

Fakultät: Informationstechnik
Studiengang: Technische Informatik

Studienarbeit
Robert Czech
Mat.Nr.: 733363

Modellierung eines mobilen, stehenden Pendels inklusive Regelung mit MATLAB

Betreuer: Prof. Dipl.-Ing. Reinhard Keller

Kurzfassung

In der Studienarbeit wurde ein mobiles, inverses Pendel in MATLAB modelliert. Für die Modellierung wurden die Bewegungsgleichungen mit Hilfe der Lagrange-Mechanik hergeleitet und in einem MATLAB-Simulink Blockschaltbild realisiert. Die Gleichungen beinhalteten jedoch eine algebraische Schleife die gelöst wurde.

Um die Ergebnisse besser zu visualisieren wurde eine 3D-Animation des Mobilen Pendels erstellt. Hierfür wurde eine Level 1 S-Function erstellt, um diese direkt in MATLAB-Simulink einzubinden und die Animation parallel der Simulation darzustellen. Um zu vermeiden, dass die Simulation schneller als in Echtzeit erfolgt wurde die Animation erweitert, um die Simulationsgeschwindigkeit auf Wunsch abzubremesen.

Für die Regelung wurde aus den Daten des Beschleunigungssensors der Neigungswinkel des Pendels berechnet, unter Berücksichtigung, dass Beschleunigungen in Fahrtrichtung auftreten und Messfehler verursachen können. Der Neigungswinkel wurde ferner auch über die Daten des Gyroskops durch Integration berechnet. Über die Fusion beider Ergebnisse konnten Ungenauigkeiten bei der Berechnung über den Beschleunigungssensor und der Nachteil der relativen und über Zeit driftenden Fehler der Berechnung über die Winkelgeschwindigkeit kompensiert werden.

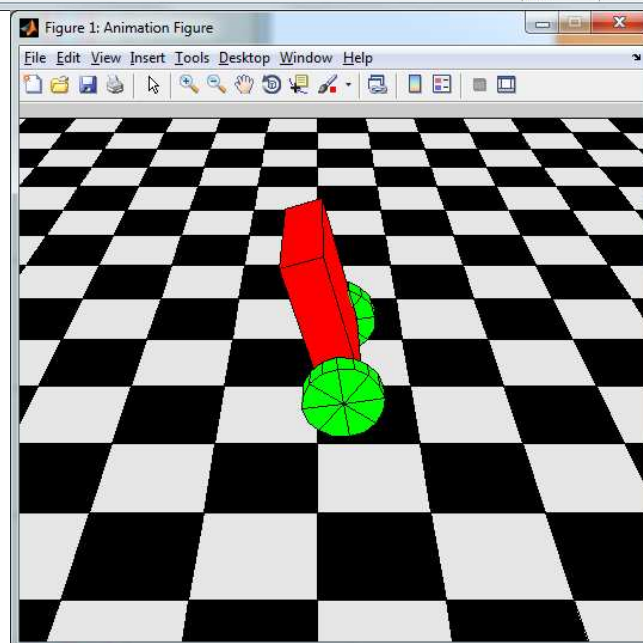
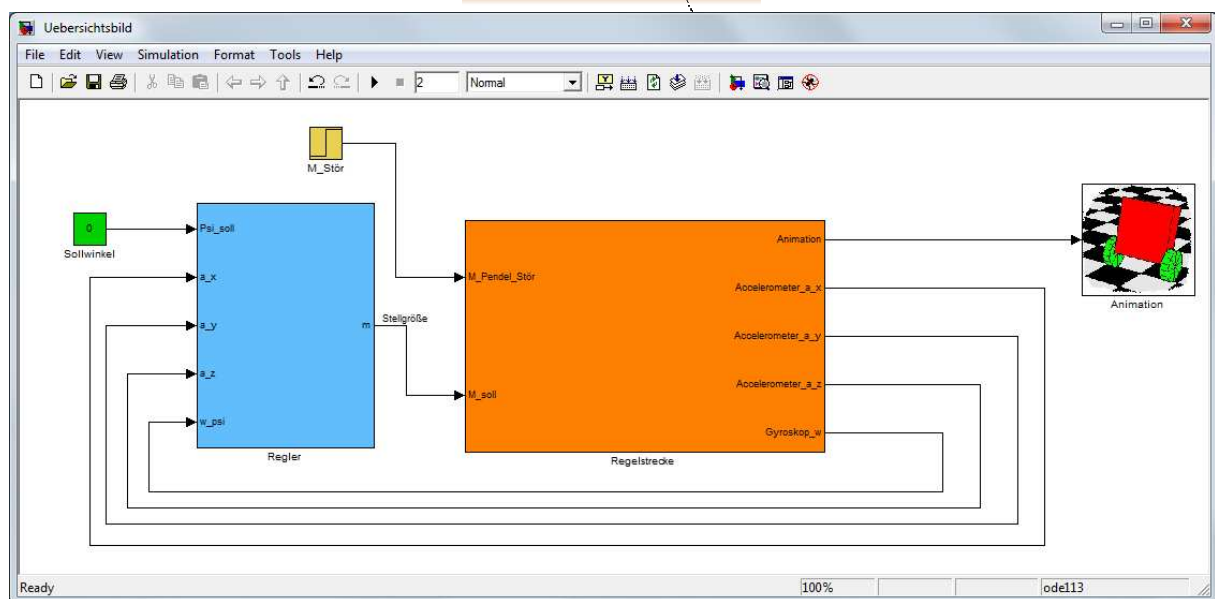
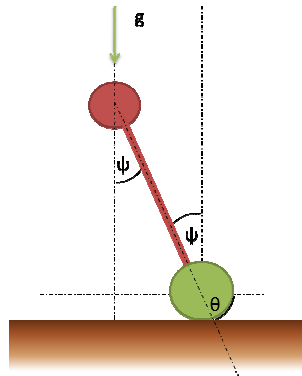
Bei der Bewertung der Beobachtbarkeit des berechneten Modells stellte sich heraus, dass nur der Neigungswinkel und dessen Winkelgeschwindigkeit mit den vorgesehenen Sensoren beobachtet und somit geregelt werden können. Das Modell wurde linearisiert und vereinfacht. An Hand der resultierenden Gleichung wurden die Regelungsparameter für die Ausregelzeiten 10, 50 und 100ms berechnet und simuliert. Schließlich wurde jede dieser Konfigurationen unter Einwirkung einer Störgröße simuliert.

Inhaltsverzeichnis

1	Projektübersichtsbild.....	1
2	Analyse.....	2
2.1	Motivation	2
2.2	Das physikalische Modell.....	3
2.2.1	Physikalische Grundlagen	4
2.2.2	Lagrange Mechanik.....	5
2.3	Regelungstechnik.....	8
2.3.1	Zustandsraumdarstellung	8
2.3.2	Rang einer Matrix.....	10
2.3.3	Steuerbarkeit.....	10
2.3.4	Zustandsregler	11
2.3.5	Zustandsbeobachter	14
2.4	MATLAB	17
2.4.1	Erstellen eines Simulink Modells mit Subsystemen	17
2.4.2	S-Function	21
3	Darstellung des Sollkonzepts	24
3.1	Die Regelstrecke.....	25
3.1.1	Der Motor.....	25
3.1.2	Die Mechanik des Pendels	25
3.1.3	Die Störgröße	25
3.1.4	Der Animationsvektor	25
3.1.5	Beschleunigungssensor	25
3.1.6	Gyroskop	25
3.2	Die Animation	26
3.3	Der Regler.....	26
4	Beschreibung der Realisierung.....	27
4.1	Die Animation	27
4.1.1	Die Eingabemaske der Animation.....	27
4.1.2	Die S-Funktion der Animation.....	28
4.1.3	Die Real-time Synchronisation	36
4.1.4	Parameter der Animation	37
4.1.5	Test der Animation.....	37
4.2	Das physikalische Modell.....	38
4.2.1	Berechnen der Bewegungsgleichungen	39
4.3	Das MATLAB-Simulink Modell.....	44
4.3.1	Der Motor	45
4.3.2	Das Pendel.....	46
4.3.3	Der Beschleunigungssensor	48
4.3.4	Das Gyroskop.....	50
4.3.5	Parameter der Regelstrecke.....	51
4.4	Die Regelung mit MATLAB	52
4.4.1	Feststellen der Steuerbarkeit	53
4.4.2	Feststellen der Beobachtbarkeit	54
4.4.3	Berechnen des Neigungswinkels aus den Beschleunigungen.....	55
4.4.4	Sensorfusion	60
4.4.5	Auslegen des Zustandsreglers	61

4.4.6	Die Parameter des Reglers	68
5	Ergebnisbewertung und Ausblick	69
6	Anhang:	74
6.1	Blockschaltbild der Regelstrecke	74
6.2	Blockschaltbild der Regelstrecke (ohne algebraische Schleife).....	75

1 Projektübersichtsbild



2 Analyse

Um die Studienarbeit durchzuführen ist die Einarbeitung in diverse Themengebiete notwendig. In diesem Kapitel wird auf die Motivation und die notwendigen Grundlagen eingegangen.

2.1 Motivation

In seiner Studienarbeit hat Herr Jean Khoury mit der Entwicklung eines mobilen, stehenden Pendels begonnen und hierfür die Mechanik, Elektronik und ein Betriebssystem erstellt.

Das mobile, stehende Pendel soll nach Fertigstellung als Anschauungsobjekt an Veranstaltungen und Präsentationen der Hochschule Esslingen gezeigt werden. Es soll dazu dienen potentiellen Studenten die Technische Informatik in einem praktischen, wie auch anschaulichen, Beispiel zu zeigen, und so motivieren, ein solches Studium an der Hochschule Esslingen zu absolvieren.

Ferner soll mit dieser Studienarbeit die Modellbasierte Entwicklung und somit der aktuelle Trend in der Industrie schnell und kostengünstig zu entwickeln dargestellt werden.

2.2 Das physikalische Modell

Das inverse Pendel wird in verschiedenen Publikationen, Skripten und Büchern beschrieben [4, 5, 6]. Jedoch wird meist die Variante mit Wagen und Stab gewählt.

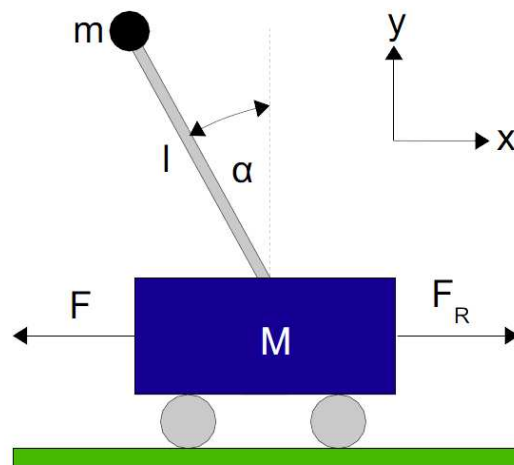


Abbildung 1: Modell des inversen Pendels mit Wagen und Stab [6]

Bei dieser Variante ist die Masse m über einen dünnen, idealerweise massefreien Stab mit einem idealerweise reibungsfreien Gelenk verbunden. Am Wagen wird entweder über die Räder oder über links und rechts angebrachte Seilzüge eine Kraft F auf den Wagen ausgeübt. Fällt das Pendel nach links oder rechts um, so kann der Wagen M über die Kraft F wieder unter das Pendel geschoben werden. Es entsteht der gleiche Effekt, wie wenn man einen Besenstiel auf dem Finger balanciert.

Die in der Studienarbeit betrachtete Variante unterscheidet sich jedoch von der mit Wagen und Stab dadurch, dass der Stab statt über ein Gelenk direkt über Elektromotoren an den Rädern befestigt ist.

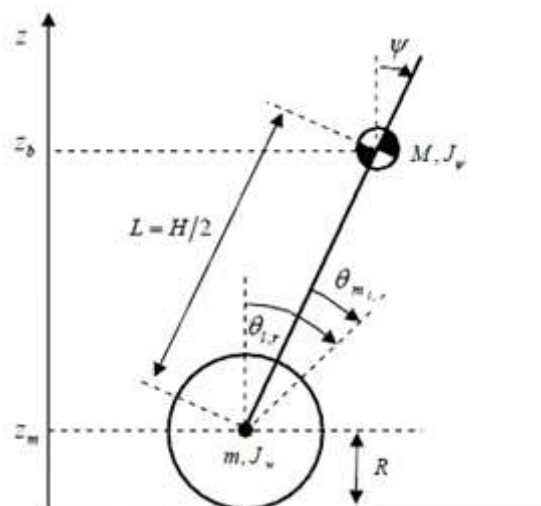


Abbildung 2: Mobiles, inverses Pendel [5]

Der wesentliche Unterschied ist somit, dass hier beim einbringen einer Kraft, bzw. in diesem Fall eines Momentes, dieses auf das Pendel und auf die Räder wirkt, statt ausschließlich auf den Wagen.

2.2.1 Physikalische Grundlagen

In diesem Kapitel soll auf die notwendigen Grundlagen, sowie auf die Lösung von Bewegungsgleichungen mit Hilfe der Lagrange-Mechanik eingegangen werden.

2.2.1.1 Massenträgheitsmoment

Bei der Längsbewegung (Translation) muss eine Beschleunigung gegen die Masse wirken, um deren Geschwindigkeit zu ändern. Bei der Rotation ist hierfür nicht nur die Masse, sondern auch noch der Abstand vom Drehpunkt von Bedeutung. Die Masse wird bei der Rotation zu einem Trägheitsmoment. (m = Masse in kg, r = Abstand vom Drehpunkt in m)

$$J = m * r^2 \quad (2.2.1)$$

Massenträgheitsmomente für Punktmassen können über die oben genannte Formel berechnet werden. Ferner kann jedes Objekt in Punktmassen zerlegt werden und über die Summe das Massenträgheitsmoment des Objekts berechnet werden. Für grundlegende geometrische Körper existieren zu diesem Zweck bereits vorgefertigte Formeln.

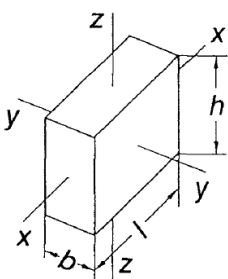
	Quader	$J_x = \frac{1}{12} m (b^2 + h^2)$ $J_y = \frac{1}{12} m (l^2 + h^2)$ $J_z = \frac{1}{12} m (l^2 + b^2)$
--	--------	--

Abbildung 3: Beispiel für quaderförmige Objekte mit homogener Masseverteilung [1]

Die Formeln sind für den Schwerpunkt des Objektes berechnet. Wird das Objekt um einen anderen Punkt gedreht, so muss der Satz von Steiner angewendet werden. Dieser fügt dem Trägheitsmoment des Objektes noch das entstehende Massenträgheitsmoment durch den Abstand vom Drehpunkt hinzu.

Das Massenträgheitsmoment wird benötigt, da bei der Lagrange-Mechanik mit Punktmassen gerechnet wird. Erst durch hinzufügen der Trägheitsmomente der bewegten Körper erreicht man Bewegungsgleichungen, welche die Form des mechanischen Modells berücksichtigen.

2.2.1.2 Kinetische Energie

Als kinetische Energie wird die Energie bezeichnet, welche in der Masse eines Körpers in Form einer Bewegung gespeichert ist. Die gesamte kinetische Energie eines Körpers lässt sich in Translations- und Rotationsbewegung aufteilen. Diese können wie folgt berechnet werden. (m = Masse in kg, v = Geschwindigkeit in m/s, J = Massenträgheitsmoment in kg m², ω = Winkelgeschwindigkeit in rad/s):

$$E_{Kin_{ges}} = E_{Kin_{trans}} + E_{Kin_{rot}} \quad (2.2.2)$$

$$E_{Kin_{trans}} = \frac{1}{2} m v^2 \quad (2.2.3)$$

$$E_{Kin_{rot}} = \frac{1}{2} J \omega^2 \quad (2.2.4)$$

2.2.1.3 Potentielle Energie

Die potenzielle Energie bezeichnet die Lageenergie eines Körpers und wird nahe der Erdoberfläche mit folgender Formel berechnet. (m = Masse in kg, g = Erdbeschleunigung = $9,81\text{m/s}^2$, h = Hub in m)

$$E_{\text{Pot}} = m g h \quad (2.2.5)$$

2.2.2 Lagrange-Mechanik

In sämtlichen Dokumentationen zu inversen Pendeln werden die physikalischen Bewegungsgleichungen über die Lagrange-Mechanik ermittelt.

Die Vorgehensweise, um mit Hilfe der Lagrange-Mechanik an die Bewegungsgleichungen zu gelangen, wird hier an dem Beispiel der Hangabtriebskraft beschrieben. Sie hat die bekannte Lösung: $F_H = m g \sin \varphi$

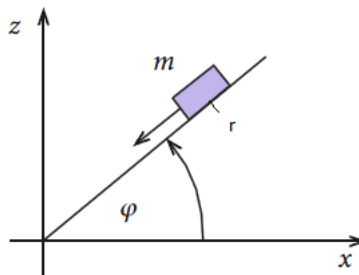


Abbildung 4: Beispiel Hangabtriebskraft [2]

2.2.2.1 Die Zwangsbedingungen des Systems

Zwangsbedingungen sind Einschränkungen in der Bewegung eines Massepunktes. Im Beispiel kann die Masse m als Massepunkt betrachtet werden. Ein Massepunkt hat 3 Freiheitsgrade er kann sich im dreidimensionalen Raum in jede Richtung bewegen. Im Beispiel existiert jedoch zum Beispiel die Y-Richtung nicht, somit gilt:

$$y = 0 \quad (2.2.6)$$

Ferner kann sich die Masse m nur auf der Ursprungsgeraden bewegen. Der Abstand vom Ursprung wird mit r bezeichnet. Es gilt:

$$x = r \cos \varphi \quad (2.2.7)$$

$$y = r \sin \varphi \quad (2.2.8)$$

Die letzten beiden Gleichungen beschreiben zusammen eine Zwangsbedingung. Durch eine Zwangsbedingung wird der Freiheitsgrad eines Systems um 1 reduziert. Im Beispiel besitzt die Masse m nur noch einen Freiheitsgrad, da die ursprünglichen 3 Freiheitsgrade um 2 Zwangsbedingungen 2.2.6 und 2.2.7/2.2.8 reduziert wurden. Es bleibt der Abstand vom Ursprung r als einziger Freiheitsgrad.

2.2.2.2 Die generalisierten Koordinaten

Unter generalisierten Koordinaten versteht man die Größen, mit denen unter Berücksichtigung der Zwangsbedingungen alle anderen Größen des Systems beschrieben werden können. Im Beispiel ist der Abstand r eine generalisierte Koordinate und auf Grund der Einfachheit des Systems auch die einzige. Da außerdem $y = 0$ und $\varphi = \text{const.}$ gilt, können die Koordinaten x , y und z alleine aus r berechnet werden.

2.2.2.3 Die Lagrange-Funktion

Die Lagrange-Funktion basiert auf allen kinetischen Energien, also der Summe aller Rotationsenergien T_2 und Translationsenergien T_1 , abzüglich aller potentieller Energien V .

$$L_{\text{Lagrange}} = T_1 + T_2 - V \quad (2.2.9)$$

Im Beispiel bildet sich die Lagrange-Funktion aus der kinetischen Energie in der Masse m und der potentiellen Energie der Masse m mit der Höhe z .

$$L_{\text{Lagrange}} = \frac{1}{2} m \dot{r}^2 - m g r \sin \varphi \quad (2.2.10)$$

2.2.2.4 Die Bewegungsgleichungen

Um aus der Lagrange-Funktion die Bewegungsgleichungen der generalisierten Koordinaten zu berechnen, muss die Lagrange-Gleichung 2. Art angewendet werden.

$$\frac{d}{dt} \frac{\partial L_{\text{Lagrange}}}{\partial \dot{q}_i} - \frac{\partial L_{\text{Lagrange}}}{\partial q_i} \quad (2.2.11)$$

Die generalisierte Koordinate q_i steht für die i 'te generalisierte Koordinate. Im Beispiel berechnet sich die Bewegungsgleichung für r wie folgt:

$$\frac{\partial L_{\text{Lagrange}}}{\partial \dot{r}} = m \dot{r} \quad (2.2.12)$$

Die Gleichung 2.2.12 ist die partielle Ableitung nach der ersten Ableitung der Koordinate r .

$$\frac{d}{dt} \frac{\partial L_{\text{Lagrange}}}{\partial \dot{r}} = m \ddot{r} \quad (2.2.13)$$

Die Gleichung 2.2.13 ist die Ableitung von 2.2.12 nach der Zeit.

$$\frac{\partial L_{\text{Lagrange}}}{\partial r} = -m g \sin \varphi \quad (2.2.14)$$

Die Gleichung 2.2.14 ist die partielle Ableitung von 2.2.10 nach r .

$$\frac{d}{dt} \frac{\partial L_{\text{Lagrange}}}{\partial \dot{r}} - \frac{\partial L_{\text{Lagrange}}}{\partial r} = m \ddot{r} - (-m g \sin \varphi) = 0 \quad (2.2.15)$$

$$m \ddot{r} = -m g \sin \varphi \quad (2.2.16)$$

Die Gleichung 2.2.16 ist Gleichung 2.2.15 umgestellt nach $m \ddot{r}$. Diese Kraft zeigt vom Ursprung weg und ist deshalb negativ. Die Masse m beschleunigt wie nicht anders erwartet Richtung Ursprung und ist somit äquivalent zu der bekannten Lösung. Der Aufwand, auf diesem Weg die Hangabtriebskraft zu berechnen, ist unverhältnismäßig größer als mit Hilfe der Newton-Mechanik. Steigt jedoch die Komplexität des Systems, wie zum Beispiel bei einem inversen Pendel, so bietet die Lagrange-Mechanik einen brauchbaren Ansatz.

Beinhaltet das System mehrere generalisierte Koordinaten, wie zum Beispiel bei dem inversen Pendel, so müssen die Schritte von Kapitel 2.2.2.4 für jede dieser Koordinaten durchgeführt werden, um alle Bewegungsgleichungen zu erhalten.

2.3 Regelungstechnik

Für die Regelung des mobilen, inversen Pendels sind Grundlagen, wie zum Beispiel Methoden, eine Strecke auf Steuerbarkeit und Beobachtbarkeit zu prüfen, sowie Methoden, einen Zustandsregler auszulegen notwendig. Auf genau diese Grundlagen soll in diesem Kapitel eingegangen werden.

SISO (<i>Single Input Single Output</i>)	Ein System mit einem Ein- und einem Ausgang
MISO (<i>Multiple Input Single Output</i>)	Ein System mit mehreren Eingängen aber nur einem Ausgang
SIMO (<i>Single Input Multiple Output</i>)	Ein System mit einem Eingang aber mehreren Ausgängen
MIMO (<i>Multiple Input Multiple Output</i>)	Ein System mit mehreren Ein- und mehreren Ausgängen
Zustand	Mit einem Zustand wird bei einem Modell der Ausgang eines Speichers, also in der Regel der Ausgang eines Integrators beschrieben.

Tabelle 1: Verwendete Begriffe und Abkürzungen

2.3.1 Zustandsraumdarstellung

In der klassischen Regelungstechnik werden in der Regel SISO-Systeme behandelt und über lineare Übertragungsfunktionen im Laplace-Raum beschrieben.

$$G(s) = \frac{s^2 - 2s + 1}{s^3 + 2s^2 + 3s + 1} \quad (2.3.1)$$

In der modernen Regelungstechnik werden auch MIMO-Systeme behandelt, hier reicht die Beschreibung mit Übertragungsfunktionen nicht aus. Über die Zustandsraumdarstellung kann das System dagegen mit wenigen Angaben vollständig beschrieben werden. Ferner kann über die Zustandsraumdarstellung das Problem im anschaulicheren Zeitbereich statt im Frequenzbereich betrachtet werden. Auch für die spätere Implementierung in einen Mikrocontroller bietet die Zustandsraumdarstellung die Grundlage für den notwendigen Algorithmus.

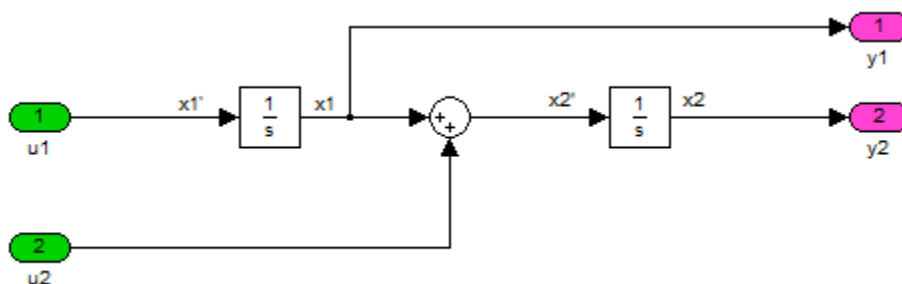


Abbildung 5: Einfaches Beispiel für ein MIMO-System

Das System in Abbildung 5 kann über 4 Übertragungsfunktionen ($u_1 \rightarrow y_1$, $u_1 \rightarrow y_2$, $u_2 \rightarrow y_1$, $u_2 \rightarrow y_2$) beschrieben und mit Hilfe des Überlagerungssatzes berechnet werden. Einfacher kann jedoch die Zustandsraumdarstellung angefertigt werden.

Hierzu wird die Differenzialgleichung für jeden Zustand (x_1, x_2) aus dem Blockschaltbild abgelesen. Vor jedem Integrator befindet sich dafür die erste Ableitung des entsprechenden Zustandes. Im Beispiel wird x_1' nur durch u_1 bestimmt, hingegen x_2' aus x_1 plus u_2 . Um alle Gleichungen in Abhängigkeit aller Zustände und Eingänge darzustellen, werden die resultierenden Gleichungen um diese mit jeweils dem Faktor 0 erweitert. Anschließend werden diese nach den Zuständen und anschließend den Eingängen sortiert.

$$x_1' = 0 x_1 + 0 x_2 + 1 u_1 + 0 u_2 \quad (2.3.2)$$

$$x_2' = 1 x_1 + 0 x_2 + 0 u_1 + 1 u_2 \quad (2.3.3)$$

Wie bereits die Gleichungen für die Zustände aus dem Blockschaltbild gelesen wurden, können auch die Gleichungen für die Ausgänge (y_1, y_2) herausgelesen werden.

$$y_1 = 1 x_1 + 0 x_2 + 0 u_1 + 0 u_2 \quad (2.3.4)$$

$$y_2 = 0 x_1 + 1 x_2 + 0 u_1 + 0 u_2 \quad (2.3.5)$$

Die 4 Gleichungen des Beispiels stellen nun ein Gleichungssystem dar:

	x_1	x_2	u_1	u_2
x_1'	0	0	1	0
x_2'	1	0	0	1
y_1	1	0	0	0
y_2	0	1	0	0

Aus dem Gleichungssystem können die 4 Zustandsmatrizen abgelesen werden:

A Systemmatrix

Sie beschreibt, welche Zustände von welchen Zuständen wie abhängen. Die Eigenwerte dieser Matrix sind die Polstellen des Systems.

B Steuermatrix

Sie beschreibt, wie die Eingänge auf welche Zustände wirken.

C Beobachtungsmatrix

Sie beschreibt, welche Zustände wie auf die Ausgänge wirken.

D Durchgangsmatrix

Sie beschreibt, welche Eingänge direkt auf die Ausgänge wirken.

Mit MATLAB kann über den Befehl *linmod* aus einem mit Simulink erstellten Blockschaltbild genau diese Matrizen erzeugt werden. Handelt es sich hierbei um ein nicht lineares Modell, so wird dieses zusätzlich am Arbeitspunkt linearisiert.

2.3.2 Rang einer Matrix

Für den Nachweis von Steuerbarkeit und Beobachtbarkeit muss der Rang einer Matrix gebildet werden. Dieser Rang beschreibt die Anzahl verschiedener Zeilen im Bezug darauf, dass eine Zeile nicht durch die Summe einer oder mehrerer anderer Zeilen bzw. durch deren Vielfache gebildet werden kann. Eine Methode, den Rang zu berechnen, ist die Matrix nach dem Gauß'schen Eliminationsverfahren in eine Stufenform zu bringen. Kann dieses Verfahren erfolgreich durchgeführt werden, also die Dreiecksmatrix gebildet werden, so hat die Matrix den vollen Rang, sonst hat die Matrix den Rang der Anzahl an Zeilenvektoren, welche ungleich 0 sind.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 5 & 4 \\ 0 & 10 & 2 \end{pmatrix} \sim \begin{pmatrix} 1 & 2 & 3 \\ 0 & 5 & 4 \\ 0 & 0 & -6 \end{pmatrix} \Rightarrow \text{rang}(A) = 3$$

$$B = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 6 & 4 \\ 0 & 3 & 2 \end{pmatrix} \sim \begin{pmatrix} 1 & 2 & 3 \\ 0 & 6 & 4 \\ 0 & 0 & 0 \end{pmatrix} \Rightarrow \text{rang}(B) = 2$$

Abbildung 6: Beispiel Rangbestimmung

Mit MATLAB kann diese Berechnung mit dem Befehl `rank(B)` durchgeführt werden.

2.3.3 Steuerbarkeit

Bevor man einen Regler für ein System auslegt, muss nachgewiesen werden, dass das System steuerbar ist. Das heißt, dass mit den vorhandenen Eingängen jeder Zustand des Systems in endlicher Zeit erreicht werden kann. Ist dies der Fall, ist das System vollständig steuerbar. Nachweisen kann man dies durch berechnen der Steuerbarkeitsmatrix [4].

$$S_s = (B \ AB \ A^2B \ \dots \ A^{n-1}B) \quad (2.3.6)$$

Hat S_s den vollen Rang, so ist das System steuerbar. Wurde das System ungünstig modelliert, kann es dazu kommen, dass ein eigentlich steuerbares System nicht den vollen Rang erreicht. Dies ist zum Beispiel der Fall, wenn aus der Winkelgeschwindigkeit des Drehwinkels eines Rades über jeweils einen eigenen Integrator der Ort und der Drehwinkel berechnet werden. Jeder der beiden Integratoren stellt einen Zustand dar, jedoch kann kein Winkel unabhängig vom Ort angesteuert werden. Die Steuerbarkeitsmatrix wird deshalb nicht den vollen Rang besitzen. Um den Ort richtig zu modellieren sollte er direkt über den Radius des Rades aus dem Drehwinkel des Rades berechnet werden. Durch diese Änderung stellt der Ort keinen eigenen Zustand dar und der volle Rang der Steuerbarkeitsmatrix wird erreicht.

Mit MATLAB kann die Steuerbarkeitsmatrix mit dem Befehl `ctrb(A,B)` berechnet werden, wobei A die Systemmatrix und B die Steuermatrix darstellt. In der MATLAB-Hilfe[8.1] wird jedoch von der Verwendung dieses Verfahrens abgeraten, da es numerisch nicht stabil ist. Es wird geraten den Befehl `ctrbf(A, B, C)` zu verwenden.

2.3.4 Zustandsregler

Der Zustandsregler ist ein für jeden Zustand unterlagerter P-Regler. Da jeder Zustand zum Eingang zurückgekoppelt wird, kann durch die verwendeten Faktoren jede Kombination von Polstellen erreicht werden. Es wird den Entwickler ermöglicht eigeninstabile Systeme durch Polvorgabe zu stabilisieren. Um den Zustandsregler auszulegen wird die Regelstrecke oft in Regelungsnormform dargestellt und hierfür der Zustandsregler berechnet. Der Regler kann jedoch für jede Darstellung des Systems ausgelegt werden, was in den folgenden Kapitel gezeigt wird.

2.3.4.1 Zustandsregler zur Regelungsnormform

Eine Übertragungsfunktion wie zum Beispiel Gleichung 2.3.7 mit den Faktoren a_0 bis a_n bestimmt die Stabilität des Systems.

$$G_S = \frac{b_3 s^3 + b_2 s^2 + b_1 s + b_0}{s^3 + a_2 s^2 + a_1 s + a_0} \quad (2.3.7)$$

Die Nullstellen des Nennerpolynoms, müssen alle negativ sein. Ein eigeninstabiles System hat jedoch auch positive Nullstellen, was einen Betrieb ohne Regelung verhindert. Die Gleichung 2.3.1 kann nach dem Schema von Abbildung 7 in der Regelungsnormform dargestellt werden.

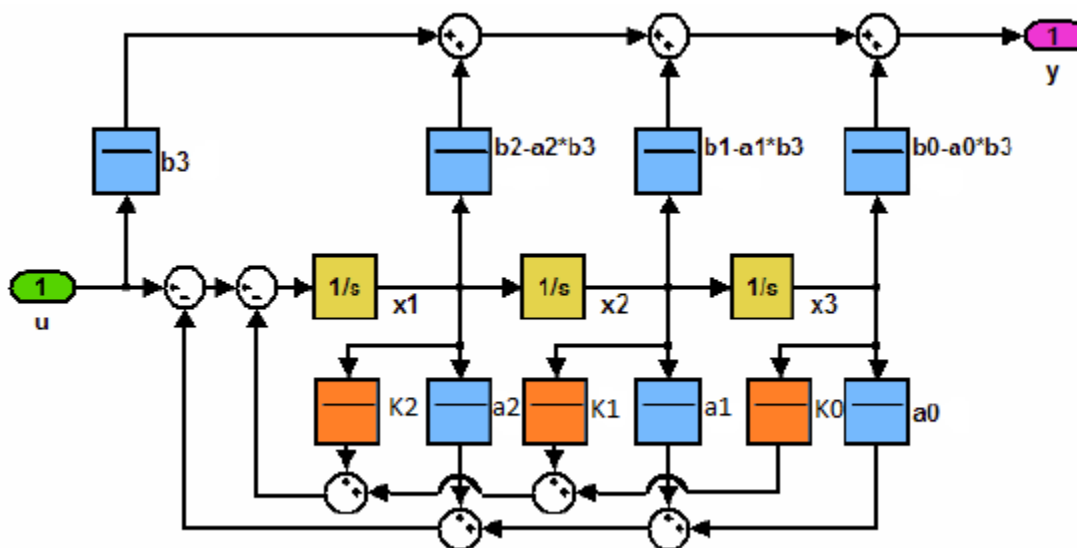


Abbildung 7: Blockschaltbild Regelungsnormform (allgemeiner Fall: $m = n$) [9]

Koppelt man über die Faktoren K_0 bis K_n jeden Zustand zum Eingang zurück, erhält man die Gleichung 2.3.8.

$$G_S = \frac{b_3 s^3 + b_2 s^2 + b_1 s + b_0}{s^3 + (K_2 + a_2) s^2 + (K_1 + a_1) s + (K_0 + a_0)} \quad (2.3.8)$$

Wie man sehen kann erhält nun jeder Faktor a_0 bis a_n einen eigenen Faktor K_0 bis K_n mit dem nun das Nennerpolynom zu jedem beliebigen Polynom gleicher Ordnung verändert werden kann. Hierzu wählt man Wunschpole, errechnet das Wunschpolynom und durch Koeffizientenvergleich die Faktoren K_0 bis K_n .

2.3.4.2 Zustandsregler zu jedem Modell

Die Zustände in der Regelungsnormalform entsprechen nicht zwingend einer physikalischen Größe des Systems. Deshalb haben diese Zustände für einen Betrachter wenig Aussagekraft. Man kann ein System auf verschiedenste Arten darstellen und das gleiche Übertragungsverhalten erreichen. Das für den Betrachter geläufigste Modell ist das physikalische Modell. Hier stellen die Zustände echte physikalische Größen wie Ort, Winkel, Geschwindigkeit, etc. dar. Auch an diesem Modell kann der Zustandsregler realisiert werden, indem die dort berechneten Zustände zum Eingang zurückgekoppelt werden. Der Preis hierfür ist jedoch, dass nur in der Regelungsnormalform die Faktoren a_0 bis a_n und K_0 bis K_n paarweise auftreten und deshalb für alle anderen Modelle die Konstellationen dieser Faktoren ausgerechnet werden müssen. Ein Beispiel hierfür zeigt Abbildung 8.

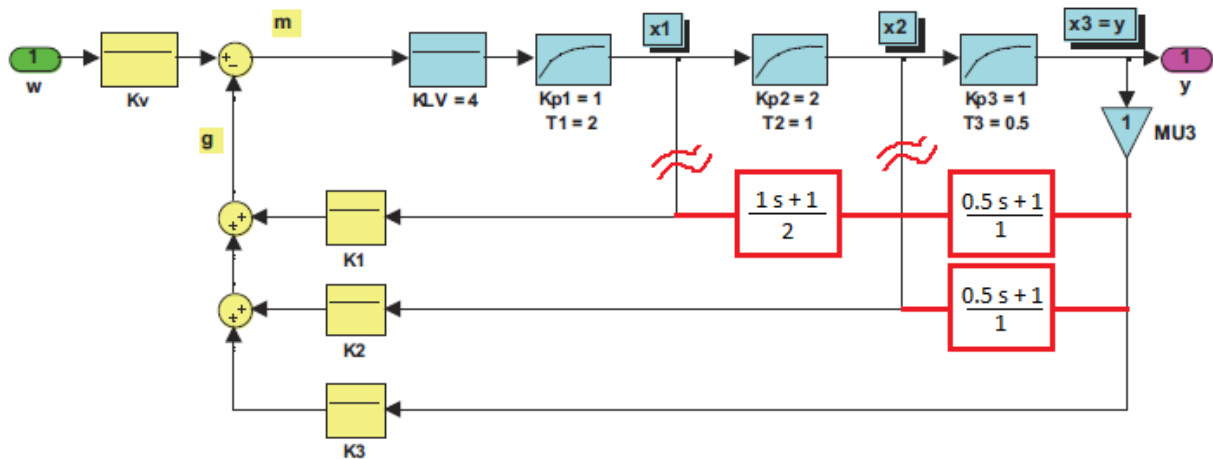


Abbildung 8: Beispiel allgemeines Blockschaltbild mit Zustandsrückführung [9]

Der Ausgang jedes PT1-Elementes wird über einen Faktor zurückgekoppelt. Der so entstehende geschlossene Regelkreis bzw. dessen Übertragungsfunktion muss zur Bestimmung der Faktoren K_1 bis K_3 neu berechnet werden. Für die Berechnung werden Rückkoppelstellen zum Ausgang verschoben und die Umkehrfunktion für die übersprungenen Blöcke hinzugefügt. Somit werden alle Rückkopplungen zu einer einzigen zusammengefasst.

$$G(s) = \frac{y(s)}{w(s)} = \frac{K_v K_{lv} \frac{1}{2s+1} \frac{2}{1s+1} \frac{1}{0.5s+1}}{1 + K_{lv} \frac{1}{2s+1} \frac{2}{1s+1} \frac{1}{0.5s+1} \left(\frac{K_1}{2} (1s+1)(0.5s+1) + K_2(0.5s+1) + K_3 \right)} \quad (2.3.9)$$

Division von Nenner und Zähler durch $\frac{1}{2s+1} \frac{2}{1s+1} \frac{1}{0.5s+1}$ ergibt:

$$G(s) = \frac{K_v K_{lv}}{(2s+1) \frac{1}{2} (1s+1) \left(\frac{1}{2} s + 1 \right) + K_{lv} \left(\frac{K_1}{2} (1s+1) \left(\frac{1}{2} s + 1 \right) + K_2 \left(\frac{1}{2} s + 1 \right) + K_3 \right)} \quad (2.3.10)$$

Den Nenner ausmultipliziert ergibt:

$$G(s) = \frac{2K_v K_{lv}}{s^3 + \left(\frac{K_1 K_{lv}}{2} + \frac{7}{2} \right) s^2 + \left(K_{lv} \left(\frac{3}{2} K_1 + K_2 \right) + \frac{7}{2} \right) s + 2K_{lv} \left(\frac{K_1}{2} + K_2 + K_3 \right) + 1} \quad (2.3.11)$$

Wie man in Gleichung 2.3.11 sehen kann, so sind die Faktoren des Nennerpolynoms jeweils von mehreren Rückkoppelfaktoren abhängig. Dennoch können auch bei der resultierenden Übertragungsfunktion über den Koeffizientenvergleich die Wunschpole vorgegeben werden.

Ein weiterer Vorteil dieser Variante, den Zustandsregler auszulegen, besteht darin, dass die über Sensoren gemessenen Zustände den Zuständen entsprechen, welche in der Regelung verwendet werden. Eine Umrechnung der gemessenen Zustände über eine Transformationsmatrix in die Zustände der Regelungsnormalform ist somit nicht notwendig.

Mit MATLAB können die Faktoren K mit dem Befehl `place(A, B, p)` berechnet werden [8.2]. A ist die Systemmatrix, B die Steuermatrix und p ein Vektor mit den Wunschknoten (z.B. $p = [-1 \ -2 \ -3]$). Dieser Befehl kann auch auf MIMO-Systeme angewendet werden, unter der Voraussetzung, dass der Rang der Steuermatrix der Größe der Systemmatrix entspricht. Von dem ebenfalls verfügbaren Befehl `acker` wird von MATLAB abgeraten, da er nur für SISO-Systeme ausgelegt und numerisch nicht sehr stabil ist.

2.3.4.3 Problem der bleibenden Regelabweichung

Der Zustandsregler ist vom Prinzip her ein für jede Zustandsgröße unterlagerter P-Regler. Daher hat er auch das für P-Regler übliche Problem der bleibenden Regelabweichung, wenn die Sollgröße ungleich 0 ist.

Um dieser Tatsache entgegen zu wirken, wird dem Sollwert ein Verstärkungsfaktor nachgeschaltet. Dieser Faktor, in der Literatur Vorfilter genannt [4], hat die Aufgabe, für einen Arbeitspunkt die Verstärkung des Gleichanteils des Regelkreises auf 1 bringen, also den Sollwert so anzupassen, dass dieser trotz bleibender Regelabweichung erreicht wird. Der notwendige Wert für K_v lässt sich berechnen, nachdem die Rückkoppelfaktoren berechnet und hinzugefügt wurden und K_v auf 1 gesetzt wurde. Über den MATLAB Befehl `dcbain(sys)` kann dann die Verstärkung des Gleichanteils berechnet werden [8.3], wobei `sys` die Übertragungsfunktion von Stellgrößeneingang zu der zu regelnden Ausgangsgröße darstellt. Das notwendige K_v ist der Kehrwert des Funktionsaufrufs, es kompensiert so Verstärkungen oder Dämpfungen des Sollwertes auf 1. Diese Kompensation funktioniert jedoch nur an einem Arbeitspunkt und führt bei Störgrößeneinflüssen zu Abweichungen vom Sollwert.

Eine Verbesserung ist es, der Verstärkung K_v einen I-Anteil parallel zu schalten, um die Regelabweichung zu kompensieren [9]. Die Zeitkonstante muss jedoch deutlich langsamer als der Regler selbst ausgelegt werden, da er nur die bleibende Regelabweichung über die Zeit kompensieren soll. Eine weitere Möglichkeit stellt der PI-Zustandsregler dar, welcher in dieser Studienarbeit jedoch nicht weiter behandelt wird.

2.3.5 Zustandsbeobachter

Da bei einem Zustandsregler alle Zustände zurückgekoppelt werden müssen und diese oft nicht messbar oder die notwendigen Sensoren zu teuer sind, müssen die Zustände auf einem anderen Weg ermittelt werden. Ein Zustandsbeobachter ist ein solcher Weg. Er beinhaltet ein Modell des Prozesses und lässt dieses quasi parallel zum realen Prozess laufen, das heißt das Modell erhält die gleichen Stellgrößen wie der reale Prozess und errechnet die Zustandsänderungen. Der Ausgang des Modells und der gemessene Ausgang des Prozesses können so verglichen und die Differenz negativ, mit einem bestimmten Faktor, auf die Eingänge der Integratoren für die Zustände zurückgekoppelt werden. Solange der Ausgang des Modells vom dem des echten Prozesses abweicht, werden so alle Zustände korrigiert. Durch diese Korrektur erhält man zusätzlich zum Ausgang des realen Prozesses die inneren Zustände, welche für den Zustandsregler verwendet werden können.

2.3.5.1 Beobachtbarkeit

Leider kann ein Zustandsbeobachter nicht auf jeden Prozess angewendet werden. Ein Prozess muss hierfür beobachtbar sein. Das heißt die gemessene Größe am Ausgang muss auch von der zu bestimmenden Zustandsgröße abhängig sein. Um dies festzustellen kann die Beobachtbarkeitsmatrix aufgestellt werden. Hat diese Matrix den vollen Rang, wird das Kalman-Kriterium für die Beobachtbarkeit erfüllt. Die Beobachtbarkeitsmatrix wird wie folgt gebildet [4]:

$$S_B = \begin{pmatrix} C \\ C A \\ C A^2 \\ \vdots \\ C A^{n-1} \end{pmatrix} \quad (2.3.12)$$

Aus folgenden Gründen kann bei einem System die Beobachtungsmatrix nicht vollständig und somit der Prozess nicht beobachtbar sein [4]:

- Eigenvorgänge des Prozesses die nicht mit dem Ausgang verbunden sind können nicht beobachtet werden.
- Zwei parallele Teilsysteme mit denselben dynamischen Eigenschaften sind nicht vollständig beobachtbar.
- Lassen sich in der Übertragungsfunktion eines Eingrößensystems Pole gegen Nullstellen kürzen, so kann man die mit den Polen verbundenen Eigenvorgänge nicht beobachten (bzw. steuern)

In MATLAB kann diese Matrix durch den Befehl `obsv(A,C)` realisiert werden, wobei A die Systemmatrix und C die Beobachtungsmatrix ist. Laut der MATLAB-Hilfe reicht dieser Test jedoch nicht aus, um die Beobachtbarkeit sicher auszuschließen und begründet dies mit numerischen Problemen bezüglich Ungenauigkeiten.

Zitat MATLAB-Hilfe [8.4]: „*obsv is here for educational purposes and is not recommended for serious control design.*”

Die MATLAB-Hilfe verweist deshalb auf einen erweiterten, aussagekräftigeren Befehl *obsvf(A,B,C)*, wobei A die Systemmatrix, B die Steuermatrix und C die Beobachtungsmatrix ist.

2.3.5.2 Zustandsbeobachter in Beobachternormalform

Auch für den Zustandsbeobachter gibt es, wie für den Zustandsregler, eine besondere Darstellungsform, mit welcher die Rückkoppelfaktoren für den Zustandsbeobachter leicht berechnet und somit die Wirkungsweise ersichtlich ist.

$$G_s = \frac{b_0}{s^3 + a_2 s^2 + a_1 s + a_0} \quad (2.3.13)$$

Eine Übertragungsfunktion wie zum Beispiel Gleichung 2.3.13 kann in der Form von Abbildung 9 dargestellt werden.

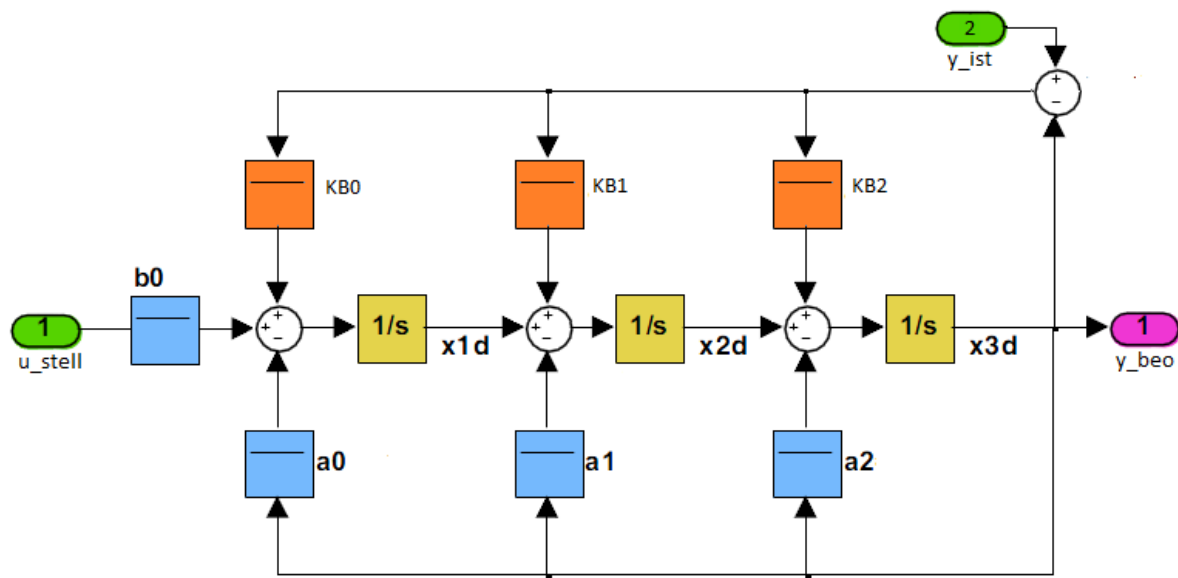


Abbildung 9 : Zustandsbeobachter in Beobachternormalform [9]

Der in Abbildung 9 gezeigte Zustandsbeobachter wird mit dem Eingang u_{stell} , dem Stellwert des Zustandsreglers, verbunden. Der zweite Eingang y_{ist} ist der Ausgang des realen Prozesses. Der Ausgang y_{beo} ist die beobachtete Regelgröße und wird von y_{ist} abgezogen. Die resultierende Differenz wird vor jeden Zustand, über die Faktoren KB_0 bis KB_2 zurückgekoppelt. Weicht die beobachtete Regelgröße von der echten Regelgröße ab, so werden die Zustände des Zustandsbeobachters um die Differenz, multipliziert mit einem Faktor korrigiert. So nehmen alle Zustände genau den Wert an, welchen sie im realen Prozess ebenfalls haben. Um die Rückkoppelfaktoren zu bestimmen, wird die Übertragungsfunktion von u_{stell} nach y_{beo} benötigt. Für diese kann y_{ist} auf 0 gesetzt werden. Damit werden ähnlich der Regelungsnormalform die Faktoren KB_0 bis KB_2 jeweils parallel zu den Faktoren a_0 bis a_2 geschaltet. Wie auch zuvor bei der Regelungsnormalform können nun die Faktoren KB_0 bis KB_1 direkt aus dem Nennerpolynom bestimmt werden.

$$G_s = \frac{b_3 s^3 + b_2 s^2 + b_1 s + b_0}{s^3 + (K_{B2} + a_2) s^2 + (K_{B1} + a_1) s + (K_{B0} + a_0)} \quad (2.3.14)$$

Wieder müssen Wunschpole festgelegt werden, welche jedoch schneller, also weiter auf der linken Seite der S-Halbebene liegen sollten als die des Zustandsreglers. Die Wunschpole werden ausmultipliziert und über den Koeffizientenvergleich werden die Faktoren KB0 bis KB2 berechnet.

2.3.5.3 Zustandsbeobachter zu jedem Modell

Wie auch bei der Regelungsnormalform kann auch der Zustandsbeobachter an jedem anderen Modell realisiert werden. Das heißt, man kann das Modell in Regelungsnormalform anfertigen oder direkt ein berechnetes physikalisches Modell verwenden. Je nach dem muss bei dem gewählten Modell die Differenz zwischen beobachteter Regelgröße und Ist-Regelgröße vor jeden Zustand zurückgekoppelt werden, wobei hier wieder die Übertragungsfunktion neu aufgestellt werden muss, da nur in der Beobachtungsnormalform die Rückkopplungsstellen so gewählt sind, dass die Faktoren KB und a paarweise auftreten. Wie die neue Übertragungsfunktion relativ einfach aufgestellt werden kann ist in Kapitel 2.3.4.2 bereits erwähnt.

MATLAB bietet hier, wie auch bei der Regelungsnormalform, den Befehl $place(A', C', p)$, wobei A' für die transponierte¹ Systemmatrix, C' für die transponierte Beobachtungsmatrix und p für die Wunschpole steht. Ferner ist zu beachten, dass bei diesem Befehl genau die K Faktoren berechnet werden, welche direkt vor den reinen Integrator gekoppelt werden müssen. Bei einem Modell kann es jedoch auftreten, dass die Integratoren selbst einen Beiwert besitzen und deshalb die durch den Befehl $place$ errechneten Faktoren ergänzt werden müssen [9].

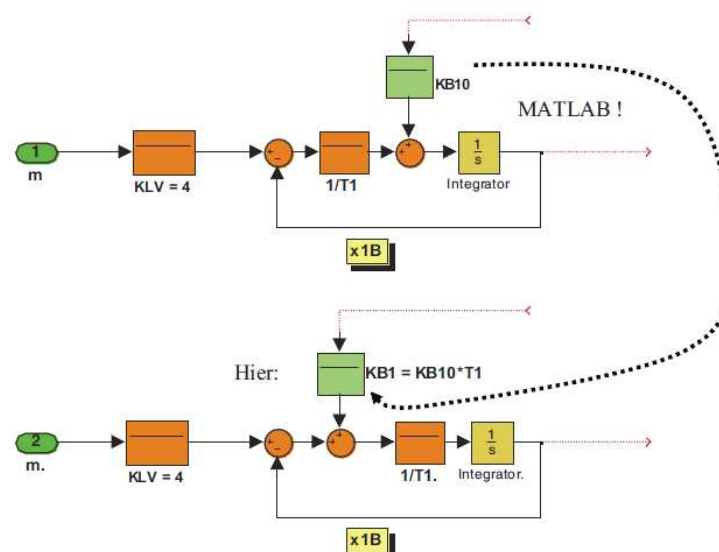


Abbildung 10: Anpassung Rückkoppelfaktoren für den Zustandsbeobachter[9]

¹ Transponieren einer Matrix bedeutet, dass Zeilen und Spalten getauscht werden [7].

2.4 MATLAB

Bei der Studienarbeit wird die reale Regelstrecke - das inverse Pendel - modelliert, eine Zustandsregelung mit Zustandsbeobachter ausgelegt und das Modell in Kombination mit der Regelung simuliert und in 3D visualisiert. Für diese Aufgaben wird das Programm MATLAB der Firma MathWorks verwendet. In diesem Kapitel wird auf Grundlagen zu diesem Programm im Zusammenhang zur Studienarbeit eingegangen.

2.4.1 Erstellen eines Simulink Modells mit Subsystemen

Um in einem Simulink-Modell die Übersicht zu behalten, ist es erforderlich, dieses hierarchisch in Subsysteme zu untergliedern. So befindet sich auf der obersten Ebene jeweils nur ein Block für den Regler und die zu regelnde Strecke. Die Zusammenhänge innerhalb der Regelstrecke können betrachtet werden, wenn man deren Subsystemblock öffnet, indem man ihn doppelt anklickt oder, wenn das Subsystem maskiert wurde, über die Menüauswahl „look under Mask“ (siehe Abbildung 14).



Abbildung 11: Schaltfläche des Library Browsers

Wie alle anderen Simulink-Komponenten findet man im Library Browser unter „Simulink/Ports & Subsystems“ den Subsystem-Block, welcher in ein Simulink-Modell (*.mdl) gezogen werden kann.

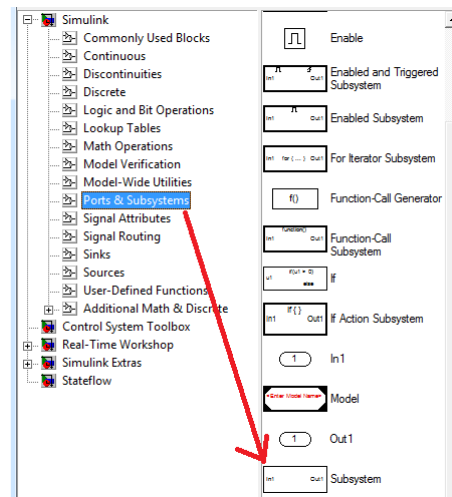


Abbildung 12: Fundort des Subsystem-Blocks

So kann jeweils ein Subsystem-Block für Regler und Regelstrecke erstellt werden.

2.4.1.1 Ein- und Ausgänge eines Subsystems

Die verfügbaren Ein- und Ausgänge eines Subsystems sind genau die In- und Out-Blöcke innerhalb des Subsystems. Im Beispiel in Abbildung 13 wird der Eingang „eing1“ und der Ausgang „ausg1“ am Subsystemblock angeboten

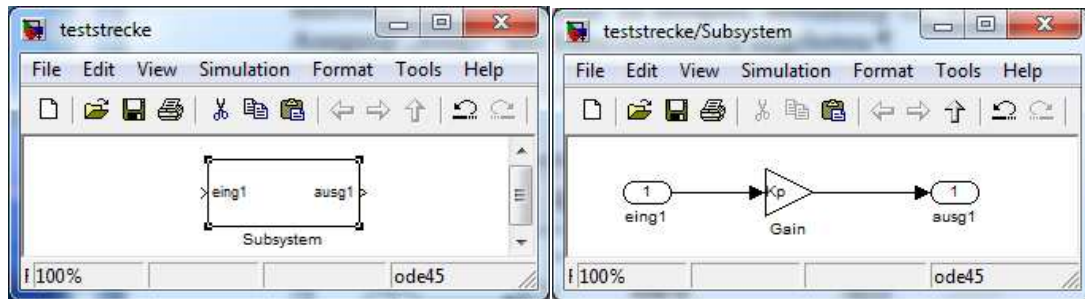


Abbildung 13: Beispiel für den Inhalt eines Subsystems

2.4.1.2 Subsystemparameter

Mit einem Rechts-Klick auf das so erstellte Subsystem können diesem über den Menüpunkt „Edit Mask“ Parameter hinzugefügt werden.

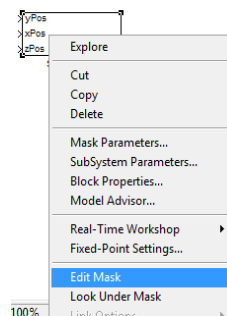


Abbildung 14: Dropdown-Menü-Eintrag 'Edit Mask' eines Subsystems

Dort kann unter dem ersten Reiter „Icons&Ports“ die Darstellung des Subsystemblocks angepasst werden. In Abbildung 15 wird ein Bild als Inhalt für die Darstellung des Subsystemblocks hinzugefügt.

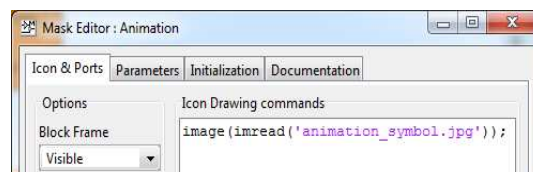


Abbildung 15: Reiter "Icon&Ports" des Mask Editors

Über die Maske können dem Subsystem Parameter hinzugefügt werden. Als Prompt sollte der Text eingegeben werden, welcher über bzw. bei dem Eingabefeld angezeigt werden soll. Als Variable sollte ein Name gewählt werden, welcher jedoch nur innerhalb des Dialoges gültig ist. Bei Typ kann zwischen den Steuerelementen „edit“ (Eingabefeld), „popup“ (Dropdown-Menü) und „checkbox“ (Haken) gewählt werden.

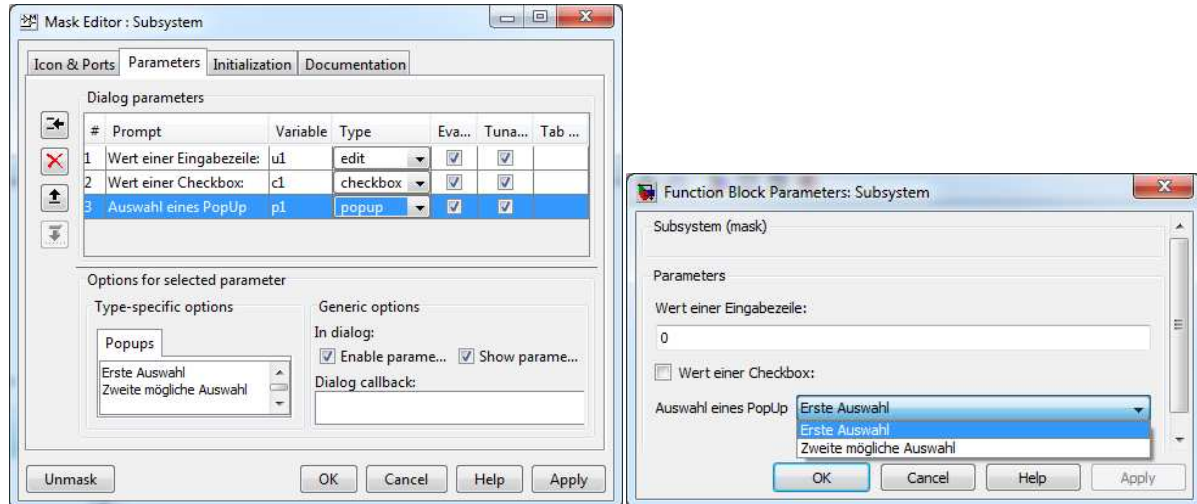


Abbildung 16: Mask Editor, anlegen von Parametern (links Konfiguration, rechts Ergebnis)

Die so angelegten Dialogvariablen sollten nicht direkt verwendet werden. Sie sollten zuvor in Variablen des Workspaces des Subsystems kopiert werden. Dies kann unter dem Reiter „Initialization“ realisiert werden.

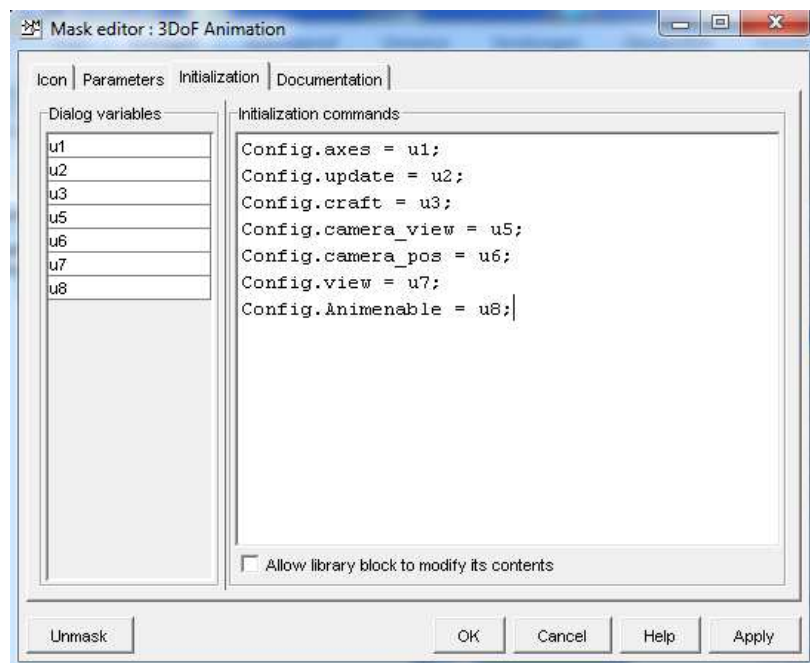


Abbildung 17: Zusammenfassen der Parameter zu einer Struktur bei der Initialisierung

Hier können die Dialogvariablen einfachen Variablen oder mehrere einer Struktur zugewiesen werden. Die Struktur bietet sich an, wenn mit den Daten einer S-Function übergeben werden sollen (siehe Kapitel 2.4.2.3).

Über den Reiter „Documentation“ kann eine Überschrift bei den Parametern, eine allgemeine Beschreibung des Subsystems, sowie ein Hilfe Text im HTML-Stil hinzugefügt werden

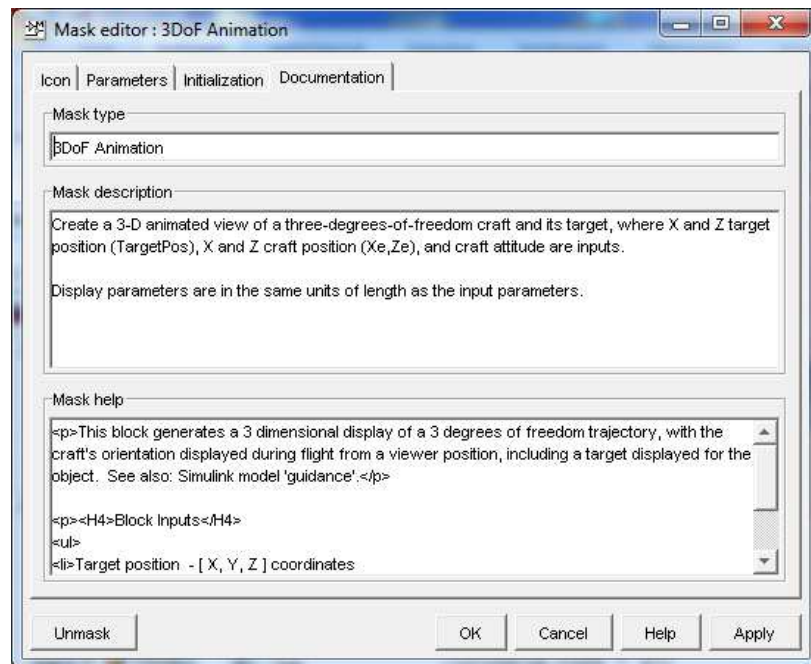


Abbildung 18: Dokumentation eines Subsystem-Blocks

2.4.2 S-Functions

Eine Methode mit MATLAB eine 3D-Animation zu erstellen besteht darin, eine S-Function zu erstellen. Eine S-Function ist eine in MATLAB geschriebene Funktion (innerhalb eines M-Files) bei dem definierte Rückgabewerte und Parameter verwendet werden. So kann diese Funktion einem S-Function-Block in Simulink zugewiesen werden und so in jedem Iterationsschritt der Simulation abgearbeitet werden. MATLAB bietet verschiedene S-Function Varianten an. In dieser Studienarbeit wurde die Variante der Level-1 S-Function gewählt. Für diese werden die folgenden Funktionen benötigt.

2.4.2.1 Die Hauptfunktion

```
function [sys,x0,str,ts] = mein3d(t,x,u,flag,Config)
switch flag,

    % Initialization
    case 0,
        [sys,x0,str,ts]=mdlInitializeSizes(Config);

    % Derivatives
    case 1,
        sys=mdlDerivatives(t,x,u, Config);

    % Update
    case 2,
        sys=mdlUpdate(t,x,u, Config);

    % Outputs
    case 3,
        sys=mdlOutputs(t,x,u, Config);

    % Outputs_NextTimeHit
    case 4,
        sys=mdlGetTimeOfNextVarHit(t,x,u,Config);

    % Terminate
    case 9,
        sys=[];

    % Unexpected flags
    otherwise
        DASTudio.error('Simulink:blocks:unhandledFlag', num2str(flag));

end
```

Diese Funktion wird von Simulink über den S-Function-Block vor, während und nach der Simulation aufgerufen. Der Name der Funktion ist wie bei M-Files üblich der Dateiname, ohne Datei-Endung (*.m). Im Parameter t wird die Simulationszeit, im Vektor x die Zustände, im Vektor u die Eingangsgrößen und im Parameter flag ein Indikator für die gewünschte Aktion übergeben. Diese Parameter müssen bei einer Level-1 S-Function immer vorhanden sein. Der Parameter Config hingegen ist ein benutzerdefinierter Parameter. Er wird nicht über den Eingang sondern über die Parameterliste beim Erstellen des S-Function-Blockes zugewiesen, siehe Kapitel 2.4.2.3.

2.4.2.2 Die Initialisierung

Als ersten Aufruf initialisiert Simulink die S-Funktion. Hierzu ruft es die S-Function mit dem Wert 0 für den Parameter flag auf. Die Hauptfunktion wiederum ruft die Initialisierungsfunktion auf. Dort werden wichtige Informationen zu Ein- bzw. Ausgängen des Blockes angegeben. Ferner besteht hier die Möglichkeit die Ausführung vorzubereiten, zum Beispiel bei einer 3D-Animation die Koordinaten der Objekte zu berechnen, ein Fenster zu öffnen, etc.

```
% Initialization
[sys,x0,str,ts]=mdlInitializeSizes(Config);

% Set Sizes Matrix
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 0;
sizes.NumInputs = 7;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);

% initialise the initial conditions
x0 = [];

% str is always an empty matrix
str = [];

% initialise the array of sample times
ts = [-2 0]; % variable sample time
```

Über die Struktur sizes wird die Breite des Eingangsvektors und Ausgangsvektors parametrisiert. Mit diesen Angaben wird so die Darstellung des S-Function-Blocks in Simulink beeinflusst.

2.4.2.3 M-File einem S-Function-Block zuweisen

Nachdem das M-File erstellt wurde, muss es einem S-Function-Block zugewiesen werden. Diesen findet man im Library Browser unter „User-Defined Functions/S-Function“.

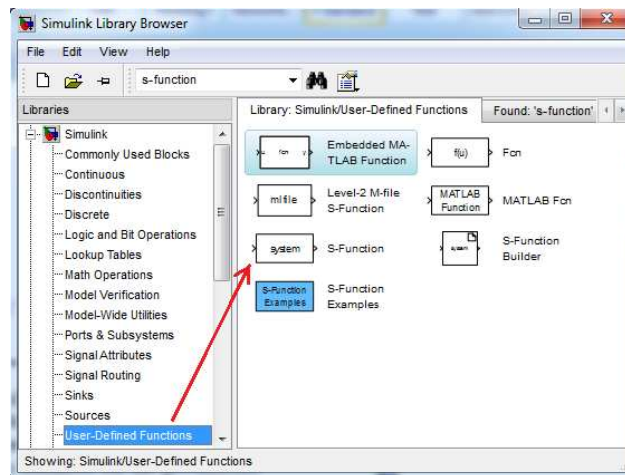


Abbildung 19: Fundort S-Function Block

Hat man diesen in sein Simulink Modell gezogen, kann er nach einem Doppelklick konfiguriert werden. Als Namen für die S-Function muss der Dateiname des zuvor erstellten M-Files (ohne die Dateiendung) gewählt werden. Werden benutzerdefinierte Parameter verwendet können diese unter „S-function parameters“ angegeben werden, in der Reihenfolge, in der sie in der Funktion verwendet werden.

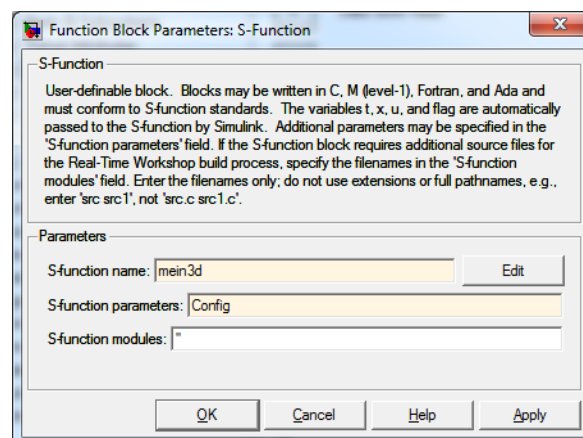


Abbildung 20: Einem S-Function-Block ein m-File zuweisen

Nach dem das M-File ausgewählt wurde, werden die Eingänge angezeigt, welche über die Initialisierung im M-File vorgegeben werden. Die benutzerdefinierten Parameter können wie in Kapitel 2.4.1.2 über die Eingabemaske des Subsystems werden.

3 Darstellung des Sollkonzepts

Für die Studienarbeit werden drei wesentliche Teile benötigt. Zuerst wird das Modell der Regelstrecke erstellt, also das Pendel inklusive des Motors und der Sensoren.

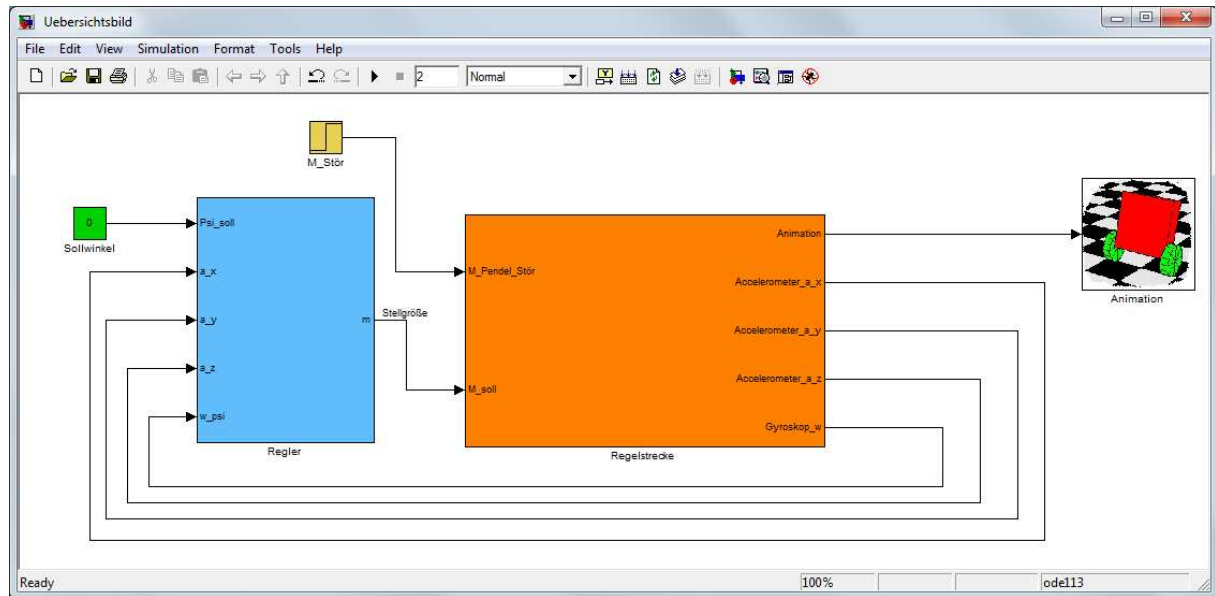


Abbildung 21: Übersicht Sollkonzept

Um die Bewegungen des Modells zu visualisieren wird eine 3D-Animation erstellt und mit dem Animationsvektor des Modells verbunden.

Um das Modell zu regeln, wird ein Zustandsregler hinzugefügt, welcher alle im Modell verfügbaren Sensorinformationen erhält und die Stellgröße für den Motor vorgibt.

3.1 Die Regelstrecke

Die Regelstrecke besteht aus den Komponenten Motor, dem Pendel selbst, einem Beschleunigungssensor und einem Gyroskop. Ferner werden die Koordinaten und Winkel zu einem Vektor zusammen gefasst, um diese der Animation zur Verfügung zu stellen.

3.1.1 Der Motor

Das Modell des Pendels kann sich nur vorwärts bzw. rückwärts bewegen. Daher werden die beiden im realen vorhandene Motoren zu einem einzigen zusammengefasst. Hierdurch gibt es für das Modell nur eine Stellgröße. Beim realen Pendel wird das durch die Regelung angeforderte Moment symmetrisch auf beide Motoren verteilt. Für Drehbewegungen kann es auch gewichtet auf die Motoren verteilt werden. Der modellierte Motor besitzt eine einstellbare Sättigung, mit welcher das geforderte Moment begrenzt wird, zusätzlich wird das geforderte Moment über ein PT1-Element verzögert am Ausgang ausgegeben. Die hierfür verantwortliche Zeitkonstante kann wie auch die Grenzen der Sättigung in der Eingabemaske vorgegeben werden. Bei dem Modell des Motors wird auf Reibungen und eine lastabhängige Gegenkopplung verzichtet.

3.1.2 Die Mechanik des Pendels

In diesem Block befindet sich das mechanische Modell. Es wird mit Hilfe der Lagrange-Mechanik berechnet und die resultierenden Bewegungsgleichungen in einem Simulink-Blockschaltbild realisiert. In diesem Modell werden weder Luftwiderstand noch Reibungen berücksichtigt.

3.1.3 Die Störgröße

Als Störgröße existiert ein Moment gegen das Pendel in Vorwärtsrichtung. Über dieses können Kräfte gegen das Pendel simuliert und die Reaktion der Strecke untersucht werden.

3.1.4 Der Animationsvektor

Um die Bewegung des Pendels graphisch darzustellen, wird der Ort zwischen den beiden Rädern, der Neigungswinkel ψ (Psi) des Pendels, die Fahrtrichtung über den Winkel ϕ (Phi) zur X-Achse und der Drehwinkel θ (Theta) der beiden Räder zu einem Vektor zusammengefasst. Dieser Vektor ist ausschließlich für die Animation gedacht, da diese Daten im realen Modell nicht über Sensoren erfasst werden und somit nicht verfügbar sind. Sie reichen jedoch aus, um das Pendel in jedem dem Modell möglichen Zustand darzustellen.

3.1.5 Beschleunigungssensor

Für den Beschleunigungssensor wird aus dem Neigungswinkel ψ des Pendels, der Erdbeschleunigung, der Zentrifugalbeschleunigung und der Beschleunigung in Fahrtrichtung die Beschleunigungen in X-, Y-, Z-Richtung des Sensorchips berechnet. Zusätzlich wird den Messwerten ein Rauschen überlagert, welches parametrisiert werden kann. Die Sensorsignale werden in der Realität gefiltert und unterliegen einer Verzögerung. Diese wird über ein PT1-Element nachgebildet, deren Zeitkonstante in der Eingabemaske vorgegeben werden kann.

3.1.6 Gyroskop

Das Gyroskop misst die Winkelgeschwindigkeit des Neigungswinkels ψ . Diesem Signal wird ein Rauschen überlagert, welches parametrisiert werden kann. Das Sensorsignal wird über ein PT1-Element verzögert ausgegeben. Die Zeitkonstante hierfür kann in der Eingabemaske vorgegeben werden.

3.2 Die Animation

In der Animation wird das Pendel zur aktuellen Simulation dargestellt. Aus einer Animation lassen sich Bewegungsabläufe anschaulicher als aus mehreren Diagrammen interpretieren.

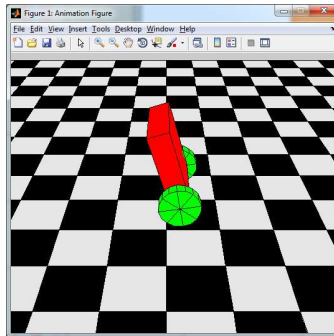


Abbildung 22: Animation

Für die Animation wird das Pendel auf einer strukturierten Oberfläche dargestellt, um so die Bewegungen in Fahrtrichtung und die dazugehörigen Radbewegungen besser zu visualisieren. Das Pendel selbst wird vereinfacht aus dem Pendelkörper und den beiden Rädern dargestellt. Details wie zum Beispiel der Akku, oder die Elektronik werden nicht visualisiert. Die Animation wird über den Eingangsvektor, bestehend aus der Fahrtrichtung als Winkel zur positiven X-Achse, dem Neigungswinkel des Pendels, dem Drehwinkel jedes Rades und der Koordinate des Punktes zwischen den Rädern, gesteuert. Mit der Animation kann so jeder mögliche Bewegungsablauf dargestellt werden.

3.3 Der Regler

Für die Regelung wird ein Zustandsregler eingesetzt. Der Neigungswinkel wird einerseits aus den Daten des Beschleunigungssensors und andererseits durch Integration der Daten des Gyroskops berechnet. Beide Ergebnisse werden zusammengeführt, um den relativen Winkel aus den Daten des Gyroskops zu korrigieren. Der so ermittelte Winkel und seine Winkelgeschwindigkeit wird über den Zustandsregler geregelt.

Der Regler hat als Sollgrößen den Neigungswinkel des Pendels nach Benutzerwunsch, und die Winkelgeschwindigkeit des Neigungswinkels mit 0 rad/s.

4 Beschreibung der Realisierung

Für die Realisierung wurden die Teile, Regelung, Regelstrecke und Animation erstellt. Die Animation ist unabhängig von den beiden anderen Teilen realisierbar. Um die Regelung zu erstellen musste zuerst das Modell der Regelstrecke erstellt und hinsichtlich Steuerbarkeit und Beobachtbarkeit bewertet werden.

4.1 Die Animation

Für die Animation wurde die Level-1 S-Function (siehe Kapitel 2.4.2) verwendet. Um die Entwicklungszeit zu reduzieren wurde das MATLAB-Beispiel „aero_guidance“ verwendet und modifiziert. Der S-Function-Block selbst wurde in einem Subsystem untergebracht, um die Parametrierung über eine Eingabemaske zu ermöglichen. Ferner wurde das Subsystem „Real-time Synchronisation“ hinzuzufügen. Mit diesem ist es möglich die Berechnungszeit der Simulation abzubremesen, so dass die Animation in Echtzeit dargestellt wird.

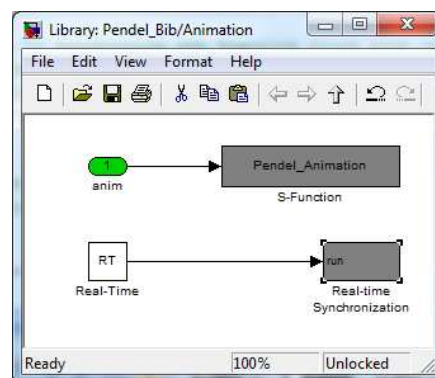


Abbildung 23: Inhalt der Animation

Der Eingang *anim* ist ein Vektor aus folgenden Signalen:

- X-Koordinate des Punktes zwischen den Rädern
- Y-Koordinate des Punktes zwischen den Rädern
- Z-Koordinate des Punktes zwischen den Rädern
- Neigungswinkel ψ des Pendels in rad
- Winkel der Fahrtrichtung ϕ in rad
- Drehwinkel des linken Rades θ_l in rad
- Drehwinkel des rechten Rades θ_r in rad

4.1.1 Die Eingabemaske der Animation

In der Eingabemaske sind die in der Tabelle 2 aufgeführten Eingaben realisiert. Diese werden alle, bis auf einen Parameter, einer Struktur „Config“ zugewiesen, um diese der S-Funktion zu übergeben. Der einzige Parameter, welcher nicht der Struktur Config zugewiesen wird, ist die Auswahl „Auf Echtzeit verlangsamen“. Diese wird direkt der Variablen RT zugewiesen, welche das Subsystem „Real-Time Synchronisation“ aktiviert, bzw. deaktiviert.

4.1.2 Die S-Funktion der Animation

Für die S-Function wurde die existierende S-Function „aero_guidance/3DoF Animation“ verwendet in der eine zielsuchende Rakete in 3D dargestellt wird. Diese Animation verwendet eine Level-1 S-Function (siehe Kapitel 2.4.2). Um statt der Rakete das mobile, inverse Pendel zu animieren mussten die einzelnen Funktionsaufrufe angepasst werden.

4.1.2.1 Initialisierung

In der Initialisierung wird über `Simsizes` festgelegt, dass der Eingangsvektor eine Breite von 7 hat. Ferner wird angegeben, dass die S-Function keine Ausgänge und auch keine Zustände besitzt. Schließlich wird über den Rückgabewert `ts` mit `[-2 0]` festgelegt, dass die Animation in dynamischen Zeitabständen aktualisiert werden soll.

```
function [sys,x0,str,ts]=mdlInitializeSizes(Config)

% Set Sizes Matrix
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 0;
sizes.NumInputs = 7;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1; % at least one sample time is needed
sys = simsizes(sizes);

% initialise the initial conditions
x0 = [];

% str is always an empty matrix
str = [];

% initialise the array of sample times
ts = [-2 0]; % variable sample time
```

Die Animation wurde so erstellt, dass sie abgeschaltet werden kann. Hierzu wird die Variable `Config.Animenable`, welche in der Eingabemaske gesetzt wird, ausgewertet. Ist diese nicht gesetzt, wird die Initialisierung abgebrochen.

```
if ~Config.Animenable
    return
end
```

Über den Befehl `findobj` wird überprüft, ob das Animationsfenster bereits geöffnet ist. Ist es vorhanden, wird dieses ausgewählt, ansonsten wird ein neues Fenster erstellt.

```
% Initialisiere Fenster
h_f=findobj('type','figure','Tag','3DOF anim');
if isempty(h_f)
    h_anim=figure;
else
    h_anim=h_f;
end
```


Im Folgenden wird das gefundene Fenster für die Animation parametrisiert. Welche Eigenschaften für das Fenster eingestellt werden können, kann der MATLAB-Hilfe entnommen werden [8.5].

```
set(h_anim,'name','Animation Figure', ...
    'renderer','zbuffer','resize','off', ...
    'position',[500, 200, 525, 450], ...
    'Tag','3DOF anim');

if ~isempty(h_anim)
    h_del = findobj(h_anim,'type','axes');
    delete(h_del);
    figure(h_anim);
end
```

Anschließend wird das alte Achsen-Objekt gelöscht und das Animationsfenster als aktuelles Fenster ausgewählt. Im nächsten Schritt wird das Achsen-Objekt neu initialisiert. Welche Eigenschaften den Achsen zugewiesen werden können, kann der MATLAB-Hilfe entnommen werden [8.6].

```
% Initialisiere Achsen
handle.axes(1)=axes;
axis(Config.axes);

set(handle.axes(1),'visible','on','xtick',[],'ytick',[],'ztick',[],'box',
    'on', ...
    'dataaspectratio',[1 1 1], ...
    'projection','pers', ...
    'units','normal', ...
    'position',[0.1 0.1 0.75 0.75], ...
    'Color',[.8 .8 .8], ...
    'drawmode','fast');
```

Um den Boden zu zeichnen, wird eine doppelte Schleife durchlaufen. Bei jedem Durchlauf einer Spalte bzw. Zeile wird die Variable Merker zwischen 0 und 1 gewechselt. Mit diesem Modifikator kann jede Platte im Wechsel hell bzw. dunkel gezeichnet werden. Die Variable count wird benötigt, um jede Fliese separat dem Array handle.floor zuzuweisen.

```
% Zeichne den Boden
fsize = Config.f_size;
count = 1;
Merker = 1;
for yf = Config.axes(1):fsize:Config.axes(2)-fsize
    count = count + 1;
    for xf = Config.axes(3):fsize:Config.axes(4)-fsize
        handle.floor(count) = patch([xf xf xf+fsize xf+fsize],...
                                     [yf yf+fsize yf+fsize yf],...
                                     [0 0 0 0], [0.9 0.9 0.9].*Merker);

        set(handle.floor(count),'userdata',...
            get(handle.floor(count), 'vertices'));
        Merker = ~Merker & 1;
    end
    Merker = ~Merker & 1;
end
```

In der Initialisierung werden die Vertex-Informationen der Objekte, das Pendel und die beiden Räder berechnet und in den Benutzerdaten gesichert. Aus dieser Quelle kann während der Aktualisierung das Objekt geladen, über Matrix-Operationen gedreht, verschoben und anschließend dargestellt werden. Die Vertex-Informationen für das Pendel und für ein Rad werden über die Funktionen *body_shape* und *wheel* berechnet. Die Daten für ein Rad werden nur einmal berechnet und nach einer Verschiebung nach links oder rechts dem entsprechenden Handle zugewiesen.

```
% Zeichne Pendel in die Benutzerdaten
[xcraft,ycraft,zcraft]=body_shape(Config);
handle.craft = patch(xcraft,ycraft,zcraft,[1 0 0]);
set(handle.craft,'userdata',get(handle.craft,'vertices'))
set(handle.craft,'facecolor',[1 0 0], ...
    'edgecolor',[0 0 0], ...
    'erasemode','nor','clipping','off');

[xwheel,ywheel,zwheel]=wheel(Config);
% Zeichne Rad links

handle.wheel_l = patch(xwheel,ywheel+Config.W/2+Config.R_WIDTH/2, ...
    zwheel,[0 1 0]);
set(handle.wheel_l,'userdata',get(handle.wheel_l,'vertices'))
set(handle.wheel_l,'facecolor',[0 1 0], ...
    'edgecolor',[0 0 0], ...
    'erasemode','nor', ...
    'clipping','off');

% Zeichne Rad rechts
handle.wheel_r = patch(xwheel,ywheel-Config.W/2-Config.R_WIDTH/2, ...
    zwheel,[0 1 0]);
set(handle.wheel_r,'userdata',get(handle.wheel_r,'vertices'))
set(handle.wheel_r,'facecolor',[0 1 0], ...
    'edgecolor',[0 0 0], ...
    'erasemode','nor', ...
    'clipping','off');

% Set Handles of graphics in Figure UserData
set(h_anim,'userdata',handle);
```

Über den abschließenden Befehl *set* werden die erzeugten Daten mit dem Fenster verknüpft und die Initialisierung abgeschlossen.

4.1.2.2 Berechnen der Vertex-Informationen für das Pendel

Der Pendelkörper wird aus einem Quader, bestehend aus 6 rechteckigen Flächen erstellt. Aus den Variablen für die Tiefe des Pendels *Config.D*, der Breite des Pendels *Config.W* und der Höhe des Pendels *Config.H*, werden die Koordinaten des vorderen, rechten und unteren Punktes und des hinteren, linken und oberen Punktes generiert. Aus den Kombinationen dieser beiden Koordinaten kann jeder Eckpunkt des Quaders erzeugt werden. Die Koordinaten eines Eckpunktes werden in 3 Matrizen, X-Koordinaten, Y-Koordinaten und Z-Koordinaten, gehalten. Die Koordinaten für eine Fläche befinden sich in einer Spalte, zum Beispiel alle X-Koordinaten für die erste Fläche in der ersten Spalte der Matrix für die X-Koordinaten. Jede weitere Spalte ist eine weitere Fläche. Über die 3 Matrizen wird so jede Koordinate jeder Fläche gehalten.

```
function [x,y,z]=body_shape(Config)

x = [];
y = [];
z = [];

% Erstellen eines Quaders
z1 = 0;
x1 = Config.D/2;
y1 = Config.W/2;

z2 = Config.H;
x2 = -Config.D/2;
y2 = -Config.W/2;

%Vorne + Hinten
x = [x [x1 x1 ; x2 x2 ;x2 x2;x1 x1] ];
y = [y [y1 y1 ; y1 y1 ;y2 y2;y2 y2] ];
z = [z [z1 z2 ; z1 z2 ;z1 z2;z1 z2] ];
%Links + Rechts
x = [x [x1 x2 ; x1 x2 ;x1 x2;x1 x2] ];
y = [y [y1 y1 ; y2 y2 ;y2 y2;y1 y1] ];
z = [z [z1 z1 ; z1 z1 ;z2 z2;z2 z2] ];
%Oben + Unten
x = [x [x1 x1 ; x2 x2 ;x2 x2;x1 x1] ];
y = [y [y1 y2 ; y1 y2 ;y1 y2;y1 y2] ];
z = [z [z1 z1 ; z1 z1 ;z2 z2;z2 z2] ];
```

In der Funktion *body_shape* werden die Matrizen mit den Koordinaten durch Anfügen neuer Flächen erweitert. MATLAB gibt bei dieser Variante im Editor eine Warnung aus, da es effizienter ist die Größe der Matrizen x, y, z zu Beginn bereits vorzugegeben. Da diese Funktion jedoch nur einmal ausgeführt werden muss, kann auf eine Indizierung verzichtet werden.

4.1.2.3 Berechnen der Vertex-Informationen für ein Rad

Ein Rad wird in der Animation aus mehreren Keilen zusammengesetzt, siehe Abbildung 24.



Abbildung 24: Segment eines Rades

Ein Keil selbst wird aus 4 viereckigen Flächen dargestellt, deren Eckpunkte abhängig vom späteren Ort berechnet werden. Da einige Eckpunkte von mehreren Flächen verwendet werden, werden diese zuerst berechnet und anschließend den Matrizen für die Koordinaten x , y und z zugewiesen. Die Variable *Config.R* beinhaltet den Radius eines Rades und wird wie die Radbreite in *Config.R_WIDTH* über die Eingabemaske vorgegeben. Über die Schleife wird aus den Keilen ein vollständiges Rad.

```
function [x,y,z]=wheel(Config) % Konfigurationsdaten
x = [];
y = [];
z = [];

% Erstellen des linken Rades
rb = Config.R_WIDTH; % Radbreite
stepsize = pi/4;
for i = 0:stepsize:2*pi-stepsize
    Rcosi = Config.R*cos(i);
    Rcross2 = Config.R*cos(i+stepsize/2);
    Rcross = Config.R*cos(i+stepsize);
    Rsini = Config.R*sin(i);
    Rsins2 = Config.R*sin(i+stepsize/2);
    Rsins = Config.R*sin(i+stepsize);

    %Vorne
    x = [x [0 ; Rcosi; Rcross2 ; Rcross]];
    y = [y [-rb/2; -rb/2 ; -rb/2 ; -rb/2]];
    z = [z [0 ; Rsini; Rsins2 ; Rsins]];

    %Hinten
    x = [x [0 ; Rcosi; Rcross2 ; Rcross]];
    y = [y [rb/2 ; rb/2 ; rb/2 ; rb/2]];
    z = [z [0 ; Rsini; Rsins2 ; Rsins]];

    %Lauffläche
    x = [x [Rcosi; Rcross2 ; Rcross2 ; Rcosi]];
    y = [y [rb/2 ; rb/2 ; -rb/2; -rb/2]];
    z = [z [Rsini; Rsins2 ; Rsins2 ; Rsini]];

    x = [x [Rcross2; Rcross ; Rcross ; Rcross2]];
    y = [y [rb/2 ; rb/2 ; -rb/2; -rb/2]];
    z = [z [Rsins2; Rsins ; Rsins ; Rsins2]];
end
```

Über die Variable *stepsize* wird vorgegeben aus wie vielen Winkeln ein Rad zusammengesetzt werden soll.

4.1.2.4 Aktualisierung der Animation

Über die Funktion *mdlUpdate* werden zyklisch die Eingangssignale verarbeitet und in diesem Zug die Animation aktualisiert. Um das richtige Fenster und dessen Inhalt zu aktualisieren wird es über den Befehl *findobj* gesucht und nur fortgefahren, wenn es vorhanden ist.

```
function mdlUpdate(t,x,u,Config)

% Handle der Fensterobjekte erhalten
handle = get(findobj('type','figure','Tag','3DOF anim'),'userdata');

if isempty(findobj('type','figure','Tag','3DOF anim'))
    %figure has been manually closed
    return
end
```

Während der Initialisierung wurden die Vertexe aller Objekte an ihrem Ursprungsort und ihrer Ursprungsorientierung berechnet. Um diese zu animieren müssen die Objekte gedreht und verschoben werden. Um eine Drehung zu realisieren muss zuerst eine Rotationsmatrix aus dem gewünschten Drehwinkel berechnet werden. Diese wird dann via Matrix-Multiplikation mit der Matrix aller Vertexe des Objekts verrechnet. Hierbei ist zu beachten, dass die Drehung immer um den Ursprung erfolgt, daher sollte sich das Objekte zuerst dort befinden, gedreht werden und anschließend an den gewünschten Ort verschoben werden.

```
% Rotations-Matrix um das linke Rad (Theta_l)
cTheta_l = cos(u(6));
sTheta_l = sin(u(6));
rot_rad_l = [cTheta_l 0 sTheta_l ; ...
             0 1 0 ; ...
             -sTheta_l 0 cTheta_l];

% Rotations-Matrix um das rechte Rad (Theta_r)
cTheta_r = cos(u(7));
sTheta_r = sin(u(7));
rot_rad_r = [cTheta_r 0 sTheta_r ; ...
             0 1 0 ; ...
             -sTheta_r 0 cTheta_r];

% Rotations-Matrix um die Radachse (Psi)
cpsi = cos(u(4));
spsi = sin(u(4));
rot_psi = [cpsi 0 spsi; ... % Rotationsmatrix um die Radachse
           0 1 0; ...
           -spsi 0 cpsi];

% Rotations-Matrix um die Körperachse (Phi)
cphi = cos(u(5));
sphi = sin(u(5));

rot_body = [cphi -sphi 0; ... % Rotationsmatrix um die Körperachse
            sphi cphi 0; ...
            0 0 1];
```

Die Variablen *u(1)* bis *u(7)* stehen für die Eingangssignale, genau in der Reihenfolge, in welcher sie sich im Eingangsvektor befinden.

Nachdem die Rotationsmatrizen berechnet wurden, werden die Vertex-Informationen aus den Userdaten der einzelnen Objekte geladen. Für die Translation der Objekte wird jeweils eine Matrix mit der gleichen Größe wie die Vertex-Matrix des jeweiligen Objekts erstellt und via Matrixmultiplikation alle Werte der ersten Zeile auf die Verschiebung in X, der zweiten auf die Verschiebung in Y und der dritten auf die Verschiebung in Z-Richtung gesetzt. Dies ist notwendig, da eine Matrix-Addition nur mit Matrizen gleicher Größe möglich ist.

```
% Lade Vertexinformation linkes Rad
vert_rad_l = get(handle.wheel_l, 'userdata');
[a_rad_l, b_rad_l] = size(vert_rad_l);
% Erstellen der Translationsmatrix
Trans_rad_l = ([ u(1) u(2) u(3)] * ones(1, a_rad_l))';
% Anwenden der Matrixoperationen
dum_rad_l = (rot_body * (rot_rad_l * vert_rad_l'))' + Trans_rad_l;
set(handle.wheel_l, 'vertices', dum_rad_l);

% Lade Vertexinformation rechtes Rad
vert_rad_r = get(handle.wheel_r, 'userdata');
[a_rad_r, b_rad_r] = size(vert_rad_r);
% Erstellen der Translationsmatrix
Trans_rad_r = ([ u(1) u(2) u(3)] * ones(1, a_rad_r))';
% Anwenden der Matrixoperationen
dum_rad_r = (rot_body * (rot_rad_r * vert_rad_r'))' + Trans_rad_r;
set(handle.wheel_r, 'vertices', dum_rad_r);

% Lade Vertexinformation Pendelkörper
vert_body = get(handle.craft, 'userdata');
[a_body, b_body] = size(vert_body);
% Translationsmatrix
Trans_body = ([ u(1) u(2) u(3)] * ones(1, a_body))';
% Rotation um die Radachse (Psi)
dum1 = rot_psi * vert_body';
% Rotation um die Körperachse (Phi)
dum2 = rot_body * dum1;
% Translation
dum_body = dum2' + Trans_body;
%dum = (vert + [u(1) u(2) u(3)] * ones(1, a))
set(handle.craft, 'vertices', dum_body);
```

Über die erstellten Matrizen kann nun das Objekt in den gewünschten Richtungen gedreht, anschließend verschoben und über den Befehl `set` in den Speicher für `'vertices'` geschrieben werden. Die ursprünglichen Daten werden somit nicht verändert und es können immer absolute Bewegungen durchgeführt werden.

Je nach Benutzervorgaben wird im Folgenden aus der der Variablen *Config.camera_view* ausgewertet, ob der Benutzer eine feste Kamera oder eine frei bewegbare in der Eingabemaske gewünscht hat.

```
% Setzen der Position und Orientierung der Kamera
switch Config.camera_view
    case 1
        % Feste Kamera Position
        set(handle.axes(1), 'cameraupvector', [0 0 1], ...
            'cameraposition', Config.camera_pos, ...
            'cameratarget', [0 0 Config.R], ...
            'cameraviewangle', Config.view);
    case 2
        % Frei Kamera Positon Kamera nur von Benutzer gesteuert
end

% Aktualisieren des Fensters erzwingen
drawnow
```

Hat er eine feste Kamera gewählt, wird den Achsen eine Kamera zugewiesen, welche zur Normalen des Bodens ausgerichtet ist, sich an der in der Eingabemaske gewählten Position *Config.camera_pos* befindet, auf den Punkt zwischen den beiden Rädern sieht und den über *Config.view* in der Eingabemaske festgelegten Öffnungswinkel verfügt.

Hat der Benutzer dagegen die freie Kamera gewählt, werden keine Kameraeinstellungen vorgenommen. Der Benutzer kann in diesem Fall über die Standardschaltflächen des Fensters alle Einstellungen manuell vornehmen.

```
% Aktualisieren des Fensters erzwingen
drawnow
```

Abschließend wird über den Befehl *drawnow* die Aktualisierung abgeschlossen und das Fenster neu gezeichnet.

4.1.2.5 Die Zeit bis zur nächsten Aktualisierung

Die Funktion *mdlGetTimeOfNextVarHit* wird von Simulink aufgerufen, da bei der Initialisierung die Samplezeit mit [-2 0] angegeben wurde. Diese Vorgabe entspricht einer variablen Zeit, also wird die S-Function regelmäßig über diese Funktion abgefragt, zu welchem Zeitpunkt die *mdlUpdate* Funktion aufgerufen werden soll. In der Studienarbeit wird in dieser Funktion immer der aktuelle Zeitpunkt plus den in der Eingabemaske festgelegten Wert *Config.update* zurückgegeben. Somit kann die vom Benutzer gewünschte Wiederholrate realisiert werden.

```
function sys=mdlGetTimeOfNextVarHit(t,x,u,Config)
    sys = t+Config.update;
```

4.1.3 Die Real-time Synchronisation

Die Real-time Synchronisation befindet sich im Subsystem der Animation und ist selbst ebenfalls ein Subsystem mit dem Inhalt aus Abbildung 25.

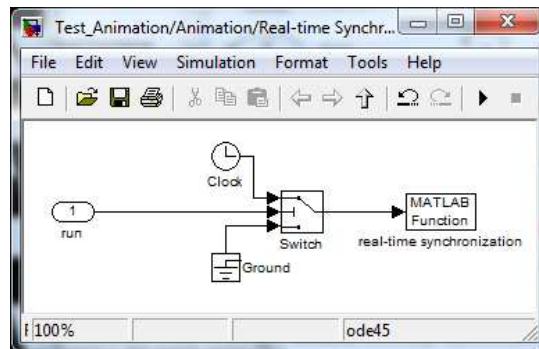


Abbildung 25: Inhalt der Real-time Synchronisation

Der Schalter gibt für $run = 0$ ebenfalls 0 am Ausgang aus, ansonsten die aktuelle Simulationszeit. Diese weitergeleitete Information wird in der MATLAB-Funktion, *waitfortreal.m* weiterverarbeitet.

```
function waitfortreal(tsim)
% This function is intended to synchronize the Simulink clock with real
% time. When called with t = 0, the beginning time is established.
% Subsequent calls, with tsim > 0, enter a busy wait state until the
% real-time clock catches up to tsim. It is assumed that the Simulink
% clock, tsim, is faster than real time.

% Stan Quinn December 1996
% Copyright (c) 1995-97 by The MathWorks, Inc.
% $Revision: 1.1 $

global waitfortrealTstart
time = clock;
if (tsim == 0)
    waitfortrealTstart = time;
else
    while etime(clock, waitfortrealTstart) < tsim
        end
    end
end
```

In dieser Funktion wird zum Startzeitpunkt der Simulation die aktuelle, echte Zeit in der Variablen *waitfortrealTstart* gesichert. Nach dem Start der Simulation wird bei jedem Aufruf so lange eine While-Schleife abgearbeitet, bis die reale Zeit, also die reale Zeit zwischen dem Start der Simulation und der aktuellen realen Zeit, kleiner als die Simulationszeit ist. Somit wird, wenn die Simulation zu schnell berechnet wurde, MATLAB so lange beschäftigt bis beide Zeiten wieder identisch sind.

4.1.4 Parameter der Animation

Achsen Grenzen [xmin xmax ymin ymax zmin zmax]:	Größe des Koordinatensystems, in dem das Pendel dargestellt wird
Zeitintervall zwischen Aktualisierungen [s]:	Simulationszeit nach der die Animation aktualisiert wird.
Kamera Art:	Auswahl „Feste Position“ oder „Frei bewegbar“. Legt fest, wie die Animation betrachtet wird.
Ort der Kamera [xc yc zc]:	Ort für die Kamera, wenn „Feste Position“ gewählt wurde
Öffnungswinkel der Kamera [Grad]:	Öffnungswinkel der Kamera
Animation Einschalten:	Auswahl, ob die Animation angezeigt werden soll. Kann rechenintensive oder lange Simulationen beschleunigen.
Auf Echtzeit verlangsamen:	Auswahl, ob die Simulationszeit auf die echte Zeit herunter gebremst werden soll
Fließengröße in [m]:	Größe der Struktur des Bodens
Rad Radius [m]:	Radius eines Rades
Rad Breite [m]:	Breite eines Rades
Höhe Pendel [m]:	Länge des Pendels
Weite Pendel [m]:	Breite des Pendels
Dicke Pendel [m]:	Dicke des Pendels

Tabelle 2: Parameter der Animation

4.1.5 Test der Animation

Um die Animation zu testen wurde in einem Simulink-Modell feste Funktionswerte übergeben, so dass das Pendel im Kreis fährt, über eine Sinusfunktion nach hinten bzw. vorne kippt, sich in Fahrtrichtung dreht und die beiden Räder sich mit einer unterschiedlichen Geschwindigkeit drehen.

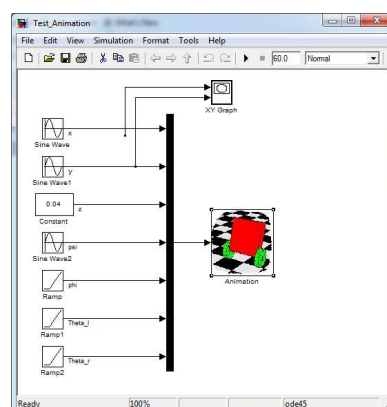


Abbildung 26: Test der Animation

Dieses Modell findet man unter :

\02 Matlab Modell\01 Animation\Test_Animation.mdl

Die Animation selbst wurde in einer Bibliothek gespeichert:

\02 Matlab Modell\Pendel_Bib.mdl

4.2 Das physikalische Modell

Für die Studienarbeit wird das mobile, inverse Pendel nur in 2 Dimensionen betrachtet und geregelt. In diesem Teil der Realisierung werden die Bewegungsgleichungen mit Hilfe der Lagrange-Mechanik berechnet und anschließend in einem Simulink-Modell umgesetzt.

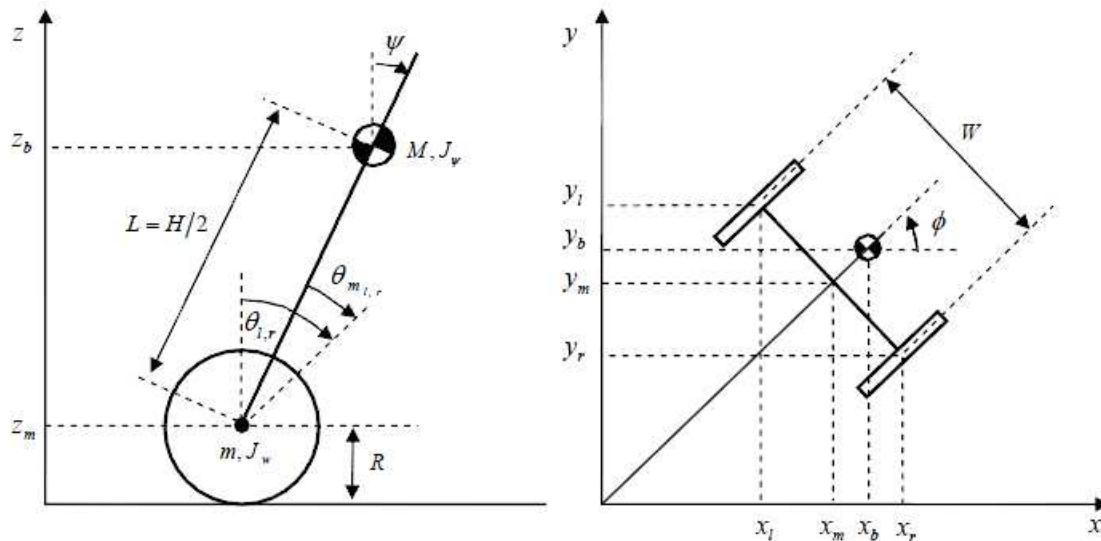


Abbildung 27 : Zeichnung Inverses Pendel [5]

Für die Berechnung und die Modellierung in nur 2 Dimensionen wurden die in Tabelle 3 angegebenen Größen verwendet.

Naturkonstanten		
g	Erdbeschleunigung mit 9,81 auf der Erdoberfläche	[m/s ²]
Bemaßung		
R	Radius eines Rades	[m]
L	Abstand des Pendelschwerpunktes von der Radachse	[m]
Winkel		
ψ	Der Neigungswinkel	[rad]
θ	Der Drehwinkel der beiden Räder	[rad]
Massen		
m_b	Masse des Pendelkörpers	[kg]
m_r	Masse der beiden Räder	[kg]
Trägheitsmomente		
J_ψ	Massenträgheitsmoment des Pendelkörpers um die Radachse	[kgm ²]
J_θ	Massenträgheitsmoment der beiden Räder	[kgm ²]
Koordinaten		
x_r	Ort des Rades auf der X-Achse	[m]
z_r	Ort des Rades auf der Z-Achse	[m]
x_b	Ort des Pendelschwerpunktes auf der X-Achse	[m]
z_b	Ort des Pendelschwerpunktes auf der Z-Achse	[m]

Tabelle 3: Bezeichnung der Größen

4.2.1 Berechnen der Bewegungsgleichungen

Die Bewegungsgleichungen werden, wie in Kapitel 2.2.2 beschrieben, berechnet. Für die zweidimensionale Darstellung kann das gesamte Modell auf 2 Punktmassen reduziert werden: die beiden Räder und der Pendelkörper. Ein solches System ist vom Grad 6, das heißt es besitzt 6 Freiheitsgrade:

$$S = 3N = 3 \cdot 2 = 6 \quad (4.2.1)$$

4.2.1.1 Die Zwangsbedingungen des Systems

Diese 6 Freiheitsgrade können durch Zwangsbedingungen reduziert werden. Beim inversen Pendel sind dies:

$$z_r = R \quad (4.2.2)$$

Die Räder befinden sich immer auf dem Boden in der Höhe des Radius eines Rades.

$$y_r = 0 \quad (4.2.3)$$

$$y_b = 0 \quad (4.2.4)$$

Das Pendel kann sich nicht in die Tiefe bewegen, daher ist die Y-Koordinate immer 0.

$$z_b = z_r + L \cos(\psi) \quad (4.2.5)$$

$$x_b = x_r + L \sin(\psi) \quad (4.2.6)$$

Der Schwerpunkt des Pendels kann sich nur in einem festen Abstand um die Radachse bewegen. Die Gleichungen 4.2.5 und 4.2.6 beschreiben somit nur eine Zwangsbedingung. Über die 4 Zwangsbedingungen wird das System von 6 auf 2 Freiheitsgrade reduziert.

4.2.1.2 Die generalisierten Koordinaten

Durch die 2 Freiheitsgrade ergibt sich, dass das System vollständig mit 2 generalisierten Koordinaten beschrieben werden kann.

$$q_1 = \psi \quad (4.2.7)$$

Die Gleichung 4.2.7 beschreibt mit dem Neigungswinkel ψ die erste generalisierte Koordinate q_1 .

$$q_2 = \theta \quad (4.2.8)$$

Die Gleichung 4.2.8 beschreibt mit dem Drehwinkel der Räder θ die zweite generalisierte Koordinate q_2 .

$$x_r = R q_2 \quad (4.2.9)$$

$$z_r = R \quad (4.2.10)$$

Der Ort der Radachse kann mit den Gleichungen 4.2.9 und 4.2.10 beschrieben werden.

Der Ort des Pendelschwerpunkts kann mit den Gleichungen 4.2.11 und 4.2.12 beschrieben werden.

$$x_b = R q_2 + L \sin(q_1) \quad (4.2.11)$$

$$z_b = R + L \cos(q_1) \quad (4.2.12)$$

Zur Berechnung der kinetischen Energie werden jeweils die Geschwindigkeiten benötigt. Diese ergeben sich aus den Ableitungen.

$$\dot{x}_r = R \dot{q}_2 \quad (4.2.13)$$

$$\dot{z}_r = 0 \quad (4.2.14)$$

$$\dot{x}_b = R \dot{q}_2 + L \cos(q_1) \dot{q}_1 \quad (4.2.15)$$

$$\dot{z}_b = -L \sin(q_1) \dot{q}_1 \quad (4.2.16)$$

4.2.1.3 Die Lagrange-Funktion

Da die generalisierten Koordinaten Winkel sind, muss zu diesen das Massenträgheitsmoment hinzugefügt werden. Dieses kann entweder berechnet oder an den realen Komponenten gemessen werden. Somit bildet sich die Lagrange-Funktion aus den kinetischen Energien, den Rotationsenergien abzüglich der potenziellen Energien.

$$Lagrange = T_1 + T_2 - V_2 \quad (4.2.17)$$

Für das inverse Pendel bildet sich die Lagrange-Funktion nach Gleichung (4.2.18).

$$Lagrange = \frac{1}{2} m_r \dot{x}_r^2 + \frac{1}{2} m_b \dot{x}_b^2 + \frac{1}{2} m_b \dot{z}_b^2 + \frac{1}{2} J_\psi \dot{\psi}^2 + \frac{1}{2} J_\theta \dot{\theta}^2 - m_b g L (1 + \cos(\psi)) \quad (4.2.18)$$

In der Gleichung 4.2.19 werden alle Koordinaten durch die generalisierten Koordinaten beschrieben, bzw. deren Ableitungen beschrieben.

$$Lagrange = \frac{1}{2} m_r (R \dot{q}_2)^2 + \frac{1}{2} m_b (L \cos(q_1) \dot{q}_1 + R \dot{q}_2)^2 + \frac{1}{2} m_b (-L \sin(q_1) \dot{q}_1)^2 + \frac{1}{2} J_\psi \dot{q}_1^2 + \frac{1}{2} J_\theta \dot{q}_2^2 - m_b g L (1 + \cos(q_1)) \quad (4.2.19)$$

Anschließend werden die Quadrate der Geschwindigkeiten ausmultipliziert.

$$Lagrange = \frac{1}{2} m_r R^2 \dot{q}_2^2 + \frac{1}{2} m_b L^2 \cos^2(q_1) \dot{q}_1^2 + m_b L R \cos(q_1) \dot{q}_1 \dot{q}_2 + \frac{1}{2} m_b R^2 \dot{q}_2^2 + \frac{1}{2} m_b L^2 \sin^2(q_1) \dot{q}_1^2 + \frac{1}{2} J_\psi \dot{q}_1^2 + \frac{1}{2} J_\theta \dot{q}_2^2 - m_b g L (1 + \cos(q_1)) \quad (4.2.20)$$

Nach dem Sortieren nach den generalisierten Koordinaten und den gemischten Gliedern

$$\begin{aligned} \text{Lagrange} = & \dot{q}_2^2 \frac{1}{2} (m_r R^2 + J_\theta + m_b R^2) + \dot{q}_1^2 \frac{1}{2} (m_b L^2 (\cos^2(q_1) + \sin^2(q_1)) + J_\psi) \\ & + m_b L R \cos(q_1) \dot{q}_1 \dot{q}_2 - m_b g L (1 + \cos(q_1)) \end{aligned} \quad (4.2.21)$$

kann durch Anwendung von $1 = \sin^2(x) + \cos^2(x)$ die Gleichung 4.2.21 zu Gleichung 4.3.22 vereinfacht werden.

$$\begin{aligned} \text{Lagrange} = & \dot{q}_2^2 \frac{1}{2} (m_r R^2 + J_\theta + m_b R^2) + \dot{q}_1^2 \frac{1}{2} (m_b L^2 + J_\psi) \\ & + m_b L R \cos(q_1) \dot{q}_1 \dot{q}_2 - m_b g L (1 + \cos(q_1)) \end{aligned} \quad (4.2.22)$$

4.2.1.4 Die Bewegungsgleichungen

Es handelt sich bei der Lagrange-Funktion um eine der 2. Art. Diese wird gelöst, indem sie partiell nach der ersten Ableitung der gesuchten generalisierten Koordinate und anschließend nach der Zeit abgeleitet wird und davon die partielle Ableitung der Lagrange-Funktion nach der generalisierten Koordinate abgezogen wird. Der Lösungsweg der generalisierten Koordinate q_1 sieht wie folgt aus.

$$\frac{\partial \text{Lagrange}}{\partial \dot{q}_1} = \dot{q}_1 (m_b L^2 + J_\psi) + m_b L R \cos(q_1) \dot{q}_2 \quad (4.2.23)$$

$$\frac{d}{dt} \frac{\partial \text{Lagrange}}{\partial \dot{q}_1} = \ddot{q}_1 (m_b L^2 + J_\psi) + m_b L R \cos(q_1) \ddot{q}_2 - m_b L R \sin(q_1) \dot{q}_2 \dot{q}_1 \quad (4.2.24)$$

$$\frac{\partial \text{Lagrange}}{\partial q_1} = -m_b L R \sin(q_1) \dot{q}_1 \dot{q}_2 + m_b g L \sin(q_1) \quad (4.2.25)$$

$$\begin{aligned} \frac{d}{dt} \frac{\partial \text{Lagrange}}{\partial \dot{q}_1} - \frac{\partial \text{Lagrange}}{\partial q_1} = & \ddot{q}_1 (m_b L^2 + J_\psi) + m_b L R \cos(q_1) \ddot{q}_2 \\ & - m_b L R \sin(q_1) \dot{q}_2 \dot{q}_1 + m_b L R \sin(q_1) \dot{q}_1 \dot{q}_2 \\ & - m_b g L \sin(q_1) \end{aligned} \quad (4.2.26)$$

Das Ergebnis der Lösung kann weiter vereinfacht werden.

$$\begin{aligned} \frac{d}{dt} \frac{\partial \text{Lagrange}}{\partial \dot{q}_1} - \frac{\partial \text{Lagrange}}{\partial q_1} = & \ddot{q}_1 (m_b L^2 + J_\psi) + m_b L R \cos(q_1) \ddot{q}_2 \\ & - m_b g L \sin(q_1) \end{aligned} \quad (4.2.27)$$

Durch Umstellen der Gleichung 4.2.27 nach \ddot{q}_1 , erhält man die Bewegungsgleichung für \ddot{q}_1 bzw. $\ddot{\psi}$:

$$\ddot{q}_1 = \frac{m_b g L \sin(q_1) - m_b L R \cos(q_1) \ddot{q}_2}{m_b L^2 + J_\psi} = \ddot{\psi} \quad (4.2.28)$$

Der Lösungsweg der generalisierten Koordinate q_2 sieht wie folgt aus:

$$\frac{\partial \text{Lagrange}}{\partial \dot{q}_2} = \dot{q}_2(m_r R^2 + J_\theta + m_b R^2) + m_b L R \cos(q_1) \dot{q}_1 \quad (4.2.29)$$

$$\begin{aligned} \frac{d}{dt} \frac{\partial \text{Lagrange}}{\partial \dot{q}_2} &= \ddot{q}_2(m_r R^2 + J_\theta + m_b R^2) + m_b L R \cos(q_1) \ddot{q}_1 \\ &\quad - m_b L R \sin(q_1) \dot{q}_1^2 \end{aligned} \quad (4.2.30)$$

$$\frac{\partial \text{Lagrange}}{\partial q_2} = 0 \quad (4.2.31)$$

$$\begin{aligned} \frac{d}{dt} \frac{\partial \text{Lagrange}}{\partial \dot{q}_2} &= \ddot{q}_2(m_r R^2 + J_\theta + m_b R^2) + m_b L R \cos(q_1) \ddot{q}_1 \\ &\quad - m_b L R \sin(q_1) \dot{q}_1^2 \end{aligned} \quad (4.2.32)$$

Durch Umstellen der Gleichung 4.2.32 nach \ddot{q}_2 , erhält man die Bewegungsgleichung für \ddot{q}_2 bzw. für $\ddot{\theta}$:

$$\ddot{q}_2 = \frac{m_b L R (\sin(q_1) \dot{q}_1^2 - \cos(q_1) \ddot{q}_1)}{m_r R^2 + J_\theta + m_b R^2} = \ddot{\theta} \quad (4.2.33)$$

4.2.1.5 Erweitern der Gleichungen um Stell- und Störgröße

Die beiden Gleichungen 4.2.28 und 4.2.33 beschreiben die Bewegungsgleichungen des mobilen inversen Pendels in dem Fall, dass keine externe Kraft auf das System wirkt. In dem späteren Modell wird jedoch über Elektromotoren ein Moment auf das Pendel und die Räder eingebracht. Dieses wirkt in positiver Drehrichtung auf die Räder, und in negativer Richtung auf den Neigungswinkel des Pendelkörpers. Gleichung 4.2.32 wird daher um das Motormoment erweitert.

$$\begin{aligned} \frac{d}{dt} \frac{\partial \text{Lagrange}}{\partial \dot{q}_1} &= \ddot{q}_2(m_r R^2 + J_\theta + m_b R^2) + m_b L R \cos(q_1) \ddot{q}_1 \\ &\quad - m_b L R \sin(q_1) \dot{q}_1^2 - M_m \end{aligned} \quad (4.2.34)$$

Umgestellt nach \ddot{q}_2 bzw. nach $\ddot{\theta}$ ergibt sich:

$$\ddot{q}_2 = \frac{m_b L R (\sin(q_1) \dot{q}_1^2 - \cos(q_1) \ddot{q}_1) + M_m}{m_r R^2 + J_\theta + m_b R^2} = \ddot{\theta} \quad (4.2.35)$$

Mit dem gleichen Verfahren kann auch Gleichung 4.2.27 um das Motormoment und die Störgröße erweitert werden, die ausschließlich gegen das Pendel wirkt.

$$\begin{aligned} \frac{d}{dt} \frac{\partial \text{Lagrange}}{\partial \dot{q}_1} - \frac{\partial \text{Lagrange}}{\partial q_1} &= \ddot{q}_1(m_b L^2 + J_\psi) + m_b L R \cos(q_1) \ddot{q}_2 \\ &\quad - m_b g L \sin(q_1) + M_m + M_s \end{aligned} \quad (4.2.36)$$

Umgestellt nach \ddot{q}_1 bzw. nach $\ddot{\psi}$ ergibt:

$$\ddot{q}_1 = \frac{m_b g L \sin(q_1) - m_b L R \cos(q_1) \ddot{q}_2 - M_m - M_s}{m_b L^2 + J_\psi} = \ddot{\psi} \quad (4.2.37)$$

4.2.1.6 Lösen der algebraischen Schleife

Die Gleichungen 4.2.35 und 4.2.37 können zwar in MATLAB-Simulink umgesetzt und simuliert werden, dennoch gibt MATLAB bei Verwendung eine Warnung aus, dass im Modell eine algebraische Schleife vorhanden ist. Diese ergibt sich daraus, dass der Neigungswinkel durch die Winkelbeschleunigung der Räder beeinflusst wird und die Winkelbeschleunigung der Räder durch die Winkelbeschleunigung des Neigungswinkels beeinflusst wird. Beide Gleichungen sind vom Ergebnis der andern abhängig. Derartige Gleichungen können zwar von MATLAB iterativ gelöst werden, eine Darstellung ohne algebraische Schleife kann jedoch die Simulationsgeschwindigkeit steigern und das manuelle Erstellen der Zustandsraumdarstellung vereinfachen. Um die Schleife zu durchbrechen muss eine der Gleichungen unabhängig von der anderen berechnet werden. Dies wird erreicht indem Gleichung 4.2.35 in 4.2.37 eingesetzt wird.

$$\ddot{q}_1(m_b L^2 + J_\psi) = m_b g L \sin(q_1) - M_m - M_s - m_b L R \cos(q_1) \frac{m_b L R (\sin(q_1) \dot{q}_1^2 - \cos(q_1) \ddot{q}_1) + M_m}{m_r R^2 + J_\theta + m_b R^2} \quad (4.2.38)$$

$$\ddot{q}_1(m_b L^2 + J_\psi) = m_b g L \sin(q_1) - M_m - M_s - \frac{m_b^2 L^2 R^2 (\sin(q_1) \cos(q_1) \dot{q}_1^2 - \cos^2(q_1) \ddot{q}_1) + M_m m_b L R \cos(q_1)}{m_r R^2 + J_\theta + m_b R^2} \quad (4.2.39)$$

$$\begin{aligned} \ddot{q}_1(m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) &= m_b g L \sin(q_1)(m_r R^2 + J_\theta + m_b R^2) \\ &\quad - M_m(m_r R^2 + J_\theta + m_b R^2) \\ &\quad - M_s(m_r R^2 + J_\theta + m_b R^2) \\ &\quad - m_b^2 L^2 R^2 \sin(q_1) \cos(q_1) \dot{q}_1^2 \\ &\quad + m_b^2 L^2 R^2 \cos^2(q_1) \ddot{q}_1 \\ &\quad - M_m m_b L R \cos(q_1) \end{aligned} \quad (4.2.40)$$

$$\begin{aligned} \ddot{q}_1[(m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2 \cos^2(q_1)] &= \\ &\quad (m_b g L \sin(q_1) - M_s)(m_r R^2 + J_\theta + m_b R^2) \\ &\quad - M_m[(m_r R^2 + J_\theta + m_b R^2) + (m_b L R \cos(q_1))] \\ &\quad - m_b^2 L^2 R^2 \sin(q_1) \cos(q_1) \dot{q}_1^2 \end{aligned} \quad (4.2.41)$$

$$\ddot{q}_1 = \frac{(m_b g L \sin(q_1) - M_s)(m_r R^2 + J_\theta + m_b R^2) - M_m [(m_r R^2 + J_\theta + m_b R^2) + (m_b L R \cos(q_1))] - m_b^2 L^2 R^2 \sin(q_1) \cos(q_1) \dot{q}_1^2}{[(m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2 \cos^2(q_1)]} \quad (4.2.42)$$

Durch erneutes Umstellen der Gleichung 4.2.38 nach \ddot{q}_1 bzw. nach $\ddot{\psi}$ erhält man die von $\ddot{\theta}$ unabhängige Beschreibung des Neigungswinkels 4.2.42. Aus diesen Gleichungen kann nun das MATLAB-Simulink-Modell erstellt werden.

4.3 Das MATLAB-Simulink-Modell

Das Modell wurde für die Studienarbeit als Simulink-Blockschaltbild erstellt. Es besteht aus folgenden Komponenten:

- Dem Pendel, als Mechanik
- Einem Motor, als Stellgröße
- Einem Beschleunigungssensor, als Messumformer für die Beschleunigungen
- Einem Gyroskop, als Messumformer für Winkelgeschwindigkeit des Neigungswinkels.

Diese Komponenten und deren Zusammenhang ist in Abbildung 28 ersichtlich. Über den Motor wird ein Moment in der Mechanik des Pendels verursacht und dort in Kombination mit der Erdbeschleunigung in eine Bewegung umgesetzt. Aus dieser werden wiederum die Daten des Beschleunigungssensors und des Gyroscopes erzeugt.

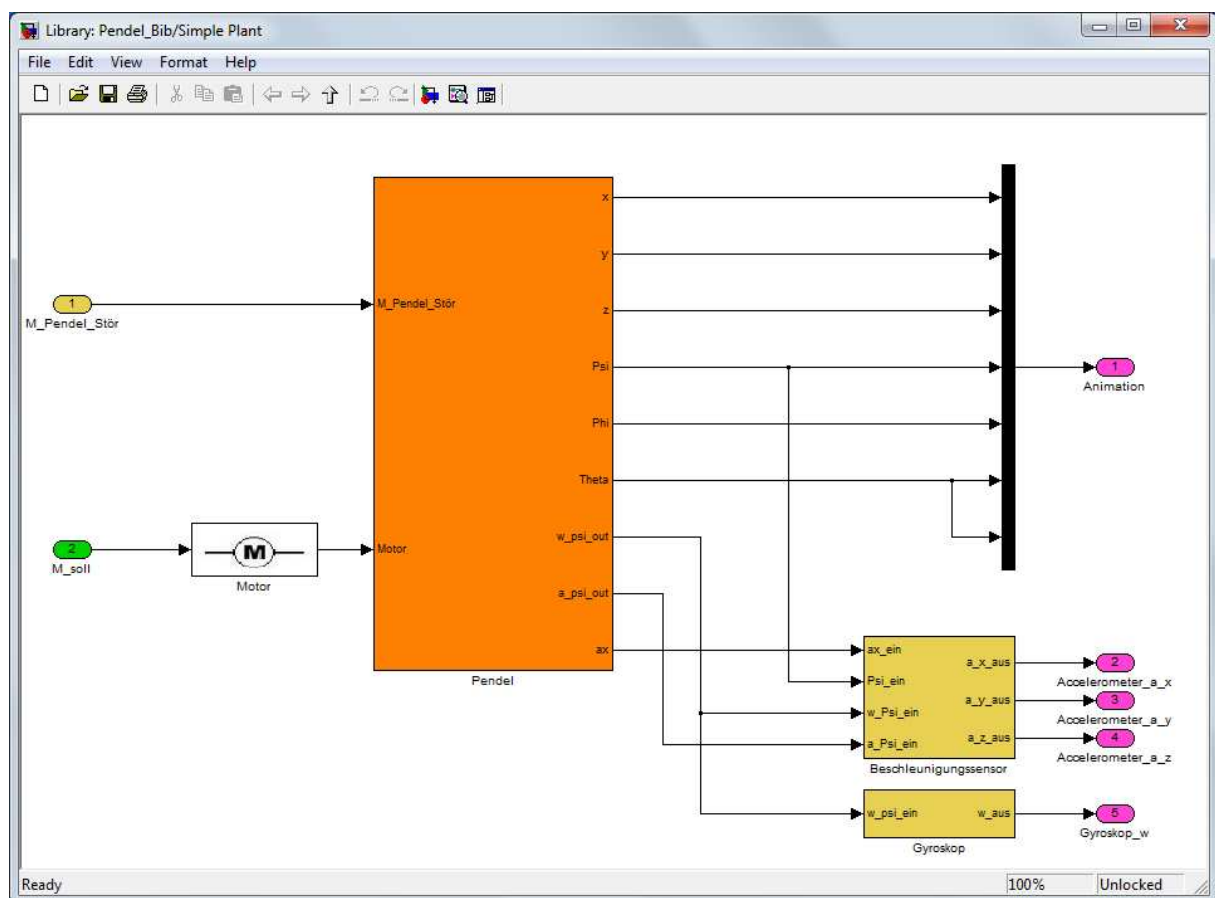


Abbildung 28: Aufbau der Regelstrecke

4.3.1 Der Motor

Der Motor ist ein vereinfachtes Modell, welches ein angefordertes Moment am Ausgang ausgibt. Um die Simulation der realen Umgebung anzunähern, wird über einen Sättigungsblock das gewünschte Moment auf das maximal bzw. minimal mögliche Moment reduziert. Über ein folgendes PT1-Element kann schließlich die Zeitkonstante festgelegt werden, welche der Motor benötigt, um das gewünschte Moment zu erreichen.

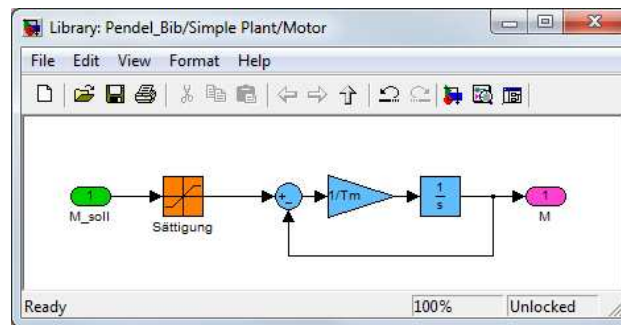


Abbildung 29: Blockschaltbild des Motors

Um die Sättigung und die Zeitkonstante festzulegen, wurde dem Motor eine Eingabemaske hinzugefügt.

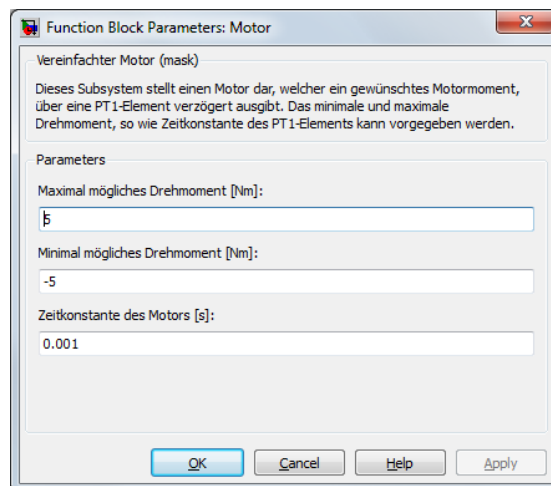


Abbildung 30: Eingabemaske für den Motor

Für die erste Simulationen kann davon ausgegangen werden, dass der Motor viel schneller als das Pendel reagiert, deshalb kann hierfür eine sehr kleine Zeitkonstante gewählt werden. Sollten die Motoren des echten Pendels nicht träger als die Mechanik des Pendels sein, so muss die Zeitkonstante angepasst und die Regelung neu ausgelegt werden. Bei dem Modell wurde auf Reibung und eine lastabhängige Gegenkopplung verzichtet.

4.3.2 Das Pendel

Für die Regelstrecke wurden die in Kapitel 4.2.1 berechneten Differenzialgleichungen verwendet. Benötigte Größen, wie zum Beispiel der Radius eines Rades, werden über die Eingabemaske eingegeben. Das gesamte Blockschaltbild der Regelstrecke kann im Anhang unter Kapitel 6.1 betrachtet werden.

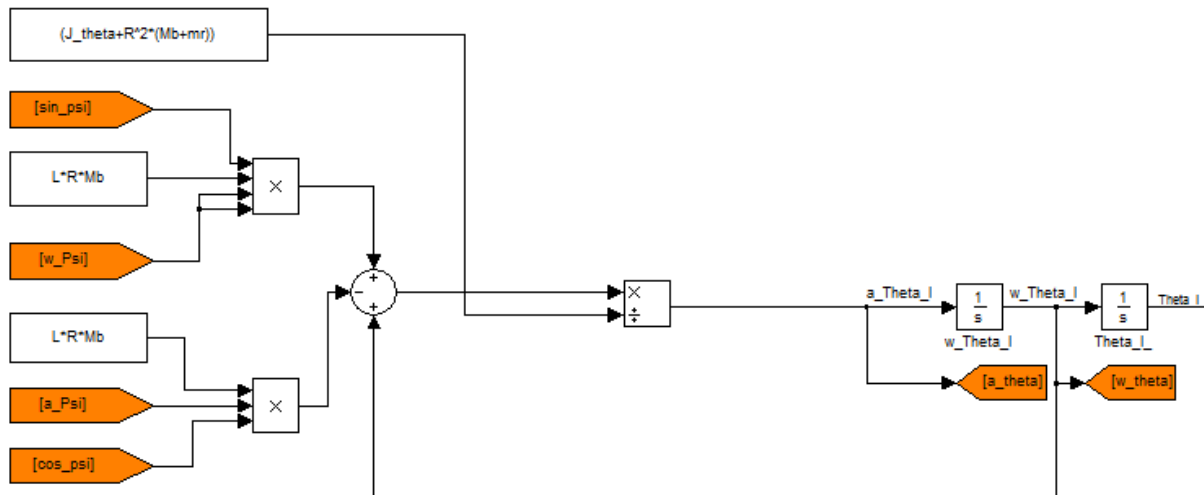


Abbildung 31: Ausschnitt aus dem Blockschaltbild für das Pendel

Aus Abbildung 31 ist ersichtlich, dass Gleichung 4.2.35 direkt umgesetzt wurde. Über den Multiplikationsblock werden die Skalare miteinander multipliziert, über den Summationsblock addiert und schließlich durch die Trägheitsmomente dividiert. Da hierbei nur eine Beschleunigung berechnet wird, muss diese zweimal integriert werden, um zuerst auf die Winkelgeschwindigkeit und schließlich auf den Winkel zu kommen. Um die Darstellung möglichst übersichtlich zu halten wurde die Signale über Goto- und From-Blöcke realisiert. Aus den beiden Differentialgleichungen auf diesem Weg das gesamte Modell erstellt.

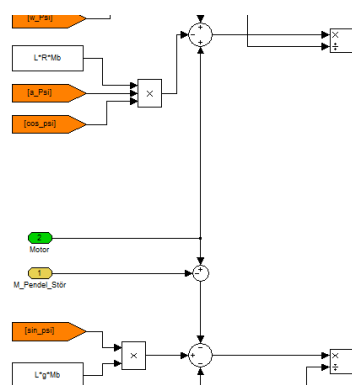


Abbildung 32: Motormoment und Störgröße

Das Motormoment wurde so modelliert, dass es in positiver Richtung auf die Räder und in negativer Richtung auf das Pendel wirkt. Die Störgröße wirkt ausschließlich in positiver Richtung auf das Pendel und simuliert hierbei zum Beispiel ein Drücken mit der Hand gegen das Pendel.

Die so erstellte Regelstrecke wurde in einer Bibliothek gespeichert:
 \02 Matlab Modell\Pendel_Bib.mdl

4.3.2.1 Das Modell ohne algebraische Schleifen

Um das Modell ohne die algebraische Schleife zu modellieren, wird für die Berechnung von $\ddot{\psi}$ statt Gleichung 4.2.36 die Gleichung 4.2.41 gewählt, welche unabhängig von $\ddot{\theta}$ berechnet werden kann, siehe Kapitel 4.2.1.6.

Die so erstellte Regelstrecke wurde in einer Bibliothek gespeichert:

\02 Matlab Modell\Pendel_Bib.mdl

4.3.2.2 Test der Regelstrecke

Über den Energieerhaltungssatz, kann nachgewiesen werden, dass ohne Einfluss der Stell- oder Störgrößen die Energie im System konstant bleibt. Als Grundlage bietet sich hier die Lagrange-Funktion, welche gerade auf der Energieerhaltung basiert. Die Umsetzung in MATLAB-Simulink wird in Abbildung 33 dargestellt.

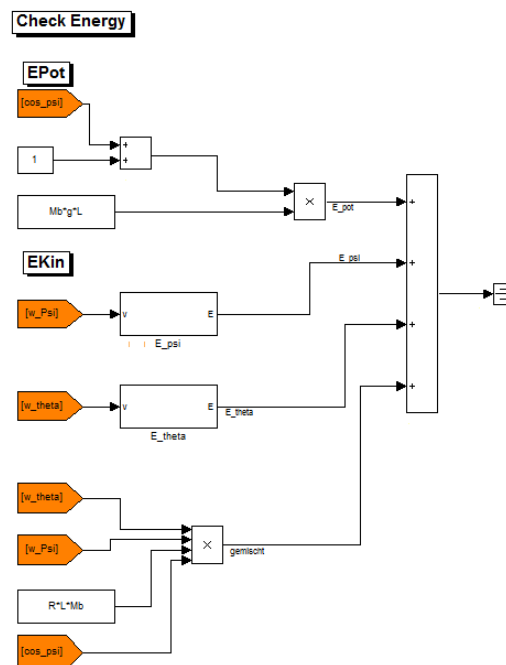


Abbildung 33: Berechnung der Energieerhaltung

Die Blöcke E_{psi} und E_{theta} sind jeweils Subsysteme, in denen die kinetische Energie des Eingangswertes berechnet wird. Die hierfür notwendigen Massen und Trägheitsmomente können über die Eingabemaske vorgegeben werden. Insgesamt kann so bis auf geringe Schwankungen in der Größenordnung von 10^{-4} die Energieerhaltung nachgewiesen werden. Da diese Schwankungen sich je nach gewähltem „Solver“² verändern, sind diese nicht modellbedingt und können die mit numerischen Fehlern begründet werden.

Diese Überprüfung befindet sich in jedem Modell der Regelstrecke, mit und ohne der algebraischen Schleife.

Die Modelle befinden sich in einer Bibliothek:

\02 Matlab Modell\Pendel_Bib.mdl

² Berechnungsmethode mit der MATLAB das Modell berechnet (ODE45, ODE113, usw.)

4.3.3 Der Beschleunigungssensor

Der Beschleunigungssensor gibt alle auf ihn wirkenden Beschleunigungen an seinen Ausgängen für die Koordinaten x, y und z aus. Die wirkenden Beschleunigungen können bezüglich der Wirkrichtung auf winkelunabhängige und winkelabhängige Beschleunigungen aufgeteilt werden.

Winkelunabhängige Beschleunigungen:

- Zentrifugalbeschleunigung
- Winkelbeschleunigung

Winkelabhängige Beschleunigungen, diese verteilen sich je nach Pendelneigung auf die X- und Z-Koordinate des Beschleunigungssensors.

- Beschleunigung in Fahrtrichtung
- Erdbeschleunigung

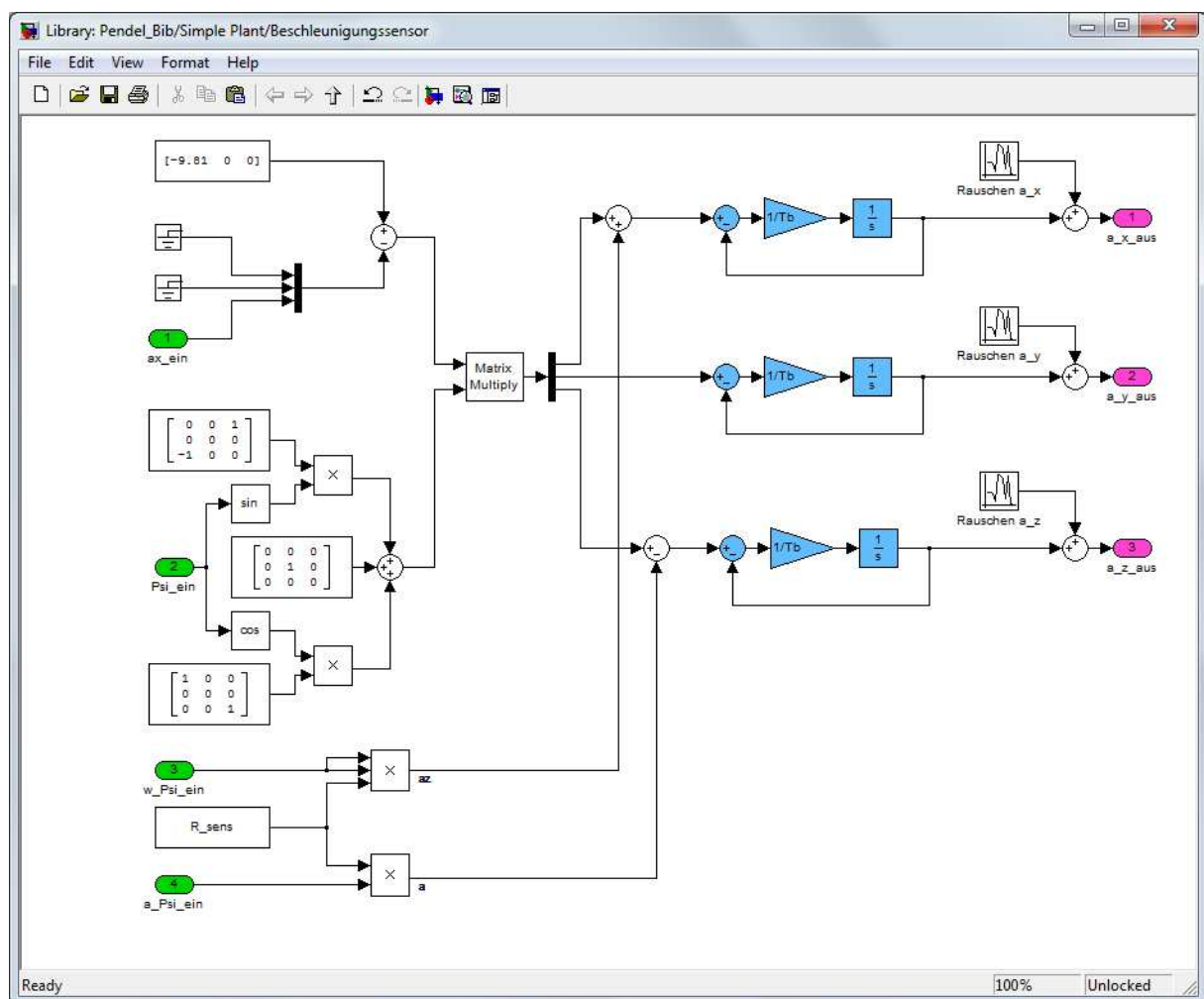


Abbildung 34: Blockschaltbild des Beschleunigungssensors

In Abbildung 34 ist das Modell des Beschleunigungssensors ersichtlich. Im oberen Teil des Blockschaltbilds werden die winkelabhängigen Beschleunigungen zu einem Vektor zusammengefasst. Zu beachten ist hierbei, dass die Z-Koordinate des Beschleunigungssensors bei einem Neigungswinkel von 0 rad entgegen der Fahrtrichtung und die X-Koordinate nach oben zeigt.

Daher wirkt die Beschleunigung in Fahrtrichtung negativ auf die Z-Achse und die Erdbeschleunigung negativ auf die X-Achse. Im mittleren Teil wird aus dem Neigungswinkel die Rotationsmatrix für eine Rotation um die Radachse gebildet und anschließend mit dem Vektor für die winkelabhängigen Beschleunigungen multipliziert. Somit werden die winkelabhängigen Beschleunigungen auf die X- und Z-Koordinate des Sensors in Abhängigkeit vom Neigungswinkel verteilt. Im unteren Teil wird aus der Winkelgeschwindigkeit des Neigungswinkels die Zentrifugalbeschleunigung in Abhängigkeit vom Einbaupunkt des Sensors berechnet. Diese wirkt immer in Richtung der X-Koordinate des Beschleunigungssensors. Die Beschleunigung auf Grund der Winkelbeschleunigung des Neigungswinkels wird ebenfalls in Abhängigkeit vom Einbaupunkt berechnet und wirkt immer entgegengesetzt der Z-Koordinate des Beschleunigungssensors. Wählt man als Einbauposition genau die Radachse, so sind die winkelunabhängigen Beschleunigungen immer 0. Um Rechenzeit im Mikrocontroller zu sparen, ist das deshalb die ideale Position für diesen Sensor. Da das Sensorsignal eventuell einer Verzögerung unterliegt, wird dieses für jede Koordinate mit einem PT1-Element nachgebildet. Um die Realität näher abzubilden, wird über einen „Random Number-Block“ Rauschen auf jede Koordinate überlagert. Bei diesem Block kann ein Mittelwert in diesem Fall immer 0 und die Varianz des Rauschens angegeben werden. Über „Seed“ wird ein Startwert vorgegeben, welcher bei allen Rauschquellen unterschiedlich sein sollte, da Rauschquellen in der Regel nie gleich rauschen.

Function Block Parameters: Beschleunigungssensor

Beschleunigungssensor (mask)

Der Beschleunigungssensor ist auf dem Pendel montiert. Bei einem Neigungswinkel von 0 rad zeigt seine X-Koordinate in Z-Richtung des Weltkoordinatensystems und seine Z-Koordinate in negativer X-Richtung des Weltkoordinatensystems. Die gemessenen Beschleunigungen sind abhängig vom Abstand des Sensors von der Radachse. Dieser Parameter, so wie die Varianz des Rauschens des Sensors, kann in dieser Maske angegeben werden. Zusätzlich kann die Verzögerungszeit angegeben werden in welcher der Sensor, Änderungen ausgiebt.

Parameters

Einbauabstand des Sensors von der Achse[m]:

Abstand_B

Varianz des Rauschens (Sigma²):

Var_rausch_B

Zeitkonstante des Beschleunigungssensors [s]:

T_B

OK Cancel Help Apply

Abbildung 35: Eingabemaske des Beschleunigungssensors

Über die Eingabemaske des Beschleunigungssensors (siehe Abbildung 35) kann der Abstand des Sensors zur Radachse, die Varianz des Rauschens des Sensors und die Zeitkonstante für die Verzögerung des Ausgangssignals vorgegeben werden.

4.3.4 Das Gyroskop

Das Gyroskop ist ebenfalls auf dem Pendel befestigt und misst die Winkelgeschwindigkeit des Neigungswinkels ψ .

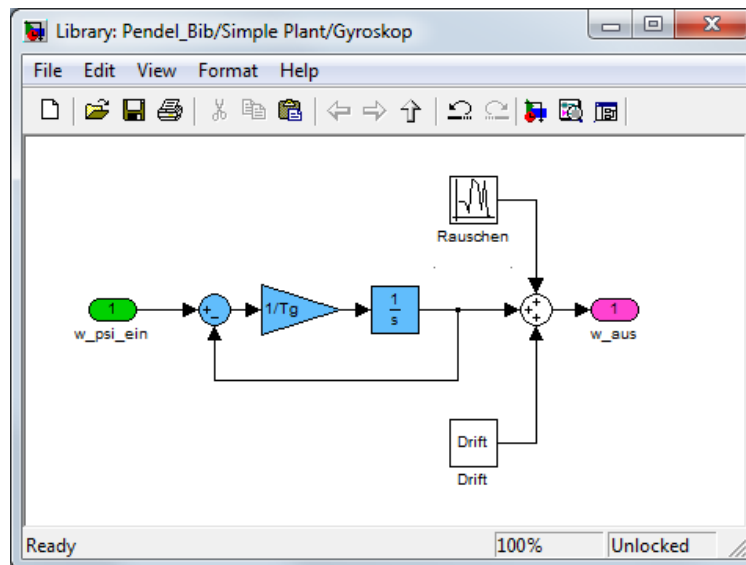


Abbildung 36: Blockschaltbild des Gyroskops

Da diese Messung eventuell einer Verzögerung unterliegt, wird eine solche im Modell durch ein PT1-Element nachgebildet. Dem resultierende Signal wird, wie bereits beim Beschleunigungssensor, ein Rauschen durch einen „Random Number-Block“ überlagert. Zusätzlich wird über die Konstante *Drift* ein Offset des Sensorsignals vorgegeben. Dieses simuliert das beanspruchungs- und temperaturbedingte Wegdriften des Nullpunktes und hat später einen permanenten Winkelfehler zur Folge, welcher über die Sensorfusion kompensiert werden muss.

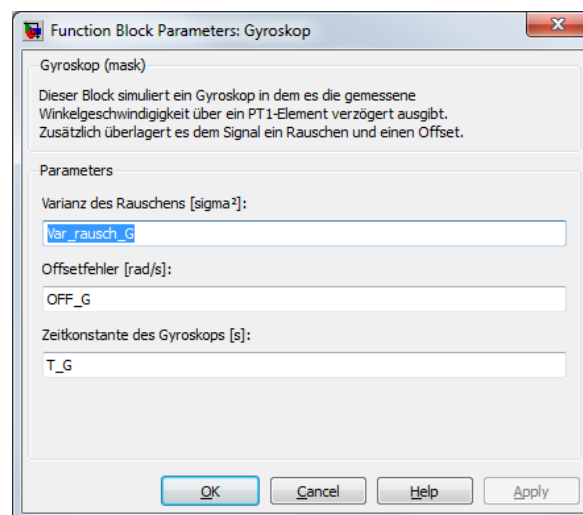


Abbildung 37: Eingabemaske für das Gyroskop

In der Eingabemaske für das Gyroskop (siehe Abbildung 37) kann die Varianz des Rauschens, der Offsetfehler in rad/s² und die Zeitkonstante für die PT1-Verzögerung angegeben werden.

4.3.5 Parameter der Regelstrecke

Erdbeschleunigung [m/s ²]:	Die durch die Gravitation hervorgerufenen Erdbeschleunigung (9,81m/s ²)
X-Koordinate zum Zeitpunkt t = 0 [m]:	Ort zu Beginn der Simulation, die Höhe Z, sowie die Tiefe Y, ist immer 0
Neigungswinkel Psi des Pendels zur Vertikalen, zum Zeitpunkt t=0 [rad]:	Der Neigungswinkel des Pendels zu Beginn der Simulation. Dieser Parameter wird zur Simulation verschiedener Szenarien benötigt.
Winkelgeschwindigkeit von Psi zum Zeitpunkt t=0 [rad/s]:	Die Winkelgeschwindigkeit des Pendels zu Beginn der Simulation. Dieser Parameter wird zur Simulation verschiedener Szenarien benötigt.
Massenträgheitsmoment um Psi [kgm ²]:	Das Massenträgheitsmoment des Pendels (siehe Kapitel 2.2.1.1). Dieser Parameter wird für die Berechnung des mechanischen Modells benötigt.
Masse des Pendel-Körpers [kg]:	Das Gewicht des Pendels. Dieser Parameter wird für die Berechnung des mechanischen Modells benötigt.
Abstand des Schwerpunktes des Pendelkörpers zur Achse [m]:	Der Abstand des Schwerpunktes des Pendelkörpers zur Radachse. Dieser Parameter wird für die Berechnung des mechanischen Modells benötigt.
Masse eines Rades [kg]:	Das Gewicht eines Rades. Dieser Parameter wird zur Berechnung des mechanischen Modells benötigt.
Radius eines Rades [m]:	Der Radius eines Rades. Dieser Parameter wird zur Berechnung des mechanischen Modells benötigt.
Massenträgheitsmoment eines Rades[kgm ²]:	Massenträgheitsmoment eines Rades (siehe Kapitel 2.2.1.1). Dieser Parameter wird für die Berechnung des mechanischen Modells benötigt.
Varianz des Rauschens des Gyroskops [rad/s]:	Die Varianz (das Quadrat der Standardabweichung) des Rauschens des Gyroskops
Offset Fehler des Gyroskops[rad/s]:	Konstanter Messfehler des Gryroskops, wird für die Simulation eines Driftfehlers benötigt.
Zeitkonstante des Gyroskops[s]:	Zeitkonstante für die PT1-Verzögerung des Gyroskops.
Varianz des Rauschens des Beschleunigungssensors [m/s ²]	Die Varianz des Rauschens des Beschleunigungssensors
Abstand des Beschleunigungssensors von der Rad-Achse: [m]	Dieser Parameter wird zur Berechnung der Zentrifugalbeschleunigung am Beschleunigungssensor benötigt.
Zeitkonstante des Beschleunigungssensors [s]:	Zeitkonstante für die PT1-Verzögerung des Beschleunigungssensors
Maximales Moment des Motors [Nm]:	Grenzen der Sättigung für das Motormoment
Zeitkonstante des Motors [s]:	Zeitkonstante für die PT1-Verzögerung des Motors

Tabelle 4: Parameter der Regelstrecke

4.4 Die Regelung mit MATLAB

Neben der Regelstrecke wurde auch der Regler in einem eigenen Subsystem modelliert. Er besteht aus folgenden Komponenten:

- Berechnung des Neigungswinkels aus den Daten des Beschleunigungssensors
- Fusion des aus den Daten des Beschleunigungssensors berechneten Neigungswinkels, mit dem aus der Winkelbeschleunigung integrierten Neigungswinkels
- Zustandsregelung

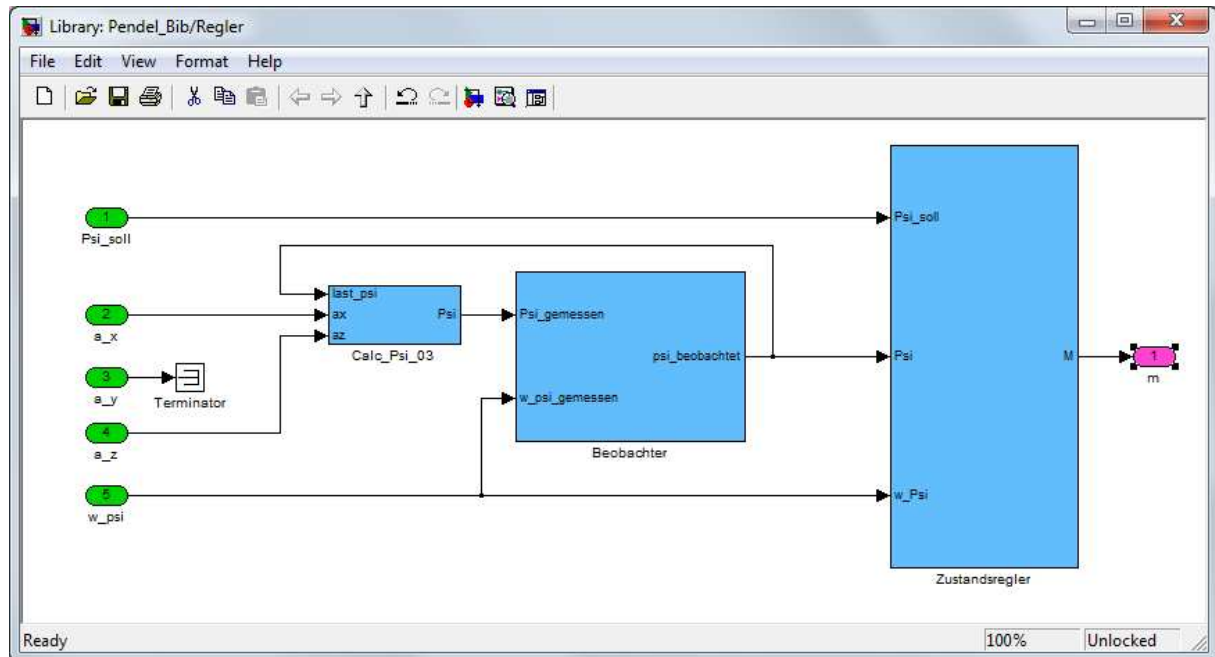


Abbildung 38: Blockschaltbild des Reglers

Für die Auslegung des Reglers musste zuerst die Steuerbarkeit und anschließend die Beobachtbarkeit des Systems festgestellt werden. Erst mit diesen Ergebnissen konnte entschieden werden in welchem Umfang die Regelung realisiert wurde.

4.4.1 Feststellen der Steuerbarkeit

Zum Feststellen der Steuerbarkeit wurde das Modell der Mechanik inklusive Motor in einer MDL-Datei eingebunden. Bei dieser Überprüfung wurde festgestellt, ob die Zustände im Pendel voneinander unabhängig über den Motor gesteuert werden können. Daher konnte auf die Sensoren bei dieser Überprüfung verzichtet werden.

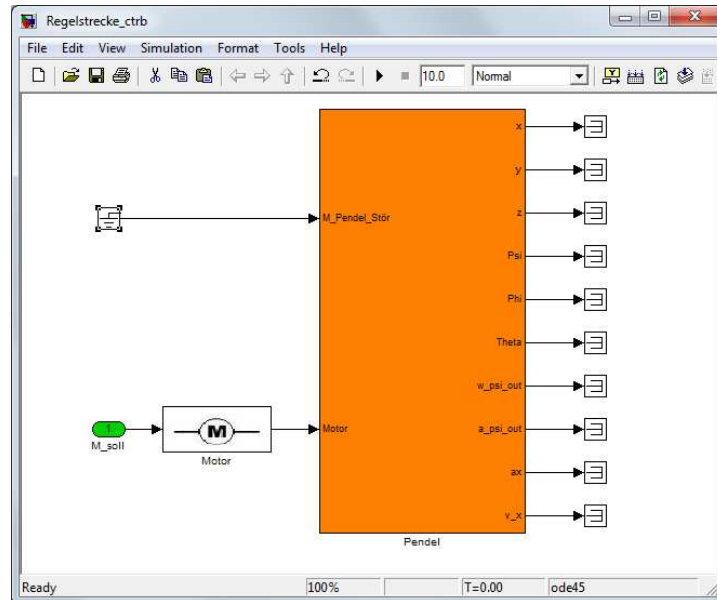


Abbildung 39: Blockschaltbild zum Untersuchen der Regelstrecke

Aus diesem Blockschaltbild wurde über den Befehl *linmod* die Zustandsraumdarstellung erzeugt. Anschließend wurde über den Befehl *ctrbf* die Steuerbarkeit überprüft.

```
sys = linmod('Regelstrecke_ctrb')

[ABAR,BBAR,CBAR,T,K] = ctrbf(sys.a, sys.b, sys.c);
display('Anzahl steuerbarer Zustände: '), sum(K)
```

Über die Summe des Rückgabevektors K kann festgestellt werden, dass die Anzahl an steuerbaren Zuständen gleich der Anzahl im System vorhandener Zustände ist. Dies bestätigt, dass alle Zustände im System voneinander unabhängig über den Motor steuerbar sind.

Das Blockschaltbild und die M-Datei sind zu finden unter:

\02 Matlab Modell\02 Regler\Regelstrecke_ctrb.mdl

\02 Matlab Modell\02 Regler\Steuerbarkeit.m

4.4.2 Feststellen der Beobachtbarkeit

Zum Feststellen der Beobachtbarkeit wurde das vollständige Modell der Regelstrecke in einer eigenen MDL-Datei untergebracht und die für den Betrieb vorgesehenen Ein- und Ausgänge über entsprechende Blöcke verbunden.

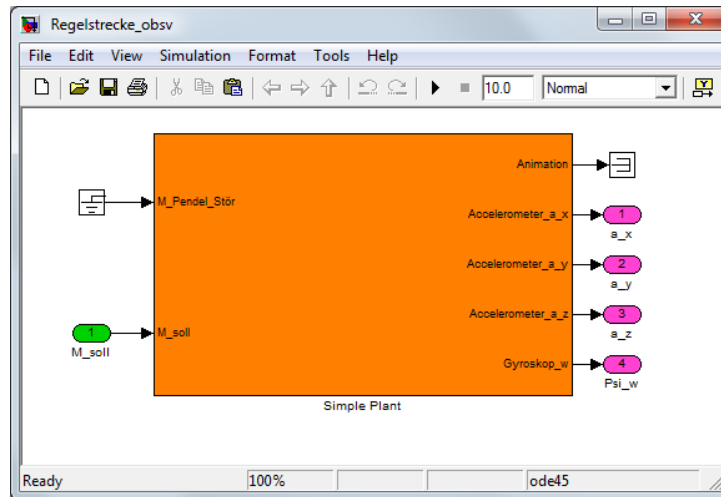


Abbildung 40: Blockschaltbild zur Überprüfung der Beobachtbarkeit

Über den Befehl *linmod* wurde wieder das Zustandsraummodell aus dem Blockschaltbild gewonnen und über den Befehl *obsvf* die Anzahl an beobachtbaren Zuständen bestimmt.

```
sys = linmod('Regelstrecke_obsv')

[ABAR,BBAR,CBAR,T,K] = obsvf(sys.a, sys.b, sys.c);
display('Anzahl beobachtbarer Zustände: '), sum(K)
```

Die Auswertung ergab, dass nur 7 der 9 vorhandenen Zustände im System beobachtbar sind. Dies ist deshalb der Fall, da im Modell keine Verkopplungen des Drehwinkels (θ) noch der Drehwinkelgeschwindigkeit ($\dot{\theta}$) der Räder zum Neigungswinkel des Pendels vorhanden sind und mit dem Beschleunigungssensor und dem Gyroskop nur Informationen über den Neigungswinkel und seine Geschwindigkeit erhalten werden können. Um nachzuweisen, dass der Drehwinkel und seine Geschwindigkeit die nicht beobachtbaren Zustände sind, wurden diese aus dem Modell entfernt und die Beobachtbarkeit erneut bestimmt.

```
sys = linmod('Regelstrecke_obsv_reduced')

[ABAR,BBAR,CBAR,T,K] = obsvf(sys.a, sys.b, sys.c);
display('Anzahl beobachtbarer Zustände: '), sum(K)
```

Von den nun 7 verbliebenen Zuständen waren dann 7 beobachtbar. Aus dieser Überprüfung folgt, dass weder der Ort noch die Geschwindigkeit dieses Modells geregelt werden kann, ohne zusätzliche Sensoren für den Ort und oder der Geschwindigkeit einzubauen.

Die Blockschaltbilder und die M-Datei sind zu finden unter:

\02 Matlab Modell\02 Regler\Regelstrecke_obsv.mdl
 \02 Matlab Modell\02 Regler\Regelstrecke_obsv_reduced.mdl
 \02 Matlab Modell\02 Regler\Beobachtbarkeit.m

4.4.3 Berechnen des Neigungswinkels aus den Beschleunigungen

Aus den Daten des Beschleunigungssensors muss der Neigungswinkel des Pendels berechnet werden. Das Ziel ist es hierbei, die Richtung der Erdbeschleunigung auszuwerten. Diese zeigt auf der Erdoberfläche in guter Näherung mit $9,81\text{m/s}^2$ zum Erdmittelpunkt. Diese Beschleunigung bildet sich auf der X- und Z-Koordinate des Beschleunigungssensors ab. Aus diesen Vektoren kann der Winkel zur X-Koordinate bestimmt werden. Jedoch ist dieser Winkel nur unter der Voraussetzung richtig, dass das mobile Pendel keine Beschleunigung in Fahrtrichtung erfährt und sich entweder der Sensor auf der Radachse befindet oder keine Drehbewegung des Pendels vorherrscht. Da diese Bedingung nur selten erfüllt ist, muss der „echte“ Neigungswinkel aus den Sensordaten heraus gerechnet werden. Bei der Berechnung wird davon ausgegangen, dass der Beschleunigungssensor an der Radachse des Pendels montiert ist, um die gemessenen Anteile durch Zentrifugalbeschleunigung und Winkelbeschleunigungen zu eliminieren. Die Beschleunigungsvektoren werden so nur noch durch die winkelabhängige Erdbeschleunigung und Beschleunigung in Fahrtrichtung gebildet.

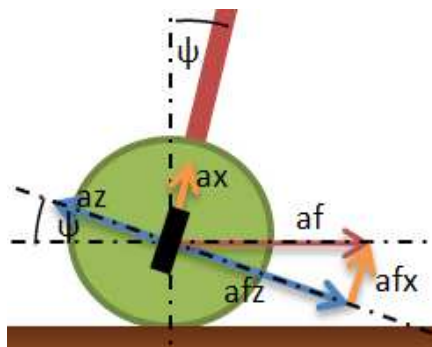


Abbildung 41: Wirkung der Beschleunigung in Fahrtrichtung

Wie aus Abbildung 41 ersichtlich wird, die Beschleunigung in Fahrtrichtung af mit dem Anteil afx , auf die Sensorkoordinate ax und mit dem Anteil afz auf die Sensorkoordinate az abgebildet. Über die trigonometrischen Funktionen können diese berechnet werden. Der schwarze Balken soll den Sensorchip, sowie dessen Montageausrichtung darstellen.

$$af \sin(\psi) = afx \quad (4.4.1)$$

$$af \cos(\psi) = afz \quad (4.4.2)$$

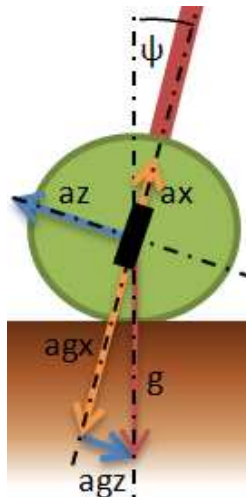


Abbildung 42: Wirkung der Erdbeschleunigung

Die Wirkung der Erdbeschleunigung wird in Abbildung 42 dargestellt. Hier wird ebenfalls der Vektor der Erdbeschleunigung g mit dem Anteil agx auf die Sensorkoordinate ax und mit dem Anteil agz auf die Sensorkoordinate az abgebildet.

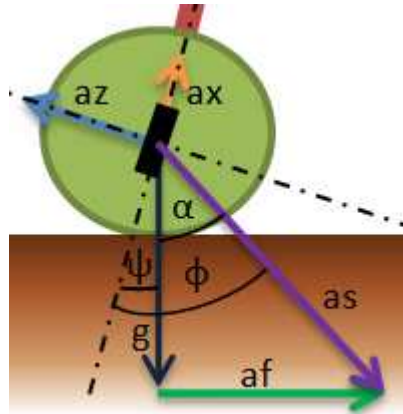
$$g \cos(\psi) = agx \quad (4.4.3)$$

$$g \sin(\psi) = agz \quad (4.4.4)$$

Die durch den Beschleunigungssensor gemessenen Beschleunigungen ergeben sich aus den Gleichungen 4.4.1-4 zur Gleichung 4.4.5.

$$\begin{bmatrix} ax \\ az \end{bmatrix} = \begin{bmatrix} af \sin(\psi) \\ -af \cos(\psi) \end{bmatrix} + \begin{bmatrix} -g \cos(\psi) \\ -g \sin(\psi) \end{bmatrix} \quad (4.4.5)$$

Die Beschleunigung af wirkt zu g immer orthogonal, daher kann der Satz des Pythagoras angewendet werden. Der Vektor as kann aus 2 Gleichungen gebildet werden. Aus der Vektoraddition der X- und Z-Koordinate des Beschleunigungssensors und der Erdbeschleunigung und der Beschleunigung in Fahrtrichtung (siehe Gleichung 4.4.6).



$$as^2 = ax^2 + az^2 = af^2 + g^2 \quad (4.4.6)$$

Der Winkel ϕ ist der Winkel zur X-Koordinate des Sensors.

$$\phi = \text{atan}\left(\frac{az}{ax}\right) \quad (4.4.7)$$

Der Winkel α ist die Verfälschung des Winkels ψ auf Grund der Beschleunigung in Fahrtrichtung.

$$\alpha = \text{atan}\left(\frac{af}{g}\right) \quad (4.4.8)$$

Abbildung 43: Bildung der Winkel

Der Betrag von af kann über Gleichung 4.4.9 berechnet werden.

$$af = \sqrt{as^2 - g^2} = \sqrt{ax^2 + az^2 - g^2} \quad (4.4.9)$$

$$\alpha = \text{atan}\left(\frac{\sqrt{ax^2 + az^2 - g^2}}{g}\right) \quad (4.4.10)$$

Bei der Berechnung kann nur der Betrag von af bestimmt werden, da nur die Existenz einer Beschleunigung in Fahrtrichtung durch die Abweichung von der reinen Erdbeschleunigung festgestellt werden kann. Die Beschleunigung in Fahrtrichtung kann jedoch auch in eine andere Richtung zeigen. Nimmt man Abbildung 43, so könnten die Zeiger g und af auch um as gespiegelt die gleichen Beschleunigungen an ax und az verursachen. Daher muss der korrigierte Winkel für beide Fälle berechnet und über den Vergleich mit dem zuletzt bekannten Winkel der wahrscheinlichste der beiden Fälle gewählt werden. Sollte keine Beschleunigung in Fahrtrichtung existieren, so liefern beide Fälle das gleiche Ergebnis, wodurch das Problem des zunächst unbekannten Winkels nach dem Einschalten des Systems gelöst wird.

$$\psi = \phi \pm \alpha = \text{atan}\left(\frac{az}{ax}\right) \pm \text{atan}\left(\frac{\sqrt{ax^2 + az^2 - g^2}}{g}\right) \quad (4.4.11)$$

Nach dem Einschalten des Systems muss deshalb das Pendel in Ruhelage initialisiert werden. Während der Fahrt wird der Neigungswinkel dann primär aus den Gyroskopdaten ermittelt und sekundär durch die berechneten Werte korrigiert.

Die Gleichung 4.4.11 wurde in MATLAB in einem Simulink-Blockschaltbild realisiert.

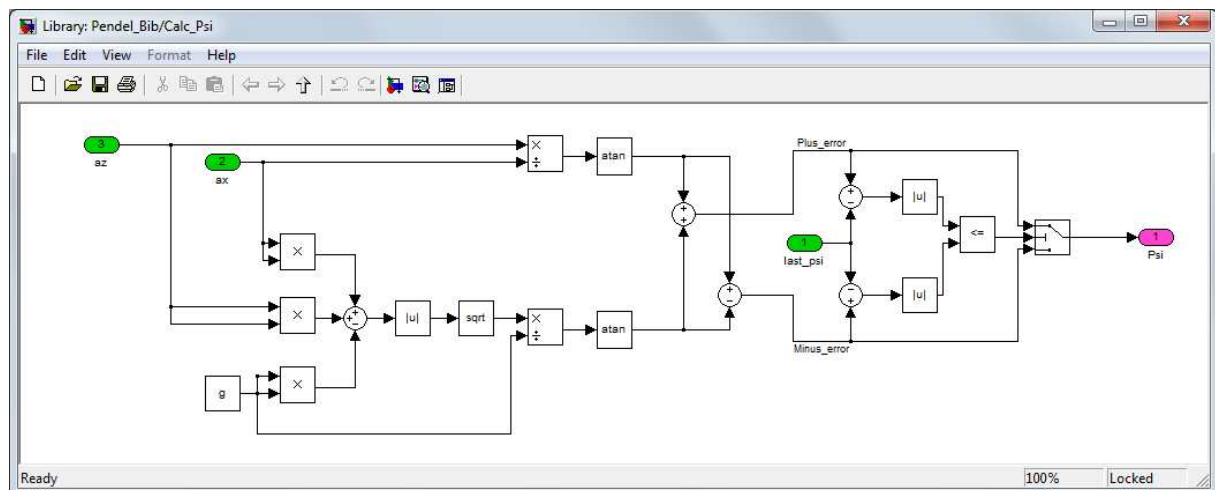


Abbildung 44: Blockschaltbild zur Berechnung des Neigungswinkels

Um die Berechnung zu überprüfen wurde das Umfallen des Pendels ohne Regelung bei einem Neigungswinkel von 0.001 rad simuliert. Bei diesem Versuch treten Beschleunigungen in Fahrtrichtung auf, da die Räder auf Grund der Gewichtskraft weggeschoben werden. Die Ergebnisse (siehe Abbildung 45) zeigen, dass nur zu einem Winkel von ca. 1 rad³ überhaupt eine der beiden Lösungen dem richtigen Wert für den Neigungswinkel entspricht.

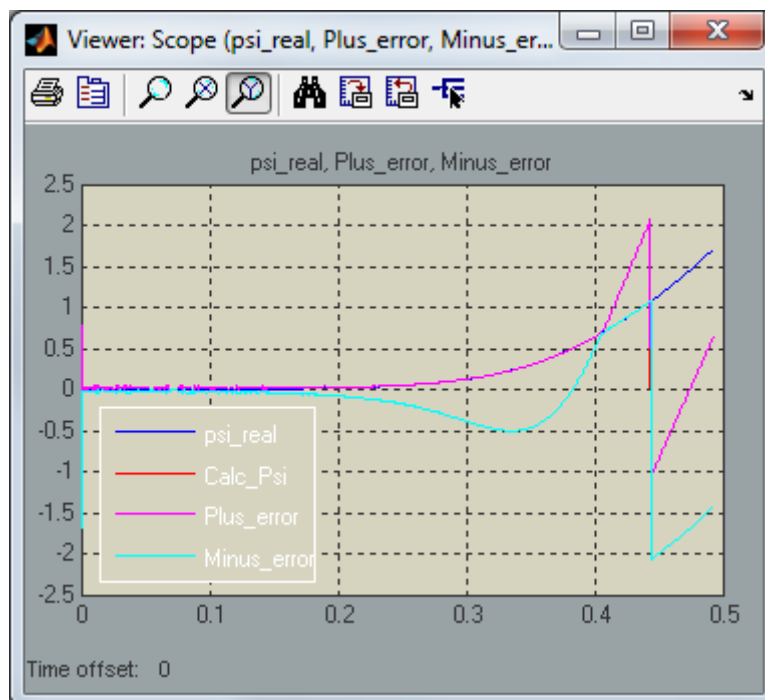


Abbildung 45: Messergebnis der Berechnung des Neigungswinkels

³ 1 rad entspricht ca. 57 Grad.

Eine weitere Möglichkeit, den Neigungswinkel zu berechnen, erhält man, indem man Gleichung 4.4.5 in die Gleichung für die X- und die Z-Koordinate des Sensors zerlegt, anschließend die Gleichung für die X-Koordinate nach af umformt 4.4.12, in die Gleichung für die Z-Koordinate einsetzt 4.4.15 und diese von MATLAB lösen lässt.

$$af = \frac{ax+g \cos(\psi)}{\sin(\psi)} \quad (4.4.12)$$

$$az = -\frac{ax+g \cos(\psi)}{\sin(\psi)} \cos(\psi) - g \sin(\psi) \quad (4.4.13)$$

$$az \sin(\psi) = -ax \cos(\psi) - g (\cos^2(\psi) + \sin^2(\psi)) \quad (4.4.14)$$

$$0 = az \sin(\psi) + ax \cos(\psi) + g \quad (4.4.15)$$

Mit MATLAB wurde diese Gleichung, über den Befehl *solve*, gelöst.

```
>> syms ax az g x, solve('az*sin(x)+ax*cos(x)+g','x')
```

$$\psi = 2 * \operatorname{atan}\left(\frac{az \pm \sqrt{ax^2 + az^2 - g^2}}{ax - g}\right) \quad (4.4.16)$$

Alternativ kann mit MATLAB über den Befehl *mupad* das Programm MuPAD gestartet werden, mit diesem Programm kann nicht nur die Lösung, sondern auch Fallunterscheidungen erhalten werden. Das Ergebnis, die Gleichung 4.4.16, wurde wiederum in ein Simulink-Blockschaltbild umgesetzt. Da auch die Lösung dieser Gleichung mehrdeutig ist, wurde wieder der dem zuletzt bekannten Winkel am nächsten liegende gewählt. Um negative Wurzelargumente zu vermeiden, wurde vor dem Radizieren der Betrag gebildet.

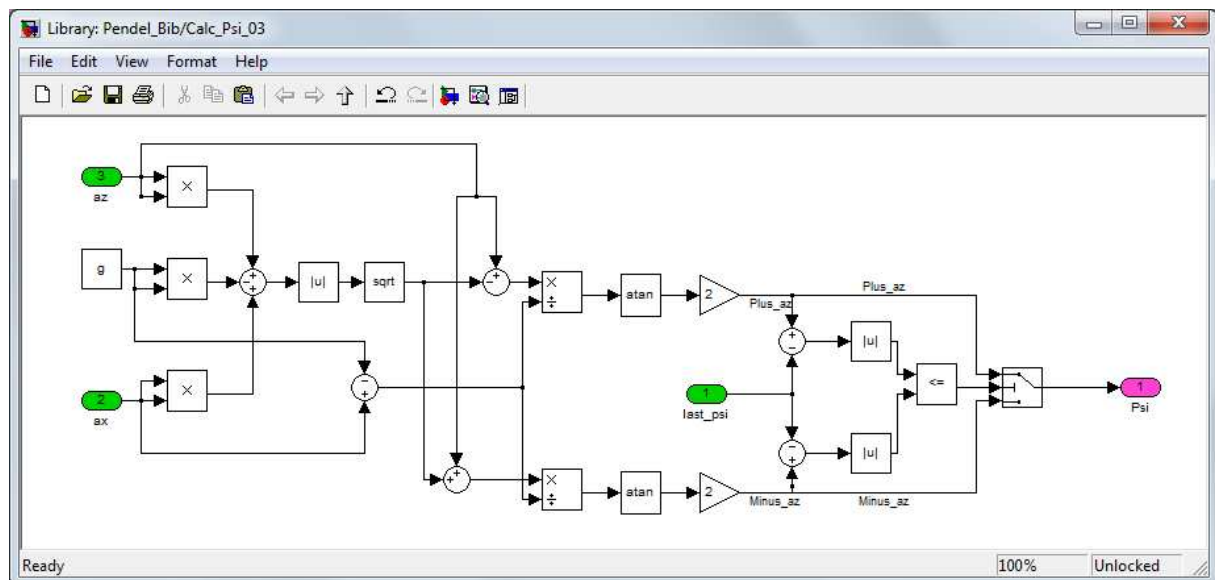


Abbildung 46: Blockschaltbild bessere Berechnung des Neigungswinkels

Testet man diese Gleichung unter den gleichen Bedingungen wie Gleichung (4.4.11), so erhält man Abbildung 47. Dort ist ersichtlich, dass sich die beiden Gleichungen abschnittsweise ergänzen und bis zu einem Winkel von π dem Neigungswinkel entsprechen.

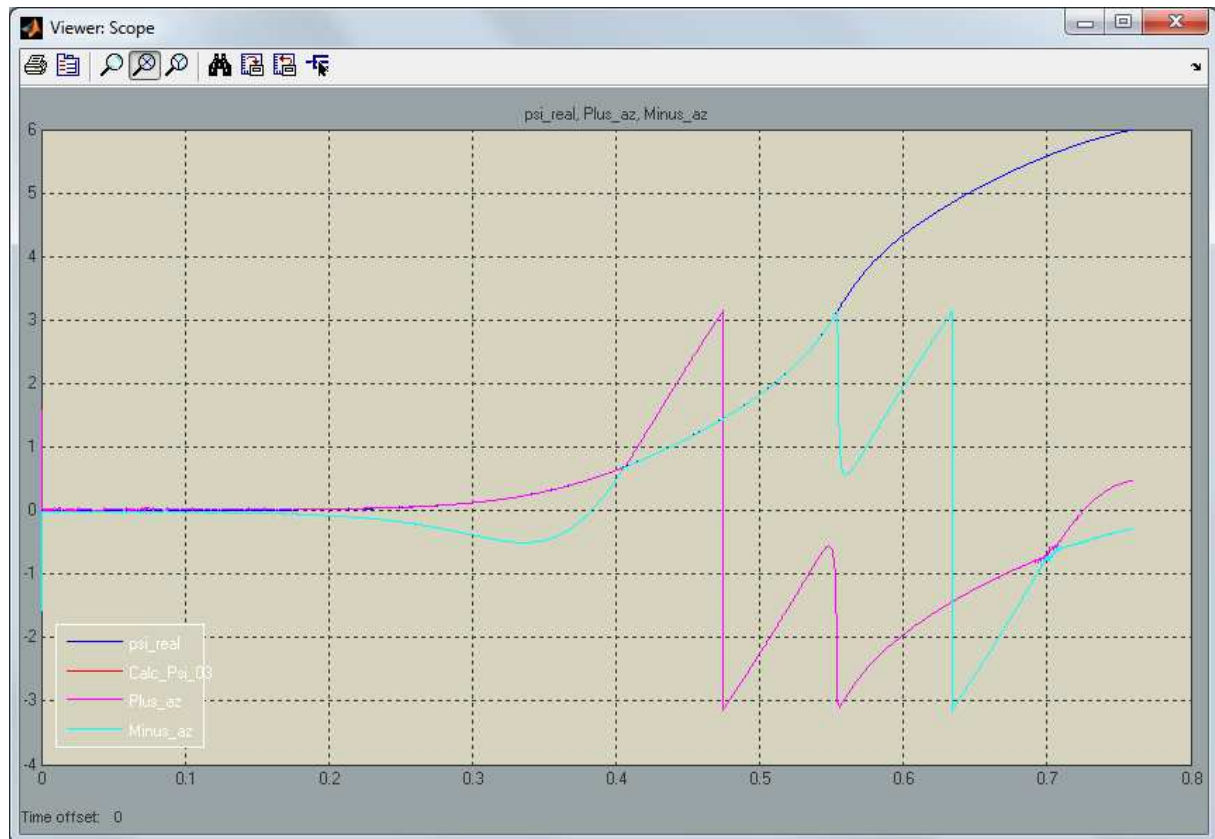


Abbildung 47: Ergebnis der besseren Berechnung des Neigungswinkels

Da für die Regelung des Pendels ein Winkel von 90° völlig ausreichend ist, bietet der zweite Algorithmus mit nur geringem Mehraufwand ein besseres Ergebnis mit mehr Reserve für Beschleunigungen in Fahrtrichtung. Daher wurde dieser Algorithmus für die weitere Arbeit gewählt.

4.4.4 Sensorfusion

Für die Regelung werden die zu regelnden Zustandsgrößen benötigt. Geht man davon aus, dass die Verzögerung im Motor, sowie im Beschleunigungssensor und Gyroskop vernachlässigbar klein sind, bleiben nur noch die Zustände für den Drehwinkel der Räder, die Drehwinkelgeschwindigkeit, der Neigungswinkel und die Neigungswinkelgeschwindigkeit. In Kapitel 4.4.2 wurde nachgewiesen, dass nur die beiden zuletzt genannten Zustände beobachtet werden können. Für diese sind jedoch Sensoren vorhanden, so dass eine Beobachtung nicht zwingend notwendig ist. Die direkte Verwendung der Sensordaten hat jedoch folgende Nachteile:

- Das Rauschen der Sensoren wird ungefiltert über die Regelung zurückgekoppelt und reduziert so die maximal mögliche Regelverstärkung. Wird diese zu groß verursachen ständige Regeleingriffe ein „unruhiges Pendel“.
- Die Bestimmung des Neigungswinkels aus den Daten des Beschleunigungssensors benötigt einen Näherungswert für den Winkel. Dieser ist nicht gegeben bzw. muss gespeichert werden.

Um den letzten Nachteil und ein Teil des ersten zu beseitigen kann die Winkelgeschwindigkeit des Neigungswinkels integriert werden, um so ebenfalls den Neigungswinkel berechnen. Diese Art der Berechnung ist jedoch relativ und konstante Messfehler verursachen über die Zeit immer größere Abweichungen. Aus diesem Grund muss der so bestimmte Winkel über den absoluten Winkel korrigiert werden. Hierzu bietet es sich an den Winkel aus der integrierten Winkelgeschwindigkeit mit dem Winkel des Beschleunigungssensors zu vergleichen und die Differenz auf den Eingang des Integrierers zurückzukoppeln.

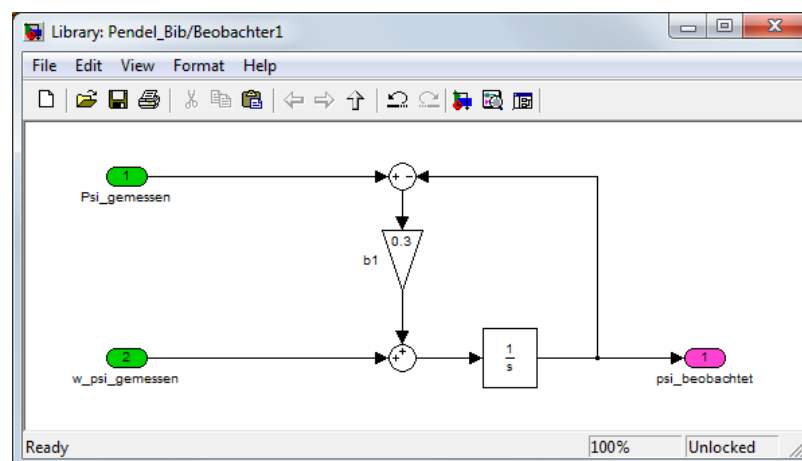


Abbildung 48: Integrieren der Winkelgeschwindigkeit zum Neigungswinkel

In Abbildung 48 ist ersichtlich, dass über den Eingang „w_psi_gemessen“ das Gyroskop angeschlossen wird und die Winkelgeschwindigkeit zum Winkel integriert wird. Der resultierende Winkel wird mit dem Eingang „Psi_gemessen“, dem über die Beschleunigungssensoren berechneten Neigungswinkel, verglichen und die Differenz über einen Faktor $b1$ gegengekoppelt. Dieser Faktor bestimmt wie lange benötigt wird, um einen Fehler zu korrigieren, was mit Gleichung 4.4.17 gezeigt werden kann.

$$\frac{\psi_{\text{beobachtet}}}{\psi_{\text{gemessen}}} = \frac{1}{s+b1} \quad \bullet \text{---} \bigcirc \quad \frac{\psi_{\text{beobachtet}}}{\psi_{\text{gemessen}}} = e^{-b1 t} \quad (4.4.17)$$

Würde man den Faktor $b1$ zu groß wählen, würden Rauschen und kurzzeitige Abweichungen durch z.B. Störungen nicht gefiltert werden. Durch das integrieren der Winkelgeschwindigkeit kann nun einerseits das Rauschen für Messwerte des Neigungswinkels deutlich reduziert werden und andererseits der zuletzt bekannte Winkel zur Berechnung des Neigungswinkels über den Beschleunigungssensor verwendet werden. Befürchtungen, ein anfänglich unbekannter Winkel könne nicht richtig berechnet werden, da ja ein Näherungswert benötigt wird, sind nur begründet, wenn das Pendel in Fahrtrichtung beschleunigt wird, sonst führen beide Berechnungen mit den Beschleunigungssensordaten zur gleichen Lösung. Wird die Regelung des Pendels also in dessen Ruhelage gestartet, kann sich der richtige Winkel einstellen.

4.4.5 Auslegen des Zustandsreglers

Der Zustandsregler soll für den Arbeitspunkt ausgelegt werden, an dem das Pendel senkrecht in Ruhe steht.

$$\psi = 0 \quad (4.4.18)$$

$$\dot{\psi} = 0 \quad (4.4.19)$$

Für den Drehwinkel der Räder und dessen Winkelgeschwindigkeit stehen bei diesem Modell weder Sensordaten noch beobachtete Werte zur Verfügung. Aus diesem Grund kann das zu regelnde Modell auf den Neigungswinkel und dessen Winkelgeschwindigkeit reduziert werden. Auf die Verzögerungen im Motor und in den Sensoren kann ebenfalls verzichtet werden, wenn diese deutlich kleiner als die der Mechanik sind. Dies lässt sich beispielhaft an der Verzögerung im Motor darstellen. Für die Übertragungsfunktion des Motors (siehe Abbildung 29) ergibt sich Gleichung 4.4.20.

$$\frac{M}{M_{soll}} = \frac{\frac{1}{t_m}}{s + \frac{1}{t_m}} \quad (4.4.20)$$

Für ein sehr kleines t_m entsteht ein Pol, der sehr weit in der linken S-Halbebene liegt und für den statischen Fall ($s \rightarrow 0$) geht die Gleichung in 1 über. Für die Mechanik des Pendels genügt es, die Bewegungsgleichung von ψ , 4.2.41 zu verwenden. Bei dieser wurde die algebraische Schleife bereits entfernt, so dass der Neigungswinkel und seine Ableitungen verwendet werden. Es ergibt sich so Gleichung 4.4.22.

$$\ddot{\psi} = \frac{(m_b g L \sin(\psi) - M_s)(m_r R^2 + J_\theta + m_b R^2)}{(m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2 \cos^2(\psi)} \dot{\psi}^2 \quad (4.4.22)$$


Da die Regelung nur am Arbeitspunkt betrachtet wird, kann die Gleichung linearisiert und vereinfacht werden. Die Ansätze 4.4.23-26 führen dann zu Gleichung 4.4.27.

$$\sin(a) \approx a \text{ für kleine } a \quad (4.4.23)$$

$$\cos(a) \approx 1 \text{ für kleine } a \quad (4.4.24)$$

$$a^2 \approx 0 \text{ für kleine } a \quad (4.4.25)$$

$$M_s = 0 \text{ keine Störung} \quad (4.4.26)$$



$$\ddot{\psi} = \frac{\psi m_b g L (m_r R^2 + J_\theta + m_b R^2) - M (m_r R^2 + J_\theta + m_b R^2 + m_b L R)}{(m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2} \quad (4.4.27)$$

$$\psi(s) s^2 = \frac{\psi(s) m_b g L (m_r R^2 + J_\theta + m_b R^2) - M(s) (m_r R^2 + J_\theta + m_b R^2 + m_b L R)}{(m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2} \quad (4.4.28)$$

Durch Transformation der Gleichung 4.4.27 in den Laplace-Raum und Umstellen der Gleichung nach dem Verhältnis Neigungswinkel zu Motormoment erhält man Gleichung 4.4.33.

$$\psi(s) s^2 ((m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2) = \psi(s) m_b g L (m_r R^2 + J_\theta + m_b R^2) - M(s) (m_r R^2 + J_\theta + m_b R^2 + m_b L R) \quad (4.4.29)$$

$$M(s) (m_r R^2 + J_\theta + m_b R^2 + m_b L R) = \psi(s) m_b g L (m_r R^2 + J_\theta + m_b R^2) - \psi(s) s^2 ((m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2) \quad (4.4.30)$$

$$M(s) (m_r R^2 + J_\theta + m_b R^2 + m_b L R) = \psi(s) [m_b g L (m_r R^2 + J_\theta + m_b R^2) - s^2 ((m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2)] \quad (4.4.31)$$

$$\frac{\psi(s)}{M(s)} = \frac{(m_r R^2 + J_\theta + m_b R^2 + m_b L R)}{m_b g L (m_r R^2 + J_\theta + m_b R^2) - s^2 ((m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2)} \quad (4.4.32)$$

$$\frac{\psi(s)}{M(s)} = \frac{\frac{(m_r R^2 + J_\theta + m_b R^2 + m_b L R)}{((m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2)}}{\frac{m_b g L (m_r R^2 + J_\theta + m_b R^2)}{((m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2)} - s^2} \quad (4.4.33)$$

$$s = \pm \sqrt{\frac{m_b g L (m_r R^2 + J_\theta + m_b R^2)}{((m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2)}} \quad (4.4.34)$$

Aus Gleichung 4.4.33 können mit Gleichung 4.4.34 die Pole berechnet werden. Diese befinden sich einmal links und einmal rechts auf der reellen Achse. Deshalb ist immer ein Pol instabil. Durch Multiplikation mit s kann vom Winkel auf die Winkelgeschwindigkeit differenziert werden. Auch die Übertragungsfunktion vom Motor zur Winkelgeschwindigkeit hat die gleichen Pole.

$$\frac{\dot{\psi}(s)}{M(s)} = \frac{\frac{(m_r R^2 + J_\theta + m_b R^2 + m_b L R)}{((m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2)} s}{\frac{m_b g L (m_r R^2 + J_\theta + m_b R^2)}{((m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2)} - s^2} \quad (4.4.35)$$

Um das Pendel zu stabilisieren, muss zumindest der positive Pol über die Rückkoppelfaktoren in die negative Halbebene verschoben werden. Im Folgenden wird davon ausgegangen, dass es sich bei der Zeitkonstante des Motors, sowie der Sensoren, um eine vernachlässigbar kleine handelt. Die Gesamtübertragungsfunktion für den Winkel bei offenem Regelkreis lautet nun:

$$\frac{\psi(s)}{M_{\text{soll}}(s)} = \frac{\frac{(m_r R^2 + J_\theta + m_b R^2 + m_b L R)}{((m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2)}}{-s^2 + \frac{m_b g L (m_r R^2 + J_\theta + m_b R^2)}{((m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2)}} \quad (4.4.37)$$

Mit der Rückkopplung des Neigungswinkels und dessen Winkelgeschwindigkeit mit den Faktoren K_2 und K_1 auf den Motor ergibt sich der geschlossene Regelkreis 4.4.38.

$$\frac{\psi(s)}{w(s)} = \frac{\frac{(m_r R^2 + J_\theta + m_b R^2 + m_b L R)}{(m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2} \cdot \frac{m_b g L (m_r R^2 + J_\theta + m_b R^2)}{(m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2} \cdot s^2}{1 + (K_1 s + K_2) \left(\frac{(m_r R^2 + J_\theta + m_b R^2 + m_b L R)}{(m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2} \cdot \frac{m_b g L (m_r R^2 + J_\theta + m_b R^2)}{(m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2} \cdot s^2 \right)} \quad (4.4.38)$$

$$\frac{\psi(s)}{w(s)} = \frac{\frac{(m_r R^2 + J_\theta + m_b R^2 + m_b L R)}{(m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2}}{\left(\frac{m_b g L (m_r R^2 + J_\theta + m_b R^2)}{(m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2} \cdot s^2 \right) + (K_1 s + K_2) \left(\frac{(m_r R^2 + J_\theta + m_b R^2 + m_b L R)}{(m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2} \right)} \quad (4.4.39)$$

$$\frac{\psi(s)}{w(s)} = \frac{\frac{(m_r R^2 + J_\theta + m_b R^2 + m_b L R)}{(m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2}}{-s^2 + \frac{K_1 (m_r R^2 + J_\theta + m_b R^2 + m_b L R)}{(m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2} s + \frac{K_2 (m_r R^2 + J_\theta + m_b R^2 + m_b L R) + m_b g L (m_r R^2 + J_\theta + m_b R^2)}{(m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2}} \quad (4.4.40)$$

Die Faktoren K_1 und K_2 in der Gleichung 4.4.41 werden durch einen Koeffizientenvergleich bestimmt. Hierfür müssen jedoch zuerst Wunschpole bzw. das gewünschte Ausregelverhalten vorgegeben werden. Eine Koeffizienten-Liste findet sich im Skript von Prof. Kull [9]. Wählt man die „Standard Tranfer Function“ (STF) und gibt eine Ausregelzeit T_{aus} von 50 Millisekunden vor, ergibt sich die Polvorgabe durch die Gleichungen 4.4.43.

$$T_{aus} = 2 T \rightarrow T = 0.025 \text{ sec.} \quad (4.4.42)$$

$$\frac{1}{2} T^2 s^2 + T s + 1 = a_2 s^2 + a_1 s + a_0 = 3.1250 \cdot 10^{-4} s^2 + 0.025 s + 1 \quad (4.4.43)$$

Durch Erweitern um $\frac{2}{T^2}$ kann die Poldarstellung zu Gleichung 4.4.44 umgeformt werden.

$$s^2 + \frac{2}{T} s + \frac{2}{T^2} = a_2 s^2 + a_1 s + a_0 = s^2 + 80 s + 3200 \quad (4.4.44)$$

Erweitert man Gleichung 4.4.40 um -1, kann ein Koeffizientenvergleich der Pole mit Gleichung 4.4.44 durchgeführt werden.

Der Faktor K_1 lässt sich mit Hilfe der Gleichungen 4.4.45-47 bestimmen.

$$-K_1 \frac{(m_r R^2 + J_\theta + m_b R^2 + m_b L R)}{((m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2)} = a_1 \quad (4.4.45)$$

$$K_1 = - \frac{((m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2)}{(m_r R^2 + J_\theta + m_b R^2 + m_b L R)} a_1 \quad (4.4.46)$$

$$K_1 = -80 \frac{1,294 \cdot 10^{-6}}{2,76 \cdot 10^{-3}} = -0.0375 \quad (4.4.47)$$

Für den Faktor K_2 gilt analog der Weg über die Gleichungen 4.4.48-50.

$$- \frac{K_2(m_r R^2 + J_\theta + m_b R^2 + m_b L R) + m_b g L (m_r R^2 + J_\theta + m_b R^2)}{((m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2)} = a_0 \quad (4.4.48)$$

$$K_2 = - \frac{a_0((m_b L^2 + J_\psi)(m_r R^2 + J_\theta + m_b R^2) - m_b^2 L^2 R^2) + m_b g L (m_r R^2 + J_\theta + m_b R^2)}{(m_r R^2 + J_\theta + m_b R^2 + m_b L R)} \quad (4.4.49)$$

$$K_1 = - \frac{3200 \cdot 1,294 \cdot 10^{-6} + 0,4374 \cdot 10^{-3}}{2,76 \cdot 10^{-3}} = -1.6588 \quad (4.4.50)$$

Nach dieser Vorgehensweise wurde die Regelung für die Ausregelzeiten 10ms, 50ms und 100ms berechnet und simuliert. Als Ausgangslage wurde ein Neigungswinkel von 0.5 rad (ca. 28,6°) gewählt. Zusätzlich wurde nach 1 Sekunde Simulationszeit eine Störgröße von 1 Nm gegen das Pendel eingebracht.

Die Blockschaltbilder und die verwendeten M-Dateien sind zu finden unter:

\02 Matlab Modell\02 Regler\Berechnung_der_Faktoren.m

\02 Matlab Modell\03 Gesamtsystem\Gesamtsystem.mdl.m

\02 Matlab Modell\03 Gesamtsystem\Analyse_des_Regelkreises.m

$$T_{\text{aus}} = 100\text{ms}$$

Parameter:

K_v : -0.4874

$K_1(\text{Psi})$: -0.0188

$K_2(w_{\text{Psi}})$: -0.5336

Gemessene Ausregelzeit: 250 ms

Überschwingweite: -0.027 rad

Ausregelzeit der Störgröße: - ms

Bleibende Regelabweichung: - rad

Diese Auslegung des Reglers erzeugt ein langsames Erreichen des Sollwinkels. Das Stellglied wird hierfür nicht an seinen Grenzen betrieben. Die Störgröße kann nicht ausgegletzt werden. Aus diesem Grund ist diese Auslegung nicht empfehlenswert.

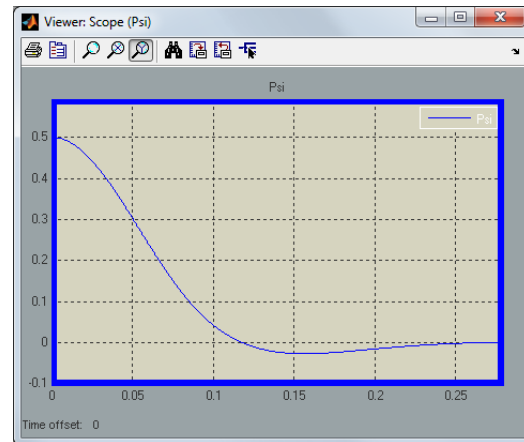


Abbildung 49: Führungsverhalten
($T_{\text{aus}}=100\text{ms}$)

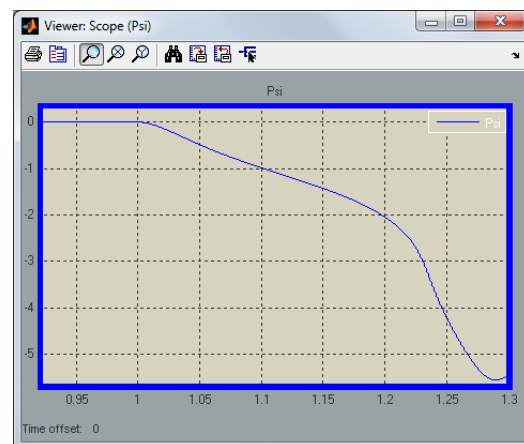


Abbildung 50: Störgrößenverhalten
($T_{\text{aus}}=100\text{ms}$)

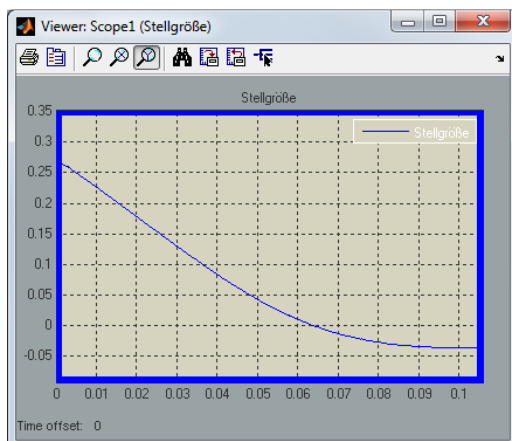


Abbildung 51: Stellgliedbelastung bei Führung
($T_{\text{aus}} = 100\text{ms}$)

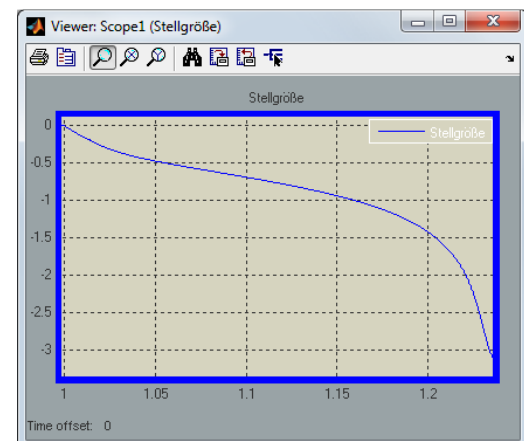


Abbildung 52: Stellgliedbelastung bei Störung
($T_{\text{aus}} = 100\text{ms}$)

$$T_{\text{aus}} = 50\text{ms}$$

Parameter:

K_v : -1.6126

$K_1(\text{Psi})$: -0.0375

$K_2(w_{\text{Psi}})$: -1.6588

Gemessene Ausregelzeit: 130 ms

Überschwingweite: 0.027 rad

Ausregelzeit der Störgröße: 200 ms

Bleibende Regelabweichung: -0.268 rad

Diese Auslegung des Reglers erzeugt ein etwas schnelles Erreichen des Sollwinkels. Das Stellglied wird hierbei nicht annähernd an seinen Grenzen betrieben, weder für die Führungs- noch Störgrößenregelung. Auch die Störgröße kann ausgeglichen werden, jedoch mit einer deutlichen bleibenden Regelabweichung.

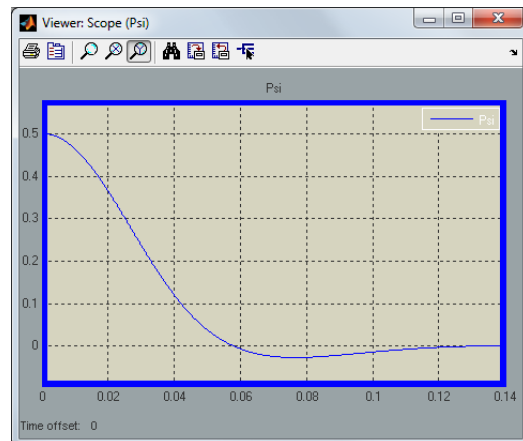


Abbildung 53: Führungsverhalten
($T_{\text{aus}}=50\text{ms}$)

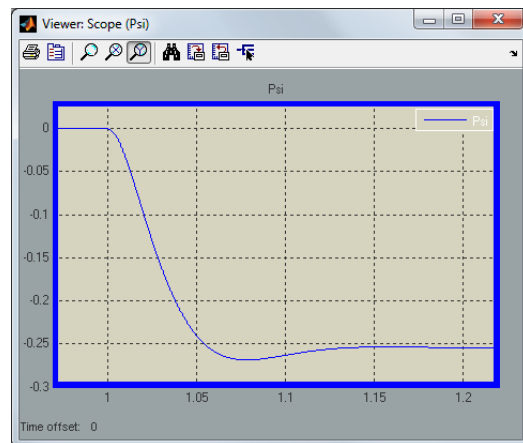


Abbildung 54: Störgrößenverhalten
($T_{\text{aus}}=50\text{ms}$)

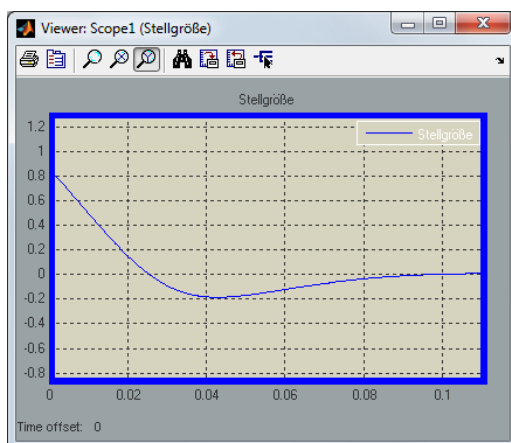


Abbildung 55: Stellgliedbelastung bei
Führung ($T_{\text{aus}} = 50\text{ms}$)

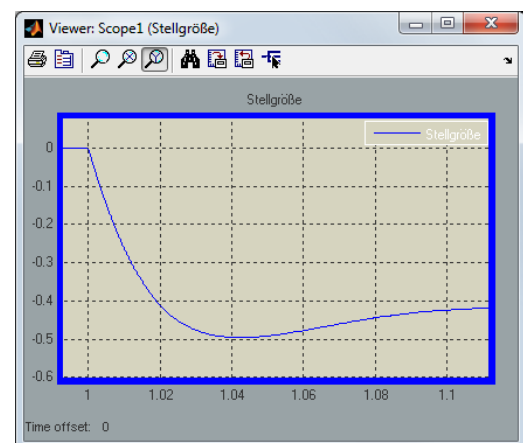


Abbildung 56: Stellgliedbelastung bei
Störung ($T_{\text{aus}} = 50\text{ms}$)

$$T_{aus} = 10ms$$

Parameter:

K_v : -37.6195
 $K_1(\Psi)$: -0.1875
 $K_2(w_\Psi)$: -37.6657

Gemessene Ausregelzeit: 30 ms
 Überschwingweite: 0.0416 rad

Ausregelzeit der Störgröße: 20 ms
 Bleibende Regelabweichung: -0.01 rad

Diese Auslegung des Reglers erzeugt ein schnelles Erreichen des Sollwinkels. Das Stellglied wird hierbei jedoch an seinen Grenzen betrieben. Die Störgröße mit nur einer geringen bleibenden Regelabweichung ausgeregelt werden.

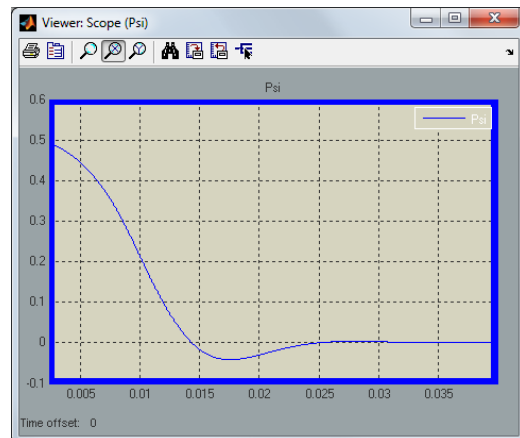


Abbildung 57: Führungsverhalten
 $(T_{aus}=10ms)$

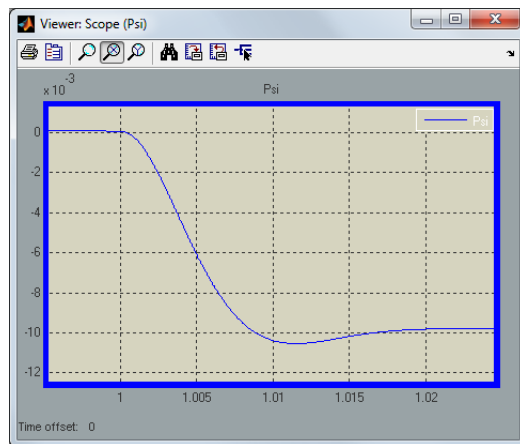


Abbildung 58: Störgrößenverhalten
 $(T_{aus}=10ms)$

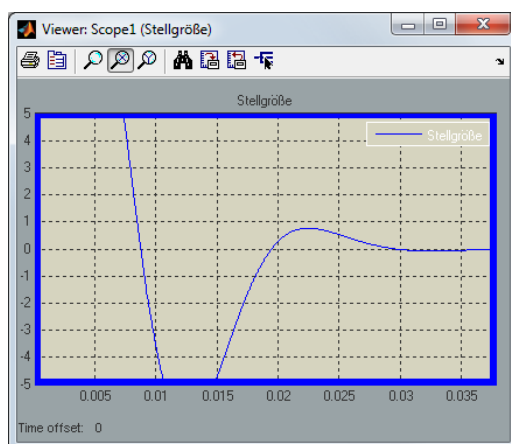


Abbildung 59: Stellgliedbelastung bei Führung
 $(T_{aus} = 10ms)$

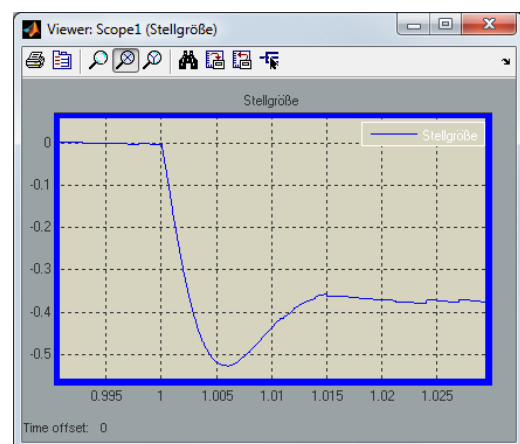


Abbildung 60: Stellgliedbelastung bei Störung
 $(T_{aus} = 10ms)$

Aus den Simulationen ergibt sich, dass die vorgegeben Ausregelzeiten und die daraus resultierenden Wunschkpole nicht genau erreicht werden, jedoch das System stabile Eigenwerte aufweist. Diese Abweichung ist auf die Linearisierung und Vereinfachung zurückzuführen.

Für einen späteren Betrieb des realen Pendels würde sich für die für dieses Modell gewählten Parametern eine Wunsch-Ausregelzeit von 10ms als sinnvoll erweisen. Die resultierende Ausregelzeit von ca. 30ms würde ein Abtastrate von <5ms erfordern.

4.4.6 Die Parameter des Reglers

Erdbeschleunigung [m/s ²]:	Die durch die Gravitation hervorgerufenen Erdbeschleunigung (9,81m/s ²)
Verstärkungsfaktor für den Vorfilter Kv:	Der Korrekturfaktor, um die bleibende Regelabweichung am Arbeitspunkt zu kompensieren
Rückkoppelfaktor K1(w_Psi):	Rückkoppelfaktor der Winkelgeschwindigkeit des Neigungswinkels auf den Motor
Rückkoppelfaktor K2(Psi):	Rückkoppelfaktor des Neigungswinkels auf den Motor
Startwert des beobachteten Winkels Psi [rad]:	Der Wert des berechneten Neigungswinkels zu Beginn der Simulation. In der Realität wird dieser Wert nach dem Einschalten in Ruhelage selbst angenommen.
Zeitkonstante für die Korrektur des berechneten Neigungswinkels [s]:	PT1-Zeitkonstante, mit der über die Daten des Gyroskops integrierten Winkel dem durch den Beschleunigungssensor bestimmten Winkel angeglichen wird.

Tabelle 5: Parameter des Reglers

5 Ergebnisbewertung und Ausblick

Mit der Studienarbeit konnte gezeigt werden, wie die Bewegungsgleichungen eines mechanischen Systems berechnet und wie eine 3D-Animation für eine MATLAB-Simulation erstellt werden kann.

Das Ergebnis der Beobachtbarkeit des errechneten Systems weicht von dem von Herrn Prof. Lunze verfassten Buch, Regelungstechnik 2 [4], ab. In dem dort aufgeführten Beispiel 3.9, wird angegeben, dass alleine durch Bestimmung des Neigungswinkels, auch die Geschwindigkeit in Fahrtrichtung beobachtet werden kann. Für die dort verwendete Zustandsraumdarstellung konnte mit MATLAB diese Aussage bestätigt werden. Da keine Dokumentation vorhanden ist wie dieses Modell erstellt wurde, muss davon ausgegangen werden, dass hier Reibungen oder der Luftwiderstand berücksichtigt wurden. Diese hätten eine geschwindigkeitsabhängige Gegenkraft zu Folge, welche auch den Neigungswinkel beeinflussen würde. Durch diesen Einfluss könnte somit die Geschwindigkeit in Fahrtrichtung beobachtet werden. Dies müsste durch weitere Arbeiten bestätigt werden.

Für den Entwurf der Regelung wurde davon ausgegangen, dass die Verzögerungen im Motor und den Sensoren vernachlässigbar klein sind. Sollte sich für die realen Parameter herausstellen, dass diese nicht deutlich schneller als die der Mechanik sind, so müssen diese in der Regelung berücksichtigt werden. In diesem Fall würde es sich lohnen, einen Zustandsbeobachter zu verwenden, um diese Zustände zu schätzen und deren Pole bei Bedarf durch Rückkopplungen schneller zu machen.

Die verwendete Zustandsregelung basiert auf einer P-Regelung, daher besteht immer eine bleibende Regelabweichung. Diese könnte in weiteren Entwicklungsschritten durch einen I-Bypass oder einen PI-Zustandsregler beseitigt werden.

Um die Regelung im realen Pendel zu nutzen muss diese diskretisiert werden. Die notwendige Abtastzeit muss zuvor aus der zeitkontinuierlichen Regelung und den Parametern des realen Pendels gewonnen werden. Für die Einbindung der diskretisierten Regelung in MATLAB bietet sich eine Level 2 S-Function an. In dieser wird eine Art C-Syntax verwendet, was eine Portierung der diskretisierten Regelung in den Mikrocontroller vereinfachen könnte.

Bei den Messungen der Zustandsregelung konnte festgestellt werden, dass der Neigungswinkel möglichst genau bestimmt werden muss. Vor Inbetriebnahme des realen Pendels sollte daher die Wirkrichtung der Sensoren und des Motors überprüft werden. Ferner sollte die Korrektur des Winkels über die Beschleunigungssensoren überprüft werden. Sollte diese abweichen, kann die Regelung nicht stabil arbeiten.

Literaturverzeichnis

- [1] Hering, Martin, Stohrer: Physik für Ingenieure, 2004, Springer-Verlag, 9. Auflage
- [2] Nolting, Wolfgang: Grundkurs Theoretische Physik 2: Analytische Mechanik, 2006, Springer-Verlag, 7. Auflage
- [3] Wolf Dieter Pietruszka: MATLAB und Simulink in der Ingenieurpraxis :Modellbildung, Berechnung und Simulation, 2006, Vieweg+Teubner, 2.Auflage
- [4] Prof. Dr. –Ing. Jan Lunze: Regelungstechnik 2, 2005, Springer-Verlag, 3. Auflage
- [5] Yoriyisa Yamamoto: NXTway-GS Model-Based Design, 2008, CYBERNET SYSTEMS CO., LTD.
- [6] Andreas Formella: Parameter Estimation des Inversen Pendels mit Hilfe eines Kalman-Filters, 2009
- [7] Lothar Papula: Mathematische Formelsammlung, 2009, Vieweg & Sohn Verlag, 9.Auflage
- [8] MathWorks, Inc.: MATLAB-Hilfe, R2009b
 - [8.1] Befehl ctrb: \00 Informationen\03 Regelungstechnik\Befehl_ctrb.xps
 - [8.2] Befehl place: \00 Informationen\03 Regelungstechnik\Befehl_place.xps
 - [8.3] Befehl dcgain: \00 Informationen\03 Regelungstechnik\Befehl_dcgain.xps
 - [8.4] Befehl obsv: \00 Informationen\03 Regelungstechnik\Befehl_obsv.xps
 - [8.5] Figure_Properties: \00 Informationen\03 Regelungstechnik\Figure_Properties.xps
 - [8.6] Axes_Properties: \00 Informationen\03 Regelungstechnik\Axes_Properties.xps
- [9] Prof. Dr. –Ing. Herman Kull: Skript Systemtechnik 2, Sommersemester 2010

Abbildungen

Abbildung 1: Modell des inversen Pendels mit Wagen und Stab [6].....	3
Abbildung 2: Mobiles, inverses Pendel [5]	3
Abbildung 3: Beispiel für quaderförmige Objekte mit homogener Masseverteilung [1]	4
Abbildung 4: Beispiel Hangabtriebskraft [2]	5
Abbildung 5: Einfaches Beispiel für ein MIMO-System	8
Abbildung 6: Beispiel Rangbestimmung.....	10
Abbildung 7: Blockschaltbild Regelungsnormalform (allgemeiner Fall: $m = n$) [9]	11
Abbildung 8: Beispiel allgemeines Blockschaltbild mit Zustandsrückführung [9]	12
Abbildung 9 : Zustandsbeobachter in Beobachternormalform [9]	15
Abbildung 10: Anpassung Rückkoppelfaktoren für den Zustandsbeobachter[9].....	16
Abbildung 11: Schaltfläche des Library Browsers	17
Abbildung 12: Fundort des Subsystem-Blocks.....	17
Abbildung 13: Beispiel für den Inhalt eines Subsystems.....	18
Abbildung 14: Dropdown-Menü-Eintrag 'Edit Mask' eines Subsystems	18
Abbildung 15: Reiter "Icon&Ports" des Mask Editors	18
Abbildung 16: Mask Editor, anlegen von Parametern(links Konfiguration, rechts Ergebnis) .	19
Abbildung 17: Zusammenfassen der Parameter zu einer Struktur bei der Initialisierung	19
Abbildung 18: Dokumentation eines Subsystem-Blocks	20
Abbildung 19: Fundort S-Function Block.....	23
Abbildung 20: Einem S-Function-Block ein m-File zuweisen	23
Abbildung 21: Übersicht Sollkonzept.....	24
Abbildung 22: Animation	26
Abbildung 23: Inhalt der Animation	27
Abbildung 24: Segment eines Rades	32
Abbildung 25: Inhalt der Real-time Synchronisation	36
Abbildung 26: Test der Animation	37
Abbildung 27 : Zeichnung Inverses Pendel [5].....	38
Abbildung 28: Aufbau der Regelstrecke.....	44
Abbildung 29: Blockschaltbild des Motors	45
Abbildung 30: Eingabemaske für den Motor	45
Abbildung 31: Ausschnitt aus dem Blockschaltbild für das Pendel	46
Abbildung 32: Motormoment und Störgröße	46
Abbildung 33: Berechnung der Energieerhaltung	47
Abbildung 34: Blockschaltbild des Beschleunigungssensors.....	48
Abbildung 35: Eingabemaske des Beschleunigungssensors.....	49
Abbildung 36: Blockschaltbild des Gyroskops.....	50
Abbildung 37: Eingabemaske für das Gyroskop	50
Abbildung 38: Blockschaltbild des Reglers	52
Abbildung 39: Blockschaltbild zum Untersuchen der Regelstrecke	53
Abbildung 40: Blockschaltbild zur Überprüfung der Beobachtbarkeit	54
Abbildung 41: Wirkung der Beschleunigung in Fahrtrichtung	55
Abbildung 42: Wirkung der Erdbeschleunigung	55
Abbildung 43: Bildung der Winkel	56
Abbildung 44: Blockschaltbild zur Berechnung des Neigungswinkels	57
Abbildung 45: Messergebnis der Berechnung des Neigungswinkels.....	57
Abbildung 46: Blockschaltbild bessere Berechnung des Neigungswinkels.....	58
Abbildung 47: Ergebnis der besseren Berechnung des Neigungswinkels	59
Abbildung 48: Integrieren der Winkelgeschwindigkeit zum Neigungswinkel.....	60
Abbildung 49: Führungsverhalten ($T_{\text{aus}}=100\text{ms}$)	65
Abbildung 50: Störgrößenverhalten ($T_{\text{aus}}=100\text{ms}$)	65
Abbildung 51: Stellgliedbelastung bei Führung ($T_{\text{aus}} = 100\text{ms}$).....	65
Abbildung 52: Stellgliedbelastung bei Störung ($T_{\text{aus}} = 100\text{ms}$).....	65
Abbildung 53: Führungsverhalten ($T_{\text{aus}}=50\text{ms}$)	66

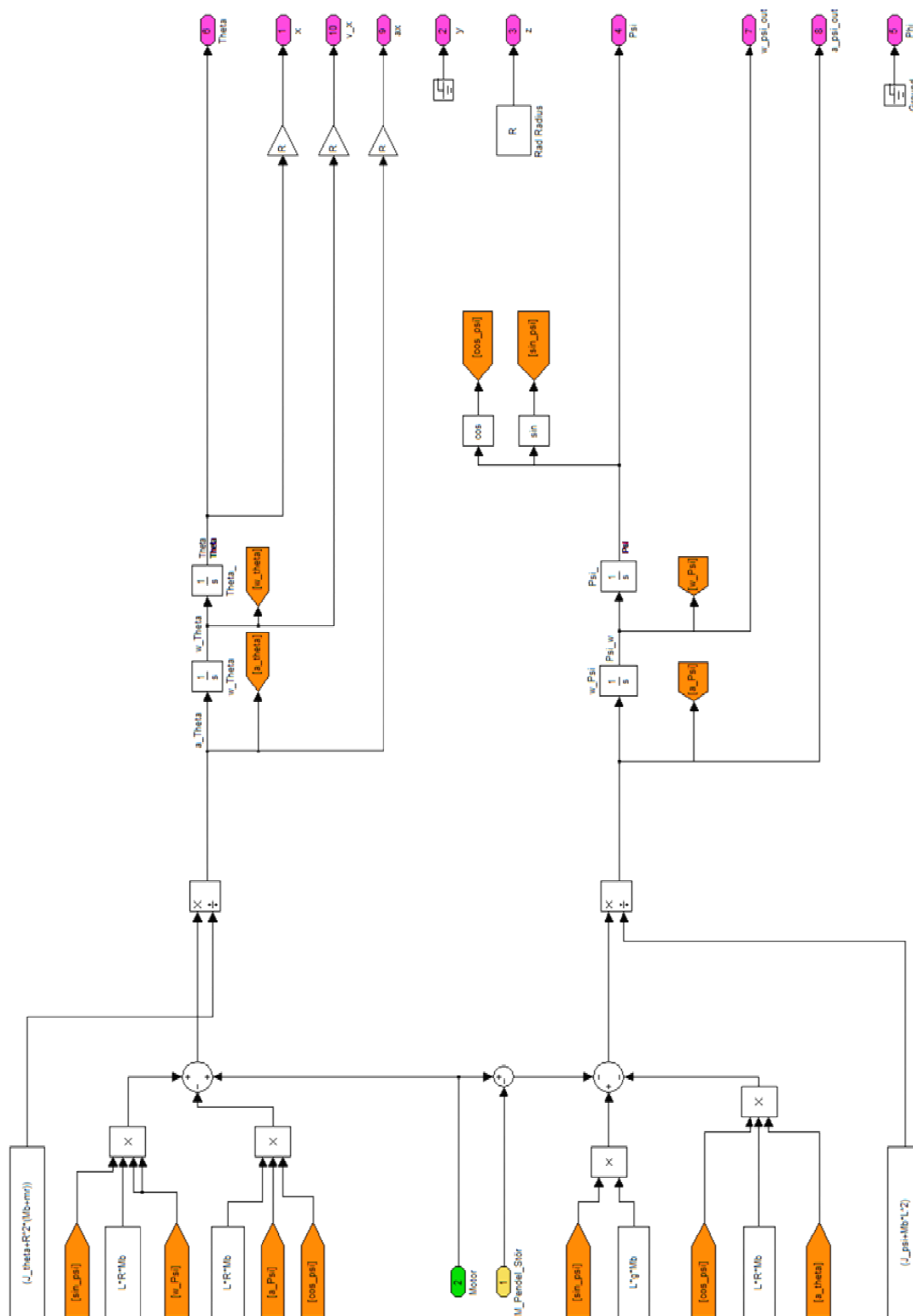
Abbildung 54: Störgrößenverhalten ($T_{\text{aus}}=50\text{ms}$)	66
Abbildung 55: Stellgliedbelastung bei Führung ($T_{\text{aus}} = 50\text{ms}$)	66
Abbildung 56: Stellgliedbelastung bei Störung ($T_{\text{aus}} = 50\text{ms}$)	66
Abbildung 57: Führungsverhalten ($T_{\text{aus}}=10\text{ms}$)	67
Abbildung 58: Störgrößenverhalten ($T_{\text{aus}}=10\text{ms}$)	67
Abbildung 59: Stellgliedbelastung bei Führung ($T_{\text{aus}} = 10\text{ms}$)	67
Abbildung 60: Stellgliedbelastung bei Störung ($T_{\text{aus}}= 10\text{ms}$)	67

Tabellen

Tabelle 1: Verwendete Begriffe und Abkürzungen.....	8
Tabelle 2: Parameter der Animation.....	37
Tabelle 3: Bezeichnung der Größen	38
Tabelle 4: Parameter der Regelstrecke.....	51
Tabelle 5: Parameter des Reglers.....	68

6 Anhang:

6.1 Blockschaltbild der Regelstrecke



6.2 Blockschaltbild der Regelstrecke (ohne algebraische Schleife)

