldsymbol{P} - \boldsymbol{P}\boldsymbol{B}(\boldsymbol{R} + \boldsymbol{B}^T \boldsymbol{P}\boldsymbol{B})^{-1} \boldsymbol{B}^T \boldsymbol{P}] \boldsymbol{A} \tag{2.52}$$

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 Einheitskreises liegen. Des weiteren lässt sich durch

$$|z| = e^{\sigma_i} \qquad | \qquad s_i = \sigma_i + \omega_i \cdot j \tag{2.53}$$

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 $\boldsymbol{K}$ bestimmt werden. Daraus kann geschlossen werden, dass

[1] Die 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 $\boldsymbol{R}$ interpretiert werden. Je größer die Elemente von $\boldsymbol{R}$ 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 $\boldsymbol{Q}$ wirkt entgegengesetzt. Das das Gütekriterium den zeitlichen Verlauf des Zustandvektors erfasst, führt eine Erhöhung der Matrix $\boldsymbol{Q}$ 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 Diagnoalelemente der Matrix $\boldsymbol{Q}$ 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 Reglergesetzt

$$\boldsymbol{u} = -\boldsymbol{K} \cdot \boldsymbol{x} \tag{2.54}$$

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 wir 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

[2] Allerdings sei angemerkt, dass sich durch die Gewichtungsmatrix $\boldsymbol{Q}$ 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.

2.4 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

$$\boldsymbol{Q} = \begin{bmatrix} \varphi_{max}^{-2} & 0 & 0 \\ 0 & u_{K,max}^{-2} & 0 \\ 0 & 0 & u_{R,max}^{-2} \end{bmatrix} \qquad \boldsymbol{R} = \begin{bmatrix} T_{M,max}^{-2} \end{bmatrix} \tag{2.55}$$

verwendet werden. Daraus resultiert das Gütekriterium

$$J = \sum_{k=0}^{\infty} \left(\frac{\varphi^2}{\varphi_{max}^2} + \frac{u_K^2}{u_{K,max}^2} + \frac{u_R^2}{u_{R,max}^2} + \frac{T_M^2}{T_{M,max}^2} \right) . \tag{2.56}$$

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

$$\lambda_1 = 0,7895 \qquad \lambda_{2,3} = 0,8830 \pm 0,0087j \tag{2.57}$$

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 Zeitpunk yy durch eine äußere Störung erregt wurde.

placeholder plot

Die Abbildungen zeigen, dass der Regler das System nicht in die Ruhelage $\boldsymbol{x} = \boldsymbol{0}$ 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

$$\hat{\boldsymbol{x}} = \begin{pmatrix} \hat{\varphi} \\ \hat{u}_K \\ \hat{u}_R \end{pmatrix} \tag{2.58}$$

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

$$\mathfrak{D}_0 \equiv \begin{cases} \boldsymbol{x}_o(k+1) = \underbrace{\begin{bmatrix} \boldsymbol{A} & \boldsymbol{0}^{3x3} \\ \boldsymbol{0}^{3x3} & \boldsymbol{I}^{3x3} \end{bmatrix}}_{\equiv \boldsymbol{A}_o} \cdot \underbrace{\begin{bmatrix} \boldsymbol{x} \\ \hat{\boldsymbol{x}} \end{bmatrix}}_{\equiv \boldsymbol{x}_0}(k) + \underbrace{\begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{0}^3 \end{bmatrix}}_{\equiv \boldsymbol{b}_o} \cdot u(k) \\ \boldsymbol{y}(k) = \underbrace{\begin{bmatrix} \boldsymbol{I}^{3x3} & \boldsymbol{I}^{3x3} \end{bmatrix}}_{\equiv \boldsymbol{C}_o} \cdot \boldsymbol{x}_o(k) \end{cases} . \tag{2.59}$$

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

$$u = \boldsymbol{k}^T \cdot \boldsymbol{y} = \boldsymbol{k}^T \cdot (\boldsymbol{x} + \hat{\boldsymbol{x}}) = \underbrace{\begin{bmatrix} \boldsymbol{k}^T & \boldsymbol{k}^T \end{bmatrix}}_{\equiv \boldsymbol{k}_o^T} \cdot \boldsymbol{x}_o \tag{2.60}$$

Hiermit kann der Regelkreis geschlossen werden und es resultiert das System

$$\overline{\mathfrak{D}}_o = \begin{cases} \boldsymbol{x}_o(k+1) = \begin{bmatrix} \boldsymbol{A} - \boldsymbol{b}_o \boldsymbol{k}_o^T & -\boldsymbol{b}_o \boldsymbol{k}_o^T \\ \boldsymbol{0}^{3x3} & \boldsymbol{I}^{3x3} \end{bmatrix} \cdot \boldsymbol{x}_o(k) \\ \boldsymbol{y}(k) = \boldsymbol{C}_o \cdot \boldsymbol{x}_o(k) \end{cases} . \tag{2.61}$$

Für den geschlossenen Regelkreis ergeben sich die Eigenwerte

$$\lambda_1 = 0,7895 \qquad \lambda_{2,3} = 0,8830 \pm 0,0087j \qquad \lambda_{4,5,6} = 1\,. \tag{2.62}$$

Diese zeigen, dass die urpsrünglichen 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 $\boldsymbol{x}$ zu ermitteln wird das System $\overline{\mathfrak{D}}_o$ in die kanonische

Normalform transformiert.

$$\tilde{\overline{\boldsymbol{A}}}_o = \boldsymbol{V}_o^{-1}\overline{\boldsymbol{A}}_o\boldsymbol{V}_o = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_6 \end{bmatrix} \qquad \boldsymbol{x}_o = \boldsymbol{V}_o \cdot \tilde{\boldsymbol{x}}_o \tag{2.63}$$

Interessant ist hierbei die Form der Transformationsmatrix

$$\boldsymbol{V}_o = \begin{bmatrix} & 0 & 0 & 0 \\ \boldsymbol{V} & 0 & 0 & 0 \\ & \gamma_1 & \gamma_2 & \gamma_3 \\ \boldsymbol{0}^{3x3} & & \boldsymbol{I}^{3x3} & \end{bmatrix}. \tag{2.64}$$

Zunächst geht aus den unteren drei Zeilen hervor, dass es sich bei den Messabweichungen $\hat{\boldsymbol{x}}$ um kanonische Zustandsvariablen handelt. Des weiteren bleiben die Eigenvektoren $\boldsymbol{V}$ 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 des Schwungmasse ein. Unter der Annahme, dass die Messabweichungen konstant sind, führen diese zu einer verbleibenden Bewegung der Schwungmasse. Deshalb muss die vorherige Stabilitäsaussage auf das Gebiet der Zustands- und Stellgrößen begrenzt werden für welche die Stellgröße nicht begrenzt ist. Beispielsweise kann das System nicht mehr stabilsiert werden, wenn der Messfehler so groß ist, dass $\boldsymbol{\gamma} \cdot \hat{\boldsymbol{x}}$ die maximale Drehzahl des Motors überschreitet.

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

Kapitel 3

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.

3.1 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 Rechenezit 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 Gesamtsymtems mit sich. Zunächst existiert eine Vielzahl von Werkzeugen für die Entwicklung von Embbeded-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 eigebettet 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 Betriebssystem auf das Zeitverhalten der Regelung kritisch betrachtet werden. Einerseits kann die Abtastung an äquidistanten Stützstellen nicht mehr garantiert werden. Anderseits 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.

3.1.1 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

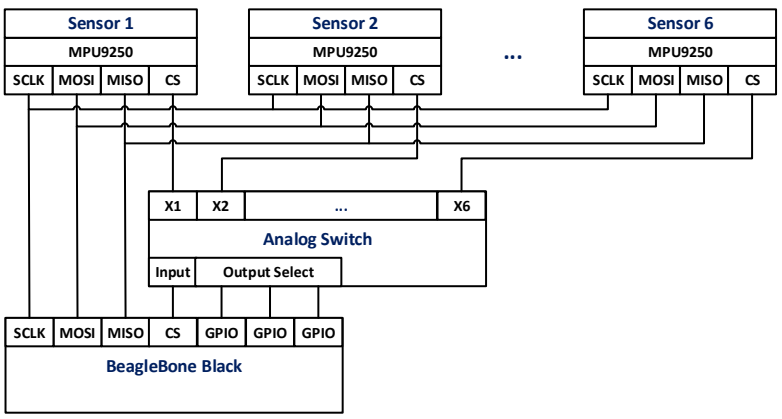

Abbildung 3.1: Blockschaltbild Sensorkommunikation, Quelle: eigene Darstellung

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 betet 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 Konfigurationsoptionnen bietet. Die Klasse nutzt die SPI-Schnittstelle um die Sensoren in der *init()*-Methode zu konfigurieren. Anschießend 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 Konstruktur 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.

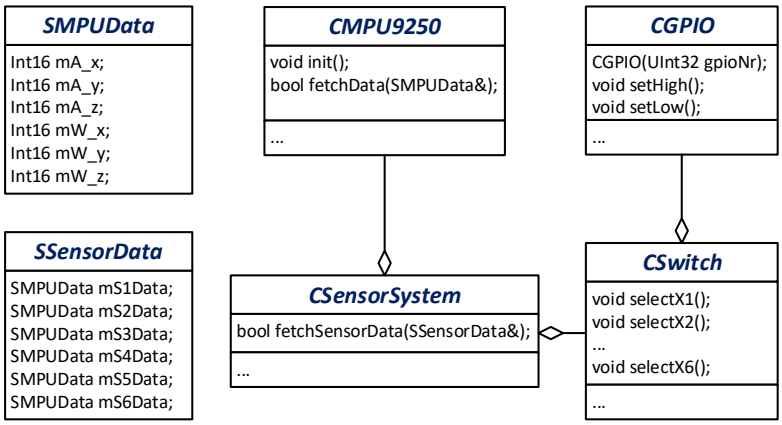

Abbildung 3.2: Klassendiagramm der Sensorschnittstelle, Quelle: eigene Darstellung

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.

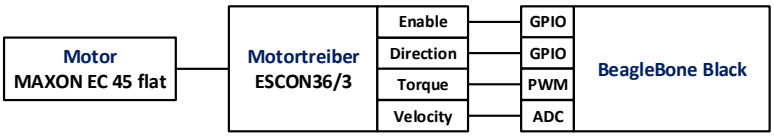

Abbildung 3.3: Blockschaltbild der Motoransteuerung, Quelle: eigene Darstellung

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.

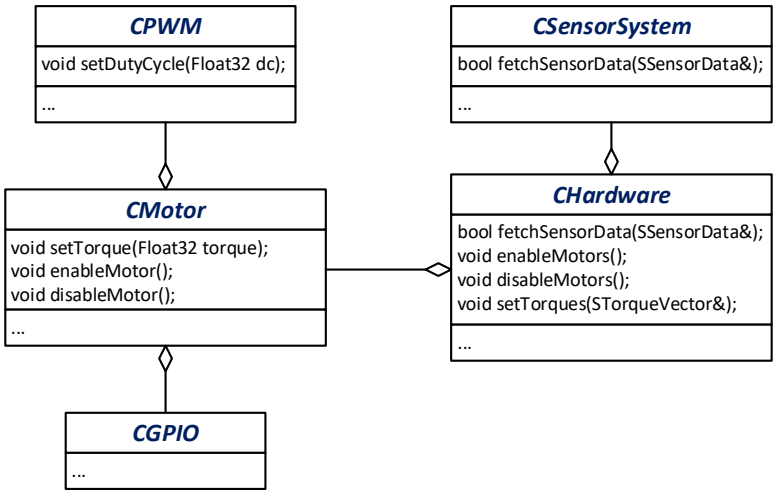

Abbildung 3.4: Klassendiagramm der Hardwareansteuerung, Quelle: eigene Darstellung

3.2 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.

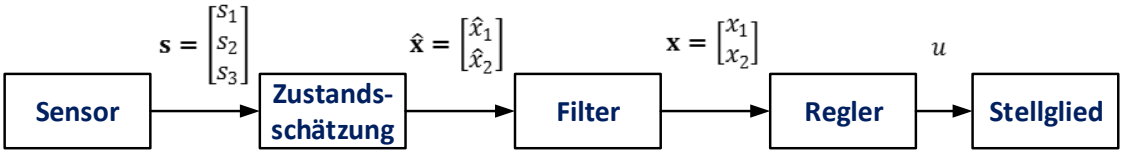

Abbildung 3.5: Blockschaltbild des Signalflusses, Quelle: eigene Darstellung

Für die Implementierung werden die Signale als Datenobjekte implementiert. Das heßt 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 Intanszierungen 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 Typenhierachie 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 TActionHodler<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 Aktionobjektes *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 *TLinHierachy*, mit dem folgenden Prototyp.

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

Der erste Templateparameter ist die Typenliste, welche die zu generierende Typenhierachie 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 TLinHierachy<TTypeList<T1, T2>, Unit, Root>
3   : public Unit<T1, TLinHierachy<T2, Unit, Root> >
4 {};
```

Diese Instanziierung wird solange genutzt bist das Ende der ursprünglichen Typenliste erreicht ist. Die instantiierte Klasse von *TLinHierachy*, ist eine leere Klasse die von *Unit* erbt. *Unit* wird wiederum mit *TLinHierachy* 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 *TLinHierachy* mit einer Typeliste der Länge eins instantiiert wird.

```
1 template<class T, template<class, class> class Unit, class Root>
2 class TLinHierachy<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 = TLinHierachy<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 Hierachie 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();
```

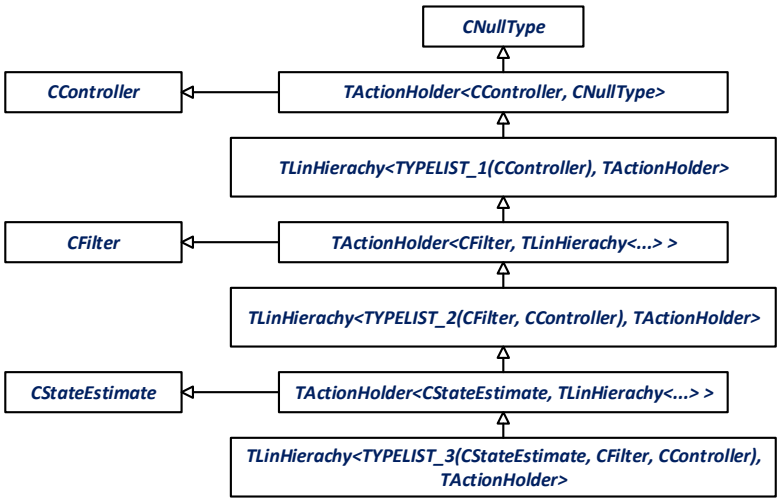

Abbildung 3.6: Klassendiagramm der generierten Hierarchie, Quelle: eigene Darstellung

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);

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.

3.3 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 gemessen 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 Plattform. Hierfür wird ein WebSocketServer verwendet (siehe Abschnitt...). Durch die Verwendung der

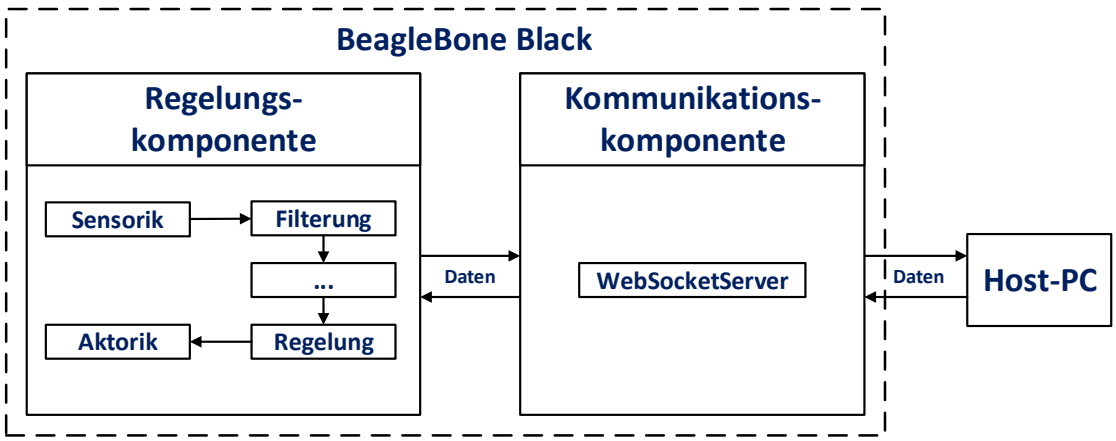

Abbildung 3.7: Blockschaltbild Gesamtsystem, Quelle: eigene Darstellung

Komponentenarchitektur enstehen 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 letzendlichen 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.

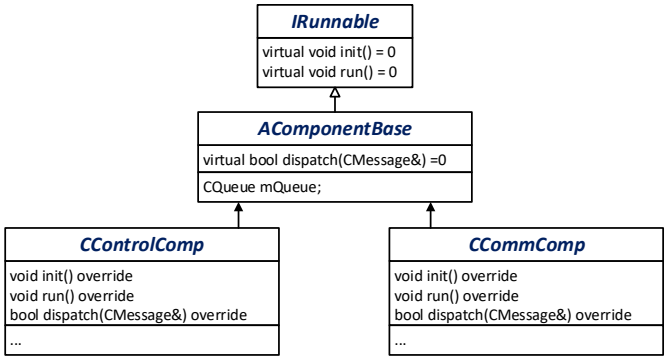

Abbildung 3.8: Klassendiagramm der Komponenten, Quelle: eigene Darstellung

Im nächsten Schritt muss ein Weg zum Datenaustausch zwischen den Komponenten etabliert werden. Einerseits müssen Messdaten und Signale aus dem Regelkreis von der Regelungs- and 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 Eregnis 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 Eingangspuffers 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.

3.4 Entwurf der Regelungskomponente

Die Aufgabe der Regelungskomponente besteht darin den Kontroll- und Signalfluss der verschiedenen Versuche zu steuern. Diese Umsetzung erfolgt mit Hilfe eines Zustandautomats, 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.

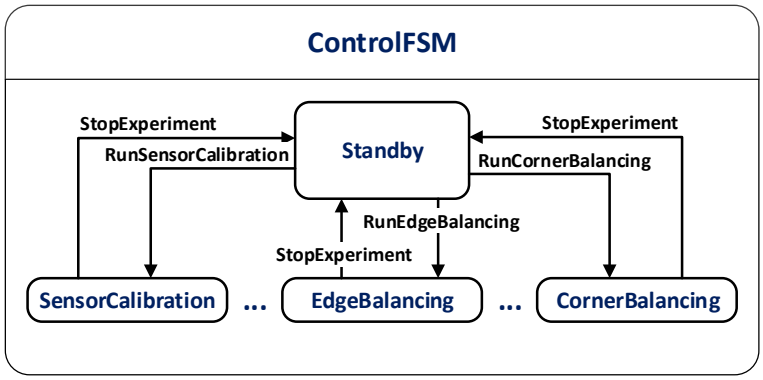

Abbildung 3.9: Zustandsdiagramm der Kontrol-Komponente, Quelle: eigene Darstellung

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 Funktionenzeiger, der auf die Methode des entsprechenden Zustandes gerichtet ist.

Als Basis der Typenhierachie 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 Projektsverlauf 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 Zustandmaschine 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 Templatehierachie verwendet. Hierfür müss 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(mg, 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 Zustandsmachine 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 entgegenimmt. 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  }
```

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 Würfelkörper. Unter dem Würfelkörper 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 Würfelkörper 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 $\boldsymbol{I}^{Ri/Mi}$ der Schwungmassen werden mit Hilfe der CAD-Anwendung ermittelt, wobei die Trägheitstensoren relativ zu den Punkten M_i berechnet werden.

$$m_R = 0,155[kg] \tag{4.1}$$

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.

$$\boldsymbol{I}^{R1/M1} = \begin{pmatrix} 3.358 \cdot 10^{-4} & 2,641 \cdot 10^{-11} & 0 \\ 2,651 \cdot 10^{-11} & 1,961 \cdot 10^{-4} & 4,527 \cdot 10^{-9} \\ 0 & 4.527 \cdot 10^{-9} & 1,691 \cdot 10^{-4} \end{pmatrix} [kg \cdot m^2] \tag{4.2}$$

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

$$\begin{aligned}
\boldsymbol{I}^{R1/M1} &= \begin{pmatrix} I_{11}^{R1} & 0 & 0 \\ 0 & I_{22}^{R1} & 0 \\ 0 & 0 & I_{33}^{R1} \end{pmatrix} = \begin{pmatrix} 3.358 \cdot 10^{-4} & 0 & 0 \\ 0 & 1,961 \cdot 10^{-4} & 0 \\ 0 & 0 & 1,691 \cdot 10^{-4} \end{pmatrix} [kg \cdot m^2] \\
\boldsymbol{I}^{R2/M2} &= \begin{pmatrix} I_{11}^{R2} & 0 & 0 \\ 0 & I_{22}^{R2} & 0 \\ 0 & 0 & I_{33}^{R2} \end{pmatrix} = \begin{pmatrix} 1,691 \cdot 10^{-4} & 0 & 0 \\ 0 & 3.358 \cdot 10^{-4} & 0 \\ 0 & 0 & 1,961 \cdot 10^{-4} \end{pmatrix} [kg \cdot m^2] \\
\boldsymbol{I}^{R3/M3} &= \begin{pmatrix} I_{11}^{R3} & 0 & 0 \\ 0 & I_{22}^{R3} & 0 \\ 0 & 0 & I_{33}^{R3} \end{pmatrix} = \begin{pmatrix} 1,961 \cdot 10^{-4} & 0 & 0 \\ 0 & 1,691 \cdot 10^{-4} & 0 \\ 0 & 0 & 3.358 \cdot 10^{-4} \end{pmatrix} [kg \cdot m^2]
\end{aligned} \tag{4.3}$$

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

Berechnung der Trägheitsmomente in den folgenden Abschnitten.

$$m_K = 1.07[kg] \tag{4.4}$$

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

Für die Masse m des Gesamtsystems folgt

$$m = 1,532[kg]. \tag{4.6}$$

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

$$\boldsymbol{c} = {}^{K}\begin{pmatrix} -6,61 \\ -6,60 \\ -6,57 \end{pmatrix} [cm] \approx {}^{K}\begin{pmatrix} l_C \\ l_C \\ l_C \end{pmatrix} \quad | \quad l_C = 6,6[cm] \tag{4.7}$$

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.

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

4.2 Untersuchung der Kinematik

4.2.1 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 $\boldsymbol{a}_1$, $\boldsymbol{a}_2$ und $\boldsymbol{a}_3$ definiert wird. Das Würfelgehäuse verfügt über drei rotatorische Freiheitsgrade, welche durch die Winkel φ_1, φ_2 und φ_3 beschrieben werden.

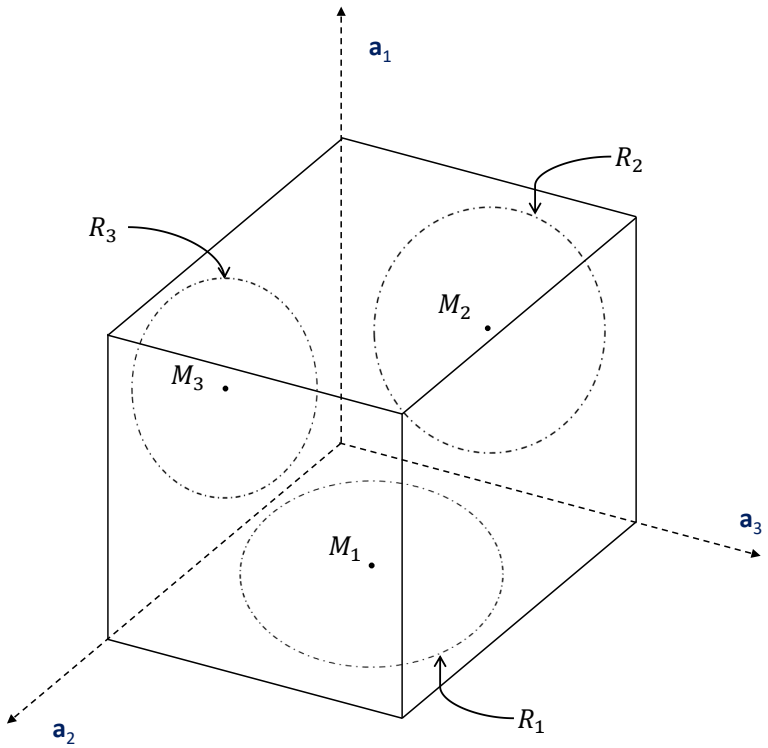

Abbildung 4.1: Mechanischer Aufbau der Würfelseite, Quelle: eigene Darstellung

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

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

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

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

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

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

$$ {}^{C}\boldsymbol{v} = {}^{K}\Big({}^{C}\boldsymbol{P}^{K} \,.\, {}^{C}\boldsymbol{v} \Big) = {}^{K}\boldsymbol{v} \qquad | \qquad {}^{C}\boldsymbol{P}^{K} = \begin{pmatrix} c_{\varphi_3} & s_{\varphi_3} & 0 \\ -s_{\varphi_3} & c_{\varphi_3} & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.11} $$

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.

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

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.

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

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.

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

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

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

digkeiten ${}^A\boldsymbol{\omega}_i^j$ 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.

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

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

$${}^A\boldsymbol{\omega}^K = u_1 \cdot \boldsymbol{k}_1 + u_2 \cdot \boldsymbol{k}_2 + u_3 \cdot \boldsymbol{k}_3 \quad \rightarrow \; {}^A\boldsymbol{\omega}_1^K = \boldsymbol{k}_1, {}^A\boldsymbol{\omega}_2^K = \boldsymbol{k}_2, {}^A\boldsymbol{\omega}_3^K = \boldsymbol{k}_3 \tag{4.18}$$

$${}^A\boldsymbol{\omega}_4^K = 0, {}^A\boldsymbol{\omega}_5^K = 0, {}^A\boldsymbol{\omega}_6^K = 0 \tag{4.19}$$

$${}^A\boldsymbol{\omega}^{R_1} = u_4 \cdot \boldsymbol{k}_1 + u_2 \cdot \boldsymbol{k}_2 + u_3 \cdot \boldsymbol{k}_3 \quad \rightarrow \; {}^A\boldsymbol{\omega}_1^K = 0, {}^A\boldsymbol{\omega}_2^K = \boldsymbol{k}_2, {}^A\boldsymbol{\omega}_3^K = \boldsymbol{k}_3 \tag{4.20}$$

$${}^A\boldsymbol{\omega}_4^K = \boldsymbol{k}_1, {}^A\boldsymbol{\omega}_5^K = 0, {}^A\boldsymbol{\omega}_6^K = 0 \tag{4.21}$$

$${}^A\boldsymbol{\omega}^{R_2} = u_1 \cdot \boldsymbol{k}_1 + u_5 \cdot \boldsymbol{k}_2 + u_3 \cdot \boldsymbol{k}_3 \quad \rightarrow \; {}^A\boldsymbol{\omega}_1^K = \boldsymbol{k}_1, {}^A\boldsymbol{\omega}_2^K = 0, {}^A\boldsymbol{\omega}_3^K = \boldsymbol{k}_3 \tag{4.22}$$

$${}^A\boldsymbol{\omega}_4^K = 0, {}^A\boldsymbol{\omega}_5^K = \boldsymbol{k}_2, {}^A\boldsymbol{\omega}_6^K = 0 \tag{4.23}$$

$${}^A\boldsymbol{\omega}^{R_3} = u_1 \cdot \boldsymbol{k}_1 + u_2 \cdot \boldsymbol{k}_2 + u_6 \cdot \boldsymbol{k}_3 \quad \rightarrow \; {}^A\boldsymbol{\omega}_1^K = \boldsymbol{k}_1, {}^A\boldsymbol{\omega}_2^K = \boldsymbol{k}_2, {}^A\boldsymbol{\omega}_3^K = 0 \tag{4.24}$$

$${}^A\boldsymbol{\omega}_4^K = 0, {}^A\boldsymbol{\omega}_5^K = 0, {}^A\boldsymbol{\omega}_6^K = \boldsymbol{k}_3 \tag{4.25}$$

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 Würfelkörper 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 $\boldsymbol{T}^{Ri/Mi}$ der Schwungmassen R_i um ihren jeweiligen Drehpunkt M_i ermittelt werden. Hierfür muss einerseits das Drehmoment $\boldsymbol{T}_M^{Ri/Mi}$ des antreibenden Motors und das verzögernde Reibmoment $\boldsymbol{T}_R^{Ri/Mi}$ beachtet werden.

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

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

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

Der Würfelkörper wird einerseits durch das Gravitationsmoment $\boldsymbol{T}K/O_G$ und die resultierenden Momente der Schwungmassen $\boldsymbol{T}Ri/Mi$, welche in umgekehrter Richtung auf den Würfel wirken, beeinflusst. Das Gravitationsmoment hängt von der Gewichtskraft $\boldsymbol{G} = {}^A\begin{pmatrix} -m \cdot g \\ 0 \\ 0 \end{pmatrix}$ und der Position des Schwerpunktes *bsc* ab.

$$\boldsymbol{T}_G^{K/O} = \boldsymbol{c} \times \boldsymbol{G} = {}^K\begin{pmatrix} l_C \\ l_C \\ l_C \end{pmatrix} \times {}^K\begin{pmatrix} -m \cdot g \cdot c_{\varphi_2} \cdot c_{\varphi_3} \\ -m \cdot g \cdot c_{\varphi_2} \cdot s_{\varphi_3} \\ -m \cdot g \cdot s_{\varphi_2} \end{pmatrix} = -m \cdot l_C \cdot g \, {}^K\begin{pmatrix} s_{\varphi_2} + c_{\varphi_2} s_{\varphi_3} \\ -s_{\varphi_2} + c_{\varphi_2} c_{\varphi_3} \\ -c_{\varphi_2} c_{\varphi_3} - c_{\varphi_2} \cdot s_{\varphi_3} \end{pmatrix} \tag{4.29}$$

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

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.

$$F_1 = \langle {}^A\boldsymbol{\omega}_1^K, \boldsymbol{T}^{K/O}\rangle + \sum_{i=1}^{3}\langle {}^A\boldsymbol{\omega}_1^{Ri}, \boldsymbol{T}^{Ri/Mi}\rangle = m \cdot l_C \cdot g(-s_{\varphi_2} - c_{\varphi_2}s_{\varphi_3}) - T_{M1} + C_\psi(u_4 - u_1)$$

(4.31)

$$F_2 = \langle {}^A\boldsymbol{\omega}_2^K, \boldsymbol{T}^{K/O}\rangle + \sum_{i=1}^{3}\langle {}^A\boldsymbol{\omega}_2^{Ri}, \boldsymbol{T}^{Ri/Mi}\rangle = m \cdot l_C \cdot g(s_{\varphi_2} - c_{\varphi_2}c_{\varphi_3}) - T_{M2} + C_\psi(u_5 - u_2)$$

(4.32)

$$F_3 = \langle {}^A\boldsymbol{\omega}_3^K, \boldsymbol{T}^{K/O}\rangle + \sum_{i=1}^{3}\langle {}^A\boldsymbol{\omega}_3^{Ri}, \boldsymbol{T}^{Ri/Mi}\rangle = m \cdot l_C \cdot g(c_{\varphi_2}c_{\varphi_3} + c_{\varphi_2}s_{\varphi_3}) - T_{M3} + C_\psi(u_6 - u_3)$$

(4.33)

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

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

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

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

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

Wobei $\boldsymbol{\alpha}$ und $\boldsymbol{\omega}$ die Winkelbeschleunigung- bzw. geschwindigkeit des Körpers und $\boldsymbol{I}$ 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.

$${}^A\boldsymbol{\alpha}^K = \frac{{}^Ad}{dt}\,{}^A\boldsymbol{\omega}^K, {}^A\boldsymbol{\alpha}^{R1} = \frac{{}^Ad}{dt}\,{}^A\boldsymbol{\omega}^{R1}, {}^A\boldsymbol{\alpha}^{R2} = \frac{{}^Ad}{dt}\,{}^A\boldsymbol{\omega}^{R2}, {}^A\boldsymbol{\alpha}^{R3} = \frac{{}^Ad}{dt}\,{}^A\boldsymbol{\omega}^{R3} \tag{4.38}$$

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

$$\boldsymbol{I}^{Ri/Mi} = \begin{pmatrix} I_{11}^{Ri} & 0 & 0 \\ 0 & I_{22}^{Ri} & 0 \\ 0 & 0 & I_{33}^{Ri} \end{pmatrix} \tag{4.39}$$

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

Der Trägheitstensor $\boldsymbol{I}^{K/O}$ des Würfelkörpers um den Punkt O setzt sich aus mehreren Komponenten zusammen. Einerseits wird er durch den Trägheitstensor $\boldsymbol{I}^{GH/O}$ des Gehäuses um den Punkt O beeinflusst, dieser Tensor beschreibt die Trägheitseigenschaften des Würfels ohne die Schwungmassen und kann mit Hilfeder CAD-Anwendung ermittelt werden, andererseits beeinflussen die Schwungmassen das Trägheitsmoment des Würfelkörpers. 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 Würfelkörper 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 $\boldsymbol{I}^{K/O}$ gleich der Summe von $\boldsymbol{I}^{GH/O}$ und den Trägheitstensoren der drei Massepunkte M_i mit der Masse m_R um O, wobei $\boldsymbol{r}_i$ den Ortsvektor des Punktes M_i beschreibt.

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

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

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 $\boldsymbol{c}$ aus Perspektive des Inertialsystems lediglich eine Komponente in Richtung von $\boldsymbol{a}_1$ besitzt.

$${}^A\begin{pmatrix} |\boldsymbol{c}| \\ 0 \\ 0 \end{pmatrix} \stackrel{!}{=} {}^K\boldsymbol{P}^A \cdot {}^K\begin{pmatrix} l_C \\ l_C \\ l_C \end{pmatrix} \quad \rightarrow \quad \varphi_{10} = 0, \quad \varphi_{20} = -2 \cdot atan(\sqrt{2} - \sqrt{3}) \approx 0.62, \quad \varphi_{30} = \frac{-\pi}{4} \tag{4.43}$$

$$\hat{\varphi}_i = \varphi_{i0} + \bar{\varphi}_i \tag{4.44}$$

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

$$\begin{aligned}
\hat{u}_1 &= \dot{\varphi}_2 \cdot s_{\varphi_{30}} + \dot{\varphi}_1 \cdot c_{\varphi_{20}} c_{\varphi_{30}} \\
\hat{u}_2 &= \dot{\varphi}_2 \cdot c_{\varphi_{30}} - \dot{\varphi}_1 \cdot c_{\varphi_{20}} s_{\varphi_{30}} \\
\hat{u}_3 &= \dot{\varphi}_3 + \dot{\varphi}_1 \cdot s_{\varphi_{20}} \\
\hat{u}_4 &= \dot{\varphi}_2 \cdot s_{\varphi_{30}} + \dot{\varphi}_1 \cdot c_{\varphi_{20}} c_{\varphi_{30}} + \dot{\psi}_1 \\
\hat{u}_5 &= \dot{\varphi}_2 \cdot c_{\varphi_{30}} - \dot{\varphi}_1 \cdot c_{\varphi_{20}} s_{\varphi_{30}} + \dot{\psi}_2 \\
\hat{u}_6 &= \dot{\varphi}_3 + \dot{\varphi}_1 \cdot s_{\varphi_{20}} + \dot{\psi}_3
\end{aligned} \tag{4.45}$$

$$ {}^{A}\hat{\boldsymbol{\omega}}^{K} = {}^{K}\begin{pmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_3 \end{pmatrix}, \quad {}^{A}\hat{\boldsymbol{\omega}}^{R1} = {}^{K}\begin{pmatrix} \hat{u}_4 \\ \hat{u}_2 \\ \hat{u}_3 \end{pmatrix}, \quad {}^{A}\hat{\boldsymbol{\omega}}^{R2} = {}^{K}\begin{pmatrix} \hat{u}_1 \\ \hat{u}_5 \\ \hat{u}_3 \end{pmatrix}, \quad {}^{A}\hat{\boldsymbol{\omega}}^{R3} = {}^{K}\begin{pmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_6 \end{pmatrix} \tag{4.46}$$

Insbesondere die Winkelbeschleunigungen werden durch die linearisierten Winkelgeschwindigkeiten stark vereinfacht.

$$ {}^{A}\hat{\boldsymbol{\alpha}}^{K} = \frac{{}^{A}d}{dt}\, {}^{A}\hat{\boldsymbol{\omega}}^{K} = {}^{K}\begin{pmatrix} \ddot{\varphi}_2 \cdot s_{\varphi_{30}} + \ddot{\varphi}_1 \cdot c_{\varphi_{20}} c_{\varphi_{30}} \\ \ddot{\varphi}_2 \cdot c_{\varphi_{30}} - \ddot{\varphi}_1 \cdot c_{\varphi_{20}} s_{\varphi_{30}} \\ \ddot{\varphi}_3 + \ddot{\varphi}_1 \cdot s_{\varphi_{20}} \end{pmatrix} = {}^{K}\begin{pmatrix} \dot{\hat{u}}_1 \\ \dot{\hat{u}}_2 \\ \dot{\hat{u}}_3 \end{pmatrix} \tag{4.47}$$

$$ {}^{A}\hat{\boldsymbol{\alpha}}^{R1} = \frac{{}^{A}d}{dt}\, {}^{A}\hat{\boldsymbol{\omega}}^{R1} = {}^{K}\begin{pmatrix} \ddot{\varphi}_2 \cdot s_{\varphi_{30}} + \ddot{\varphi}_1 \cdot c_{\varphi_{20}} c_{\varphi_{30}} + \ddot{\psi}_1 \\ \ddot{\varphi}_2 \cdot c_{\varphi_{30}} - \ddot{\varphi}_1 \cdot c_{\varphi_{20}} s_{\varphi_{30}} \\ \ddot{\varphi}_3 + \ddot{\varphi}_1 \cdot s_{\varphi_{20}} \end{pmatrix} = {}^{K}\begin{pmatrix} \dot{\hat{u}}_4 \\ \dot{\hat{u}}_2 \\ \dot{\hat{u}}_3 \end{pmatrix} \tag{4.48}$$

$$ {}^{A}\hat{\boldsymbol{\alpha}}^{R2} = \frac{{}^{A}d}{dt}\, {}^{A}\hat{\boldsymbol{\omega}}^{R2} = {}^{K}\begin{pmatrix} \ddot{\varphi}_2 \cdot s_{\varphi_{30}} + \ddot{\varphi}_1 \cdot c_{\varphi_{20}} c_{\varphi_{30}} \\ \ddot{\varphi}_2 \cdot c_{\varphi_{30}} - \ddot{\varphi}_1 \cdot c_{\varphi_{20}} s_{\varphi_{30}} + \ddot{\psi}_2 \\ \ddot{\varphi}_3 + \ddot{\varphi}_1 \cdot s_{\varphi_{20}} \end{pmatrix} = {}^{K}\begin{pmatrix} \dot{\hat{u}}_1 \\ \dot{\hat{u}}_5 \\ \dot{\hat{u}}_3 \end{pmatrix} \tag{4.49}$$

$$ {}^{A}\hat{\boldsymbol{\alpha}}^{K} = \frac{{}^{A}d}{dt}\, {}^{A}\hat{\boldsymbol{\omega}}^{K} = {}^{K}\begin{pmatrix} \ddot{\varphi}_2 \cdot s_{\varphi_{30}} + \ddot{\varphi}_1 \cdot c_{\varphi_{20}} c_{\varphi_{30}} \\ \ddot{\varphi}_2 \cdot c_{\varphi_{30}} - \ddot{\varphi}_1 \cdot c_{\varphi_{20}} s_{\varphi_{30}} \\ \ddot{\varphi}_3 + \ddot{\varphi}_1 \cdot s_{\varphi_{20}} + \ddot{\psi}_3 \end{pmatrix} = {}^{K}\begin{pmatrix} \dot{\hat{u}}_1 \\ \dot{\hat{u}}_2 \\ \dot{\hat{u}}_6 \end{pmatrix} \tag{4.50}$$

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

$$ \hat{\boldsymbol{T}}_{*}^{R1/M1} = -\, {}^{A}\hat{\boldsymbol{\alpha}}^{R1} \cdot \boldsymbol{I}^{R1/M1} = {}^{K}\begin{pmatrix} -\dot{\hat{u}}_4 \cdot I_{11}^{R1} \\ -\dot{\hat{u}}_2 \cdot I_{22}^{R1} \\ -\dot{\hat{u}}_3 \cdot I_{33}^{R3} \end{pmatrix} \tag{4.51}$$

$$ \hat{\boldsymbol{T}}_{*}^{R2/M2} = -\, {}^{A}\hat{\boldsymbol{\alpha}}^{R2} \cdot \boldsymbol{I}^{R2/M2} = {}^{K}\begin{pmatrix} -\dot{\hat{u}}_1 \cdot I_{11}^{R2} \\ -\dot{\hat{u}}_5 \cdot I_{22}^{R2} \\ -\dot{\hat{u}}_3 \cdot I_{33}^{R2} \end{pmatrix} \tag{4.52}$$

$$\hat{\boldsymbol{T}}_*^{R3/M3} = -\,{}^A\hat{\boldsymbol{\alpha}}^{R3} \cdot \boldsymbol{I}^{R3/M3} = {}^K\begin{pmatrix} -\hat{\dot{u}}_1 \cdot I_{11}^{R3} \\ -\hat{\dot{u}}_2 \cdot I_{22}^{R3} \\ -\hat{\dot{u}}_6 \cdot I_{33}^{R3} \end{pmatrix} \tag{4.53}$$

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

$$\hat{\boldsymbol{T}}_*^{K/O} = -\,{}^A\hat{\boldsymbol{\alpha}}^{K} \cdot \boldsymbol{I}^{K/O} = -\,{}^K\begin{pmatrix} \hat{\dot{u}}_1 \cdot I_{11}^K + \hat{\dot{u}}_2 \cdot I_{21}^K + \hat{\dot{u}}_3 \cdot I_{31}^K \\ \hat{\dot{u}}_1 \cdot I_{12}^K + \hat{\dot{u}}_2 \cdot I_{22}^K + \hat{\dot{u}}_3 \cdot I_{32}^K \\ \hat{\dot{u}}_1 \cdot I_{13}^K + \hat{\dot{u}}_2 \cdot I_{23}^K + \hat{\dot{u}}_3 \cdot I_{33}^K \end{pmatrix} \tag{4.54}$$

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

$$\hat{F}_1^* = \langle \hat{\boldsymbol{T}}_*^{K/O}, {}^A\boldsymbol{\omega}_1^K \rangle + \sum_{i=1}^{3} \langle \hat{\boldsymbol{T}}_*^{Ri/Mi}, {}^A\boldsymbol{\omega}_1^{Ri} \rangle = -\hat{\dot{u}}_1 (I_{11}^K + I_{11}^{R2} + I_{11}^{R3}) - \hat{\dot{u}}_2 \cdot I_{21}^K - \hat{\dot{u}}_3 \cdot I_{31}^K \tag{4.55}$$

$$\hat{F}_2^* = \langle \hat{\boldsymbol{T}}_*^{K/O}, {}^A\boldsymbol{\omega}_2^K \rangle + \sum_{i=1}^{3} \langle \hat{\boldsymbol{T}}_*^{Ri/Mi}, {}^A\boldsymbol{\omega}_2^{Ri} \rangle = -\hat{\dot{u}}_1 \cdot I_{12}^K - \hat{\dot{u}}_2 (I_{22}^K + I_{22}^{R1} + I_{33}^{R3}) - \hat{\dot{u}}_3 \cdot I_{13}^K \tag{4.56}$$

$$\hat{F}_3^* = \langle \hat{\boldsymbol{T}}_*^{K/O}, {}^A\boldsymbol{\omega}_3^K \rangle + \sum_{i=1}^{3} \langle \hat{\boldsymbol{T}}_*^{Ri/Mi}, {}^A\boldsymbol{\omega}_3^{Ri} \rangle = -\hat{\dot{u}}_1 \cdot I_{13}^K - \hat{\dot{u}}_2 \cdot I_{23}^K - \hat{\dot{u}}_3 (I_{33}^K + I_{33}^{R1} + I_{33}^{R2}) \tag{4.57}$$

$$\hat{F}_4^* = \langle \hat{\boldsymbol{T}}_*^{K/O}, {}^A\boldsymbol{\omega}_4^K \rangle + \sum_{i=1}^{3} \langle \hat{\boldsymbol{T}}_*^{Ri/Mi}, {}^A\boldsymbol{\omega}_4^{Ri} \rangle = -\hat{\dot{u}}_4 \cdot I_{11}^{R1} \tag{4.58}$$

$$\hat{F}_5^* = \langle \hat{\boldsymbol{T}}_*^{K/O}, {}^A\boldsymbol{\omega}_5^K \rangle + \sum_{i=1}^{3} \langle \hat{\boldsymbol{T}}_*^{Ri/Mi}, {}^A\boldsymbol{\omega}_5^{Ri} \rangle = -\hat{\dot{u}}_5 \cdot I_{22}^{R2} \tag{4.59}$$

$$\hat{F}_6^* = \langle \hat{\boldsymbol{T}}_*^{K/O}, {}^A\boldsymbol{\omega}_6^K \rangle + \sum_{i=1}^{3} \langle \hat{\boldsymbol{T}}_*^{Ri/Mi}, {}^A\boldsymbol{\omega}_6^{Ri} \rangle = -\hat{\dot{u}}_6 \cdot I_{33}^{R3} \tag{4.60}$$

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 $\boldsymbol{T}_G^{K/O}$ linearisiert werden, da die restlichen Momente bereits in linearer Form vorliegen.

$$\hat{\boldsymbol{T}}_G^{K/O} = -m \cdot g \cdot l_C \; {}^K\begin{pmatrix} (c_{\varphi 20} - s_{\varphi 20} c_{\varphi 30})\bar{\varphi}_2 + c_{\varphi 20} c_{\varphi 30} \bar{\varphi}_3 \\ (-c_{\varphi 20} - s_{\varphi 20} c_{\varphi 30})\bar{\varphi}_2 - c_{\varphi 20} s_{\varphi 30} \bar{\varphi}_3 \\ (s_{\varphi 20} s_{\varphi 30} + s_{\varphi 20} c_{\varphi 30})\bar{\varphi}_2 + (c_{\varphi 20} s_{\varphi 30} - c_{\varphi 20} c_{\varphi 30})\bar{\varphi}_3 \end{pmatrix} \tag{4.61}$$

$$\hat{F}_1 = -m \cdot g \cdot l_C(c_{\varphi 20} - s_{\varphi 20}c_{\varphi 30})\bar{\varphi}_2 + c_{\varphi 20}c_{\varphi 30}\bar{\varphi}_3 - T_{M1} + C_\psi(\hat{u}_4 - \hat{u}_1) \tag{4.62}$$

$$\hat{F}_1 = -m \cdot g \cdot l_C[(-c_{\varphi 20} - s_{\varphi 20}c_{\varphi 30})\bar{\varphi}_2 - c_{\varphi 20}s_{\varphi 30}\bar{\varphi}_3] - T_{M2} + C_\psi(\hat{u}_5 - \hat{u}_2) \tag{4.63}$$

$$\hat{F}_3 = -m \cdot g \cdot l_C[(s_{\varphi 20}s_{\varphi 30} + s_{\varphi 20}c_{\varphi 30})\bar{\varphi}_2 + (c_{\varphi 20}s_{\varphi 30} - c_{\varphi 20}c_{\varphi 30})\bar{\varphi}_3] - T_{M3} + C_\psi(\hat{u}_6 - \hat{u}_3) \tag{4.64}$$

$$\hat{F}_4 = T_{M1} - C_\psi(\hat{u}_4 - \hat{u}_1) \tag{4.65}$$

$$\hat{F}_5 = T_{M2} - C_\psi(\hat{u}_5 - \hat{u}_2) \tag{4.66}$$

$$\hat{F}_6 = T_{M3} - C_\psi(\hat{u}_6 - \hat{u}_3) \tag{4.67}$$

An dieser Stelle können nun prinzipiell nach Kanes Gleichung

$$F_i + F_i^* = 0 \tag{4.68}$$

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.

$$\boldsymbol{u}_K \equiv \begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_3 \end{bmatrix} \qquad \boldsymbol{u}_R \equiv \begin{bmatrix} \hat{u}_4 \\ \hat{u}_5 \\ \hat{u}_6 \end{bmatrix} \tag{4.69}$$

$$\begin{aligned} \boldsymbol{F}_K^* = \begin{bmatrix} -F_1^* \\ -F_2^* \\ -F_3^* \end{bmatrix} &= \begin{bmatrix} \hat{\dot{u}}_1(I_{11}^K + I_{11}^{R2} + I_{11}^{R3}) + \hat{\dot{u}}_2 \cdot I_{21}^K + \hat{\dot{u}}_3 \cdot I_{31}^K \\ \hat{\dot{u}}_1 \cdot I_{12}^K + \hat{\dot{u}}_2(I_{22}^K + I_{22}^{R1} + I_{33}^{R3}) + \hat{\dot{u}}_3 \cdot I_{13}^K \\ \hat{\dot{u}}_1 \cdot I_{13}^K + \hat{\dot{u}}_2 \cdot I_{23}^K + \hat{\dot{u}}_3(I_{33}^K + I_{33}^{R1} + I_{33}^{R2}) \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} I_{11}^K + I_{11}^{R1} + I_{11}^{R2} & I_{21}^K & I_{31}^K \\ I_{12}^K & I_{22}^K + I_{22}^{R1} + I_{22}^{R3} & I_{32}^K \\ I_{13}^K & I_{23}^K & I_{33}^K + I_{33}^{R1} + I_{33}^{R2} \end{bmatrix}}_{\equiv \boldsymbol{I}_K} \cdot \underbrace{\begin{bmatrix} \hat{\dot{u}}_1 \\ \hat{\dot{u}}_2 \\ \hat{\dot{u}}_3 \end{bmatrix}}_{\equiv \dot{\boldsymbol{u}}_K} \end{aligned} \tag{4.70}$$

$$\boldsymbol{F}_R^* = \begin{bmatrix} -F_4^* \\ -F_5^* \\ -F_6^* \end{bmatrix} = \begin{bmatrix} \hat{\dot{u}}_4 \cdot I_{11}^{R1} \\ \hat{\dot{u}}_5 \cdot I_{22}^{R2} \\ \hat{\dot{u}}_6 \cdot I_{33}^{R3} \end{bmatrix} = \underbrace{\begin{bmatrix} I_1^{R1}1 & 0 & 0 \\ 0 & I_{22}^{R2} & 0 \\ 0 & 0 & I_{33}^{R3} \end{bmatrix}}_{\equiv \boldsymbol{I}_R} \cdot \underbrace{\begin{bmatrix} \hat{\dot{u}}_4 \\ \hat{\dot{u}}_5 \\ \hat{\dot{u}}_6 \end{bmatrix}}_{\equiv \dot{\boldsymbol{u}}_R} \tag{4.71}$$

$$\begin{aligned} \boldsymbol{F}_K &= {}^K\begin{pmatrix} m \cdot l_C \\ m \cdot l_C \\ m \cdot l_C \end{pmatrix} \times \underbrace{{}^K\begin{pmatrix} -g \cdot c_{\varphi 20}c_{\varphi 30} \\ g \cdot c_{\varphi 20}s_{\varphi 30} \\ -g \cdot s_{\varphi 20} \end{pmatrix}}_{\equiv \boldsymbol{g}} + C_\psi \cdot \begin{bmatrix} \hat{u}_4 - \hat{u}_1 \\ \hat{u}_5 - \hat{u}_2 \\ \hat{u}_6 - \hat{u}_3 \end{bmatrix} - \underbrace{\begin{bmatrix} T_{M1} \\ T_{M2} \\ T_{M3} \end{bmatrix}}_{\equiv \boldsymbol{T}_M} \\ &= \underbrace{\begin{bmatrix} 0 & -m \cdot l_C & m \cdot l_C \\ m \cdot l_C & 0 & -m \cdot l_C \\ -m \cdot l_C & m \cdot l_C & 0 \end{bmatrix}}_{\equiv \boldsymbol{G_{LA}}} \cdot \boldsymbol{g} + C_\psi \cdot \boldsymbol{u}_R - C_\psi \cdot \boldsymbol{u}_K - \boldsymbol{T}_M \end{aligned} \tag{4.72}$$

$$\boldsymbol{F}_R = \begin{bmatrix} F_4 \\ F_5 \\ F_6 \end{bmatrix} = \sum_{i=1}^{3} {}^K\boldsymbol{T}^{Ri/Mi} = \begin{bmatrix} T_{M1} \\ T_{M2} \\ T_{M3} \end{bmatrix} - C_\psi \cdot \begin{bmatrix} \hat{u}_4 - \hat{u}_1 \\ \hat{u}_5 - \hat{u}_2 \\ \hat{u}_6 - \hat{u}_3 \end{bmatrix} \tag{4.73}$$

Nach Kanes Gleichung gilt $F_i = -F_i^*$, somit muss auch $\boldsymbol{F}_K = \boldsymbol{F}_K^*$ und $\boldsymbol{F}_R = \boldsymbol{F}_R^*$ gelten. Daraus resultieren die folgenden Zusammenhänge.

$$\dot{\boldsymbol{u}}_K = \boldsymbol{I}_K^{-1} \cdot \boldsymbol{F}_K^* = \boldsymbol{I}_K^{-1} \cdot \boldsymbol{F}_K = \boldsymbol{I}_K^{-1} \cdot \boldsymbol{G}_{LA} \cdot \boldsymbol{g} - C_\psi \cdot \boldsymbol{I}_K^{-1} \cdot \boldsymbol{u}_K + C_\psi \cdot \boldsymbol{I}_K^{-1} \cdot \boldsymbol{u}_R - \boldsymbol{I}_K^{-1} \cdot \boldsymbol{T}_M$$

(4.74)

$$\dot{\boldsymbol{u}}_R = \boldsymbol{I}_R^{-1} \cdot \boldsymbol{F}_R^* = \boldsymbol{I}_R^{-1} \cdot \boldsymbol{F}_K = C_\psi \cdot \boldsymbol{I}_R^{-1} \cdot \boldsymbol{u}_K - C_\psi \cdot \boldsymbol{I}_R^{-1} \cdot \boldsymbol{u}_R + \boldsymbol{I}_R^{-1} \cdot \boldsymbol{T}_M \tag{4.75}$$

Wenn nun ein Zustandsvektor $\boldsymbol{x} \equiv \begin{bmatrix} \boldsymbol{g} & \boldsymbol{u}_K & \boldsymbol{u}_R \end{bmatrix}^T$ und ein Eingangsvektor $\boldsymbol{u} \equiv \boldsymbol{T}_M$ definiert werden, geben die obigen Gleichungen bereits zwei Drittel der gesuchten Zustandsraumdarstellung wieder. Der letzte Teil ergibt sich durch die Ableitung des Gravitationsmomentes $\boldsymbol{g}$ in Bezug auf das körperfeste System K. Nach Kane gilt für einen Vektor $\boldsymbol{p}$ der in einem Bezugssystem B fixiert ist, welches sich mit der Winkelgeschwindigkeit ${}^A\boldsymbol{\omega}^B$ in dem Bezugssystem A bewegt:

$$\frac{{}^A d\boldsymbol{p}}{dt} = {}^A\boldsymbol{\omega}^B \times \boldsymbol{p} \tag{4.76}$$

Dieser Zusammenhang kann genutzt werden um die Ableitung von $\boldsymbol{g}$ mit Respekt zu K zu bestimmen. Und zwar ist der Gravitationsvektor $\boldsymbol{g}$ im Intertialsystem A fixiert. Des weiteren bewegt sich das Intertialsystem A aus Perspektive des Würfelkörpers mit der Geschwindigkeit ${}^K\boldsymbol{\omega}^A$, welche gleich der richtungsinvertierten Winkelgeschwindigkeit ${}^A\boldsymbol{\omega}^K$ des Würfels ist. Somit gilt für die erste Ableitung von $\boldsymbol{g}$ in K:

$$\begin{aligned} \frac{{}^K d\boldsymbol{g}}{dt} = {}^K\boldsymbol{\omega}^A \times \boldsymbol{g} = -\,{}^A\boldsymbol{\omega}^A \times \boldsymbol{g} &= {}^K\begin{pmatrix} -g \cdot c_{\varphi 20} c_{\varphi 30} \\ g \cdot c_{\varphi 20} s_{\varphi 30} \\ -g \cdot s_{\varphi 20} \end{pmatrix} \times {}^K\begin{pmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_3 \end{pmatrix} \\ &= \underbrace{\begin{bmatrix} 0 & g \cdot s_{\varphi 20} & g \cdot c_{\varphi 20} s_{\varphi 30} \\ -g \cdot s_{\varphi 20} & 0 & g \cdot c_{\varphi 20} c_{\varphi 30} \\ -g \cdot c_{\varphi 20} s_{\varphi 30} & -g \cdot c_{\varphi 20} c_{\varphi 30} & 0 \end{bmatrix}}_{\dot{\boldsymbol{g}}_{LA}} \cdot \boldsymbol{u}_K \end{aligned} \tag{4.77}$$

Diese Umformung macht sich analog zu der Berechnung von $\boldsymbol{G}_{LA}$ zu Nutze, dass Kreuzprodukte als lineare Abbildung dargestellt werden können. Nun können die Gleichungen (4.77), (4.74) und (4.75) zu einer Zustandsraumdarstellung zusammengeführt werden.

$$\underbrace{\begin{bmatrix} \dot{\boldsymbol{g}} \\ \dot{\boldsymbol{u}}_K \\ \dot{\boldsymbol{u}}_R \end{bmatrix}}_{\equiv \dot{\boldsymbol{x}}} = \underbrace{\begin{bmatrix} 0^{3x3} & \dot{\boldsymbol{g}}_{LA} & 0^{3x3} \\ \boldsymbol{I}_K^{-1} \cdot \boldsymbol{G}_{LA} & -C_\psi \cdot \boldsymbol{I}_K^{-1} & C_\psi \cdot \boldsymbol{I}_K^{-1} \\ 0^{3x3} & C_\psi \cdot \boldsymbol{I}_R^{-1} & -C_\psi \cdot \boldsymbol{I}_R^{-1} \end{bmatrix}}_{\equiv \boldsymbol{A}} \cdot \underbrace{\begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{u}_K \\ \boldsymbol{u}_R \end{bmatrix}}_{\equiv \boldsymbol{x}} + \underbrace{\begin{bmatrix} 0^{3x3} \\ -\boldsymbol{I}_K^{-1} \\ \boldsymbol{I}_R^{-1} \end{bmatrix}}_{\equiv \boldsymbol{B}} \cdot \underbrace{\begin{bmatrix} T_{M1} \\ T_{M2} \\ T_{M3} \end{bmatrix}}_{\equiv \boldsymbol{u}} \tag{4.78}$$