Ausarbeitung zum Projektseminar Echtzeitsysteme, WS16/17

Team KnightRiders
Projektseminar eingereicht von
Daniel Hueske, Nicolas Himmelmann, Moritz Weissenberger,
Christian Müller, Alexander Altmann
am 12. April 2017

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachgebiet Echtzeitsysteme

Elektrotechnik und Informationstechnik (FB18)

Zweitmitglied Informatik (FB20)

Prof. Dr. rer. nat. A. Schürr
Merckstraße 25
64283 Darmstadt

www.es.tu-darmstadt.de

Gutachter: Géza Kulcsar
Betreuer: Géza Kulcsar

Erklärung zum Projektseminar

Hiermit versichere ich, das vorliegende Projektseminar selbstständig und ohne Hilfe Dritter angefertigt zu haben. Gedanken und Zitate, die ich aus fremden Quellen direkt oder indirekt übernommen habe, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen und wurde bisher nicht veröffentlicht.

Ich erkläre mich damit einverstanden, dass die Arbeit auch durch das Fachgebiet Echtzeitsysteme der Öffentlichkeit zugänglich gemacht werden kann.

Darmstadt, den 12. April 2017

(Daniel Hueske, Nicolas Himmelmann, Moritz Weissenberger, Christian Müller, Alexander Altmann)

Inhaltsverzeichnis

1 Einleitung

1.1 Zielsetzung des Seminars

Ziel des Projektseminars Echtzeitsysteme ist es, ein mit Sensorik ausgestattetes Modellfahrzeug zu programmieren, sodass es autonom und möglichst sicher verschiedene Szenarien absolvieren kann.
Welche Gruppe dabei die beste Lösung erarbeitet hat, wird anschließend in einem Live-Wettbewerb im Hans-Busch-Institut entschieden. Außerdem soll man während des Seminars verschiedene Soft Skills erlernen und ausbauen, dazu zählen die Gruppenorganisierung und Zeitplanung eines Projekts.

1.2 Abgrenzung der Aufgabenstellung

Eine feste Aufgabenstellung, die alle Gruppen absolvieren mussten, gab es nicht. Nur eine Teilaufgabe musste von allen Gruppen implementiert werden: Das autonome Fahren eines Rundkurses ohne Hindernisse. Die weiteren Aufgaben konnten nach Absprache mit dem Betreuer Géza Kulcsar frei festgelegt werden. Unsere Gruppe entschied sich dabei für „Autonomes Einparken“ sowie „Rundkurs mit Hindernissen“.

1.3 Organisation unserer Gruppe

Der Regelungstechniker unserer Gruppe hat während des Seminars einen fundierten mathematischen Kaskadenregler für das Auto ausgelegt, während die anderen 4 Teilnehmer sich aufgeteilt haben: Zwei haben sich mit dem Thema „Autonomes Einparken“ beschäftigt und die anderen zwei mit dem Rundkurs mit Hindernissen.
In regelmäßigen Abständen hat unsere Gruppe sich dann getroffen und den aktuellen Stand besprochen.

2 Fahrzeug, ROS, Sensorik

2.1 Das Fahrzeug

Das Fahrzeug besteht aus einem Chassis der Firma Tamiya, worauf sich ein Aufbau befindet, der die Hardware der Plattform beinhaltet. Zu Beginn des Seminars haben wir uns zunächst mit dem Fahrverhalten des Fahrzeugs vertraut gemacht und verschiedene Eigenschaften vermessen.
Zur Steuerung des Fahrzeugs steht das ROS-Paket „PSES_Basis“ zur Verfügung, welches eine grafische Oberfläche enthält, mit der man Befehle an das Fahrzeug senden kann.
Die in dieser Oberfläche verwendete Größe für die Lenkung ist nicht ein Winkel in Grad, sondern Ticks im Intervall von [-50, 50]. Dabei entspricht ein eingestellter Wert von 50 einem Volleinschlag nach links und -50 einem Volleinschlag nach rechts.

2.1.1 Lenkwinkel-Mapping und Vermessung

Da die im folgenden Kapitel 3 erläuterte Regelung jedoch Lenkwinkel in Grad als Stellgröße ausgibt, war es nötig, ein Mapping der Lenk-Ticks auf den vom Servo eingestellten Winkel in Grad zu bestimmen. Dabei haben wir das Auto für jeden fünften Lenk-Tick mehrmals einen Halbkreis fahren lassen und haben aus dem Mittelwert der Durchmesser dann den Lenkwinkel bestimmt. Die Werte dazwischen haben wir in Matlab mit einem neuronalen Netz interpoliert.
Dabei ist uns aufgefallen, dass die Durchmesser der Wendekreise nach links und rechts extrem variieren, was wir dann in unserem Algorithmus berücksichtigen mussten.
Desweiteren haben wir noch verschiedene andere Eigenschaften des Autos vermessen, darunter Länge, Breite, Höhe, Spurweite und Radius der Reifen.

2.2 ROS-Grundlagen

Die auf dem Auto verwendete Software ist ROS („Robot Operating System“). ROS ist eine Middleware, also ein Softwarepaket, das auf einem anderen Betriebssystem (in diesem Fall Lubuntu) läuft und verschiedene Funktionalitäten bereitstellt. Außerdem gibt es bereits viele fertige Pakete, die Algorithmen aus dem Bereich der Robotik und Lokalisierung implementieren. Ein sinnvoller Einstieg sind die ROS-Tutorials unter http://wiki.ros.org/ROS/Tutorials.
Für unsere Lösung der Aufgaben haben wir jedoch hauptsächlich eigenen Code geschrieben, und nur das Message-Passing-System von ROS benutzt.

2.2.1 Message-Passing-System

Der wahrscheinlich wichtigste Teil von ROS ist das Message-Passing-System. Es arbeitet nach dem Publish/Subscribe-Prinzip, das heißt, dass verschiedene ausführbare Software-Komponenten untereinander asynchron Nachrichten austauschen können. Das funktioniert folgendermaßen: Es gibt Topics, was soviel heißt wie „Kanal“ oder „Thema“. Auf diesem Topic werden Nachrichten („Messages“) ausgetauscht, welche immer von einem vorher bestimmten Typ sind (z.B. std_msgs::String oder sensor_msgs::LaserScan).
Ein ROS-Node kann sich nun als Publisher (Versender) oder Subscriber (Empfänger), oder beides, an dem Topic „anmelden“. Sobald ein Publisher eine Nachricht auf einem Topic veröffentlicht, werden alle Subscriber benachrichtigt und im Code wird das zu der Subscription gehörende Callback aufgerufen, in dem man die Nachricht weiterverarbeiten kann.

2.2.2 Organisierung unserer ROS-Nodes

Ziel war es, unsere ROS-Nodes so zu implementieren, dass jeder Node eine Klasse darstellt, welche einen Zustand und einen Haupttimer besitzt. In diesem Haupttimer wird der Ablauf des jeweiligen Algorithmus kontrolliert und bei Bedarf können andere Timer gestartet und gestoppt werden, die andere Aufgaben erledigen sollen.
Wir haben für unser Projekt folgende ROS-Nodes angelegt:

- KinectProcessing
  Ist dafür zuständig, jeden neu empfangenen LaserScan zu filtern und in kartesische Koordinaten umzurechnen (dazu mehr in Kapitel 4).

- RCBoard
  Ist dafür zuständig, die Signale einer Funkfernbedienung und vom Drehgeber (siehe Abschnitt 2.3.1) zu verarbeiten, die von einem Arduino geliefert werden.

- WallFollow
  Enthält die Regelung für den Wandabstand, welche bei Bedarf aktiviert oder deaktiviert werden kann. Siehe Kapitel 3.

- AvoidObstacles
  Enthält unseren Algorithmus für das Erkennen und Umfahren von Hindernissen, sowie für das Erkennen von Kurven. Siehe Kapitel 4.

- Parking
  Enthält unseren Algorithmus für das autonome parallele Einparken. Siehe Kapitel 5.

Man kann per RC-Fernbedienung oder über die PSES-App einen Modus auswählen, der aktiviert werden soll. Folgende Möglichkeiten gibt es in unserem ROS-Package:

- Roundtrip
  Startet die Regelung des Wandabstandes, und hält gleichzeitig Ausschau nach Hindernissen und Kurven. Wird dieser Modus gewählt, fährt das Fahrzeug mit maximaler Geschwindigkeit.

- Roundtrip with Obstacles
  Dasselbe wie bei „Roundtrip“, aber mit einer langsameren Geschwindigkeit, da dieser Modus für den Hindernisrundkurs gedacht ist.

- Park Car
  Stoppt das Auto, falls es gerade fährt, und sucht mit langsamer Geschwindigkeit nach einer Parklücke. Dabei fährt es solange geradeaus, bis mit dem rechten Ultraschallsensor eine Parklücke gefunden wurde.

2.3 Sensorik

2.3.1 Hall-Sensor ersetzt durch Drehgeber

Den in [1] Kapitel 1.3.2 erklärten Hall-Sensor des Fahrzeugs haben wir für unsere Zwecke durch einen kontinuierlichen Drehgeber vom Typ KY-040 ersetzt. Diesen haben wir an einen Arduino Nano Mikrocontroller angeschlossen, welcher über die serielle Schnittstelle mit dem Mainboard

kommuniziert und die gezählten Pulse übermittelt. Der im Auto vorhandene Hall-Sensor liefert nur einen Impuls pro Achtelumdrehung des Rades, der von uns eingebaute Drehgeber hat jedoch eine Auflösung von 40 Ticks/360°. Demnach kann die gefahrene Strecke damit genauer bestimmt werden. Diese Maßnahme ist wahrscheinlich nicht nötig, wenn man die gefahrene Strecke zwischen den Impulsen mit einem Schätzer ermittelt, welcher auf Basis der Geschwindigkeit und anderen Informationen arbeitet.

2.3.2 Kinect-Tiefenbild als Laserscan und Filterung

Die auf dem Fahrzeug vorhandene Microsoft Kinect liefert ein Tiefenbild in verschiedenen Auflösungen, in dem die Entfernungen der Objekte im Raum durch Graustufen dargestellt sind. Für unsere Zwecke hat die niedrigste Auflösung ausgereicht (SD, 512x424 Pixel), bei höheren Auflösungen wurde die Verarbeitungszeit des Bildes zu groß. Das hat vor allem bei höheren Geschwindigkeiten den Nachteil, dass die im Algorithmus verwendeten Daten nicht mehr mit der realen Situation übereinstimmen.
Das ROS-Paket „depthimage_to_laserscan“ nimmt einen Ausschnitt aus diesem Tiefenbild und liefert einen Laserscan als ROS-Sensor-Message-Typ „sensor_msgs::LaserScan“. Der dafür verwendete Ausschnitt kann in der Launch-File mit bestimmten Parametern festgelegt werden (siehe Launch-File im Anhang).
Ein Laserscan ist im Prinzip nichts anderes, als ein Array mit Entfernungswerten. Zwischen diesen „Entfernungsstrahlen“ liegt ein Winkel, welcher sich aus den Angaben im Datenblatt ermitteln lässt: Winkel zwischen zwei Strahlen = Gesamtanzahl der Strahlen / Maximaler Blickwinkel der Kamera.
Dadurch, dass das ursprüngliche Tiefenbild weder gefiltert noch anderweitig bearbeitet ist, ist der gelieferte Laserscan mit verschiedenen Fehlern behaftet:

- Auftreten von Fehlern bei Sprüngen in den Entfernungswerten:
  Bei diesen Sprüngen werden, z.B. an Kanten, neben den korrekten Werten auch falsche Werte gemessen. Diese Werte sind aber im Vergleich zu den korrekten Werten in der Unterzahl, weshalb man sie gut mit einem genügend großen Median-Filter filtern kann.
  Der von uns gewählte Filter hat eine Größe von 41 Werten (empirisch bestimmt), d.h. bis zu 20 fehlerhafte Werte können korrigiert werden. Die dabei entstehende Unschärfe war für unseren Algorithmus nicht relevant, da die Kanten erhalten bleiben.

- Minimale und maximale Entfernungswerte:
  Der Laser-Scan kann Werte unterhalb sowie oberhalb einer bestimmten Entfernung nicht mehr korrekt messen, das äußert sich in NaN-Werten bei der Ausgabe. Für unseren Algorithmus war es ausreichend, diese Werte auf die maximale Entfernung zu setzen.

- Verhalten bei Glas:
  Glas ist problematisch, da die Kinect eine Infrarot-Kamera ist und die Glasscheibe somit nicht als Wand erkennt, sondern als freien Platz. Das kann Probleme zur Folge haben, wenn der Abschnitt im Scan als Gang betrachtet wird. In der Live-Demo wurden deshalb Pappkartons vor den Scheiben platziert. Eine mögliche Verbesserung wäre, hier einen Algorithmus zu finden, der die Glasscheibe erkennt und als Wand behandelt.

- Systematische Fehler in den Messwerten (regelmäßige NaN-Werte):
  Bei der Untersuchung der Laserscan-Werte ist aufgefallen, dass in regelmäßigen Abständen NaN-Werte im Scan enthalten waren. Des Weiteren waren immer gleich viele Werte am linken und am rechten Rand des Scans fehlerbehaftet. Die Ursachen dafür konnten wir

leider nicht feststellen, aber der oben erwähnte Median-Filter hat diese Fehler ebenfalls behoben.

2.3.3 Ultraschallsensoren

Die von den Ultraschallsensoren gelieferten Werte sind ungefiltert, deshalb haben wir auch hier zunächst einen Median-Filter in Planung gehabt. Da die Werte aber in der Praxis für unsere Zwecke stabil und präzise genug waren, haben wir uns gegen eine Filterung entschieden und die Zeit in dringendere Aufgaben investiert.

3 Erste Aufgabe: Regelung des Wandabstandes für den Rundkurs

Dieses Kapitel soll die Problematiken sowie die verschiedenen in unserer Gruppe erarbeiteten Lösungsansätze für die Pflichtaufgabe „Rundkurs ohne Hindernisse“ aufzeigen. Dazu soll im Unterkapitel 3.1 kurz die eigentliche Aufgabenstellung erläutert werden. Im nächsten Unterkapitel (Kapitel 3.2) werden die für das System “Auto” wichtigsten Elemente benannt, welche zum Lösen des Problems von großer Bedeutung sind. Aufbauend auf diesen Überlegungen werden in Kapitel 3.3 bis 3.6 die in diesem Seminar erarbeiteten Lösungsansätze sowie deren Problematiken diskutiert und anhand von mathematischen Gleichungen und Grafiken beschrieben. Dabei dient das Kapitel 3.3 zum PID-Regler, sowie das Kapitel 3.4 zum PI-Regler nur als Grundlage für den danach folgenden Kaskadenregler und sind nicht für eine Implementierung geeignet.
Der Anhang der Ausarbeitung beinhaltet außerdem wichtige mathematische Herleitungen und Grafiken sowie Code von Matlab-Modellen, welche für die einzelnen Lösungsansätze von Interesse sind.
Ziel ist, dass der Leser möglichst gut nachvollziehen kann, wie das Vorgehen bei der Reglerauslegung war und mit dem Kaskadenregler eine gute Grundlage hat, die ausgebaut werden kann.

3.1 Erläuterung der Aufgabenstellung und Grundlagen des Modells “Auto”

3.1.1 Erläuterung der Aufgabenstellung

Die Pflichtaufgabe besteht aus dem erfolgreichen Befahren einer Rundstrecke ohne Hindernisse, wobei das Auto nicht von Hand gesteuert werden soll, sondern “autonom”. Im Hinblick auf den Wettbewerb sollte dies in möglichst kurzer Zeit geschehen. Mit einer Rundstrecke ist eine Fahrbahn gemeint, welche sich aus vier geraden Teilstücken zusammensetzt, wobei zwei Teilstücke über eine 90-Grad-Kurve verbunden sind. Somit ergibt sich ein Rechteck als Rundkurs. Abbildung 3.1 stellt dieses Szenario dar.

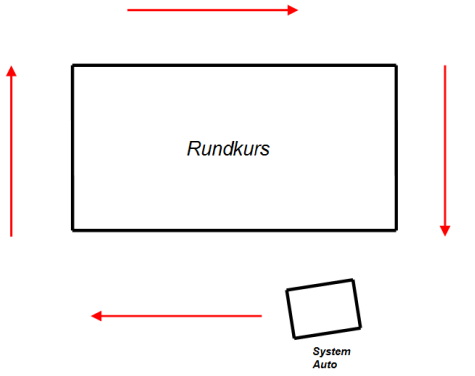

Abbildung 3.1: Darstellung der Pflichtaufgabe

Um diesen Rundkurs ohne Hindernisse möglichst schnell und “autonom” passieren zu können, sollte das Auto die entstehenden Schwankungen durch Störungen der Lenkung, Anfangswerteinflüsse oder Messungenauigkeiten gering halten. Ein möglicher Lösungsansatz für die Pflichtaufgabe ist die Entwicklung eines Reglers, welcher auf Grund von Messungen wichtiger Größen am Auto, den Abstand zur Wand bei jeder Geschwindigkeit immer auf einem konstanten Wert halten soll. Durch eine geschickte Auslegung des Reglers und unter Beachtung von gewissen Maßnahmen, können auch Störeinflüsse gering gehalten werden.

Den Einsatz eines Reglers bezüglich eines zu regelnden Systems zeigt Abbildung 3.2. Dies ist ein geschlossener Regelkreis, welcher einen Regler und ein System enthält. Das System entspricht in unserem Fall dem “Auto”, welches mit Hilfe einer Übertragungsfunktion (algebraische Gleichung) beschrieben wird. Diese spiegelt die dynamischen Eigenschaften des Autos wieder. Die Aufgabe des Reglers besteht nun darin, aufgrund von Messwerten des Autos (z.B. Wandabstand, Kurswinkel, etc. ...) und den dazugehörigen Vergleichswerten (Sollwerte) den Lenkwinkel (Eingangsgröße des Autos) so vorzugeben, dass die Messwerte möglichst schnell den Vergleichswerten entsprechen und diese auch beibehalten werden. Somit könnte zum Beispiel der Wandabstand zum Auto gemesen und mit einem Vergleichswert einem entsprechenden Regelalgorithmus übergeben werden.

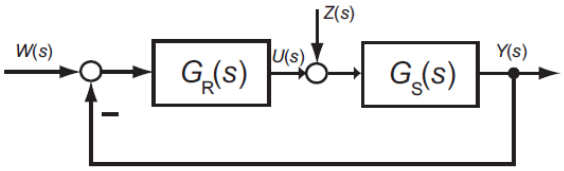

Abbildung 3.2: Darstellung System mit Regler - geschlossener Regelkreis

Ein solches Vorgehen bedarf jedoch einer guten Modellierung des Systems. Bei dem Seminar wurde vom Fachgebiet Regelungstechnik und Mechatronik das Modell eines Fahrzeugs mit Ackermann-Lenkung zur Verfügung gestellt, wodurch der Fokus auf die Auslegung des Reglers sowie dazugehörigen Elemente gelegt werden konnte.

Da die Modellierung im kontinuierlichen Zeitbereich vorliegt, jedoch aufgrund der Messwerte eine zeitdiskrete Modellierung von nöten ist, wird im folgenden Unterkapitel nochmals auf die Modellierung des Systems eingegangen.

3.1.2 Grundlagen des Modells “Auto”

Modell des zeitkontinuierlichen Systems

Da die ausführliche Herleitung im zur Verfügung gestellten Dokument von Dr.-Ing. Eric Lenz und Dipl.-Ing. Tim Hansen bereitgestellt wurde, werden hier nur die Formeln zur Systemdarstellung und das entsprechende Strukturbild angegeben.

Die Systemdarstellung im Zustandsraummodell lautet wie folgt:

$$\begin{bmatrix} \dot{y} \\ \dot{\varphi}_k \end{bmatrix} = \begin{bmatrix} 0 & v \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} y \\ \varphi_k \end{bmatrix} + \begin{bmatrix} v \cdot \frac{l_H}{l} \\ \frac{v}{l} \end{bmatrix} \cdot \varphi_L^* \,, \tag{1}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} y \\ \varphi_k \end{bmatrix} . \tag{2}$$

Gleichung (1) wird als Zustandsdifferentialgleichung bezeichnet und (2) als Ausgangsgleichung. Die Eingangsgröße φ_L^* ist der Lenkwinkel des Autos und die Ausgangsgröße y der Abstand zwischen Auto und Wand. Die Bedeutung der restlichen Parameter und Zustandsgrößen können in der bereitgestellten Herleitung nachgelesen werden.

Die Systemdarstellung über die Übertragungsfunktion ist in Gleichung (3) dargestellt und beschreibt die Division zwischen Ausgangs- und Eingangsgröße im Laplacebereich, also $\frac{Y(s)}{\varphi_L^*(s)}$:

$$G(s) = \frac{v \cdot l_H}{l} \cdot \frac{\frac{v}{l_H} + s}{s^2} \quad . \tag{3}$$

Das zu diesem System gehörende Strukturbild ist in Abbildung 3.3 nochmals gezeigt.

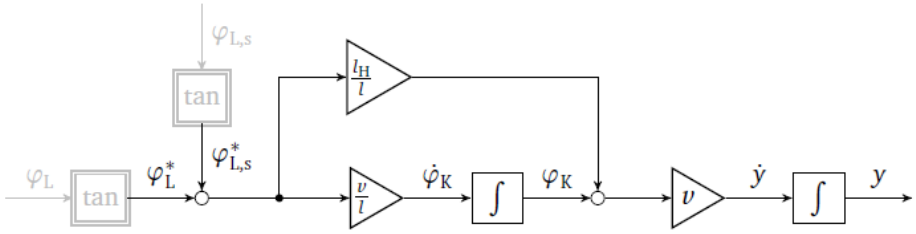

Abbildung 3.3: Strukturbild der zeitkontinuierlichen Systemdarstellung

Die angegebenen Gleichungen beschreiben das dynamische Verhalten des Systems "Auto" und können als Modell in Abbildung 3.2 für die Auslegung eines der Aufgabe entsprechenden Reglers genutzt werden. Da jedoch die benötigten Messwerte des Systems im zeitdiskreten vorliegen, wird eine dazugehörige Modellierung benötigt. Darauf wird im Folgenden eingegangen.

Modell des zeitdiskreten Systems

Das zeitdiskrete Zustandsraummodell berechnet sich aus dem zeitkontinuierlichen Modell mit Hilfe der Formeln aus Kapitel 10.2. von [3] auf Seite 82. Hierbei ist anzumerken, dass ein allgemeines Single-Input-Single-Output (SISO) Zustandsraummodell im zeitkontinuierlichen durch die Gleichungen (4) und (5) beschrieben wird.

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A} \cdot \boldsymbol{x}(t) + \boldsymbol{b} \cdot u(t) \ , \tag{4}$$

$$y(t) = \boldsymbol{c}^T \cdot \boldsymbol{x} + d \cdot u(t) \ . \tag{5}$$

Im Folgenden wird nun auf die Berechnungen der entsprechenden Elemente des dazugehörigen zeitdiskreten Modells eingegangen, welches durch die Gleichungen (6) und (7) beschrieben wird.

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_d \cdot \boldsymbol{x}_k + \boldsymbol{b}_d \cdot u_k \ , \tag{6}$$

$$y_k = \boldsymbol{c}^T{}_d \cdot \boldsymbol{x}_k + d_d \cdot u_k \ . \tag{7}$$

Die Matrix $\boldsymbol{A}_d$ ergibt sich aus Gleichung (8), wobei T für die Abtastzeit des Systems steht. Der Vektor $\boldsymbol{c}^T$ verändert sich nicht und auch der Durchgriff d bleibt im diskreten Fall derselbe ($\boldsymbol{c}_d^T = \boldsymbol{c}^T$ und $d_d = d$). Der Vektor $\boldsymbol{b}_d$ kann mittels Gleichung (9) berechnet werden.

$$\boldsymbol{A}_d = e^{\boldsymbol{A}T} \ , \tag{8}$$

$$\boldsymbol{b}_d = \int_0^T e^{\boldsymbol{A}v} \, \boldsymbol{b} \, dv \ . \tag{9}$$

Für das betrachtete zeitkontinuierliche System "Auto" können die zur Transformation benötigten Matrizen aus den Gleichungen (1) und (2) abgelesen werden. Somit ergibt sich für die Matrix $\boldsymbol{A}_d$ Folgendes:

$$\boldsymbol{A}_d = \exp\left(\begin{bmatrix} 0 & v \\ 0 & 0 \end{bmatrix} T\right) = \begin{bmatrix} 1 & v \cdot T \\ 0 & 1 \end{bmatrix} \ .$$

Durch Anwenden der Gleichung (9) erhält man für den diskretisierten Eingangsvektor des betrachtete Systems:

$$\boldsymbol{b}_d = \int_0^T \exp\left(\begin{bmatrix} 0 & v \\ 0 & 0 \end{bmatrix} \nu\right) \cdot \begin{bmatrix} v \cdot \frac{l_H}{l} \\ \frac{v}{l} \end{bmatrix} d\nu = \begin{bmatrix} \frac{v\, l_H T}{l} + \frac{1}{2} \cdot \frac{v^2 T^2}{l} \\ \frac{v\, T}{l} \end{bmatrix} .$$

Die genaue Berechnung der Matrix $\boldsymbol{A}_d$ und des Vektors $\boldsymbol{b}_d$ sind im Anhang (Kapitel A.3) beschrieben. Dies kann bei Interesse gelesen werden, jedoch ist an dieser Stelle nur das Ergebnis von Bedeutung.

Da die Größen $\boldsymbol{c}^T$ und d identisch bleiben ergibt sich unter Berücksichtigung des zeitkontinuierlichen Systems (2):

$$\begin{aligned} \boldsymbol{c}_d^T &= \begin{bmatrix} 1 & 0 \end{bmatrix} , \\ d_d &= 0 , \end{aligned}$$

für die entsprechende zeitdiskrete Darstellung.

Als nächster Schritt wird noch das zeitdiskrete System in Form einer Übertragungsfunktion benötigt. Die verschiedenen Darstellungsformen des selben Systems werden für die unterschiedlichen Ansätze zur Reglerauslegung benötigt.

Um eine Übertragungsfunktion für das diskrete System zu erhalten, kann zum Beispiel das zeitdiskrete Zustandsraummodell in eine Übertragungsfunktion überführt werden. Hierzu kann Gleichung (10) verwendet werden, welche aus Kapitel 10.5 von [3] entnommen wurde.

$$G(z) = \boldsymbol{c}_d^T \left(z \cdot \boldsymbol{I}_n - \boldsymbol{A}_d\right)^{-1} \boldsymbol{b}_d + d_d \ . \tag{10}$$

Damit erhält man die Übertragungsfunktion des zeitdiskreten Systems “Auto” zu:

$$G(z) = \left(\frac{v\, l_H\, T}{l} + \frac{v^2 T^2}{2\, l}\right) \frac{z - \left(\frac{2\, l_H\, T - v\, T^2}{2\, l_H\, T + v\, T^2}\right)}{(z-1)^2} \ . \tag{11}$$

Die genaue Berechnung kann ebenfalls im Anhang nachvollzogen werden.

Mit den beiden Systemdarstellungen im zeitdiskreten werden im Weiteren mehrere Ansätze zur Reglerauslegung vorgestellt und diskutiert. Diese basieren auf der schon gezeigten Abbildung 3.2. Doch vorerst wird im nächsten Kapitel auf zwei wichtige Elemente eingegangen, welche ein Regler beinhalten sollte um möglichst gute Ergebnisse zu liefern.

3.2 Bedeutung des I- und D-Anteils für das System “Auto”

Ziel ist es nun einen Regelalgorithmus zu entwerfen, welcher das Auto möglichst schnell auf einen gewünschten Abstand zur Wand bringt, diesen hält und dabei wenig bis keine Schwingungen um diesen Wert aufweist. Dies soll wenn möglich für verschiedene Anfangswerte möglich sein. Des Weiteren soll der geschlossene Regelkreis (Abbildung 3.2) stabiles Verhalten zeigen. Gleichung (11) besitzt zwei Pole in Eins, weshalb das eigentliche System “Auto” (ohne Regler) instabil ist (siehe [3] Kapitel 7).

3.2.1 Bedeutung des I-Anteils

Um sich dem Ziel nähern zu können, wird nochmals Abbildung 3.3 betrachtet. Hier ist zu erkennen, dass auf die Lenkung Störungen wirken, welche natürlich auch bei der diskreten Betrachtung vorhanden sind. Diese führen dazu, dass sich nicht der gewünschte Abstand zur Wand einstellt, sondern eine gewisse Differenz vorhanden sein kann ("bleibende Regelabweichung"). Damit jedoch der gewünschte Abstand zur Wand für verschiedene Stör- und Führungsgrößen erreicht werden kann, muss der Regelalgorithmus einen Integrator (I-Anteil) besitzen. Der Einfluss der Störung bezogen auf den gewünschten Wandabstand verschwindet. Der mathematische Beweis für diesen Effekt kann in [6] in Kapitel 6.3 auf Seite 71 nachgelesen werden.

3.2.2 Bedeutung des D-Anteils

In diesem Unterkapitel soll kurz auf den Zusammenhang zwischen Schnelligkeit des Reglers und einem differenzierenden Anteil (D-Anteil) im Regler eingegangen werden.

Anhand des ersten Ansatzes zur Reglerauslegung (PID-Regler), welcher im Kapitel 3.3 diskutiert wird, ist erkennbar, dass ein D-Anteil im Regler zu einer schnelleren Reaktionsfähigkeit des Reglers führt als ohne Verwendung eines D-Anteils. Dies hängt mit dem Strukturbild beziehungsweise dem Zusammenhang zwischen den Differentialgleichungen des Systems zusammen. Bei Betrachtung der Abbildung 3.3 ist erkennbar, dass eine Differentiation des Abstandes zur Wand eine Information über den Kurswinkel liefert. Dieser wiederum gibt an, in welchem Winkel sich das Auto zur Wand befindet. Bei einem Kurswinkel von 0° steht das Auto parallel zur Wand. Mithilfe dieser zusätzlichen Information kann nicht nur auf eine Veränderung des Abstandes zur Wand, sondern auch auf eine Veränderung des Kurswinkels reagiert werden, womit eine schnellere Reaktion des Reglers auf Veränderungen möglich ist. Dies begründet auch den Einsatz eines Zustandsreglers mit erweitertem I-Anteil, welcher in Kapitel 3.4 näher beschrieben wird.

Die Abbildungen 3.9 und 3.10 stellen den Unterschied bei einer Regelung für das System "Auto" ohne und mit differenzierenden Anteil im Regelalgorithmus dar. Dieser Unterschied ist besonders gut bei Anfangswerten zu beobachten, welche ungleich Null sind. Die Abbildung 3.9 wurde mit einem diskreten PI-Regler simuliert und die Abbildung 3.10 mit einem diskreten PI-Zustandsregler, welcher als zusätzliche Information den Kurswinkel erhält. Wegen des Strukturbildes unterscheiden sich beide Ansätze in der Information über den Kurswinkel. Es ist deutlich zu erkennen, dass der diskrete PI-Regler deutlich länger zum Einschwingen benötigt und nur eine sehr langsame Reaktionsfähigkeit aufweist. Solche großen Schwingungen um den Endwert, wie es in Abbildung 3.9 dargestellt ist, sollten unbedingt vermieden werden, da beim Rundkurs der Abstand zur Wand nach oben hin durch eine weitere Wand beschränkt sein wird und somit Kollisionen nicht auszuschließen sind.

3.3 Diskrete PID-Reglerauslegung (1. Ansatz)

3.3.1 Problematiken des Wurzelortskurvenverfahrens bei Betrachtung diskreter Systeme

Aus dem vorherigen Kapitel ist hervorgegangen, dass der Regler einen differenzierenden und einen integrierenden Anteil besitzen sollte. Somit ist der einfachste Ansatz, einen PID-Regler für das diskrete System zu entwerfen. Zur Auslegung kann zum Beispiel das Wurzelortskurvenverfahren verwendet werden, welches in [3] in Kapitel 8.1 für diskrete Systeme ausführlich diskutiert ist und deshalb an dieser Stelle nicht nochmals erläutert wird. Jedoch soll kurz die

Idee des Verfahrens erklärt werden und in Zusammenhang damit das bevorzugte Stabilitätsgebiet im diskreten Fall genannt werden.

Allgemein wird das dynamische Verhalten eines Systems durch die Lage der Polstellen der Übertragungsfunktion beschrieben. Um somit eine gute Dynamik für das System Auto mit Regler zu erhalten müssen die Polstellen des geschlossenen Regelkreises (Abbildung 3.2) betrachtet werden. Die Wurzelortskurve stellt nun die Pole des geschlossenen Regelkreises in Abhängigkeit eines Verstärkungsfaktors in einer komplexen Ebene dar. Das heißt anhand des Verstärkungsfaktors der Übertragungsfunktion des Reglers können die Pole des geschlossenen Regelkreises verschoben werden und somit in einen Bereich der komplexen Ebene gelegt werden, der wiederum die gewünschte Dynamik hervorruft.

Von Interesse ist im Weiteren der Bereich, in dem die Pole des geschlossenen Regelkreises liegen dürfen, um eine gute Dynamik für das System zu erzeugen. Im diskreten Fall dürfen die Pole nur innerhalb des Einheitskreises der komplexen Ebene liegen, um die Stabilität des Gesamtsystems zu gewährleisten. Um möglichst kein Überschwingen zu erhalten, sollten die Pole auf der reellen Achse liegen. Liegen die Pole jedoch auf der reellen Achse im Bereich zwischen 0 und -1 so führt dies zu zusätzlichen Schwingungen (Störungen) bei der Sprungantwort. Um Überschwingen zu vermeiden, dürfen deshalb die Pole des geschlossenen Regelkreises nur auf der reellen positiven Achse innerhalb des Einheitskreises liegen. Der entstehende Effekt ist in Abbildung 3.4 und 3.5 nochmals dargestellt. Die hier dargestellten Simulationen wurden mit einem diskreten geschlossenen Regelkreis der Form $G(z) = \frac{k}{z-p_1}$ erzeugt.

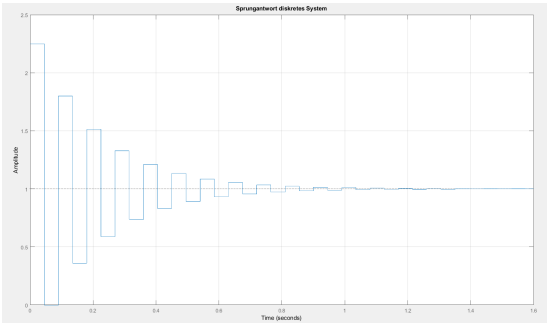

Abbildung 3.4: Sprungantwort: Pole zwischen -1 und 0

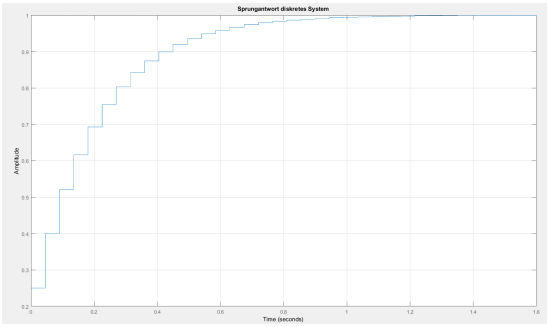

Abbildung 3.5: Sprungantwort: Pole zwischen 0 und 1

Abbildung 3.4 zeigt den Effekt von zusätzlichen Schwingungen auf der Sprungantwort und Abbildung 3.5 zeigt die Sprungantwort bei Pollagen auf der reellen Achse zwischen 0 und 1. Im linken Bild liegt der einzige Pol p_1 des geschlossenen Regelkreises bei -0.8 und im rechten Bild bei 0.8. Die Abtastzeit wurde auf 45ms gesetzt.

Angenommen, das in dem Seminar betrachtete System Auto müsste einem dem linken Bild ähnlichen Verlauf folgen, so würde dies sicherlich der Lenkmechanik auf Dauer nicht zu gute kommen und könnte unter Umständen auch bleibende Schäden verursachen. Aus diesem Grund soll ein PID-Regler für das System entworfen werden, dessen Pole des geschlossenen Regelkreises auf der reellen Achse in der rechten Halbebene des Einheitskreises liegen.

3.3.2 Auslegung mittels Wurzelortskurvenverfahren

Um den PID-Regler mittels Wurzelortskurvenverfahren auszulegen, wird vorerst die Übertragungsfunktion des Reglers benötigt, welche noch zu wählende Freiheitsgrade besitzt. Diese sollten genutzt werden, um die Dynamik des später geschlossenen Regelkreises positiv zu beeinflussen.

Ein diskreter PID-Regler besitzt die folgende z-Übertragungsfunktion:

$$G_R(z) = \frac{b_2 z^2 + b_1 z + b_0}{z(z-1)} \ ,$$

welche auch in folgender Form geschrieben werden kann:

$$G_R(z) = \frac{k(z-n_1)(z-n_2)}{z(z-1)} \ . \tag{12}$$

Aus dem vorherigen Abschnitt dieses Kapitels ist bekannt, dass das Wurzelortskurvenverfahren die Verstärkung des offenen Regelkreises als freien Parameter besitzt. Somit müssen die noch offenen Nullstellen von Gleichung (12) erst gewählt werden, bevor das Verfahren angewendet werden kann. Die Wahl beeinflusst jedoch schon das Ergebnis des geschlossenen Regelkreises, weshalb ein Gesamtbild der Wurzelortskurve für verschieden Möglichkeiten von n_1 und n_2 betrachtet werden sollte, um die freien Parameter zu wählen. Dazu wird noch die Übertragungsfunktion des diskreten Systems (11) benötigt.

Mit dem aus dem Skript [3] vermittelten Wissen zur Wurzelortskurve können grobe Skizzen der Verläufe der Wurzelortskurven für verschieden Möglichkeiten von n_1 und n_2 erstellt werden. Hierfür können die Regeln zur Konstruktion aus [7] Kapitel 2.3 verwendet werden, oder man verwendet Matlab zum Plotten verschiedener Möglichkeiten. Die Abbildungen 3.6 und 3.7 zeigen jeweils die Wurzelortskurve für eine mögliche Wahl der übrigen Freiheitsgrade.

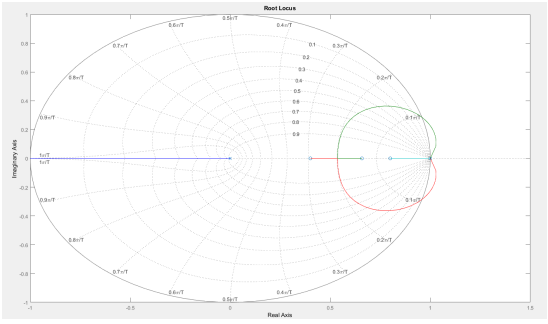

Abbildung 3.6: Wurzelortskurve: Strecke mit PID-Regler

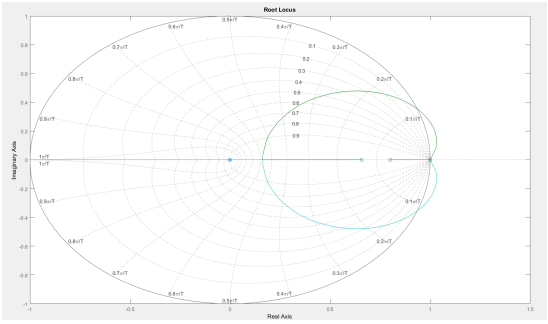

Abbildung 3.7: Wurzelortskurve: Strecke mit PI-Regler

In Abbildung 3.6 wurden die noch übrigen Nullstellen (n_1 und n_2) um die schon existierende Nullstelle der Strecke gelegt. Mit anderen Worten, eine wurde rechts von der existierenden Nullstelle auf der reellen Achse platziert und die andere links. Es ist zu erkennen, dass der Pol in Null mit steigendem Parameter k in die linke Halbebene des Einheitskreises läuft. Dadurch würde sich jedoch wieder eine Sprungantwort ergeben, welche zusätzliche Sprünge aufweist (ähnlich wie in Abbildung 3.4). Es erweist sich als schwierig, eine geeignete Wahl der freien

Parameter zu finden, welche auf eine Dynamik mit möglichst geringem Überschwingen, stationärem Verhalten und ohne zusätzliche Schwingungen (Effekt der negativen reellen Achse im Einheitskreis) führt.

Abbildung 3.7 stellt eine Wurzelortskurve dar, in der es möglich ist, alle Pole des geschlossenen Regelkreises innerhalb des Einheitskreises und auf die positive reelle Achse zu legen. Jedoch wurde dafür der Pol des PID-Reglers in Null durch eine entsprechende Wahl einer Nullstelle gekürzt, wodurch sich für die Gleichung des Reglers nur noch ein PI-Regler anstatt eines PID-Reglers ergibt. Dies ist mathematisch mit wenigen Rechenschritten im Anhang gezeigt. Die Verstärkung des offenen Regelkreises kann so gewählt werden, dass die Pole des geschlossenen Regelkreises nahe oder im Verzweigungspunkt auf der reellen Achse liegen.

3.3.3 Reduktion auf PI-Regler und dessen Simulationsergebnisse

Aufgrund der schwierigen Wahl der Freiheitsgrade eines PID-Reglers, wurde dieser auf einen PI-Regler reduziert. Somit verzichtet man jedoch gleichzeitig auf die Information des Kurswinkels, wodurch die Möglichkeit zu einer schnellen Reaktion des Regelalgorithmus auf Veränderungen verloren gehen. Deshalb beschäftigen sich die Kapitel 3.4 und 3.5 noch mit anderen Ansätzen, um die Information des Kurswinkels zu berücksichtigen.

Im Folgenden werden jedoch die Simulationsergebnisse des PI-Reglers kurz gezeigt. Die erste Nullstelle des Reglers kompensiert dessen Pol bei 0. Die zweite Nullstelle des Reglers wird so gewählt, dass diese links der Nullstelle der Strecke auf der reellen Achse liegt. Die Pole des geschlossenen Regelkreises wurden neben dem Verzweigungspunkt platziert, welcher analytisch noch gut ausgerechnet werden kann. Somit ist es möglich während der Fahrt für verschiedene Geschwindigkeiten die Reglerparameter auf dem Auto direkt neu zu bestimmen.

Ein Beispiel einer Sprungantwort ist in Abbildung 3.8 zu sehen. Der fehlende D-Anteil macht sich vor allem bei Anfangsauslenkungen der Zustände und beim realen System bemerkbar. Hier reagiert der Regelalgorithmus sehr langsam um den Endwert zu erreichen. Dies ist auch beim Vergleich von Abbildung 3.8 und Abbildung 3.9 beobachtbar. In Abbildung 3.9 wurde als Anfangswert für den Abstand 0.8 m vorgegeben und für den Kurswinkel 20°. Die Geschwindigkeit beträgt $v = 0.4\frac{m}{s}$. Dieser Fall kann zum Beispiel beim Befahren von Kurven im Rundkurs auftreten und darf deshalb nicht vernachlässigt werden.

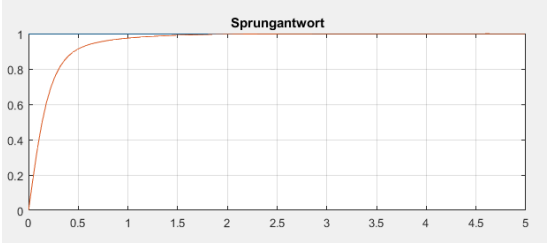

Abbildung 3.8: Sprungantwort ohne Anfangswerte

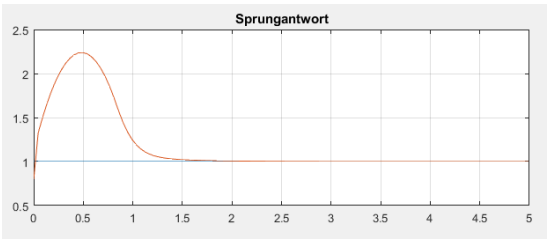

Abbildung 3.9: Sprungantwort mit Anfangswerten

Abbildung 3.10 zeigt zum Vergleich nochmal die Sprungantwort bei denselben Anfangsauslenkungen, jedoch mit Verwendung eines diskreten PI-Zustandsreglers. Hier sind die Auswirkungen des D-Anteils im Regler deutlich zu sehen.

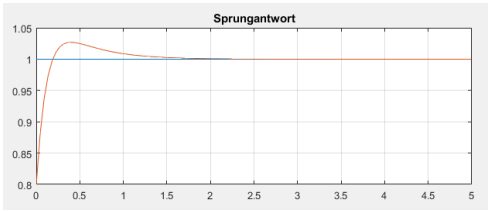

Abbildung 3.10: Sprungantwort mit Anfangswerten

Bei der Auslegung des Reglers muss darauf geachtet werden, dass die Pole des geschlossenen Regelkreises für alle Geschwindigkeiten v des Autos in dem gewünschten Bereich liegen. Dies schränkt die Wahl der Nullstelle des Reglers noch weiter ein. Im Anhang sind die zur Simulation der Sprungantworten für den PID-Regler benötigten Dateien (Matlabcode und Modell zur Simulation) angehängt. Jedoch ist es für das Verständnis und auch für die eigene Kreativität bei der Reglerauslegung von Vorteil sich selbst Gedanken über den benötigten Matlabcode zu machen.

3.4 Diskreter PI-Zustandsregler (2. Ansatz)

Aufgrund der fehlenden Information über den Kurswinkel aus dem vorherigen Ansatz wird im Folgenden ein weiterer Ansatz beschrieben, der es erlaubt, den Kurswinkel mit einzubeziehen. Hierbei handelt es sich um einen diskreten Zustandsregler mit einen zusätzlichen I-Anteil. Der Vorteil eines Zustandsreglers beim Vergleich mit einem PID-Regler liegt in dessen Methoden zur Auslegung. Es können weitaus leistungsfähigere Verfahren angewendet werden, als die Auslegung eines Reglers über das Wurzelortskurvenverfahren. In diesem Abschnitt wird zum Beispiel die Eigenwertplatzierung über die Ackermann-Formel verwendet. Mit dieser können die Pole beliebig platziert werden, ohne dass man an die Äste einer Wurzelortskurve gebunden ist. Somit hat man in der Wahl der Dynamik des geschlossenen Regelkreises deutlich mehr Möglichkeiten und muss keine Anteile herauskürzen, um eine vernünftige Sprungantwort zu erhalten (siehe PID-Regler Kapitel 3.3, Reduktion auf PI-Regler).

3.4.1 Strukturbild PI-Zustandsregler mit Strecke

Das in diesem Seminar verwendete Strukturbild zum PI-Zustandsregler ist in Abbildung 3.11 dargestellt.

In diesem bezeichnet r den Zustandsregler für die Zustände der Strecke und KI den Verstär-

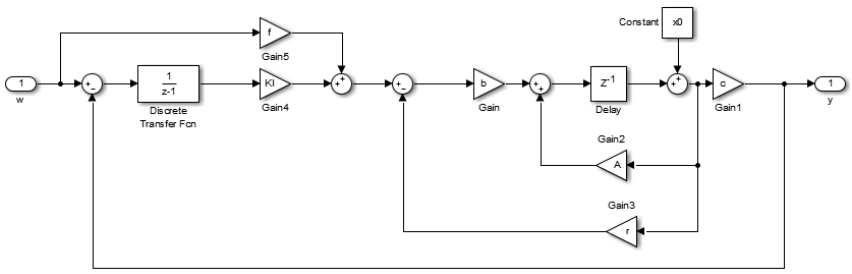

Abbildung 3.11: Strukturbild PI-Zustandsregler

kungsfaktor für den I-Anteil des Reglers. Der eigentliche I-Anteil besitzt die Übertragungsfunktion $\frac{1}{z-1}$. Das Vorfilter wird mit dem Verstärkungsfaktor f dargestellt und die Anfangswerte sind

in der Konstanten $x0$ enthalten. Die übrigen Verstärkungsfaktoren A, b und c repräsentieren die zu regelnde Strecke. Um das in Abbildung 3.11 dargestellte System zur Simulation nutzen zu können, müssen die Anfangswerte $x0$ dem Delay übergeben werden. Der Block "Constant" mit $x0$ würde wegfallen.

Um für das Gesamtsystem alle Pole des geschlossenen Regelkreises vorgeben zu können, muss zuerst ein Zustandsraummodell für das Gesamtsystem, bestehend aus Strecke und I-Anteil des Reglers, aufgestellt werden. Dies wird benötigt, um die Ackermann-Formel anwenden zu können. Dazu wird der I-Anteil des Reglers als ein weiterer Zustand angenommen und eine Differentialgleichung für diesen aufgestellt. Dieses Vorgehen entspricht dem aus Kapitel 10.9 aus dem Skript [3] und wird dort ausführlich beschrieben. Für das System "Auto" mit zusätzlichem I-Anteil ergibt sich das folgende Zustandsraummodell:

$$\begin{bmatrix} \boldsymbol{x}_{k+1} \\ \tilde{u}_{Ik+1} \end{bmatrix} = \begin{bmatrix} 1 & vT & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{x}_k \\ \tilde{u}_{Ik} \end{bmatrix} + \begin{bmatrix} \frac{vl_HT}{l} + \frac{v^2T^2}{2l} \\ \frac{vT}{l} \\ 0 \end{bmatrix} \cdot u_k + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \cdot w_k \ . \tag{13}$$

In Gleichung (13) entspricht $\tilde{u}_{Ik}$ dem Ausgang nach der Übertragungsfunktion $\frac{1}{z-1}$ in Abbildung 3.11. Die beiden Eingänge u_k und w_k können ebenfalls in dieser Abbildung wiedergefunden werden. Der Eingang u_k liegt direkt vor dem Eingangsvektor b an und w_k ist das Eingangssignal des Gesamtsystems, welches mit w in der Abbildung gekennzeichnet ist.

3.4.2 Reglerauslegung mittels Ackermann-Formel

Im weiteren Verlauf sollen für die Systemmatrix des geschlossenen Regelkreises die Eigenwerte mittels der Parameter $\boldsymbol{r}$ und KI beliebig vorgegeben werden. Dazu soll wie schon erwähnt die Ackermann-Formel (Skript [3], Kapitel 10.8 auf Seite 95) zum Einsatz kommen:

$$\boldsymbol{r}^T = \boldsymbol{q}_n^T \left(a_0^* \boldsymbol{I}_n + \cdots + a_{n-1}^* \boldsymbol{A}_d^{n-1} + A_d^n \right) \ , \tag{14}$$

wobei A_d der Systemmatrix des ungeregelten Systems entspricht und a_0^* bis a_{n-1}^* die Koeffizienten der charakteristischen Gleichung (Polynom) für die Eigenwerte des geschlossenen Regelkreises. In Gleichung (14) entspricht $\boldsymbol{q}_n^T$ der letzten Zeile der inversen Steuerbarkeitsmatrix. Auf die Begrifflichkeiten "Steuerbarkeitsmatrix" und "charakteristische Gleichung" bezüglich des betrachteten Systems soll im Folgenden kurz eingegangen werden, da diese benötigt werden, um eine Lösung für Gleichung (14) zu berechnen.

Die Nullstellen eines charakteristischen Polynoms entsprechen den Eigenwerten des dazugehörigen geschlossenen Regelkreises. Deshalb ergeben sich die Parameter a_0^* bis a_{n-1}^* aus dem Polynom mit den Wunscheigenwerten $\gamma_1, \gamma_2, \gamma_3$:

$$\begin{aligned} P_R(z) &= (z-\gamma_1) \cdot (z-\gamma_2) \cdot (z-\gamma_3) \\ &= z^3 + z^2 \cdot (-\gamma_1 - \gamma_2 - \gamma_3) + z \cdot (\gamma_1\gamma_2 + \gamma_1\gamma_3 + \gamma_2\gamma_3) + (-\gamma_1\gamma_2\gamma_3) \\ &= z^3 + z^2 \cdot a_2^* + z \cdot a_1^* + a_0^* \ . \end{aligned}$$

Die Parameter a_0^* bis a_{n-1}^* können also mittels eines Koeffizientenvergleichs bestimmt werden, wobei die Eigenwerte des geschlossenen Regelkreises vorher bekannt sein müssen beziehungsweise gewählt werden müssen. Gleichung (13) besitzt drei Zustände, womit drei Eigenwerte gewählt werden müssen.

Die Steuerbarkeitsmatrix, welche für $\boldsymbol{q}_n^T$ benötigt wird, kann nach Gleichung (15) aus dem Skript [3] Kapitel 10.7, Seite 89 berechnet werden:

$$\boldsymbol{Q}_S = \begin{bmatrix} \boldsymbol{B}_d & \boldsymbol{A}_d\boldsymbol{B}_d & \cdots & \boldsymbol{A}_d^{n-1}\boldsymbol{B}_d \end{bmatrix} . \tag{15}$$

Diese Matrix muss anschließend invertiert werden. Die letzte Zeile der invertierten Matrix entspricht der gesuchten Variable $\boldsymbol{q}_n^T$. Für unser betrachtetes System "Auto" ergibt sich für die Matrix $\boldsymbol{Q}_S$:

$$\boldsymbol{Q}_S = \begin{bmatrix} \frac{vl_HT}{l} + \frac{v^2T^2}{2l} & \frac{vl_HT}{l} + 3\frac{v^2T^2}{2l} & \frac{vl_HT}{l} + 5\frac{v^2T^2}{2l} \\ \frac{vT}{l} & \frac{vT}{l} & \frac{vT}{l} \\ 0 & -\frac{vl_HT}{l} - \frac{v^2T^2}{2l} & -2\frac{vl_HT}{l} - 4\frac{v^2T^2}{2l} \end{bmatrix} . \tag{16}$$

Diese kann entweder per Hand oder mit der Symbolic Toolbox aus Matlab ausgerechnet werden. Zur Berechnung von Gleichung (16) wurden für A_d und b_d die entsprechenden Variablen aus Gleichung (13) entnommen:

$$\begin{aligned} \boldsymbol{A}_d &= \begin{bmatrix} 1 & vT & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} , \\ \boldsymbol{b}_d &= \begin{bmatrix} \frac{vl_HT}{l} + \frac{v^2T^2}{2l} \\ \frac{vT}{l} \\ 0 \end{bmatrix} . \end{aligned}$$

Setzt man alle ermittelten Größen in die Gleichung (14) zur Berechnung der Reglerverstärkungen ein, so erhält man folgende Gleichung:

$$\boldsymbol{r}^T = \frac{l}{v^3T^3}\begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix} , \tag{17}$$

mit

$$\begin{aligned} r_1 &= -1 \cdot (1 + a_0^* + a_1^* + a_2^*) \cdot (l_H + \frac{vT}{2}) + vT(3 + a_1^* + 2a_2^*) , \\ r_2 &= -vT(a_1^* + 2a_2^* + 3) \cdot (l_H + \frac{vT}{2}) + (1 + a_0^* + a_1^* + a_2^*) \cdot (l_H + \frac{vT}{2})^2 + v^2T^2(3 + a_2^*) , \\ r_3 &= -vT(1 + a_0^* + a_1^* + a_2^*) . \end{aligned}$$

Diese Gleichung wurde für das Seminar per Hand ausgerechnet und kann verwendet werden um auf dem Auto bei Variation der Geschwindigkeit direkt die Reglerparameter für die gewählten Pole neu zu berechnen. Die Reglerparameter sind wie in Gleichung (17) zu sehen abhängig von der Geschwindigkeit des Fahrzeugs und müssen deshalb bei Änderung dieses Wertes angepasst werden.

Da es sich bei dem System aus Gleichung (13) um das System "Auto" mit zusätzlichem I-Anteil handelt, gehören die ersten zwei Reglerparamater von Gleichung (17) (r_1 und r_2) dem eigentlichen Zustandsregler für das System "Auto" an und der letzte Parameter aus (17) (r_3) entspricht der negativen Verstärkung des I-Anteils, also $-KI$ aus Abbildung 3.11. Mit anderen Worten,

r_1 und r_2 bilden die Variable r in Abbildung 3.11 und $r_3 = -1 \cdot KI$. Zum tieferen Verständnis diesbezüglich, wird auf Kapitel 10.9 in [3] verwiesen.

Um nun noch stationäres Verhalten bezüglich der Führungsgröße zu erzielen, muss das Vorfilter f entsprechend ausgelegt werden. Dies berechnet sich mit der in [3] in Kapitel 10.8.1 auf Seite 96 angegebenen Formel:

$$f = \left[\boldsymbol{c}_d^T \left(\boldsymbol{I}_n - \boldsymbol{A}_d + \boldsymbol{b}_d \boldsymbol{r}^T \right)^{-1} \boldsymbol{b}_d \right]^{-1} \tag{18}$$

Wird in diese Gleichung $\boldsymbol{A}_d$, $\boldsymbol{b}_d$, $\boldsymbol{c}_d^T$ und $\boldsymbol{r}^T$ aus den vorherigen ergebnissen eingesetzt und vereinfacht, so erhält man als einfaches Ergebnis:

$$f = r_1 \ .$$

Da sich das Vorfilter jedoch nur auf das System ohne I-Anteil bezieht, wie in Abbildung 3.11 erkennbar, ergibt sich der folgende Ansatz mit zwei Zuständen zum Lösen von Gleichung (18):

$$\begin{aligned} f &= \left[\boldsymbol{c}_d^T \left(\boldsymbol{I}_n - \boldsymbol{A}_d + \boldsymbol{b}_d \boldsymbol{r}^T \right)^{-1} \boldsymbol{b}_d \right]^{-1} \\ &= \left[\begin{bmatrix} 1 & 0 \end{bmatrix} \left[\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & vT \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} \frac{vl_HT}{l} + \frac{v^2T^2}{2l} \\ \frac{vT}{l} \end{bmatrix} \cdot \begin{bmatrix} r_1 & r_2 \end{bmatrix} \right]^{-1} \begin{bmatrix} \frac{vl_HT}{l} + \frac{v^2T^2}{2l} \\ \frac{vT}{l} \end{bmatrix} \right]^{-1} \end{aligned}$$

Von Interesse ist nun das Überschwingen sowie die Einschwingdauer für bestimmte Kombination von Polen des geschlossenen Regelkreises. Hierzu wird im nächsten Unterkapitel das Ergebnis einer Polkonfiguration betrachtet und anhand dessen die Problematik dieses Reglers erläutert.

3.4.3 Simulationsergebnisse und Problematiken des PI-Zustandsreglers

Alle ergebnisse beziehen sich auf die Wunschpole des charakteristischen Polynoms des geschlossenen Regelkreises:

$$\begin{aligned} \gamma_1 &= 0.7 \ , \\ \gamma_1 &= 0.8 \ , \\ \gamma_1 &= 0.9 \ . \end{aligned}$$

Außerdem sind die verwendeten Dateien zur Simulation im Anhang angegeben.

Zuerst wird die Sprungantwort des Systems Auto mit PI-Zustandsregler für die Geschwindigkeit $v = 0.4\frac{m}{s}$ betrachtet. Als Startwerte sind ein Abstand zur Wand von $0m$ und ein Kurswinkel von 0° vorgegeben. Abbildungen 3.12 und 3.13 zeigen die Sprungantwort einmal ohne und einmal mit Berücksichtigung eines Sättigungsgliedes für den Lenkwinkel. Betrachtet man noch zusätzlich die Stellgröße für diesen Fall (Abbildung 3.14 und 3.15) so ist erkennbar, dass der benötigte Lenkwinkel außerhalb des zulässigen Bereichs liegt. Deshalb kann es auch zusätzlich zu Effekten wie “erhöhtes Überschwingen” führen. Diese Effekte werden auch als “Windup-Effekte” bezeichnet. Dieser Begriff wird in Kapitel 9.3 im Skript [3] beschrieben und hängt mit dem I-Anteil des Reglers zusammen. Es wird zum Verständnis empfohlen sich kurz das Kapitel durchzulesen.

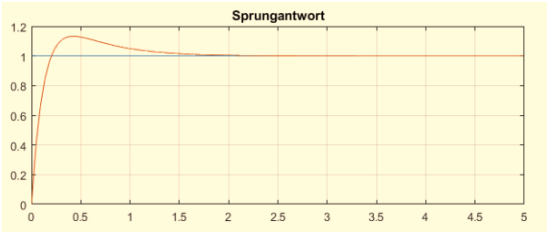

Abbildung 3.12: Sprungantwort ohne Sättigungsglied

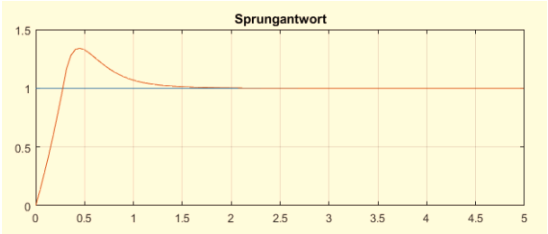

Abbildung 3.13: Sprungantwort mit Sättigungsglied

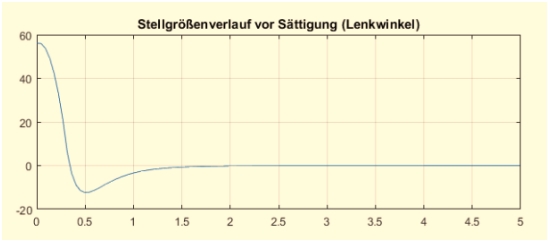

Abbildung 3.14: Stellgröße ohne Sättigungsglied

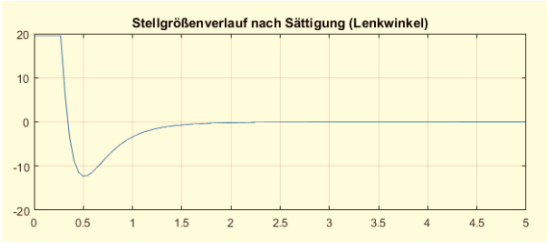

Abbildung 3.15: Stellgröße mit Sättigungsglied

In einem zweiten Beispiel wird eine Geschwindigkeit von $v = 1\frac{m}{s}$ betrachtet und als Startwerte werden ein Abstand zur Wand von $0.5m$ vorgegeben sowie ein Kurswinkel von $-20°$. Das heißt, das Auto fährt zu Beginn schräg auf die Wand zu. Abbildungen 3.16 und 3.17 zeigen die Sprungantwort einmal ohne und einmal mit Berücksichtigung eines Sättigungsgliedes für den Lenkwinkel. Betrachtet man noch zusätzlich die Stellgröße für diesen Fall (Abbildung 3.18

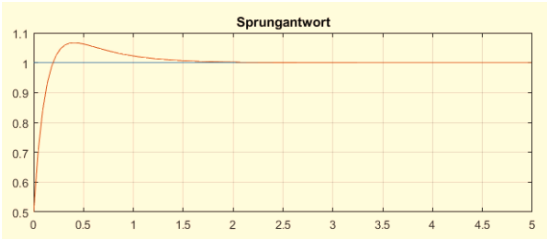

Abbildung 3.16: Sprungantwort ohne Sättigungsglied

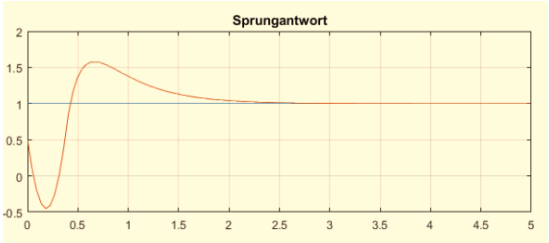

Abbildung 3.17: Sprungantwort mit Sättigungsglied

und 3.19) so ist erkennbar, dass der benötigte Lenkwinkel auch hier außerhalb des zulässigen Bereichs liegt.

Windup-Effekten entgegenzuwirken ist kein einfaches Vorgehen. Es gibt viele Methoden, um Windup-Effekte zu umgehen oder diese zu mildern. In der Dissertation [8] sind einige Methoden aufgezeigt. Jedoch stößt man oft auf mathematische Probleme, welche nicht einfach zu lösen sind, oder man benötigt viel Zeit, um sich in die entsprechende Materie einzulesen. Für diskrete PI-Regler kann ein sehr einfaches Vorgehen angewendet werden (siehe [3] Kapitel 9.3), welches auch im ersten Ansatz (Kapitel 3.3) für diese Pflichtaufgabe verwendet wurde. Für diskrete PI-Zustandsregler existiert leider keine solche einfache Lösung.

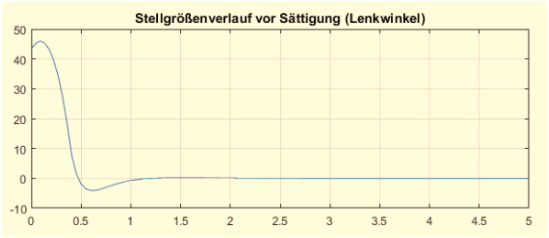

Abbildung 3.18: Stellgröße ohne Sättigungsglied

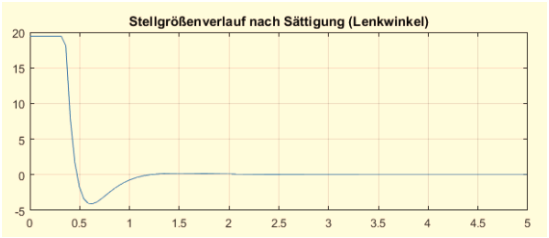

Abbildung 3.19: Stellgröße mit Sättigungsglied

Aus diesem Grund wird in Kapitel 3.5 ein weiterer Ansatz diskutiert um hohe Lenkwinkel als Stellgrößen zu vermeiden und trotzdem eine Information über den Kurswinkel zu erhalten.

3.5 Diskrete Kaskadenregelung und Ausblick für folgende Seminare (3. Ansatz)

3.5.1 Idee der Kaskadenregelung und Struktur

Die Idee eine Kaskadenregelung zu verwenden basiert auf dem im vorherigen Kapitel diskutierten Zustandsregler. Eine Zustandsregelung kann nämlich durch einfache Strukturbildumformungen auch als Kaskadenregelung interpretiert werden. Allerdings kann die in diesem Kapitel betrachtete Kaskadenregelung nicht in die in Kapitel 3.4 verwendete Zustandsregelung überführt werden. Grund dafür ist eine Veränderung in den verwendeten Übertragungsfunktionen der Regler für die Kaskadenregelung.

Der Vorteil einer Kaskadenregelung steht in Verbindung mit der Aufsplittung des zu regelnden Gesamtsystems. Das Gesamtsystem wird hierbei in Teilsysteme unterteilt, für diese jeweils ein eigener Regler implementiert wird. Das heißt, es entsteht somit eine unterlagerte Regelung. Es wird zuerst für ein “Teilsystem 1” ein Regler entworfen. Der dazugehörige geschlossene Regelkreis dient zusammen mit dem “Teilsystem 2” als Strecke, um einen zweiten Regler auszulegen. Dies kann theoretisch für beliebig viele Teilsystem durchgeführt werden. Jedoch verliert man möglicherweise irgendwann den Überblick beim Auslegen der Regler. In Abbildung 3.20 ist die Struktur der Kaskadenregelung angegeben, welche in diesem Seminar für das System “Auto” verwendet wurde.

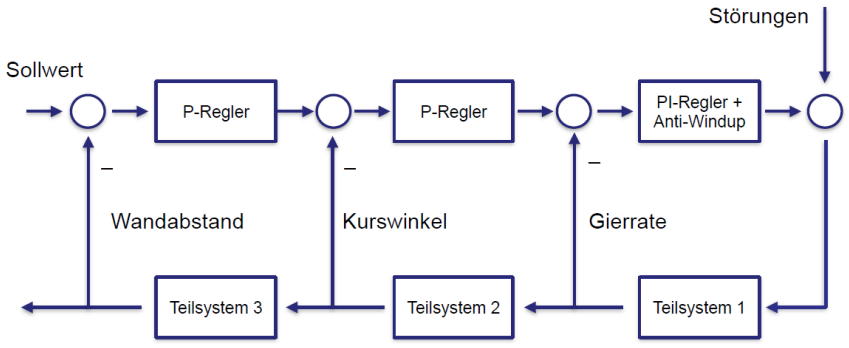

Abbildung 3.20: Struktur der Kaskadenregelung

In Abbildung 3.20 ist deutlich zu erkennen, dass für das erste Teilsystem mit der Gierrate als

Ausgang ein PI-Regler verwendet wird, um den Störungen des Lenkwinkels möglichst schnell entgegenzuwirken. Der geschlossene Regelkreis für das erste Teilsystem mit Regler wird zusammen mit dem zweiten Teilsystem verwendet, um eine Regelung für den Kurswinkel auszulegen. Im äußersten Regelkreis soll zu guter Letzt der eigentliche Wandabstand auf einen gewünschten Sollwert gebracht werden.

3.5.2 Wichtige Hinweise zur Auslegung

Bei der Auslegung ist zu beachten, dass der innerste Regelkreis bezüglich der Dynamik und des Einschwingverhaltens immer am schnellsten ausgelegt werden muss und der darauffolgende äußere Kreis langsamer. Dadurch kann gewährleistet werden, dass die einzelnen Kreise sich beim Einschwingen gegenseitig nicht stören.

Durch dieses Vorgehen kann der I-Anteil anstatt in den äußersten in den innersten Regelkreis gelegt werden, wodurch vermieden wird, dass hohe Regeldifferenzen aufintegriert werden. Dies war im zweiten Ansatz (PI-Zustandsregelung, Kapitel 3.4) der Fall, wodurch ein erhöhtes Überschwingen entsteht, obwohl Pole vorgegeben wurden, welche kein Überschwingen zur Folge haben sollten. Da in Abbildung 3.20 der I-Anteil im innersten Kreis liegt, welcher am schnellsten reagiert, wird kein hoher I-Anteil erzeugt, was wiederum eine Stellgröße in einem akzeptablen Bereich zur Folge hat und Windup-Effekte, die sich in Grenzen halten. Außerdem kann für den diskreten PI-Regler im innerste Kreis ein einfaches Element zur Reduzierung von Windup-Effekten entworfen werden. Hierfür kann nach Kapitel 9.3 in Skript [3] vorgegangen werden.

3.5.3 Auslegung des innersten Regelkreises bzgl. des Systems "Auto" (Regelung Gierrate)

Zur Auslegung des innersten Regelkreises wurde für die Strecke, welche nur aus einem Verstärkungsglied besteht, ein PI-Regler verwendet. Grund dafür ist die gewünschte stationäre Genauigkeit trotz Störungen der Lenkung. Damit ergibt sich das in Abbildung 3.21 zu sehende Strukturbild. Für die Auslegung wird, wie schon beim PID-Regler (Kapitel 3.3), das Wurzelortskurvenverfahren angewendet. Auch hier muss auf die Problematik der reellen negativen Achse des Einheitskreises geachtet werden, welche in Kapitel 3.3 schon erörtert wurde.

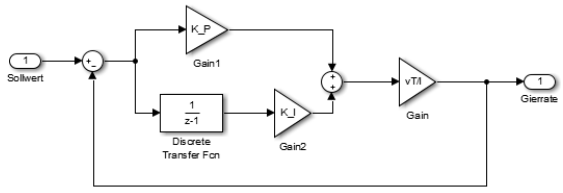

Abbildung 3.21: Struktur des inneren Kreises

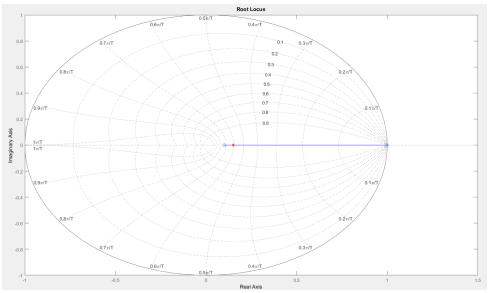

Abbildung 3.22: Wurzelortskurve des inneren Kreises

Aus Abbildung 3.21 kann auch die Übertragungsfunktion des offenen Regelkreises (19) und die dazugehörigen Wurzelortskurve (Abbildung 3.22) gezeichnet werden.

$$G_o(z) \quad = \quad k_{R,innen} \frac{vT}{l} \frac{z - n_{R,innen}}{z - 1} \quad . \tag{19}$$

Die Nullstelle des verwendeten PI-Reglers $n_{R,innen}$ wurde auf den Wert 0.1 gesetzt und der Pol des geschlossenen Regelkreises auf $z_p = 0.15$. Dadurch ergibt sich eine sehr schnelle Regelung ohne Überschwingen für den innersten Regelkreis. Die dazugehörige Verstärkung für den Regler kann über die Skalierungsgleichung aus Kapitel 2.2, Seite 27 von [7] berechnet werden:

$$\begin{aligned} |k| &= \frac{|N_0(z)|}{|Z_0(z)|} \\ &= \frac{|z_p - 1|}{|z_p - n_{R,innen}|} , \end{aligned}$$

wobei $|k| = k_{R,innen}\frac{vT}{l}$ ist. Dadurch kann $k_{R,innen}$ berechnet werden. Die entsprechenden Dateien zur Bestimmung der Wurzelortskurve des inneren Kreises sind im Anhang (Kapitel A.3) zu finden.

3.5.4 Auslegung des mittleren Regelkreises bzgl. des Systems "Auto" (Regelung Kurswinkel)

Da die Störungen bezüglich der Lenkung schon mit der innersten Schleife kompensiert werden, wird für die mittlere Schleife nur noch ein P-Regler verwendet. Weiterhin wird die Übertragungsfunktion des geschlossenen Regelkreises der inneren Schleife benötigt. Diese lässt sich mittels der Formel für die FührungsÜbertragungsfunktion aus [7], Kapitel 1 von Seite 8 berechnen und ergibt sich zu:

$$G_{w,innen} = \frac{k_{R,innen}\frac{vT}{l}}{1 + k_{R,innen}\frac{vT}{l}} \cdot \frac{z - n_{R,innen}}{z - \frac{1+n_{R,innen}k_{R,innen}\frac{vT}{l}}{1+k_{R,innen}\frac{vT}{l}}} = k_{Gier,w} \cdot \frac{z - n_{R,innen}}{z - p_{Gier,w}} . \tag{20}$$

Anhand des Strukturbildes (Abbildung 3.23) und der Wurzelortskurve des mittleren Regelkreises (Abbildung 3.24) erhält man eine Vorstellung darüber, wie der Parameter des P-Reglers zu wählen ist. Um die Wurzelortskurve in Matlab zu zeichnen, muss vorerst die Übertragungsfunktion des mittleren offenen Regelkreises der Kaskadenregelung aufgestellt werden. Dies entspricht jedoch gerade Gleichung (20) mit k_1 (Verstärkung P-Regler des mittleren Regelkreises) und einem I-Anteil multipliziert. k_1 wird in diesem Beispiel so gewählt, dass die zwei vorhandene Pole des geschlossenen Regelkreises um den Verzweigungspunkt in der rechten Halbebene auf der reellen Achse liegen. Dadurch sollte sich kein Überschwingen ergeben und durch die Lage nahe dem Nullpunkt ein schnelles Einschwingverhalten.

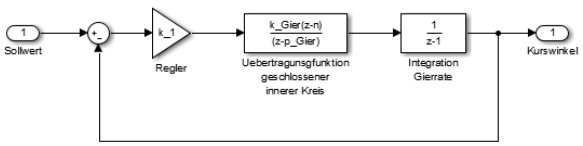

Abbildung 3.23: Struktur des mittleren Kreises

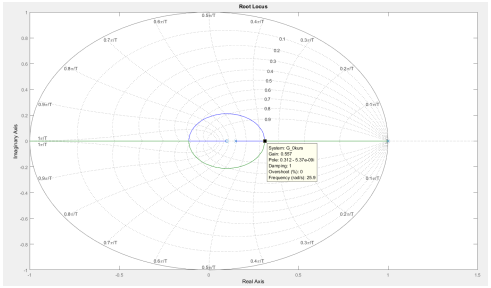

Abbildung 3.24: Wurzelortskurve des mittleren Kreises

In Abbildung 3.25 wird eine Sprungantwort für den geschlossenen mittleren Regelkreis gezeigt.

Die zur Simulation genutzten Dateien (Matlabcode und Simulinkmodell) sind ebenfalls im Anhang dargestellt.

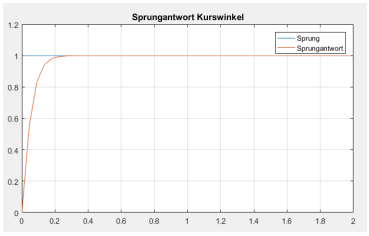

Abbildung 3.25: Sprungantwort Kaskadenregelung für Gierrate und Kurswinkel

3.5.5 Auslegung des äußeren Regelkreises bzgl. des Systems "Auto" (Regelung Wandabstand)

Anwendung des Wurzelortskurvenverfahrens und Problematiken der Pollagen

Um auch für den äußersten Regelkreis das Wurzelortskurvenverfahren anwenden zu können, wird hier ebenfalls die Übertragungsfunktion des geschlossenen mittleren Regelkreises benötigt. Diese lässt sich auf dieselbe Weise wie bei der Auslegung des mittleren Regelkreises berechnen und ist in Gleichung (21) angegeben:

$$G_{w,kurs} = k_1 k_{Gier,w} \frac{z - n_{R,innen}}{(z - p_{1,kurs,w})(z - p_{2,kurs,w})} \ , \tag{21}$$

wobei $p_{1,kurs,w}$ und $p_{2,kurs,w}$ wie folgt bestimmt werden können:

$$p_{1,kurs,w} = -1 \cdot \frac{k_1 k_{Gier,w} - p_{Gier,w} - 1}{2} + \sqrt{\left(\frac{k_1 k_{Gier,w} - p_{Gier,w} - 1}{2}\right)^2 + k_1 k_{Gier,w} n_{R,innen}} \ ,$$

$$p_{2,kurs,w} = -1 \cdot \frac{k_1 k_{Gier,w} - p_{Gier,w} - 1}{2} - \sqrt{\left(\frac{k_1 k_{Gier,w} - p_{Gier,w} - 1}{2}\right)^2 + k_1 k_{Gier,w} n_{R,innen}} \ .$$

Auch die Übertragungsfunktion des letzten Teilsystems wird für das Wurzelortskurvenverfahren benötigt. Diese kann aus dem Strukturbild (Abbildung 3.26) abgelesen werden. Gleichung (22) stellt diese Übertragungsfunktion dar:

$$G_{o,außen} = (l_H + \frac{vT}{2}) \cdot \frac{z - \frac{l_H - \frac{vT}{2}}{l_H + \frac{vT}{2}}}{z - 1} = k_{außen} \frac{z - n_{außen}}{z - 1} \ . \tag{22}$$

Für die Regelung des Abstandes zur Wand wird wieder ein P-Regler verwendet. Dies führt zu einer Übertragungsfunktion des offenen Regelkreises, welche aus der Multiplikation zwischen (21), (22) und der Verstärkung des verwendeten P-Reglers (k_2) besteht. Die somit gewonnene Funktion kann zum Zeichnen der Wurzelortskurve verwendet werden.

In Abbildung 3.27 ist ein Beispiel einer Wurzelortskurve für den geschlossenen Gesamtregelkreis gezeigt. Dabei fällt auf, dass aufgrund der Lage der Pole und Nullstellen der offenen Übertragungsfunktion keine geeignete Lage der Pole für den geschlossenen Regelkreis gefunden werden kann, die den Kriterien einer "guten Dynamik" entsprechen. An dieser Stelle nochmals der Hinweis, dass die negative reelle Achse im Einheitskreis nicht als Pollage verwendet werden sollte, da sonst zusätzliche Störungen die Sprungantwort beeinflussen. Außerdem sollte möglichst wenig Überschwingen vorhanden sein. Dies ist jedoch mit der Wurzelortskurve in Abbildung 3.27 nicht möglich.

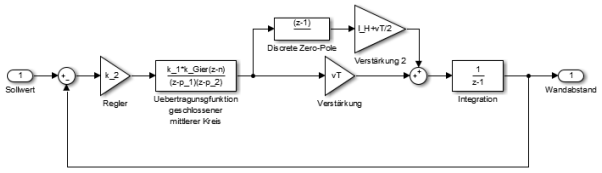

Abbildung 3.26: Struktur des äußeren Kreises

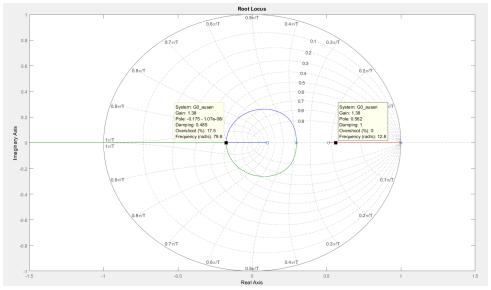

Abbildung 3.27: Wurzelortskurve des äußeren Kreises

Maßnahme zur Verbesserung der Pollagen

Eine Möglichkeit besteht darin, die Parameter des Autos zu beeinflussen (z.B. l_H) um die Nullstelle der Strecke des äußeren Teilsystems zu verschieben und somit geeignete Pollagen finden zu können. Angenommen die Größe l_H könnte zu Null gesetzt werden, dann folgt daraus eine Nullstelle der Strecke des äußeren Teilsystems bei -1, unabhängig der Geschwindigkeit des Autos v und der Abtastzeit des Systems. In Abbildung 3.28 ist eine Wurzelortskurve für ein solches Vorgehen dargestellt.

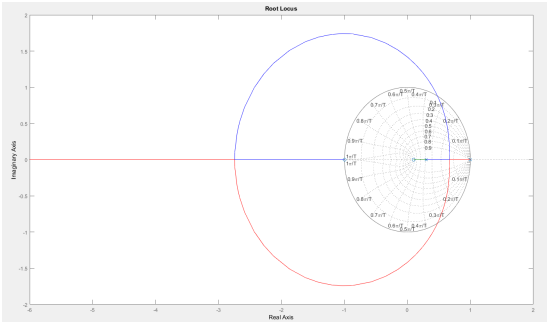

Abbildung 3.28: Wurzelortskurve des äußeren Kreises mit $l_H = 0$

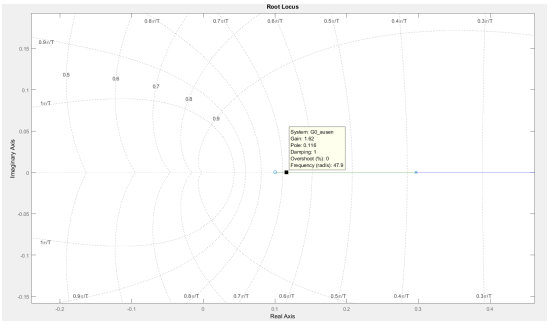

Abbildung 3.29: Wurzelortskurve des äußeren Kreises mit $l_H = 0$ (Ausschnitt grüner Verlauf)

In Abbildung 3.28 ist deutlich erkennbar, dass zum Beispiel die zwei Pole des geschlossenen Regelkreises um den Verzweigungspunkt in der rechten Halbebene des Einheitskreises auf der reellen Achse platziert werden können (roter und blauer Verlauf). Der dritte Pol (grüner Verlauf) würde niemals die negative reelle Achse erreichen, da sein Verlauf in der Nullstelle des innersten Regelkreises endet (Abbildung 3.29).

Was bedeutet jedoch eine Umrechnung auf $l_H = 0$? l_H ist der Abstand zwischen hinterem Rad des Fahrzeugs und dem Punkt, an dem die Sensoren zum Messen des Wandabstandes angebracht sind. Diese Größe wird bei der Herleitung des Zustandsraummodells verwendet und spielt deshalb eine wichtige Rolle für die Dynamik des Autos. Jedoch wäre es auch denkbar, dass die Sensoren zum Messen des Wandabstandes direkt über dem Hinterrad angebracht wären und somit ein l_H von Null folgen würde. Um dies zu verwirklichen und den Vorteil der Wurzelortskurve in Abbildung 3.28 nutzen zu können, muss jedoch die Größe l_H zu Null gesetzt werden. Dazu dürfen jedoch nur die gemessenen Größen verwendet werden, da mit ihnen die Dynamik des Fahrzeugs beeinflusst werden kann. Nicht möglich ist das einfache Umrech-

nen der Wandabstände mit Hilfe des Sinus oder Kosinus. Mit Hilfe der in Abbildung 3.30 bis 3.32 gezeigten Strukturbilder und deren Umformungen kann jedoch mit Hilfe des gemessenen Kurswinkels die Dynamik so beeinflusst werden, dass $l_H = 0$ gilt. Hierzu wurde in Abbildung 3.30 ein zusätzlicher Pfad eingefügt, welcher den Term $(z-1) \cdot l_H$ kompensiert und somit ein l_H von Null garantiert. Die Umformung zeigt, dass durch die Messung des Kurswinkels und dessen Multiplikation mit $-l_H$, wobei l_H den Wert vom realen Auto besitzen sollte, die Größe l_H zu Null gesetzt werden kann.

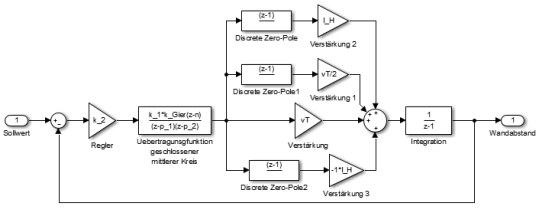

Abbildung 3.30: Strukturbildumformung um ein l_H von Null zu gewährleisten

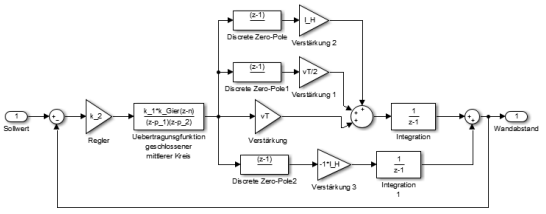

Abbildung 3.31: Strukturbildumformung um ein l_H von Null zu gewährleisten

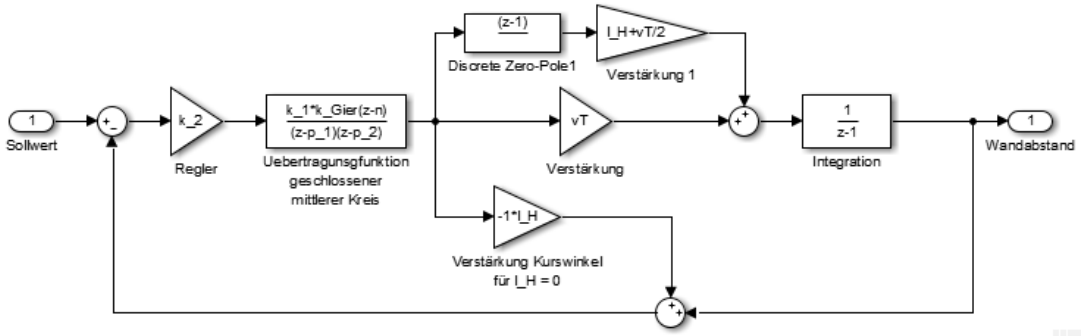

Abbildung 3.32: Strukturbildumformung um ein l_H von Null zu gewährleisten

3.5.6 Simulationsergebnisse

Die folgenden Simulationen sind mit der eben gezeigten Umrechnung von l_H durchgeführt worden. Im ersten Beispiel beträgt die Geschwindigkeit des Fahrzeugs $0.4\frac{m}{s}$ und die Startwerte des Wandabstandes und des Kurswinkels wurden beide auf Null gesetzt. Abbildung 3.33 zeigt die Sprungantwort des Systems (Abstand zur Wand gemessen vom Hinterrad) und Abbildung 3.34 zeigt die berechneten Lenkwinkel des Reglers (Stellgröße).

In Abbildung 3.33 ist kein Überschwingen erkennbar und auch die Stellgröße aus Abbildung 3.34 nimmt dank Anti-Windup-Maßnahme im innersten Regelkreis sehr schnell Werte in dem zur Verfügung stehenden Bereich an.

In einem weiteren Beispiel wurden die Anfangswerte verändert. Der Abstand zur Wand entspricht nun $0.8m$ und der Kurswinkel 20°. Damit ergeben sich die Verläufe für die Sprungantwort und Stellgröße aus Abbildung 3.35 und 3.36. Im Vergleich zu den Abbildungen 3.9

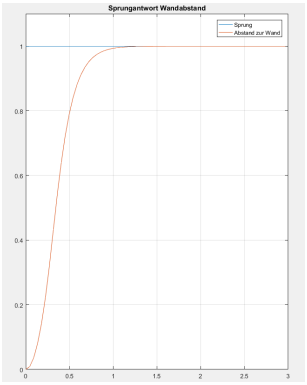

Abbildung 3.33: Sprungantwort Gesamtsystem

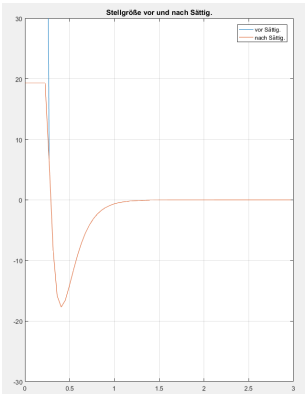

Abbildung 3.34: Stellgröße Gesamtsystem

und 3.10, welche mit den gleichen Startwerten, jedoch anderen Reglern simuliert wurden, ergibt sich für die Kaskadenregelung das beste Ergebnis. Aufgrund des D-Anteils ergibt sich eine schnelle Reaktion und Dank dem I-Anteil im innersten Regelkreis erkennt man kaum ein Überschwingen. Dagegen besitzt die Sprungantwort in Abbildung 3.9 (PI-Regler) ein sehr großes Überschwingen, welches auch der langsamen Reaktion geschuldet ist.

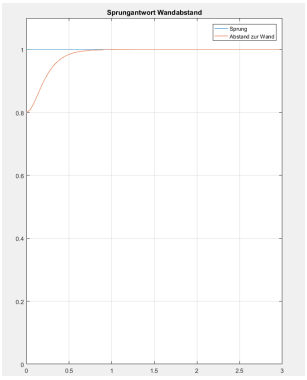

Abbildung 3.35: Sprungantwort Gesamtsystem

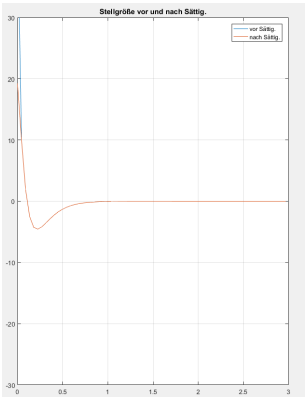

Abbildung 3.36: Stellgröße Gesamtsystem

Im nächsten Beispiel soll die Stellgröße des Zustandsreglers mit der Stellgröße des Kaskadenreglers verglichen werden. Hierzu wird eine Simulation der Kaskadenregelung mit einer Geschwindigkeit des Fahrzeugs von $v = 1\frac{m}{s}$ und den Startwerten $0.5m$ als Wandabstand und $-20°$ als Kurswinkel durchgeführt. In allen Simulationen der Kaskadenregelung ist ein Begrenzungsglied für den Lenkwinkel berücksichtigt. Die ergebnisse können in den Abbildungen 3.37 und 3.38 betrachtet werden. In den Abbildungen 3.17 und 3.19 sind ergebnisse bei Anwendung eines PI-Zustandsreglers dargestellt. Beim Vergleich dieser vier Abbildung fällt auf, dass die Stellgröße bei der Kaskadenregelung deutlich geringer ist und deshalb kaum Windup-Effekte zum tragen kommen. Diese sind bei Verwendung des PI-Zustandsreglers deutlich zu sehen. Ein Grund dafür ist die Lage des I-Gliedes in der PI-Zustandsreglelung, welche sich im äußersten Regelkreis befindet. Wie schon erwähnt, wurde das I-Glied der Kaskadenregelung in die innerste Regelschleife verschoben, womit nur noch kleine Regelabweichungen aufintegriert werden.

Die zur Simulation verwendeten Dateien wurden nicht dem Anhang hinzugefügt. Der Matlabcode der zurückliegenden Kapitel, sowie die Erklärungen in diesem Kapitel sollten bei aus-

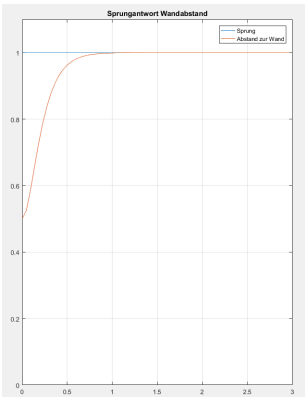

Abbildung 3.37: Sprungantwort Gesamtsystem

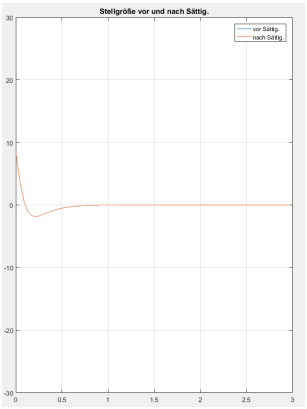

Abbildung 3.38: Stellgröße Gesamtsystem

reichender Kenntnis aus den Grundvorlesungen (Digitale Regelungstechnik 1, Regelungstechnik 1 und 2, Praktikum Matlab/Simulink 1) genügen, um selbst den Matlabcode für die Simulationen anzufertigen.

Aus den Simulationsergebnissen kann man schlussfolgern, dass die Kaskadenregelung die besten Ergebnisse bezüglich der Pflichtaufgabe liefert. Im Folgenden sollen nochmals Hinweise bezüglich der Handhabung und ein Ausblick für die folgenden Seminarteilnehmer gegeben werden.

3.5.7 Ausblick und Handhabung des Kaskadenreglers

Ein großer Nachteil der Kaskadenregelung ist die benötigte Messung des Kurswinkels. Dieser müsste noch zusätzlich über einen Kalmanfilter ermittelt werden. Ansätze dazu stehen in dem vom Fachgebiet “Regelungstechnik und Mechatronik” bereitgestellten Dokument. Außerdem muss die Größe l_H auf die Hinterachse umgerechnet werden. Dies ist ebenfalls mit Hilfe der Messung vom Kurswinkel möglich. Da jedoch der gemessene Kurswinkel meistens mit Rauschen überlagert ist, kann dies zu nicht exakten Umrechnungen der Größe l_H führen und somit zu unerwünschten Effekten in der Sprungantwort. Es bedarf einer möglichst guten Schätzung und/oder Filterung des Kurswinkels, damit ein zufriedenstellender Einsatz der Kaskadenregelung möglich ist.

Hinzu kommen außerdem noch Rauschen in der Gierrate und in dem Abstand zur Wand, welche durch die Sensoren verursacht werden. Auch weisen die Sensoren in der Regel einen Bias auf, welcher in der Regelung auch noch nicht berücksichtigt wurde.

Das Konzept und die ergebnisse der Kaskadenregelung versprechen zwar gute Simulationsergebnisse, jedoch wird hier davon ausgegangen, dass nur die Lenkung mit einer Störung beaufschlagt ist und der Kurswinkel exakt bekannt ist. Dies entspricht aber nicht den realen Bedingungen am Auto. Um dennoch das Konzept anwenden zu können müssen noch einige zusätzliche Effekte (z.B. Rauschen oder Bias von Sensoren) berücksichtigt werden.

Da zum Schluss dieses Seminars auf Grund der eben genannten Bedingungen der Einsatz des Kaskadenreglers noch nicht möglich war, wurde zum Wettbewerb der schon in den Vorjahren oft verwendete PD-Regler implementiert. Dieser soll im nächsten Abschnitt nochmals kurz vorgestellt werden.

3.6 Beim Wettbewerb verwendeter diskreter PD-Regler

3.6.1 Allgemeiner Ansatz

Der im Wettbewerb verwendete PD-Regler basiert auf der zeitdiskreten Gleichung (23) und kann mittels der Z-Transformation in den Z-Bereich transformiert werden, um das Wurzelortskurvenverfahren zur Auslegung dieses Reglers verwenden zu können.

$$u_k = K_P \cdot e + \frac{K_D}{T}(e_k - e_{k-1}) \ , \tag{23}$$

wobei u_k den Reglerausgang beschreibt und e_k den Eingang. Dementsprechend beschreibt e_{k-1} den Eingang vom vorherigen Zeitpunkt. K_P und K_D sind die Verstärkungsfaktoren der Reglerkomponenten (P- und D-Anteil) und T die Abtastzeit. Gleichung (24) zeigt die umgeformte Z-Transformation (Übertragungsfunktion des Reglers) von Gleichung (23):

$$G(z) = \frac{\frac{1}{Tk_P+k_D}}{T} \cdot \frac{z - \frac{k_D}{Tk_P+k_D}}{z} = k_R \frac{z-n_R}{z} \ . \tag{24}$$

Die Umrechnung ist im Anhang näher beschrieben.

3.6.2 Wurzelortskurvenverfahren zur Auslegung des PD-Reglers

Mit Hilfe dieser Übertragungsfunktion und der diskreten Übertragungsfunktion des Systems "Auto" (Gleichung (11)), kann im Weiteren das Wurzelortskurvenverfahren verwendet werden, um für verschieden Möglichkeiten der noch freien Reglerparameter Wurzelortskurven zu zeichnen. Anhand dieser kann eine Vorstellung über das Systemverhalten bei den unterschiedlichen Möglichkeiten der Reglerparameter übermittelt werden.

In Abbildung 3.39 ist eine Wurzelortskurve und in 3.40 die dazugehörige Sprungantwort für die Wahl $k_D = k_P = 15$ und $T = 0.045$ dargestellt.

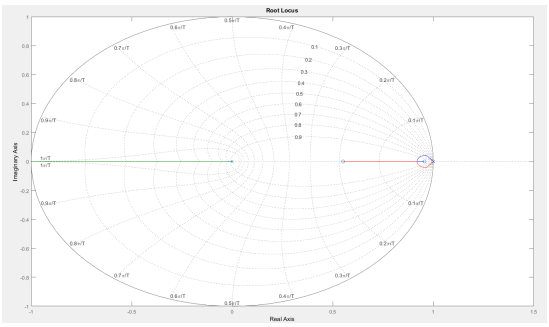

Abbildung 3.39: Wurzelortskurve mit PD-Regler

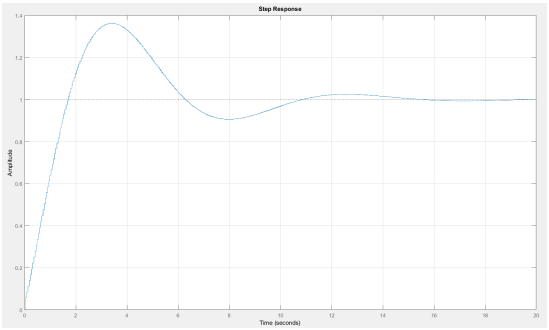

Abbildung 3.40: Sprungantwort mit PD-Regler

Um die Abbildungen richtig zu interpretieren ist noch die Pollage des geschlossenen Regelkreises wichtig. Diese können mittels Matlab bestimmt werden und liegen für das konkrete Beispiel bei:

$$\begin{aligned} p_1 &= -0,0321 \\ p_2 &= 0,9864 + \mathrm{i} \cdot 0,0304 \\ p_3 &= 0,9864 - \mathrm{i} \cdot 0,0304 \end{aligned}$$

Da p_2 und p_3 ein konjugiert komplexes Polpaar bilden, ergibt sich im Verlauf der Sprungantwort ein Überschwingen. Zum Verständnis wird auf die Literatur [3] verwiesen. Die Eigenschaften des Polpaares p_2 und p_3 spiegeln sich in der Sprungantwort des Systems (Abbildung 3.40) wieder.

Bei Betrachtung der Wurzelortskurve kann man feststellen, dass ein Pol auf der negativen reellen Achse entlang läuft. Für diese Achse wurde in Kapitel 3.3 eine Problematik angesprochen, welche jedoch in der Sprungantwort 3.40 kaum bemerkbar ist. Der Grund hierfür ist die Lage des ersten Pols p_1, welcher sich sehr nahe an der Null befindet. Je näher ein Pol an der Null liegt, desto geringer sind die Auswirkungen des Effektes der negativen reellen Achse. Diese Erkenntnis könnte genutzt werden, um die Wurzelortskurven aus den vorangegangenen Kapiteln nochmals zu überarbeiten und somit einfachere Lösungen zu finden, welche einen I- und D-Anteil als Reglerkomponente besitzen.

Die zur Simulation verwendeten Dateien wurden nicht dem Anhang hinzugefügt.

3.6.3 Nachteile und Hinweise PD-Regler

Der Ansatz des PD-Reglers hat einen großen Nachteil. Da kein I-Anteil im Regelalgorithmus enthalten ist, kann die stationäre Genauigkeit bei Störungen auf der Lenkung nicht gewährleistet werden. Das Auto hat jedoch mit diesem Regler und der Hilfe der Hinderniserkennung die Pflichtaufgabe gut genug gemeistert.

Ein weiterer Nachteil der konkreten Auslegung sind die starken Schwingungen. Diese würden vor allem bei großen Abweichungen vom Sollwert Probleme bereiten. Deshalb wurde der Regler nicht für das Befahren von Kurven verwendet. Hierfür wurde eine Steuerung benutzt die das Auto mit einem bestimmten Lenkwinkel um die Kurve fahren lässt (siehe Kapitel 4.3.3). Nach Abschluss dieses Manövers kommt der PD-Regler wieder zum Einsatz.
Außerdem kam es manchmal vor, dass der PD-Regler bei der Zufahrt auf die Wand nicht schnell genug reagiert hat. Das wird in Zusammenarbeit mit dem Code der Hinderniserkennung kompensiert, insbesondere durch den Teil zur Fahrtplanung bei Erkennung einer Wand (siehe Kapitel 4.3.2).

Die oben gezeigte Wahl von k_D und k_P ist nur für hohe Geschwindigkeiten geeignet. Um den Regler auch für verschiedene Geschwindigkeiten einsetzen zu können, müssen die Parameter in Abhängigkeit der Geschwindigkeit gewählt werden. Hierzu können wie schon in den vorherigen Kapiteln die Gleichungen der Wurzelortskurve genutzt werden, welche in [7] in den Kapiteln 2.2 und 2.3 notiert und beschrieben sind. Dabei muss aufgepasst werden, dass der Reglerpol bei Null nicht den Einheitskreis verlässt, wodurch der geschlossene Regelkreis instabil wäre.

Auch besteht die Möglichkeit, anstatt der hier angegebenen Wahl von k_P und k_D andere Werte für die Reglerparameter zu finden, welche eine bessere Dynamik zur Folge haben. Dies ist mit viel Geduld und ausprobieren verbunden.

4 Zweite Aufgabe: Autonomes Umfahren von Hindernissen

Bei dieser Aufgabe wird derselbe Rundkurs absolviert wie bei der Pflichtaufgabe. Allerdings werden auf dem Rundkurs verschiedene Hindernisse platziert, die das Fahrzeug erkennen muss und für die es entsprechende Manöver einleiten muss, um ihnen auszuweichen. Dabei haben wir zunächst am White Board Ideen für einen Algorithmus gesammelt und verfeinert, bis es dann an eine Implementierung ging.
Dabei kamen wir schnell zu dem Schluss, dass die Ultraschallsensoren für das zuverlässige Erkennen von Hindernissen nicht ausreichen, und wir deshalb für diesen Versuch die Kinect als emulierten Laserscanner miteinbezogen haben. Ausgangspunkt für diese Aufgabe ist, dass das Fahrzeug einer Wand folgt, wie beim Rundkurs ohne Hindernisse. Dieser Zustand wird nun um eine Hinderniserkennung und Pfadplanung erweitert.

4.1 Laserscan in kartesische Koordinaten umrechnen

Da die Laserscan-Daten initial als Polarkoordinaten vorliegen, erschien es uns sinnvoll, diese zuerst in ein kartesisches System umzurechnen. Das vereinfacht die Bestimmung von Koordinaten von Elementen im Sichtfeld, da man direkten Zugriff auf x- und y-Werte hat. Diese werden sehr oft benötigt, wie im weiteren Verlauf ersichtlich werden wird.

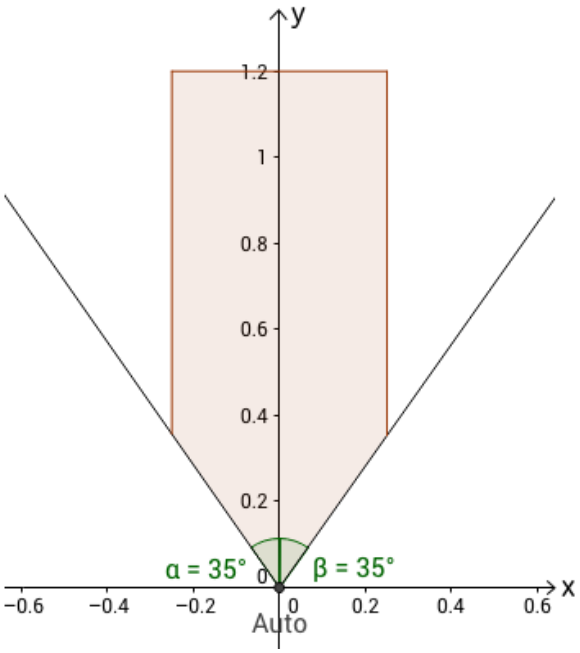

Abbildung 4.1: Kinect-Blickfeld mit eingezeichnetem Hinderniserkennungsbereich

Den in der Skizze ersichtlichen Erkennungsbereich haben wir so gewählt, da weiter außen liegende Hindernisse oder Objekte für die aktuelle Fahrt nicht relevant sind. Nur was in Fahrtrichtung direkt vor dem Fahrzeug auftaucht sollte berücksichtigt werden.

4.2 Erkennung und Klassifizierung des Hindernisses

Nachdem nun der Erkennungsbereich auf dem aktuellen Laserscan definiert ist geht es daran, während der Fahrt vor dem Fahrzeug auftauchende Hindernisse zu erkennen. Dazu wird im regelmäßig ablaufenden Timer-Callback (Intervall von 0.1s) eine Methode obstacleFound()

aufgerufen, welche überprüft, ob ein Messwert innerhalb des Erkennungsbereiches liegt. Ist dies der Fall, wird der Typ des Objekts bestimmt: Wand, Hindernis oder Nichts.

Dabei wird zuerst davon ausgegangen, dass der gefundene Index des Scans teil einer Wand ist und es wird probiert, den Winkel zu dieser Wand zu bestimmen. Dazu geht der Algorithmus in beide Richtungen über die Laserscan-Werte und kontrolliert, ob diese noch zu der Wand gehören, indem er überprüft, ob die Änderung der Werte in einem gewissen Rahmen bleibt. Solange dies der Fall ist werden diese Veränderungen in einer Liste gespeichert, um später den Gesamtgradienten der Wand zu bestimmen. Diese Schleife läuft solange, bis entweder der gerade geprüfte Index 50 cm von dem zu Beginn gefundenen Index entfernt ist, oder ein Punkt nicht mehr zu der Wand gehört. Wobei 50 cm empirisch gewählt ist, um eine Mindestlänge der Wand von einem Meter vorrauszusetzen. Wenn die Schleife abbricht, weil ein Punkt eine zu große Veränderung zum Vorgänger aufweist, wird das Hindernis nicht mehr als Wand gewertet, sondern als reguläres Hindernis. Bricht der Algorithmus normal ab, wird eine Wand erkannt und der Winkel zu dieser über den Gradienten bestimmt.

Da die Entfernung zum Schnittpunkt der Wand und der y-Achse des Autos aus dem mittleren Scanpunkt bekannt ist, kann auch der Schnittpunkt mit der x-Achse bestimmt werden. Der Winkel ergibt sich dann aus dem Arkustangens des ersten Schnittpunktes durch den zweiten.
Falls dieser Winkel so gering ist, dass die Regelung ihn von selber ausgleichen kann, wird die Wand nichtmehr als solche behandelt und es wird „Nichts“ an den Haupttimer zurückgegeben (bei uns waren das ungefähr 10°). Ansonsten wird der Winkel zurückgegeben, der später durch ein Lenkmanöver ausgeglichen wird.

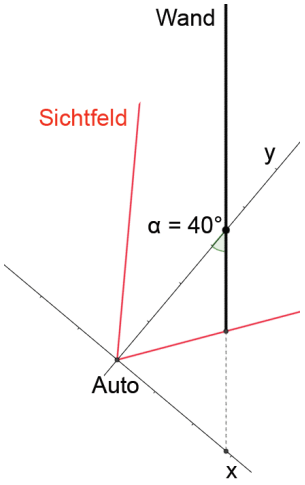

Abbildung 4.2: Beispiel der Winkelbestimmung

4.3 Fahrtplanung abhängig von der Klasse des Hindernisses

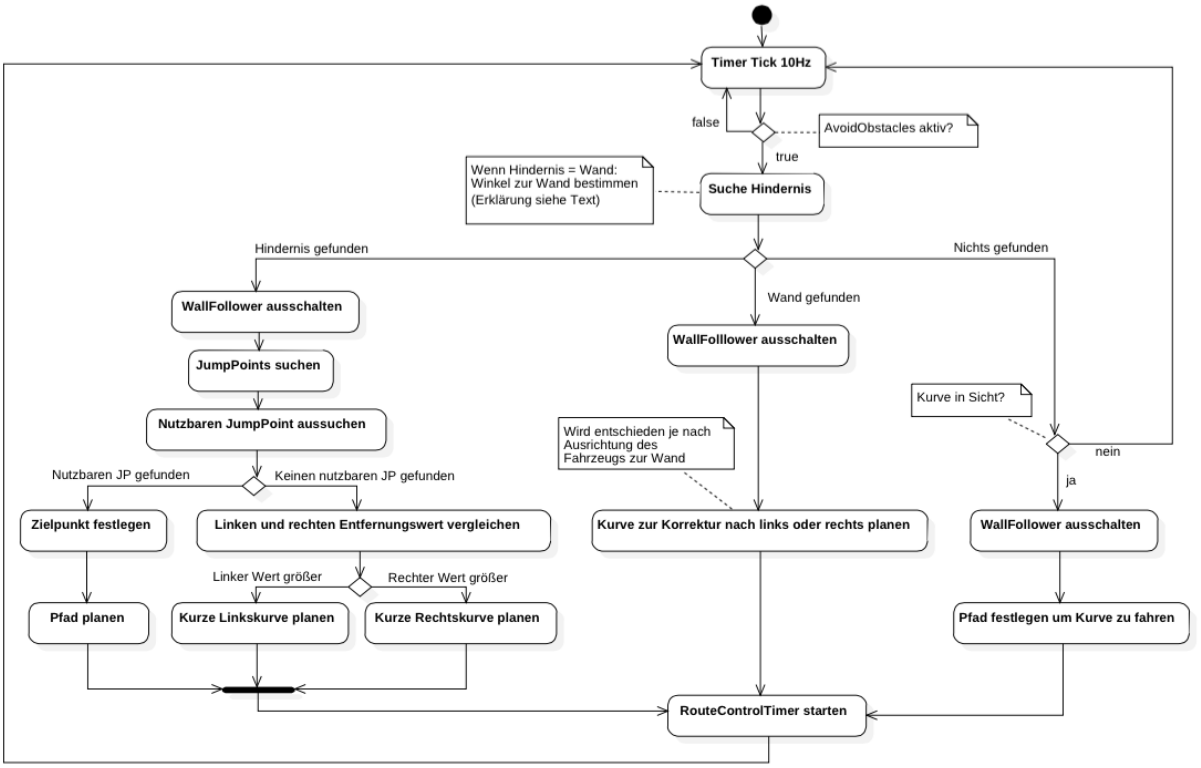

Abbildung 4.3: Zustandsdiagramm des Haupttimers

4.3.1 Fahrtplanung bei einem Hindernis

Wenn ein Hindernis erkannt wurde wird als nächstes nach dem Ende des Hindernisses gesucht. Dies geschieht über Erkennung von Sprüngen in den Werten des Scans, die im folgenden als „Jumppoints“ oder „Sprungpunkte“ bezeichnet werden. Zuerst wird eine Liste aller möglichen Jumppoints unter einer Entfernung von 1,5 Metern zum Auto erstellt und nach der Relevanz für das Ausweichen sortiert (je näher, desto wichtiger). Danach muss geprüft werden, ob neben diesen Jumppoints genügend Platz zum Ausweichen vorhanden ist. Abhängig davon, ob man den Sprungpunkt links oder rechts umfahren muss, wird in die jeweilige Richtung geprüft, ob alle weiteren Punkte eine euklidische Distanz von mindestens 50 cm zum Sprungpunkt aufweisen. Ist dies nicht der Fall wird der Jumppoint verworfen und ein weiterer wird geprüft.
Falls dann kein weiterer Jumppoint nutzbar ist wird der Entfernungswert außen links mit dem Entfernungswert außen rechts verglichen und eine feste 25° Kurve in die jeweilige Richtung geplant. Dieser Fall tritt nur extrem selten auf und hat deswegen kaum Priorität bei uns bekom-

men, weshalb die Lösung auch einer Verbesserung bedarf.
Wenn ein nutzbarer Sprungpunkt gefunden ist, wird ein Zielpunkt für die Fahrtplanung 20 cm in die Ausweichrichtung gesetzt. Dieser Zielpunkt wird im Normalfall mit einer S-Kurve angefahren, also eine Kurve gefolgt von einer Gerade und einer weiteren Kurve. Dazu werden die Geichungen (25) und (26) nach der Länge der Geraden und dem Winkel der Kurven aufgelöst. Dabei gehen wir von den minimalen Wendekreisen unseres Autos aus.
Falls das Ziel näher als der Wendedurchmesser in Y-Richtung und gleichzeitig zu weit links oder rechts liegt, kann es vorkommen, dass eine S-Kurve nicht mehr möglich ist. In diesem Fall wird nur eine Kurve und dann eine Gerade geplant. Nachteil ist, dass man nicht mehr in derselben Orientierung ankommt, wie bei Start des Ausweichmanövers. Hierfür werden die Gleichungen (27) und (28), abhängig von der Richtung der Kurve, nach Länge und Winkel aufgelöst.
In speziellen seltenen Fällen hat auch dieses Gleichungssystem keine reelle Lösung. Wenn das auftritt, wird eine kleine Strecke rückwärts als Pfad gesetzt.
Die so ermittelten Pfade werden im RouteControlTimer verwendet, der mit Hilfe des Drehgebers die Strecken abfährt.

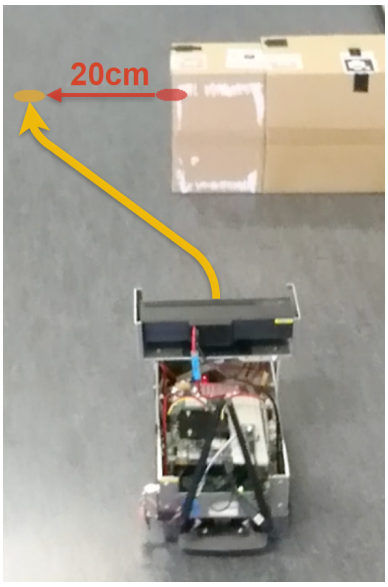

Abbildung 4.4: Fahrtplanung Hindernis

$$Y_z = (W_l + W_r) * \sin(\alpha) + S * \cos(\alpha) \tag{25}$$

$$X_z = (W_l + W_r) * (1 - \cos(\alpha)) + S * \sin(\alpha) \tag{26}$$

$$Y_z = W_{l|r} * (1 - \cos(\alpha)) + S * \sin(\alpha) \tag{27}$$

$$X_z = W_{l|r} * (1 - \cos(\alpha)) + S * \sin(\alpha) \tag{28}$$

Tabelle 4.1: Formelzeichen

Y_z	Y-Entfernung zum Zielpunkt
X_z	X-Entfernung zum Zielpunkt
W_l	minimaler linker Wendekreis
W_r	minimaler rechter Wendekreis
α	Winkel des Kreisbogens
S	Länge der Gerade

4.3.2 Fahrtplanung bei einer Wand

Für den Fall „Wand“ wird nur eine einzelne Kurve geplant. Diese ist abhängig von dem vorher ermittelten Winkel zur Wand (siehe 4.2), sodass das Fahrzeug nach dem Manöver wieder parallel zu dieser Wand steht. Diese geplante Kurve wird ebenfalls mit dem RouteControlTimer abgefahren.

4.3.3 Fahrtplanung bei keinem Hindernis

Wie aus dem Zustandsdiagramm ersichtlich wird, wird der WallFollower für die Abfahrt eines geplanten Pfades ausgeschaltet. Ist kein Hindernis in Sicht, bleibt dieser aber aktiv. Dennoch wird währenddessen kontinuierlich nach einer möglichen Kurve gesucht. Dies ist vorallem für den normalen Rundkurs ohne Hindernisse von großer Bedeutung, da dadurch viel Zeit gewonnen werden kann.
Da unser Algorithmus immer der rechten Wand folgt, erkennt er auch nur Kurven nach rechts. Dies kann aber auch ohne großen Aufwand auf die andere Richtung erweitert werden. Grundsätzlich werden die Werte von ganz rechts bis zur Mitte des Scans nach einer Wand und einem darauffolgenden Sprung abgesucht. Wenn eine Abbiegung gefunden wurde, wird eine Gerade bis zu dieser geplant, gefolgt von einer 90-Grad-Kurve.
Zunächst muss der äußerste Entfernungswert in X-Richtung unterhalb einer festgelegten Grenze liegen, damit keine Rechtskurve geplant wird während das Auto mitten im Raum steht. Ist das der Fall, wird von dem Index ausgehend zur Mitte hin kontrolliert, ob all diese Werte zu einer Wand gehören. Die Abstandswerte in X-Richtung dürfen keine zu große Steigung haben, damit gewährleistet ist, dass das Auto parallel genug zur Wand steht und diese gerade ist.
Ist die Wand noch keine 0.75 Meter lang (empirischer Wert) und ein Sprung tritt auf, wird die Kurvenplanung verworfen, da es sich auch um ein gerade umfahrenes Hindernis handeln kann. Bei diesem will man natürlich nicht immer eine 90-Grad-Kurve fahren.
Die Suche nach einer Kurve wird abgebrochen, wenn die Wand schon 2 Meter lang ist und bis dahin kein Sprung gefunden wurde, der eine Kurve signalisieren würde. Dies ist der Tatsache geschuldet, da die Werte der Kinect bei weiter entfernten Objekten schlechter auflösen.
Wenn alle Bedingungen erfüllt sind (Wandlänge: 0.75-2 Meter, Geringe Schwankung in X-Werten, Auto nah genug an der Wand) wird zuletzt noch kontrolliert, ob genügend Platz zum Abbiegen an der Kurve vorhanden ist. Dies wird auf die selbe Art geprüft wie auch schon bei den Sprungpunkten (siehe 4.3.1).

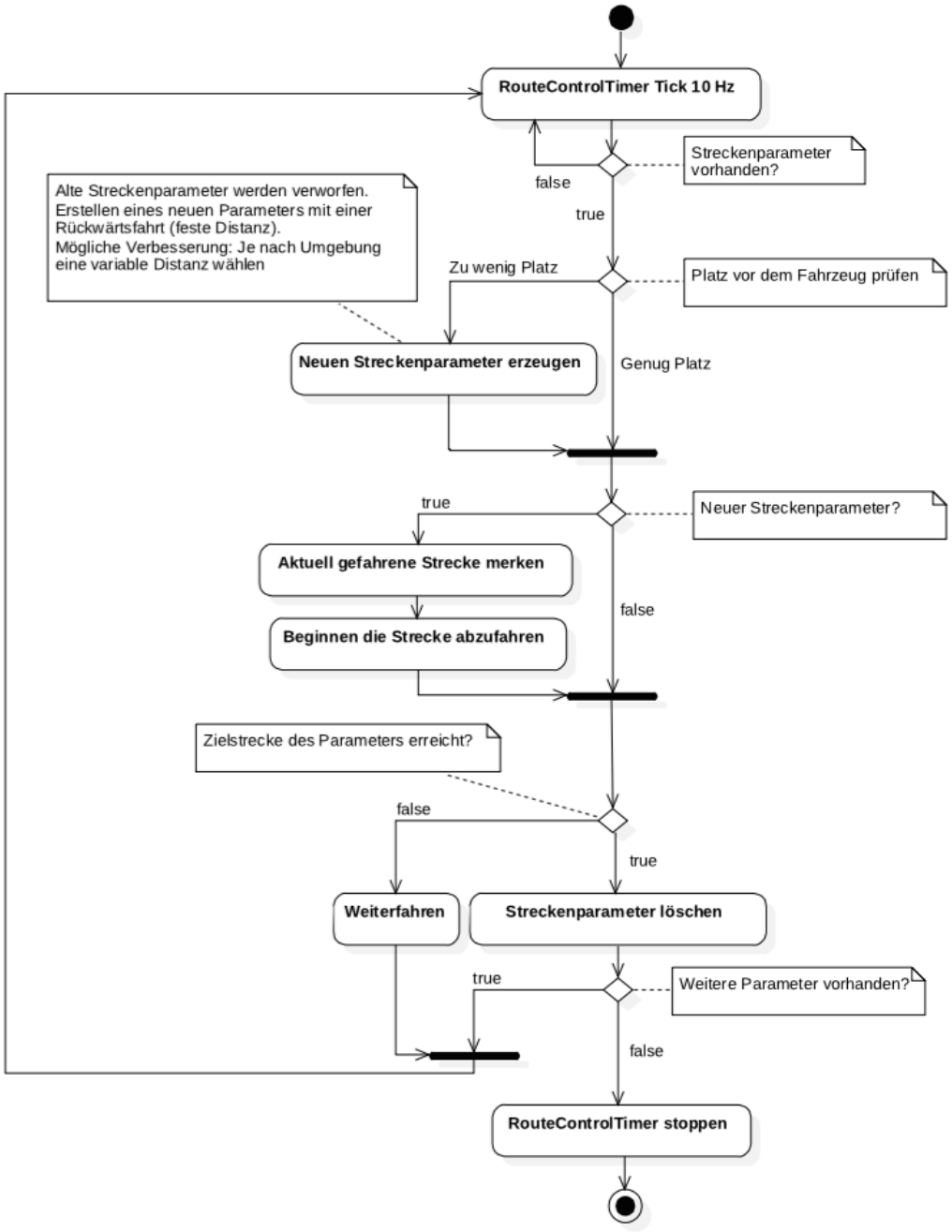

Abbildung 4.5: Zustandsdiagramm des RouteControlTimers

An den RouteControlTimer werden Streckenparameter mit den folgenden Eigenschaften übergeben: Distanz der Strecke, Lenkung während der Ausführung des aktuellen Manövers und Geschwindigkeit mit der die Strecke abgefahren wird.
Die Geschwindigkeit hängt davon ab, ob man den Rundkurs ohne oder mit Hindernissen fährt und ist für gerade Strecken höher als für Kurven. Der Wert des Lenkeinschlags ist für eine Geradeausfahrt Null, und für Kurven der der jeweiligen Richtung entsprechende Maximaleinschlag. Eine variable Lenkung belastet zwar den Servo nicht so stark und sieht auch schöner aus, aber

dadurch werden die Berechnungen für S-Kurven deutlich komplexer. Als Verbesserung könnte man dies, ebenso wie eine variable Geschwindigkeit, implementieren. Für gerade Strecken wird die Distanz des Pfades übergeben, für Kurven muss vorher noch der Winkel der Kurve mit dem Wenderadius multipliziert werden.
Der Timer fährt so lange einen Streckenparameter ab, bis die Distanz des Parameters mit der vom Drehgeber ermittelten Strecke übereinstimmt. Danach folgt der nächste Parameter, bis alle abgearbeitet sind.
Während dieser Fahrt ist es nicht nötig, die normale Hinderniserkennung laufen zu lassen, da zum Zeitpunkt der Fahrtplanung darauf geachtet wird, dass überall genügend Platz vorhanden ist. Da das Auto in der Realität leider nicht immer wie berechnet fährt, kann es sein, dass der errechnete Zielpunkt leicht von dem tatsächlichen Zielpunkt abweicht. Um zu garantieren, dass das Auto auf dem Weg nicht mit einem Gegenstand kollidiert, wird während der Fahrt der Bereich direkt vor dem Auto auf Hindernisse kontrolliert. Falls ein solches Hindernis auftaucht, werden alle Streckenparameter verworfen. Stattdessen wird eine kurze Strecke ruckwärts als neuer Pfad gesetzt. Als Verbesserung könnte man dieses Vorgehen natürlich noch optimieren.
Sobald die kurze Rückwärtsfahrt abgeschlossen ist, sucht der Algorithmus erneut nach Hindernissen und plant Pfade solange, bis das Problem umfahren ist.

5 Dritte Aufgabe: Autonomes paralleles Einparken

5.1 Vereinfachtes Modell

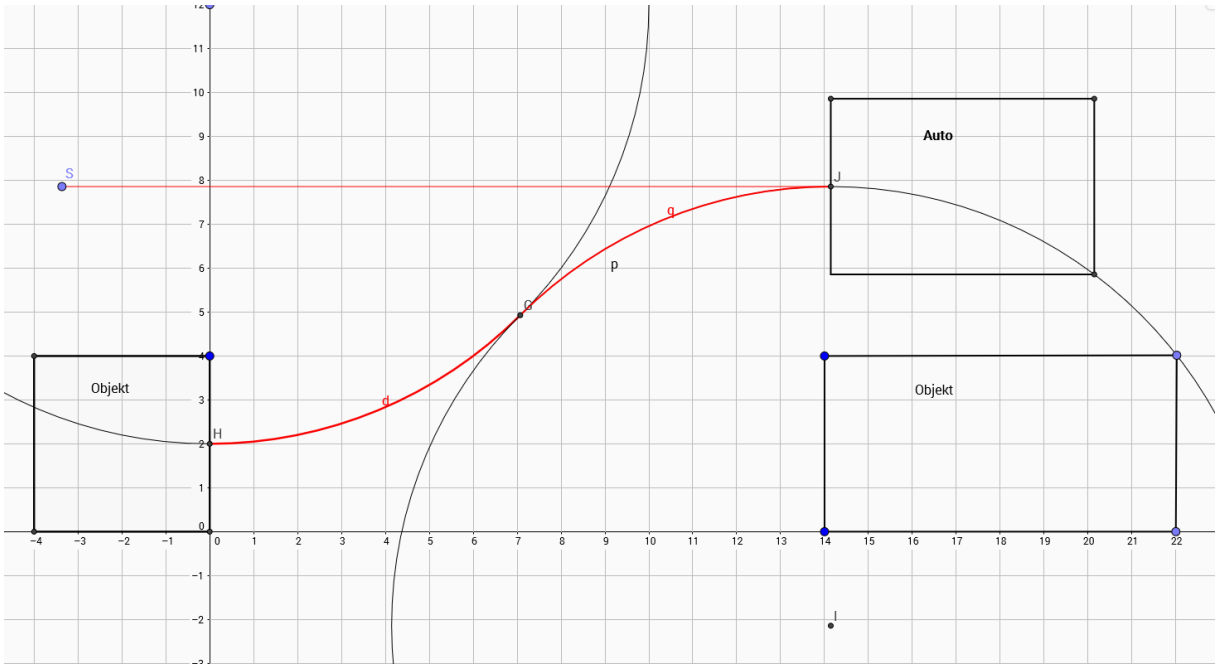

Abbildung 5.1: GeoGebra-Modell des minimalen Einparkens

Für das parallele Einparken des Autos aus der Rückwärtsfahrt wurde von uns als Grundlage das Einparken in zwei Zügen in eine minimale Parklücke genommen. Diese beschreibt die Bewegung des Autos, indem jeweils nur zwei Kurvenabschnitte gefahren werden, ohne dabei mehr als zwei Einschlagwinkel vorgeben zu müssen. Die Länge der Kurvenabschnitte ergibt sich aus den maximalen Einschlagwinkeln des Autos, sowohl nach links, als auch nach rechts. Um diese zu ermitteln, ist es notwendig den Durchmesser des Wendekreises zu kennen, den das Auto bei maximalem Einschlag beschreibt. Dieser kann sich bei Vorwärts- und Rückwärtsfahrt, als auch zu beiden Seiten hin, unterscheiden.

$$D = 2 \cdot \frac{L}{\sin(\alpha)} \tag{29}$$

In Formel (29) beschreibt D hierbei den Spurkreis und nicht den eigentlichen Wendekreis, wobei dieser Unterschied bei dem Modellfahrzeug vernachlässigbar ist. Durch Umkehr der Formel nach α ist nun der maximale Einschlagwinkel der Räder berechenbar.

Mit den vorliegenden Durchmessern ist es möglich die minimale Parklücke zu berechnen, um das Fahrzeug kollisionsfrei einzuparken.

In Abbildung 5.1 ist zu sehen, dass von einer Bahn ausgegangen wird, welche in der Mitte der Fahrzeugbreite angesetzt wird und von dort die beiden Kreisbögen beschreibt, die abgefahren werden sollen. Zur Visualisierung des Modells wird das mathematische Zeichenprogramm *GeoGebra* verwendet. Die beiden Kreisbögen liegen in einem Koordinatensystem, um hierbei deren Höhe und Länge in X- als auch in Y-Richtung zu bestimmen. Durch geeignete trigonometrische

Formeln lassen sich diese Werte für das Fahrzeug berechnen. Einfache Addition der X-Werte ergibt die minimale Parklückenlänge.

$$X_{Bogen} = \frac{D}{2} \cdot \sin(45°) \tag{30}$$

$$Y_{Bogen} = \frac{D}{2}(1 - \cos(45°)) \tag{31}$$

$$X_{Gesamt} = 2 \cdot X_{Bogen} \tag{32}$$

In Gleichungen (30) - (36) sind die Parameter der Parklücke zu erkennen. X_{Bogen} entspricht der Abszisse und Y_{Bogen} der Ordinate des Kreisbogens. X_{Gesamt} ist die Länge der minimalen Parklücke. Diese Strecke stellt den Teil der Fahrt dar, welcher bei der Parkplatzsuche in Vorwärtsrichtung abgefahren werden muss.

Anschließend ist es notwendig die Länge der Kreisbögen zu kennen, die das Auto abfahren soll. Diese sind in Formel (33) zu berechnen.

$$Bogenlänge = \frac{1}{8} \cdot 360° \cdot \frac{D}{2} \tag{33}$$

Zur Berechnung des minimalen Abstands in Y-Richtung, konkret die Breite der Parklücke, addieren sich die Breiten der Kreisbögen.

$$Y_{Gesamt} = 2 \cdot Y_{Bogen} \tag{34}$$

5.2 Erweiterung des Einparkens

Eine Erweiterung des Parkalgorithmus stellt die Herausforderung dar, aus einem beliebigen Abstand zur Wand einparken zu können und immer am gewünschten Einparkort anzukommen. Dazu muss die Bahn, welche das Auto abfährt erweitert werden. Hierbei wird als Ergänzung zwischen den minimalen Kreisbogenabschnitten eine einfache Gerade gewählt. Die Länge der Geraden ist abhängig vom Abstand zur Wand. Es ist notwendig die Parameter der Parklücke neu zu berechnen, da sich jetzt Breite und Länge der Parklücke, im Modell X- und Y-Parameter, ändern. Diese lassen sich mit folgenden Formeln verallgemeinern, um eine generische Bahn zu planen.

$$X_{Mittel} = Y_{Mittel} = \delta - 2 \cdot Y_{Gesamt} \qquad \delta \geq Y_{Gesamt} \tag{35}$$

$$X_{Gesamt} = 2 \cdot X_{Bogen} + X_{Mittel} \tag{36}$$

$$Geradenlänge = \sqrt{2} \cdot X_{Mittel} \tag{37}$$

δ beschreibt den Abstand des Autos zur Wand, welcher in jedem Fall größer als der minimale Abstand zur Wand Y_{Gesamt} sein muss. Y_{Gesamt} ist die Breite der beiden Kreisbögen, welche immer abgefahren werden. Y_{Mittel} entspricht der Ordinate des Geradenstückes im Koordinatensystem. Dieses ist gleich dem Abszissenteil der Geraden, das additiv zur Parklückenlänge zu ergänzen ist. Je größer der Abstand zur Wand ist, desto länger muss die Parklücke sein und somit auch das zu ergänzende Geradenstück.

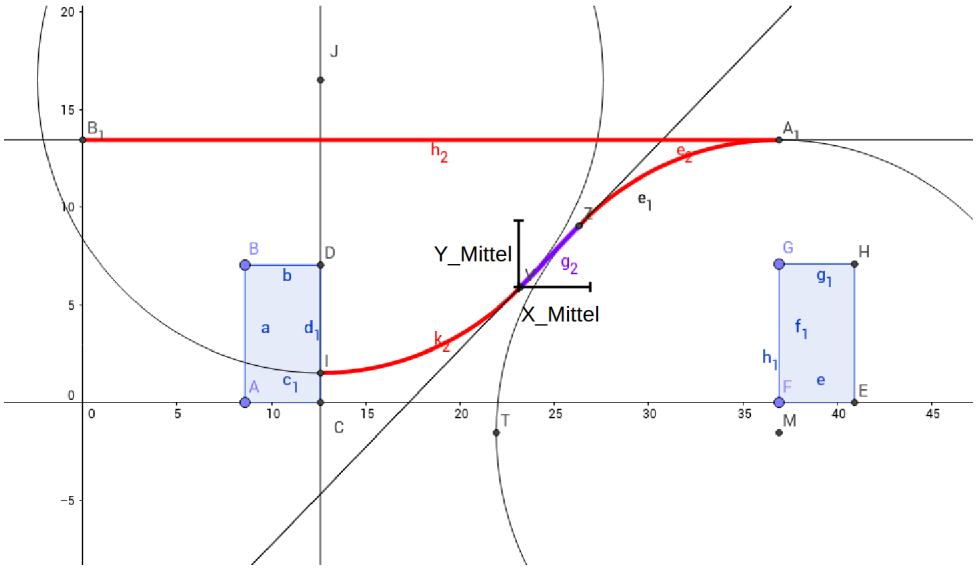

Abbildung 5.2: GeoGebra-Modell des generischen Einparkens

5.3 ROS-Implementierung

In der Implementierung wird der gesamte Ablauf des Einparkens in zwei Prozeduren unterteilt. Das Fahrzeug sucht während der Vorwärtsfahrt erst eine Parklücke und anschließend wird der eigentliche Einparkvorgang eingeleitet. Zur Ermittlung des Abstands zur Wand wird der seitlich des Fahrzeugs angebrachte Ultraschallsensor eingesetzt. Aus dieser Distanz wird nun die notwendige Parklückenlänge berechnet, die das Fahrzeug mindestens vorwärts zurücklegen muss. Ist eine Parklücke gefunden, stoppt das Fahrzeug und die Bahn wird aus den vorher berechneten Parametern bestimmt und rückwärts abgefahren. Das heißt konkret, dass erst ein Kreisabschnitt mit maximalem Lenkeinschlag nach rechts vorgegeben wird, anschließend das Geradenstück und der zweite Kreisabschnitt folgt.

Um die festgelegten Distanzen abzufahren, wurden Timer in ROS verwendet. Diese vergleichen periodisch die aktuell akkumulierte Distanz mit dem berechneten Zielwert. Die Timer terminieren jeweils sobald die entsprechenden Strecken zurückgelegt sind.

Desweiteren wurden konstante Parameter wie der Wendekreis in eine Header-Datei ausgelagert, um auch für andere Aufgaben verwendet werden zu können. Alle dynamischen Parameter werden in der eigentlichen C-Datei durch Funktionen zur Laufzeit berechnet. Dies trifft zum Beispiel auf die Strecke zwischen den Kreisbögen zu oder auch auf die Parklückenlänge.

6 Fazit

6.1 Ergebnis

Das Projektseminar ist insgesamt betrachtet sehr erfolgreich abgelaufen. Die gesetzten Ziele wurden erreicht und konnten im dafür vorgesehenen Zeitrahmen umgesetzt werden. Dabei war es sinnvoll vorab zu klären, welche Aufgaben unsere Gruppe bearbeiten will, um eine grobe Zeitplanung über das Semester hinweg einhalten zu können.
Ein großer Vorteil war die Möglichkeit, Arbeiten parallel auszuführen und Arbeitsressourcen neu zu verteilen, sobald eine Aufgabe erledigt war. Planung der Aufgaben und Organisation in der Gruppe war ein essentieller Bestandteil, um eine Umsetzung der Ziele zu gewährleisten, da ansonsten eine effektive Arbeit nicht möglich gewesen wäre.
Am Anfang stand eine gemeinsame Einarbeitung in die Fahrzeugtheorie an, als auch ein Aufsetzen von ROS und die Bereitstellung der Umgebung für jeden Computer. Dies ermöglichte einen gemeinsamen Start in das Projekt und eine gleiche Grundlage für jedes Mitglied in der Gruppe. Theorieausarbeitung und Implementierung der einzelnen Aufgaben konnten sehr gut parallelisiert werden und sind im Verhältnis zum Zeitaufwand im geplanten Rahmen gewesen.
Testphasen am eigentlichen Auto haben mehr Zeit in Anspruch genommen als dafür angedacht war, da es sich als schwierig herausstellte, erarbeitete Theorie und Implementierung auf das eigentliche Fahrzeug umzusetzen. Es war schwer abzusehen, inwieweit Fehler und Störeinflüsse auf das Fahrzeug wirken, welche in den einzelnen Modellen nicht immer realitätsnah dargestellt waren. Auch mussten verschiedene Ansätze zur Problemlösung, beispielsweise bei der Hindernisumfahrt, wieder verworfen werden, da eine tatsächliche Umsetzung entweder nicht funktioniert oder den zeitlichen Rahmen überschritten hätte.

6.2 Mögliche Verbesserungen

Alle Aufgaben waren davon beeinträchtigt, die Lenkung des Fahrzeugs zu korrigieren und auch Werte für die gefahrene Strecke zu ermitteln. Eine Lösung für die abweichende Lenkung war es den Lenkeinschlag um einen Offset zu versetzen, damit eine genauere Geradeausfahrt erreicht wird. Dabei wäre es sinnvoll, sich mit der eigentlichen physikalischen Lenkung erneut auseinanderzusetzen, um das grundsätzliche Fahrverhalten zu verbessern.
Die Sensorik, die eine gefahrene Strecke ermitteln sollte, wurde mit einem Drehgeber vereinfacht. Da die Odometrie des Drehgebers robuster gegenüber Störungen ist als die des Hall-Sensors, ist dessen Verwendung einfacher und mit weniger Arbeitsaufwand verbunden. Der Drehgeber war allerdings nicht optimal am Auto befestigt, sodass über eine bessere Montage nachgedacht werden sollte.
Desweiteren wäre es von Vorteil, die Regelung des Wandabstandes ebenfalls beim Einparken zu verwenden, um dies zu unterstützen. Dabei könnte eine Abweichung von der gewünschten zur Wand parallelen Startposition des Fahrzeugs ausgeglichen werden.

6.3 Fazit zum Projektseminar

Das Seminar ist eine gute Gelegenheit, sich mit autonomem Fahren und dessen grundlegenden Problemen und Algorithmen auseinanderzusetzen. Es bietet eine Breite an Herausforderungen in Bezug auf hardwarenahe und High-Level-Programmierung, Entwurfsphasen von eigenen Algorithmen zur Problemlösung und auch eine Tiefe in Bezug auf die Regelungstechnik.
Daher ist es von Vorteil eine heterogene Gruppe zu haben, die unterschiedliche Fachgebiete abdeckt. Kritikpunkt ist hierbei die zu Anfang nicht ersichtliche Tiefe in die Regelungstechnik, was dem Titel *Projektseminar Echtzeitsysteme* nicht ganz gerecht wird.

Insgesamt lässt das Seminar viele Freiheiten in der Umsetzung der Aufgaben zu und bietet einen ersten Einblick in das autonome Fahren sowie das praktische Arbeiten mit der damit verbundenen ROS-Umgebung.

A Anhang

A.1 Aufbau einer CMakeLists-Datei

```
cmake_minimum_required(VERSION 2.8.3)
project(<Projektname>)

## Alle Abhaengigkeiten des Pakets auflisten
find_package(catkin REQUIRED COMPONENTS
    roscpp
    roslib
    rospack
    std_msgs
    genmsg
    sensor_msgs
    message_generation)

## Angabe aller ROS-Messages, die vom Paket verwendet werden
add_message_files(
    FILES
    Example.msg
)

## Generate added messages and services
generate_messages(
    DEPENDENCIES
    std_msgs
)

## Declare a catkin package
catkin_package(
    CATKIN_DEPENDS message_runtime std_msgs)

## Build
include_directories(include)
include_directories(include ${catkin_INCLUDE_DIRS})

# 3 Zeilen fuer jeden Node
add_executable(exampleNode src/exampleNodeSrc.cpp)
target_link_libraries(exampleNode ${catkin_LIBRARIES})
add_dependencies(parking ${catkin_EXPORTED_TARGETS})
```

Abbildung A.1: Eine CMakeLists-Datei

A.2 Aufbau einer Launch-File

Listing A.1: Knightriders Launch-File

```
<?xml version="1.0" encoding="utf-8"?>
<launch>

        <arg name="camera" value="camera"/>
        <arg name="manager" value="$(arg camera)_nodelet_manager" />

        <node pkg="depthimage_to_laserscan" type="depthimage_to_laserscan" name="depthimage_to_laserscan
             " args="image:=kinect2/sd/image_depth">
                <param name="scan_height" value="20" />
                <param name="scan_time" value="0.1" />
                <param name="range_min" value="0.01" />
                <param name="range_max" value="25.0" />
                <param name="output_frame_id" value="base_footprint" />
        </node>

        <!-- Run all the knightriders nodes -->
        <node pkg="knightriders" type="rcboard"  name="rcboard"  output="screen" respawn="true" />
        <node pkg="knightriders" type="wallfollow" name="wallfollow" output="screen" respawn="true" />
        <node pkg="knightriders" type="parking" name="parking" output="screen" respawn="true" />
        <node pkg="knightriders" type="avoidobstacles" name="avoidobstacles" output="screen" respawn="true" /
             >
</launch>
```

A.3 Modellierung und Ansätze zur Reglerauslegung

A.3.1 Berechnung der Elemente einer zeitdiskreten Zustandsraumdarstellung

Zuerst soll die Matrix $\boldsymbol{A}_d$ berechnet werden, welche sich über Gleichung (8) wie Folgt bestimmen lässt:

$$\begin{aligned} \boldsymbol{A}_d &= \exp\left(\begin{bmatrix} 0 & v \\ 0 & 0 \end{bmatrix} T\right) \\ &= \exp\left(vT \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\right) . \end{aligned}$$

Um die Exponentialfunktion einer Matrix zu lösen, müssen spezielle Verfahren angewendet werden, welche zum Beispiel im Kapitel 9 von [3] nachgelesen werden können. In unserem speziellen Fall bietet sich die Berechnung der Exponentialfunktion über die Jordan'sche Normalform an. Man erkennt leicht, dass $\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ ein Jordanblock zum Eigenwert $\lambda = 0$ ist. Deshalb kann die Berechnung der Exponentialfunktion über Folgende Formel berechnet werden, welche im Skript [4] auf Seite 10 zu finden ist:

$$\exp(\boldsymbol{J}_i \cdot t) = \exp(\lambda_i t) \cdot \begin{bmatrix} 1 & t & \frac{t^2}{2} & \cdots & \frac{t^{n_i-1}}{(n_i-1)!} \\ 0 & 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \frac{t^2}{2} \\ \vdots & & \ddots & \ddots & t \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix} , \tag{38}$$

wobei

$$\boldsymbol{J}_i = \begin{bmatrix} \lambda_i & 1 & \cdots & 0 \\ 0 & \lambda_i & \ddots & \vdots \\ \vdots & & \ddots & 1 \\ 0 & \cdots & 0 & \lambda_i \end{bmatrix},$$

als Jordanblock zum Eigenwert λ_i bezeichnet wird. Mit Gleichung (38) ergibt sich für das System "Auto" für $\boldsymbol{A}_d$ folgende Rechnung:

$$\begin{aligned} \boldsymbol{A}_d &= \exp\left(vT \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \right) \\ &= e^{0 \cdot vT} \cdot \begin{bmatrix} 1 & vT \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & vT \\ 0 & 1 \end{bmatrix}. \end{aligned}$$

Als nächstes muss der Eingangsvektor $\boldsymbol{b}_d$ berechnet werden. Hierfür wird Gleichung (9) benötigt. Die benötigte Matrix $e^{\boldsymbol{A}\nu}$ entspricht der Matrix $e^{\boldsymbol{A}T}$ bis auf die Konstante T. Deshalb kann das Ergebnis $\boldsymbol{A}_d$ zur Berechnung von $\boldsymbol{b}_d$ herangezogen werden. Für $e^{\boldsymbol{A}\nu}$ ergibt sich $\begin{bmatrix} 1 & v\ \nu \\ 0 & 1 \end{bmatrix}$. Setzt man dies in Gleichung (9) ein und beachtet, dass die Integration über einen Vektor auf jedes einzelne Element angewendet werden kann, ergibt sich folgende Rechnung:

$$\begin{aligned} \boldsymbol{b}_d &= \int_0^T \begin{bmatrix} 1 & v\ \nu \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v \cdot \frac{l_H}{l} \\ \frac{v}{l} \end{bmatrix} d\nu \\ &= \int_0^T \begin{bmatrix} v \cdot \frac{l_H}{l} + \frac{v^2}{l}\nu \\ \frac{v}{l} \end{bmatrix} d\nu \\ &= \begin{bmatrix} v \cdot \frac{l_H}{l}\nu + \frac{v^2}{2l}\nu^2 \\ \frac{v}{l}\nu \end{bmatrix} \Bigg|_0^T \\ &= \begin{bmatrix} v \cdot \frac{l_H}{l}T + \frac{v^2T^2}{2l} \\ \frac{v}{l}T \end{bmatrix}. \end{aligned}$$

A.3.2 Berechnung der zeitdiskreten Übertragungsfunktion für das System "Auto"

Um die Übertragungsfunktion zu berechnen, wird Gleichung (10) verwendet und die entsprechenden ermittelten Größen des diskreten Zustandsraummodells eingesetzt. Somit erhält man die folgende Rechnung:

$$
\begin{aligned}
G(z) &= \boldsymbol{c}_d^T \left(z \cdot \boldsymbol{I}_n - \boldsymbol{A}_d\right)^{-1} \boldsymbol{b}_d + d_d \\
&= \boldsymbol{c}_d^T \begin{bmatrix} z-1 & -v\ T \\ 0 & z-1 \end{bmatrix}^{-1} \boldsymbol{b}_d \\
&= \boldsymbol{c}_d^T \cdot \frac{1}{(z-1)^2} \begin{bmatrix} z-1 & v\ T \\ 0 & z-1 \end{bmatrix} \boldsymbol{b}_d \\
&= \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \frac{1}{(z-1)^2} \begin{bmatrix} z-1 & v\ T \\ 0 & z-1 \end{bmatrix} \begin{bmatrix} \frac{v\ l_H T}{l} + \frac{1}{2} \cdot \frac{v^2 T^2}{l} \\ \frac{v\ T}{l} \end{bmatrix} \\
&= \frac{1}{(z-1)^2} \left(\left(\frac{v l_H T}{l} + \frac{v^2 T^2}{2l} \right) \cdot (z-1) + \frac{v^2 T^2}{2l} \right) \\
&= \left(\frac{v\ l_H\ T}{l} + \frac{v^2 T^2}{2\,l} \right) \frac{z - \left(\frac{2\,l_H\ T - v\ T^2}{2\,l_H\ T + v\ T^2} \right)}{(z-1)^2} \ .
\end{aligned}
$$

A.4 Diskreter PID-Regler (1. Ansatz)

A.4.1 Reduktion PID- auf PI-Regler

In diesem Abschnitt wird gezeigt, dass die Kompensation des Reglerpols in Null eines PID-Reglers mit einer entsprechenden Nullstelle auf einen PI-Rgeler führt. Hierzu wird die folgende Rechnung durchgeführt:

$$
\begin{aligned}
G_R(z) &= \frac{k_R(z-n_1)(z-n_2)}{z(z-1)} \\
&= \frac{k_R(z)(z-n_2)}{z(z-1)} \\
&= \frac{k_R(z-n_2)}{z-1} \ .
\end{aligned}
$$

Im Weiteren wird $G_R(z)$ als $\frac{Y(z)}{U(z)}$ geschrieben und die Funktion in den zeitdiskreten Bereich zurück transformiert:

$$
\begin{aligned}
Y(z) &= k_R \frac{z}{z-1} U(z) - k_R n_2 \frac{1}{z-1} U(z) \\
&\ \ \vdots \\
y(k) &= k_R \sum_{\nu=0}^{k} u_\nu - k_R n_2 \sum_{\nu=0}^{k-1} u_\nu \\
&= k_R u_k + k_R \sum_{\nu=0}^{k-1} u_\nu - k_R n_2 \sum_{\nu=0}^{k-1} u_\nu \\
y_k &= k_R u_k + k_R (1-n_2) \sum_{\nu=0}^{k-1} u_\nu \ .
\end{aligned}
$$

Die letzte Gleichung entspricht von der Form her einem PI-Regler im zeitdiskreten Bereich.

A.4.2 Matlabcode PI-Regler

Im Listing A.2 ist die Matlabdatei zum starten der Simulation dargestellt. Diese berechnet die wichtigsten Größen für die Regelung und das System "Auto" in Unterdateien, startet die Simulation und stellt die Ergebnisse in Matlabplots dar.

Listing A.2: Starten der Simulation

```
%Modell simulieren:
param_Regler
sim('sim_modell_Zustandsraummodell')
plotten_ergeb_Zustandsraummodell
```

Listing A.3 beinhaltet die zur Simulation benötigten Parameter. In den Zeilen 1 bis 7 sind die Systemparameter und die Abtastzeit angegeben. Die Anfangswerte der Strecke werden in Zeile 10 übergeben und die Matrizen des Zustandsraummodells des Systems "Auto" werden in Zeile 11 berechnet. Hierzu wird die Funktion aus Listing A.4 verwendet. In den Zeilen 15 bis 25 aus Listing A.3 werden die Pole und Nullstellen festgelegt. Nullstelle n_3 entspricht der Streckennullstelle und n_2 dem Freiheitsgrad aus dem PI-Regler. Den Verzweigungspunkt, welcher für die Lage der Pole des geschlossenen Regelkreises von Interesse ist wird in Zeile 35 berechnet. Die Formel ergibt sich aus der Regel 4 von Kapitel 2.3 auf Seite 39 in [6]. Die Berechnung der Verstärkung für den Regler in den Zeilen 55 bis 57 ergibt sich ebenfalls aus [6] aus Kapitel 2.2 Seite 27. Diese Listings könnten als grobe Vorlage dienen um eine Simulation für das eigene System mit Regler durchzuführen. Es ist jedoch für das Verständnis empfehlenswert, seinen eigenen Code zu schreiben.

Listing A.3: Parameter für die Simulation

```
1  %Systemparameter:
   T = 0.045; %Abtastzeit
   v = 0.4; %Wert zwischen 0 und 2 m/s
   v_begin = 0.1;
   v_end = 2;
6  lH = 0.175;
   l = 0.35;

   %Matrizen der Strecke bestimmen:
   x0 = [0.8; 20];
11 [Asys,Bsys,Csys,Dsys] = mat_sys(T,v,lH,l);

   %Diskretisierter offener Regelkreis:
   %Pole
   p1 = 1;
16 p2 = 1;
   p3 = 1;
   p4 = 0;

   %Nullstellen:
21 %Wahl fuer PI-Regler:
   n1 = 0;
```

```
n3 = (2*lH*T-v*T^2)/(2*lH*T+v*T^2);
%n2 = (1-n3)/2+n3;
n2 = -(1-n3)/2+n3;

n3_begin = (2*lH*T-v_begin*T^2)/(2*lH*T+v_begin*T^2);
n3_end = (2*lH*T-v_end*T^2)/(2*lH*T+v_end*T^2);

%Reglerverstaerkunug:
k = 1;

%Verzweigungspunkte:
verzw_1 = -1*(1-n3-n2)+sqrt((1-n3-n2)^2-3*n2*n3+n3+n2);
verzw_2 = -1*(1-n3-n2)-sqrt((1-n3-n2)^2-3*n2*n3+n3+n2);

%Wunschpole des geschlossenen Regelkreises bei obiger Wahl der Pole und Nullstellen:
pol_w_1 = verzw_2 -0.025;
pol_w_2 = verzw_2 + 0.025;

%offene Strecke mit Regler:
sys_0 = zpk([n1 n2 n3],[p1 p2 p3 p4],[k],T);

%Wurzelortskurve:
figure();
rlocus(sys_0);
hold on;
plot([verzw_1, verzw_2],[0,0], 'r*');
hold on;
plot([pol_w_1, pol_w_2],[0,0], 'y*');

%%%%%%%%%%%%%%
% k waehlen!!!
%%%%%%%%%%%%%%
k_zaehl = abs((pol_w_1-1)^3);
k_nenn = abs((pol_w_1-n2)*(pol_w_1-n3));
k_reg = k_zaehl/k_nenn;

%Parameter fuer Regler bestimmen:
b2 = k_reg*2*l/(v*(T*2*lH+v*T^2));
b12 = -n2;

%Sprungantwort:
sys_0 = zpk([n2 n3],[p1 p2 p3],[k_reg],T);
sys_feedb = feedback(sys_0,1);

[p_feed] = pole(sys_feedb);

figure();
step(sys_feedb);
```

```
grid on;
title('Sprungantwort');
```

Listing A.4: Funktion um die Matrizen der Strecke auszugeben

```
%Funktion um Matrizen der Strecke fuer Simulation zu berechnen:
function[Asys,Bsys,Csys,Dsys] = mat_sys(T,v,lH,l)
    Asys = [1 v*T; 0 1];
    Bsys = [(v*lH*T/l)+(v^2*T^2/(2*l)); v*T/l];
    Csys = [1 0];
    Dsys = 0;
end
```

A.4.3 Simulinkmodell zur Simulation

In Abbildung A.2 ist auch das Simulinkmodell dargestellt, welches zur Simulation des geschlossenen Regelkreises verwendet wurde. In diesem ist zu erkennen, dass nicht nur das Zustandsraummodell und der PI-Regler implementiert sind, sondern auch noch weitere Elemente. Zum Beispiel ein Anti-Windup Element, um die Stellgrößenbeschränkung der Lenkung zu berücksichtigen. Dieses wurde nach Kapitel 9.3 aus [1] entworfen. Auch wurde eine "Interpreted Matlab Function" genutzt, um das Mapping zwischen dem berechneten Lenkwinkel des Reglers und dem eigentlich Tickwert für den Servo zu berücksichtigen.

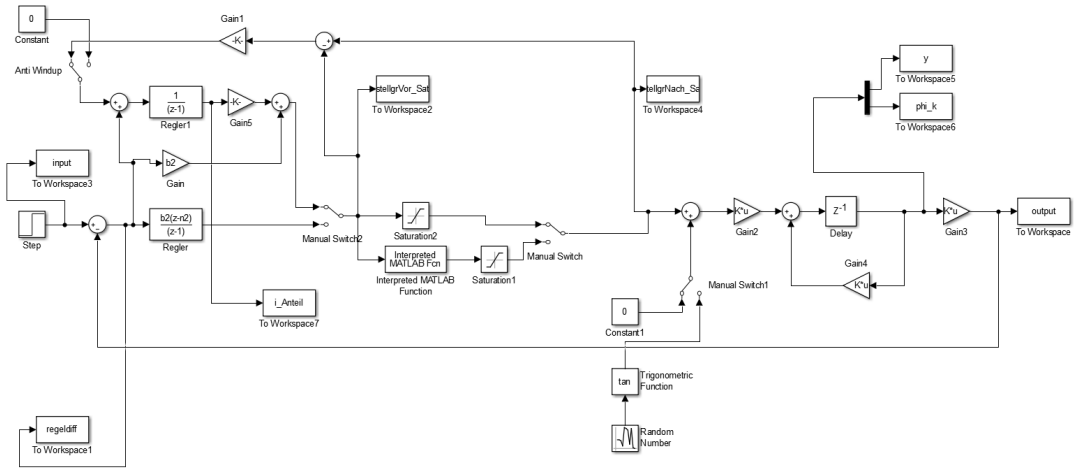

Abbildung A.2: Simulinkmodell zur Simulation

A.5 Diskreter PI-Zustandsregler (2. Ansatz)

A.5.1 Matlabcode zum Regler

Im Listing A.2 ist eine änliche Matlabdatei zum starten der Simulation dargestellt, wie sie für den diskreten PI-Zustandsregler verwendet wurde. Diese berechnet die wichtigsten Größen für

die Regelung und das System “Auto“ in Unterdateien, startet die Simulation und stellt die Ergebnisse in Matlabplots dar. Die Datei, welche die wichtigsten Grösën für die Regelung berechnet ist in Listing A.5 dargestellt.

Listing A.5: Berechnung der Reglerparameter

```
%Parameter des diskreten Modells:
T = 0.045; %Abtastzeit
v = 1; %Wert zwischen 0 und 2 m/s
lH = 0.175;
l = 0.35;

x0 = [0.5 ; -20]; %y-Startwert, phi_k-Startwert

%Pole des geregelten Systems:
p1 = 0.7;
p2 = 0.8;
p3 = 0.9;
[a0,a1,a2] = param_poly(p1,p2,p3);

%Matrizen der Strecke bestimmen:
[Asys,Bsys,Csys,Dsys] = mat_sys(T,v,lH,l);

%Matrizen des Gesamtsystems:
[Ages,Bges] = mat_gesamtsys(T,v,lH,l);

%Regler berechnen:
[q, Ages2, Ages3] = regler_eingabe_para(T,v,lH,l);
qneu = [0 0 1]*inv(ctrb(Ages, Bges));
r = regler_berechnen(q,Ages, Ages2, Ages3, a0, a1, a2);
rneu = regler_berechnen(qneu,Ages, Ages2, Ages3, a0, a1, a2);
rneu2 = l/(v^3*T^3)*[-1*(1+a0+a1+a2)*(lH+v*T/2)+v*T*(3+a1+2*a2) ,
    -1*v*T*(a1+2*a2+3)*(lH+v*T/2)+(1+a0+a1+a2)*(lH+v*T/2)^2+v^2*T^2*(3+a2) ,
    -1*v*T*(1+a0+a1+a2)];

%Pole des geschlossenen Regelkreises:
eig_Val = eig(Ages-Bges*rneu2');

%Vorfilter auslegen:
r_vorFil = rneu2(1:2);
[fneu, f] = vor_Filter(Asys,Bsys,Csys,r_vorFil);
```

Auch in dieser Datei werden wieder eine Unterdatei zum Berechnen des Zustandsraummodells für das System “Auto“ benötigt (siehe Listing A.4) und eine Unterdatei zum Berechnen “param poly“ (Zeile 13) berechnet die Parameter des charakteristischen Polynoms des geschlossenen Regelkreises und ist in Listing A.7 dargestellt. Die Funktion in Zeile 19 (“mat gesamtsys“) berechnet die Matrizen und Vektoren des Gesamtsystems (I-Anteil Regler und System “Auto“). Die Berechnungen in Zeile 22 bis 24 sind nicht relevant. Diese dienten nur zur Überprüfung der Formel in Zeile 26.

Listing A.6: Berechnung des Vorfilters

```
%Vorfilter bestimmen:
function[fneu, f] = vor_Filter(Asys,Bsys,Csys,r_vorFil)
    f = inv(Csys*inv(eye(2)-Asys+Bsys*r_vorFil')*Bsys);
    fneu = r_vorFil(1);
end
```

Listing A.7: Berechnung der Parameter des charakteristischen Polynoms

```
%Funktion um Pole in Parameter eines Polynoms umzuwandeln
function[a0, a1, a2] = param_poly(p1,p2,p3)
    a0 = -1*p1*p2*p3;
    a1 = p1*p2+p1*p3+p2*p3;
    a2 = -1*p1-1*p2-1*p3;
end
```

A.5.2 Simulinkmodell zur Simulation

In Abbildung A.3 ist das zur Simulation genutzte Simulinkmodell dargestellt. In diesem ist zu erkennen, dass noch weitere Elemente (nicht nur Regler und Strecke) eingebaut sind. Zum Beispiel eine Begrenzung des I-Gliedes. Dies war ein einfacher Versuch, um die Auswirkungen der Windup-Effekte gering zu halten. Der Auswirkungen waren jedoch nicht zufriedenstellend, da dadurch eine bleibende Regelabweichung entstanden. Auch das eingebaute Anti-Windup-Element, welches die Struktur aus Kapitel 9.3 aus [1] besitzt, erfüllt nicht die Erwartungen und sollte daher nicht verwendet werden. Ansonsten können zum Testen der stationären Genauigkeit Störungen auf die Lenkung geschaltet werden.

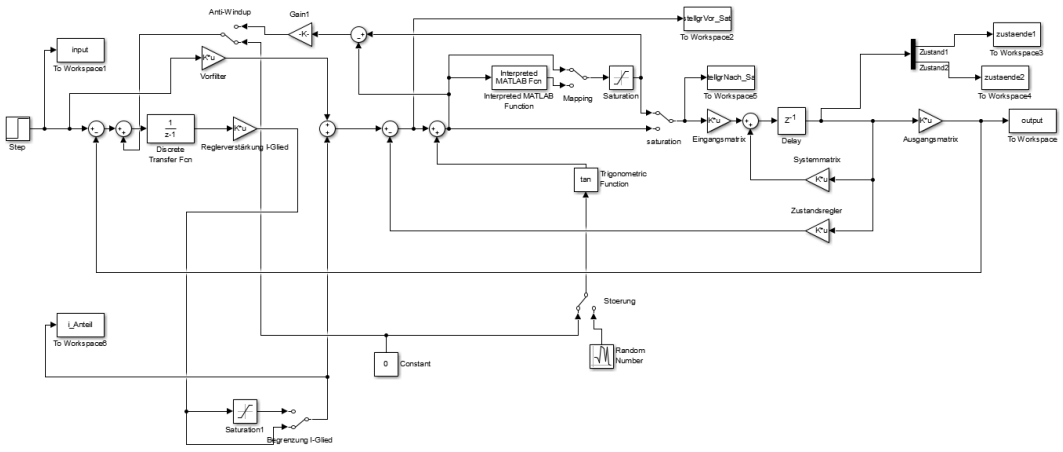

Abbildung A.3: Simulinkmodell zur Simulation

A.6 Diskrete Kaskadenregelung (3. Ansatz)

A.6.1 Matlabcode zur Auslegung der inneren Schleife

Die beiden Listings A.8 und A.9 stellen die zur Simulation und Berechnung der nötigen Parameter benötigten Funktionen bereit. Die benötigten Parameter für die Regelung werden in Listing A.9 berechnet. Dieser Code erzeugt auch die Wurzelortskurve. Das Starten der Simulation wird in Listing A.8 durchgeführt. Auch hier ist es bezüglich des Verständnisses geschickter eigene Dateien für die Simulation zu schreiben, jedoch kann der angegebene Matlabcode als grobe Orientierung verwendet werden.

Listing A.8: Starten der Simulation

```
% Simulation der Regelung fuer den Gierwinkel:
clear;
clc;

regelung_gierrate_correctMistake

% Simulation starten
%set_param('sim_regelung_gierwinkel','AlgebraicLoopSolver','LineSearch')
%set_param('sim_regelung_gierwinkel','AlgebraicLoopSolver','TrustRegion')
sim('sim_regelung_gierrate')
%sim('test')
%plotten_ergeb_test
plotten_ergeb_regelung_gierrate
```

Listing A.9: Berechnung der Reglerparameter

```
% Regelung der Gierrate:
clear;
clc;

% Parameter der Strecke:
T = 0.045; % Abtastzeit fuer aeussere Regelung
v = 0.6; % Wert zwischen 0.3 und 2 m/s
lH = 0.14;
l = 0.35;

% Anfangswert
x_0 = [0, 0]; % erster Wert fuer Abstand, zweiter Wert fuer Winkel

T_gier = 0.045; % Abtastzeit fuer Regelung der Gierrate
% Anmerkung zu T_gier: auch bei einem T_gier von 0.045 ist kaum
% Ueberschwingen vorhanden und das System ist sehr schnell eingeschwungen
% (fuer z_vzw = 0.3)

% Wahl des Verzweigungspunktes:
z_val = 0.15; % alter Wert: 0.3
```

```
% 2/3 ist neur Wert fuer Verschiebung des Referenzpunktes auf die
% Hinterachse!!!

% Aus Verzweigungspunkt berechnete Nullstelle der Uebertragungsfunktion
% G_0,G!!!
n_G = 0.1;

% Verstaerkungsparameter des PI-Reglers:
k_zaehler = abs(z_val-1);
k_nenner = abs(z_val-n_G);
k = k_zaehler/k_nenner;

k_G1 = k*1/(v*T);
k_G2 = k_G1*(1-n_G);

% Kontrolle der Pole des geschlossenen Regelkreises:
% Strecke:
z = [n_G];
p = [1];
g = k;
G_0G = zpk([z],[p],[g],T_gier);

% Feedback einfuegen:
G_wG = feedback(G_0G, 1);

% Pole des geschlossenen Regelkreises:
[p_feed] = pole(G_wG);

% WOK:
rlocus(G_0G)
grid on;
hold on;
plot(p_feed, 0, 'r*');

% Sprungantwort:
figure();
step(G_wG);
grid on;
title('Sprungantwort');
%axis([0 0.5 -0.5 1.05]);
```

A.6.2 Simulinkmodell zur Auslegung der inneren Schleife

In Abbildung A.4 zeigt das verwendete Simulinkmodell für diese Auslegung.

A.6.3 Matlabcode zur Auslegung der mittleren Schleife Schleife

Die beiden Listings A.10 und A.11 stellen die zur Simulation und Berechnung der Reglerparameter benötigten Funktionen bereit. Die benötigten Parameter für die Regelung werden in

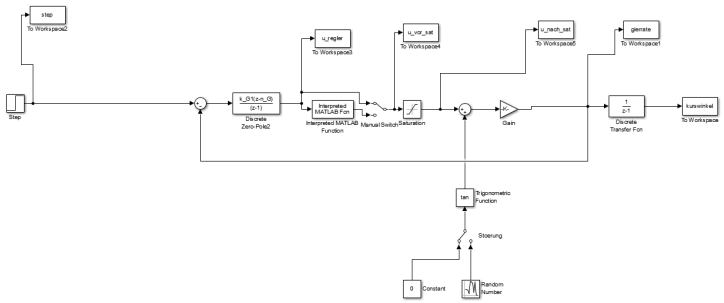

Abbildung A.4: Simulinkmodell zur Simulation

Listing A.11 berechnet. Dieser Code erzeugt auch die Wurzelortskurve. Das Starten der Simulation wird in Listing A.10 durchgeführt. Auch hier ist es für das Verständnis geschickter, eigene Dateien für die Simulation zu schreiben. Schon alleine die Benennung der vielen Variablen kann verwirrend sein, wenn man den Code nicht selbst geschrieben hat. Jedoch kann der angegebene Matlabcode als grobe Orientierung verwendet werden. Die Variable k_{G1} in Zeile 6 von Listing A.11 entspricht dem Verstärkungsfaktor $k_{R,innen}$. Ansonsten wird in Zeile 8 die Variable $p_{Gier,w}$ berechnet, welche für die Wurzelortskurve benötigt wird. Die Variable g in Zeile 13 entspricht der Verstärkung der Übertragungsfunktion des offenen mittleren Regelkreises. Die Übertragungsfunktion wird in Zeile 14 berechnet und anschließend die Wurzelortskurve gezeichnet. In Zeile 20 steht der abgelesene Verstärkungsfaktor aus der eben berechneten Wurzelortskurve. Für alle Geschwindigkeiten werden die Pole des geschlossenen Regelkreises bei dieser Verstärkung an der selben Stelle liegen. Dies hängt damit zusammen, dass die Pollage des inneren geschlossenen Regelkreises $p_{Gier,w}$ und die des I-Gliedes im mittleren Teilsystem unabhängig von der Geschwindigkeit sind. Nur die Verstärkungsfaktoren der Reglerparameter müssen für eine Änderung der Geschwindigkeit angepasst werden. Dies geschieht in Zeile 33 bis 35.

Listing A.10: Starten der Simulation

```
% Simulation der Regelung fuer die Gierrate und den Kurswinkel:
clear;
clc;

regelung_kurswinkel_correctMistake

% Simulation starten
sim('sim_regelung_gierrate_und_kurswinkel')

%plotten_ergeb_test
plotten_ergeb_regelung_kurswinkel
```

Listing A.11: Berechnung der Reglerparameter

```
% Regelung Kurswinkel
%regelung_gierrate
regelung_gierrate_correctMistake

k_k1_pred = 1;
```

```
p_gier_zaeh = 1+n_G*k_G1*v*T/l;
p_gier_nenn = 1+k_G1*v*T/l;
p_gier = p_gier_zaeh/p_gier_nenn;
z = [n_G];
p = [1, p_gier];
g_zaeh = k_k1_pred*k_G1*v*T/l;
g_nenn = 1+(k_G1*v*T/l);
g = g_zaeh/g_nenn;
G_0kurs = zpk([z],[p],[g],T_gier);

rlocus(G_0kurs)
grid on;

%Gain = 0.557
gain_abgele = 0.556;
G_0kurs_gain_abgele = zpk([z],[p],[gain_abgele],T_gier);
% Feedback einfuegen:
G_wkurs = feedback(G_0kurs_gain_abgele, 1);

[p_feed_kurswinkel_correct] = pole(G_wkurs);

% Sprungantwort:
figure();
step(G_wkurs);
grid on;
title('Sprungantwort');

k_k1_zahl = 1+k_G1*v*T/l;
k_k1_nenn = k_G1*v*T/l;
k_k1 = gain_abgele*k_k1_zahl/k_k1_nenn;


%f = f_zaehler/f_nenner;
f = 1;
```

A.6.4 Simulinkmodell zur Auslegung der mittleren Schleife

In Abbildung A.5 zeigt das verwendete Simulinkmodell für diese Auslegung.

A.7 PD-Regler (Wettbewerb)

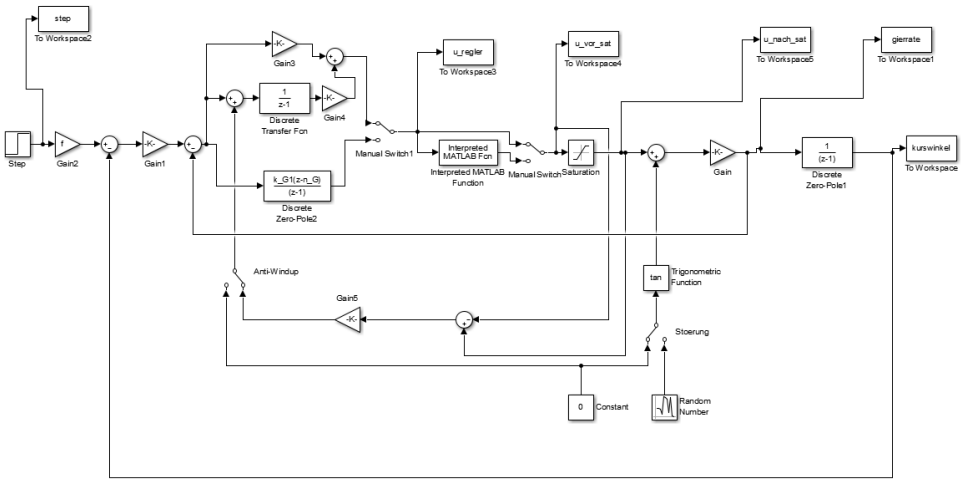

Abbildung A.5: Simulinkmodell zur Simulation

A.7.1 Umrechnung Regelalgorithmus von zeitdiskret in z-Bereich

Der Ansatz des PD-Reglers im zeitdiskreten soll hier in den z-Bereich transformiert werden. Hierzu können die Regeln aus Kapitel 4.2 von [1] verwendet werden. Die Umrechung ist im Folgenden dargestellt:

$$
\begin{aligned}
u_k &= K_P \cdot e + \frac{K_D}{T}(e_k - e_{k-1}) \\
&\vdots \\
U(z) &= k_P E(z) + \frac{k_D}{T}\frac{z-1}{z}E(z) \\
&= \left(k_P + \frac{k_D}{T}\frac{z-1}{z}\right)E(z) \ .
\end{aligned}
$$

Um die Übertragungsfunktion zu erhalten, muss der Quotient $\frac{U(z)}{E(z)}$ gebildet werden. Dadurch ergibt sich mit wenigen Umformungsschritten:

$$
\begin{aligned}
\frac{U(z)}{E(z)} &= \frac{Tk_Pz + k_D(z-1)}{zT} \\
&= \frac{z(Tk_P + k_D) - k_D}{zT} \\
&= \frac{1}{T(Tk_P + k_D)}\frac{z - \frac{k_D}{Tk_P + k_D}}{z} \\
&= k_R\frac{z - n_R}{z} \ .
\end{aligned}
$$

Literatur

[1] http://wiki.ros.org/depthimage_to_laserscan, Stand: 10.04.2017

[2] Dr.-Ing. Eric Lenz, Beschreibung Fahrzeug und ucboard, https://github.com/tud-pses/ucboard/blob/master/ucboard.pdf, Stand 26.11.2016

[3] Prof. Dr.-Ing. U. Konigorski, Digitale Regelungssysteme 1 und 2. TU Darmstadt: Regelungstechnik und Mechatronik, Sommersemester 2015.

[4] C. Voigt, J. Adamy, Formelsammlung der Matrizenrechnung. München: Oldenbourg Wissenschaftsverlag GmbH, 2007.

[5] Prof. Dr.-Ing. U. Konigorski, Mehrgrößenreglerentwurf im Zustandsraum. TU Darmstadt: Regelungstechnik und Mechatronik, Wintersemester 2015/2016.

[6] Prof. Dr.-Ing. U. Konigorski, Systemdynamik und Regelungstechnik 1. TU Darmstadt: Regelungstechnik und Mechatronik, Wintersemester 2012/2013.

[7] J. Adamy, Systemdynamik und Regelungstechnik 2. Aachen: Shaker Verlag, 2013.

[8] A. Ortseifen, Entwurf von modellbasierten Anti-Windup-Methoden für Systeme mit Stellbegrenzungen. TU Darmstadt: Regelungstheorie und Robotik, 2012.