Studienarbeit

Entwurf einer Fuzzy-Regelung eines Software-Fahrzeugmodells

Studiengang Elektrotechnik

Duale Hochschule Baden-Württemberg Mannheim

von

Hannes Reinholdt

Abgabedatum: 24.03.2017

Betreuer der Studienarbeit Prof. Dr.-Ing. Serge Zacher

Inhaltsverzeichnis

2.2 Fuzzy Regler

Der Entwurf eines Beispiels der Mixed Reality besteht aus einem Software-Modell realisiert mit MATLAB bzw. Simulink und einem Mikroprozessor-Board, dem STM32F4 Discovery, als Hardware-Regler. Resultierend soll ein Regelkreis aus Software und Hardware entstehen. Auf dem STM32F Discovery ist ein in Simulink entworfener Fuzzy-Regler zu implementiert. Dieser Fuzzy-Regler soll in beiden Modellen die Steuerung von Beschleunigung und Bremse regeln. Weitere Erläuterungen diesbezüglich sind im Kapitel 3 und dessen Unterkapitel nachzuschlagen.

Fuzzy-Regler sind der intelligenten Regelung zuzuordnen und als solche als System bzw. Element zu verstehen welches mittels mathematischen und logischen Algorithmen flexibel auf das Verhalten eines Regelkreises reagieren kann. Diese Flexibilität zeichnet sich durch eine Reaktion bei Fehlern oder einer Änderung der Parameter aus. Die intelligente Regelung wird des Weiteren in die Gruppen der modellbasierten Regelalgorithmen, zu denen der Fuzzy Regler zählt, und wissensbasierte Regelalgorithmen unterteilt. Modellbasierte Regler besitzen einen eigenen Mikroprozessor und Speicher. Resultierend sind diese Regler nicht mehr an PID-Algorithmen gebunden und es können komplexere Regelalgorithmen mit der eingangs erwähnten Flexibilität entwickelt werden. Dazu wird das Modell einer Regelstrecke in dem Regelalgorithmus integriert. (vgl. [5], S. 357) Bei einem Fuzzy-Regler wird der Regelalgorithmus durch linguistische Variablen geprägt. Dies bedeutet das Messgrößen nicht in Zahlen, sondern in Worten formuliert werden. Man spricht auch von unscharfer (im engl. fuzzy) Logik, da numerische Werte als unscharfe bzw. sich überschneidende Mengen festgelegt werden. Mittels Zugehörigkeitsfunktion werden Zusammenhänge zwischen linguistischen Variablen und numerischen Werten hergestellt. Dies wird auch als Fuzzifizierung bezeichnet. Die linguistischen Variablen werden mit den logischen Operatoren AND oder OR nach aus der Realität gewonnen Regeln verknüpft. Der Aufbau einer Regel entspricht dem „IF-THEN“ Schema. Die Gesamtheit aller Regeln wird als Regelbasis bezeichnet. Diese wird von der Inferenz ausgewertet und in eine linguistische Variable bzw. unscharfe Menge gewandelt und als Ergebnis ausgegeben. Abschließend muss das Ergebnis der Inferenz in eine scharfe Größe, sprich eine nummerische Variable für die nachfolgende Glieder des Regelkreises überführt werden. Dieser Schritt wird als Defuzzifizierung bezeichnet und kann nach unterschiedlichen Methoden durchgeführt werden. Die häufigste

verwendete Methode ist die Schwerpunktmethode nach der die Ausgangsvariablen gemeinsam als eine Fläche interpretiert werden und anschließend der Flächenschwerpunkt als Wert für die Stellgröße y gebildet wird. (vgl. [6], S. 22-31)

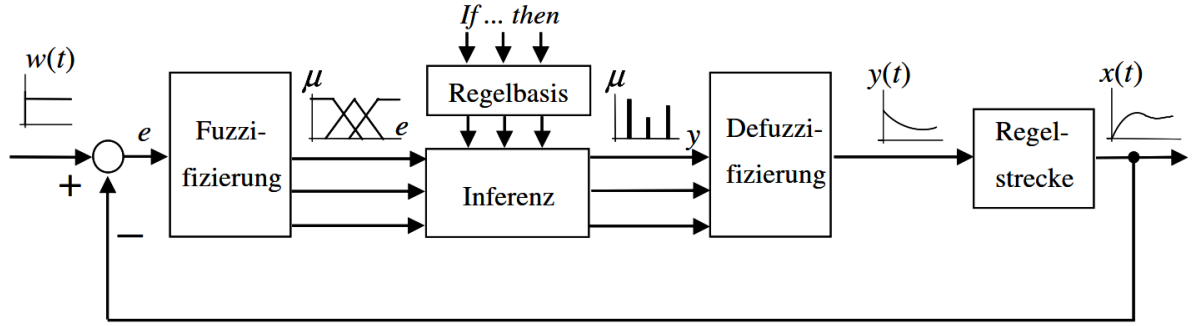

Abbildung 1: Wirkungsplan eines Regelkreises mit Fuzzy-Regler (*[5]*,S.371)

Auf tiefgreifender Erläuterungen wird an dieser Stelle verzichtet da das Thema des Fuzzy-Reglers bereits ausführlich in der Vorlesung Regelungstechnik 2 behandelt worden ist. Für weitere Informationen sind die im Quellenverzeichnis aufgeführten Quellen „Regelungstechnik für Ingenieure“ und das „Vorlesungsskript Regelungstechnik 2“ zu betrachten.

Die Vorteile eines Fuzzy-Reglers liegen im stabilen Verhalten bei nicht konstanten Parametern der Regelstrecke. Des Weiteren sind die Kosten und der Aufwand für die Entwicklung im Vergleich zu herkömmlichen Reglern niedriger. Einsatz finden Fuzzy-Regler insbesondere bei Strecken von denen man ein stabiles Verhalten bei sich ändernden Parametern erwarten wie „… z.B. Kraftfahrzeugen, Haushalts- und Medizingeräten ..“. (vgl. [5], S. 371). Jedoch müssen diese Parameter als Erfahrungswert aus einem realen Sachverhalt für die Fuzzy-Logik vorliegen. Eine willkürliche Änderung ohne Vergleich oder fehlenden Referenzwerten zu einem realen Ereignis führt den Fuzzy-Regler und seine Logik an die Grenze der erwarteten Funktionalität.

2.3 Unterschiede der verschiedenen „in-the-Loop“ Simulationen

Für eine Implementation des Fuzzy-Reglers auf das auf das STM32F4 Discovery sind verschiedene Entwicklungsschritte notwendig. Eine besondere Aufmerksamkeit gilt in der Entwicklungs- und Testphase des Modells den verschiedenen Simulationen. Die

Möglichkeit einer Simulation ist einer der Vorteile der modellbasierten Entwicklung. Entwürfe können jederzeit mittels Simulation in sich nicht verändernder Umgebung getestet werden. Somit können bereits im Entwicklungsprozess eines Produkts Rückschlüsse auf Leistung und Funktion sowie Änderungen dieser getätigt werden. Je nach Entwicklungstand werden die Simulation der Model-in-the-Loop, Software-in-the-Loop, Processor-in-the-Loop oder Hardware-in-the-Loop verwendet. Im Rahmen der Studienarbeit kommen MIL-, SIL- und PIL-Simulationen zum Einsatz. Die MIL-Simulation findet in der führen Entwurfsphase, in der Modell oder Regelkreis noch verändert werden, Verwendung. Der Regler, noch mittels Simulink realisiert, und das Softwaremodell der Regelstrecke werden in einer Schleife mit eingestellter zeitlicher Dauer simuliert. Aus der MIL-Simulation können erste Daten, Referenzwerte und Verhaltensmuster des Reglers und des Fahrzeugmodells sowie dessen zusammenwirken erschlossen werden. Diese Informationen können wiederum für Anpassungen und Optimierung verwendet werden und bilden die für die nächsten Entwicklungsschritte. (vgl. [7]) Die nächste Stufe der Entwicklung ist die SIL-Simulation. Dafür wird der Regler bzw. alle Komponenten des Reglers in ein Subsystem überführt. Dieses Subsystem wird in verwertbaren C-Code für eine später verwendete Hardware gewandelt. Der entstandene Code wird in der SIL-Simulation noch auf dem Entwicklungsrechner ausgeführt. Mittels erstellten SIL-Controller-Block wird der generierte Code mit dem Softwaremodell simuliert. Die Ergebnisse der MIL- und SIL-Simulation sollten möglichst identisch sein. Andernfalls muss die Ursache der Abweichung mittels Analyse ergründet und ggf. beseitigt werden. Der Vorteil in der SIL-Simulation besteht zum einen darin das sich noch auf keine Hardware zur Testdurchführung festgelegt werden muss und das die Kosten für einen aussagekräftigen Test aufgrund der fehlenden Hardwarenutzung niedrig sind. (vgl. [8]) Im nächsten Entwicklungsschritt wird der vom Entwicklungsrechner generierte C-Code mittels PIL-Simulation kompiliert und auf einen externen Prozessor implementiert. Der als Regler programmierte Prozessor wird dann als PIL-Block, vergleichbar wie beim Vorgehen mit dem SIL-Block, in das Software Modell integriert und zusammen mit diesem simuliert. Ein realer Prozessor ist nun in eine virtuelle Umgebung eingebettet und wird mit Werten des Softwaremodells gespeist. Im Gegensatz zu realen Messsignalen sind dies konstant und ermöglichen einen identischen Versuchsaufbau zum Testen des Prozessors. Der Prozessor verarbeitet die eingehenden Werte und beeinflusst das Software-Modell durch die daraus resultierenden Steuersignale. Durch

Abweichungen der Werte von PIL-Simulation mit den Werten der SIL- oder MIL-Simulation können Softwarefehler und Hardwareprobleme bei der Benutzung des Prozessors im Anwendungsgebiet ermittelt werden. (vgl. [9]) Als erste Entwicklungsstufe mit Komponenten der realen und der virtuellen Welt in einem Regelkreis kann die PIL-Simulation als Mixed Reality bezeichnet werden. Die nächstfolgende Stufe der HIL-Simulation findet keine Verwendung für die Bearbeitung der Studienarbeit. Aus Gründen der Vollständigkeit und im Kontext der Industrie 4.0 und der Mixed Reality wird die HIL-Simulation kurz und prägnant erläutert. Der im PIL verwendete Prozessor wird nun durch ein komplettes Hardwaresystem, zum Beispiel ein Steuergerät, ersetzt. Die Hardware wird über Ein- und Ausgänge mit einem Softwaremodell einer virtuell nachempfundenen Realität verbunden und mit den Werten sowie Eigenschaften der virtuellen Welt simuliert. Dabei besteht das Softwaremodell aus eine möglichst genaue Nachbildung der realen Situation. Dazu sind ermittelte Erfahrungs- bzw. Referenzwerte der realen Umgebung unabdingbar. Der Vorteil besteht nun im Erhalt einer konstanten bzw. einer idealen Versuchs- und Testumgebung. Da diese virtueller Natur ist entfallen kostspielige Versuchsaufbauten und mögliche Beschädigungen des zu testenden Produkts. Des Weiteren sind zeitlich verkürzte Simulationen realisierbar und bringen enorme Zeitersparnisse. (vgl. [8])

Der Simulationen mittels In-the-Loop Anwendungen sind jedoch auch Grenzen gesetzt. Zum einen kann die virtuell abgebildete Realität nur einen begrenzten Teil der realen Welt abbilden. Dieser Ausschnitt wird aus den Messwerten und Erfahrungswerten der realen Welt begrenzt. Trotz neuster Technik ist es bis heute nicht möglich die Gesamtheit eines realen Szenarios oder einer realen Welt komplett mit allen möglichen Ereignissen in ein Softwaremodell zu überführen. Gründe dafür sind ein begrenzter Speicher und begrenzte Leistung der für die Simulation notwendigen Hardware. Resultierend können Messwerte und Erfahrungswert der realen Welt nur als vereinfachte mathematische Gleichungen in ein virtuelles Softwaremodell überführt werden. Eine weitere Grenze zeigt sich in Funktionen bei denen der Mensch bzw. Kunde mit dem Modell oder Regelkreis interagiert. Daraus können nicht vorhersehbare Auswirkung entstehen die nicht von dem Softwaremodell erfasst und bearbeitete werden können.

3. Entwurf des Software Modells mit Fuzzy-Reglers

Das folgende Kapitel schildert die anfänglichen Entwürfe von beiden zur Verfügung stehenden Modellen bis hin zur Auswahl des optimierten Regelkreises für die weiterführende Bearbeitung. Dazu werden Änderungen an den ursprünglichen Regelkreisen sowie Einstellungen am Fuzzy-Regler und am Fahrzeugmodell als Regelstrecke erklärend dokumentiert und begründet. Auf die Vorstellung und Verwendung eigener Entwürfe wird in der schriftlichen Ausführung der Studienarbeit verzichtet. Gründe hierfür sind die im Vergleich zu den bereitgestellten Modellen auftretenden Defizite in Funktion und Aufbau sowie das Fehlen von eigener Erfahrung und selbst ermittelten Referenzwerten einer realen Umgebung für den Aufbau solcher Regelkreise. Eine ausführliche Erläuterung für die Notwendigkeit der hier aufgezählten Gründe wurde im Kapitel 2 und dessen Unterkapitel 2.1 und 2.3 erwähnt. Am Ende des dritten Kapitels wird die Wahl des Modells für die weiteren Simulationsschritte begründet.

3.1 Regelkreis Hindernis

Der Regelkreis bzw. das Modell Hindernis setzt sich zusammen aus der „Hindernis_Fuzzy.slx“ und der dazugehöroger „Hindernis.fis“. Die slx-Datei beinhaltet den Blockentwurf des kompletten Regelkreises und die fis-Datei beinhaltet die für den Fuzzy-Block notwendige Fuzzifizierung, Regelbasis, Inferenz und Defuzzifzierung.

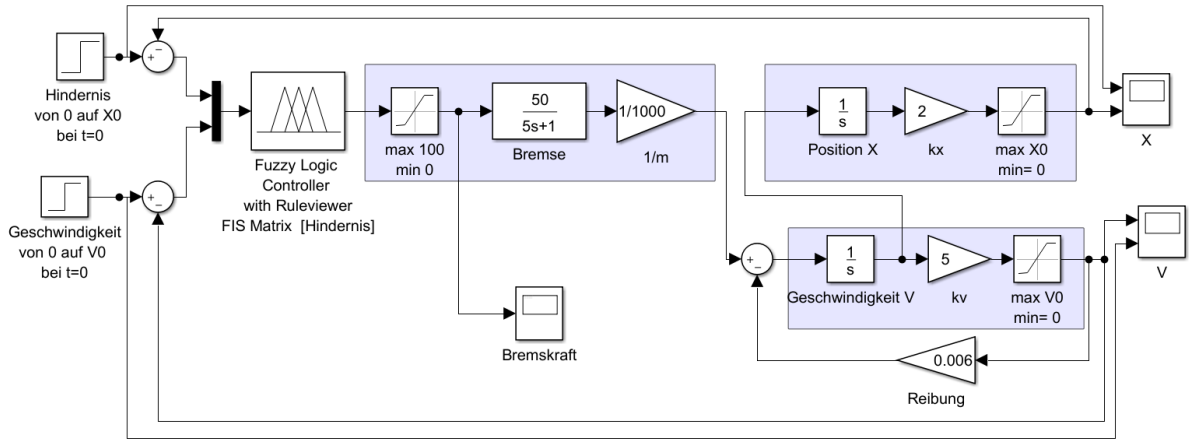

Abbildung 2: unverändertes Hindernis Modell der bereitgestellten „Hindernis_Fuzzy.slx“ Datei, dargestellt in Simulink

Der Regelkreis wird nun beginnend von links nach rechts blockweise erläutert. Aus Gründen der Verständlichkeit werden die Bezeichnungen der Schaltblöcke in der verwendeten Sprache des Programms, Englisch, bezeichnet. Die ersten beiden Schaltblöcke sind Step-Blöcke und als solche den Sources-Blöcken zugehörig. Sie stellen die Führungsgrößen des Regelkreises als Sprungfunktion da. Über das „Command Window“ von MATLAB werden der Abstand zum Hindernis als X0 und die Anfangsgeschwindigkeit des Fahrzeugs als V0 definiert und bilden die jeweilige Maximum der Step-Blöcke. Die Führungsgrößen führen in jeweils einen subtrahierenden Sum-Block. Dieser ermittelt die Regeldifferenz aus der Führungsgröße und der aus der Regelstrecke resultierenden Regelgröße für die Geschwindigkeit und den Abstand zum Hindernis. Von dort werden die beiden Regeldifferenzen über einen Mux-Block zu einem zweidimensionalen Vektor gewandelt. Da der nachfolgende „Fuzzy Logic Controller“-Block nur über einen Eingang verfügt ist die Verwendung des Mux-Blocks unumgänglich. Der in diesem Regelkreis verwendete FLC-Block ist ein „FLC with Ruleviewer-Block“. Das bedeutet das während einer Simulation das „Rule Viewer“-Fenster geöffnet wird und die Prozesse bzw. Arbeitsweise des Fuzzy-Reglers bildlich dargestellt werden.

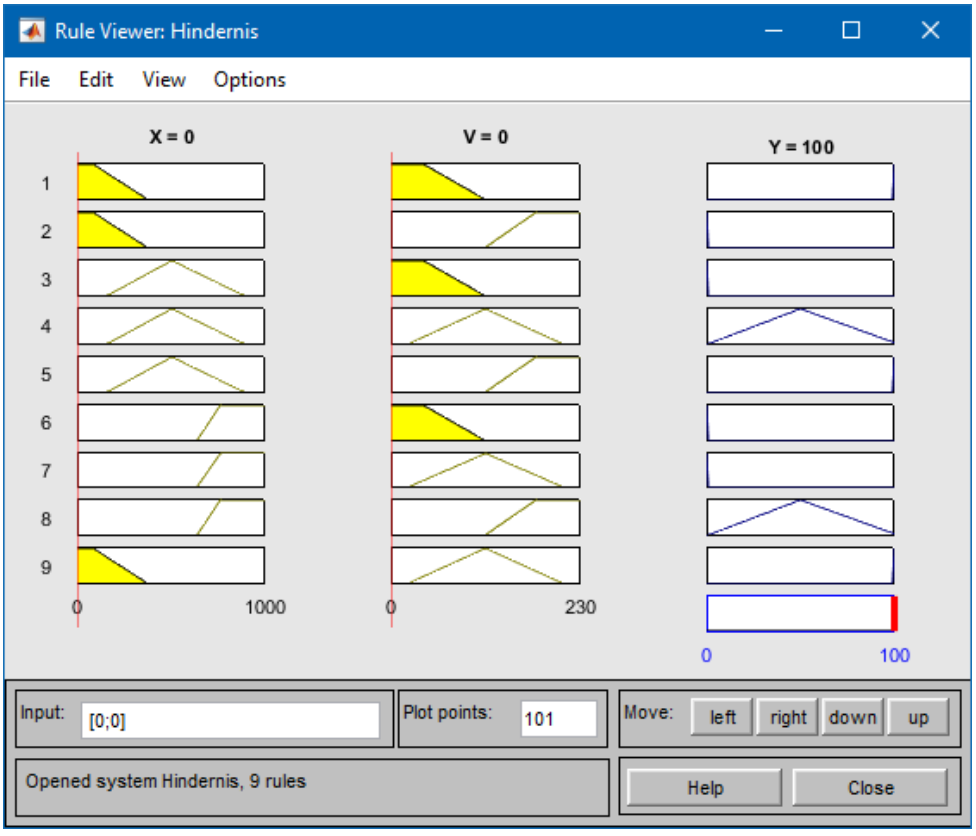

Abbildung 3: Rule Viewer während der MIL Simulation des Regelkreises Hindernis

Für eine ordnungsgemäße Funktion des FLC-Blocks muss die „Hindernis.fis“ Datei in den Regelkreis implementiert werden. Dazu ist im „Command Window“ der Befehl „fuzzy“ einzugeben. Es öffnet sich der „Fuzzy Logic Designer“. Dort sind die Spezifikationen des Fuzzy-Regler einzustellen oder bereits erstellte Spezifikation vom „Workspace“ oder einer „File“ zu importiert. Zunächst wird die bereitgestellte „Hindernis.fis“ über die Reiter „File -> Import -> From File“ in den „Fuzzy Logic Designer“ importiert.

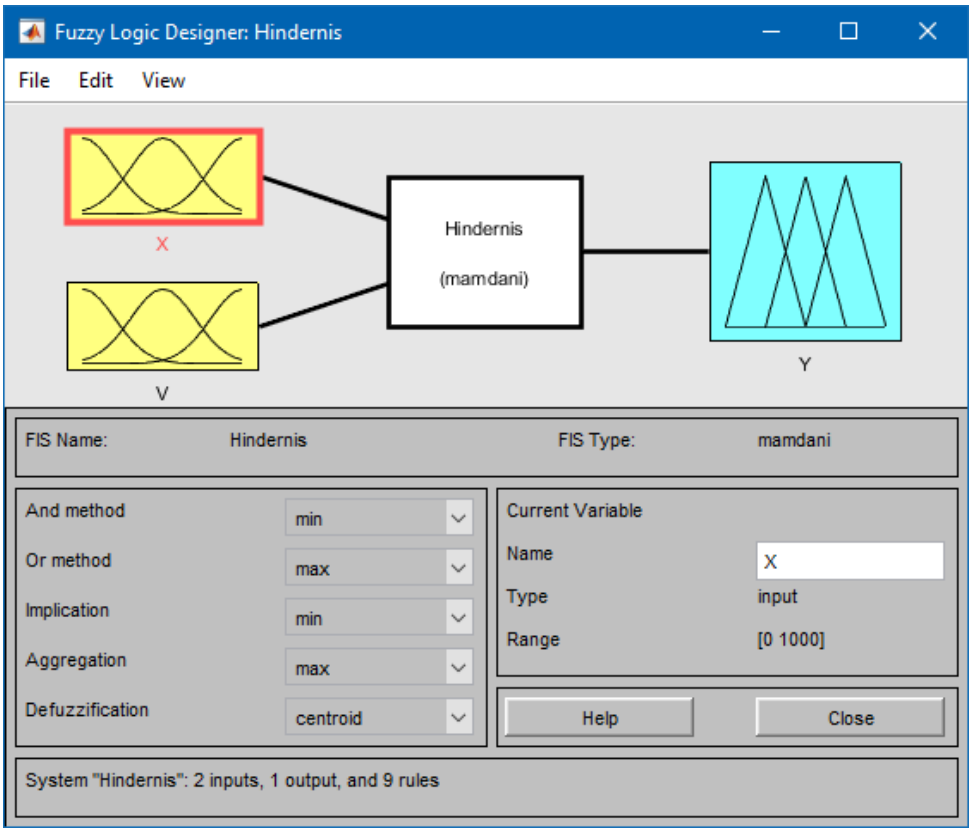

Abbildung 4: Fuzzy Logic Designer mit importierter "Hindernis.fis"

Über einen Doppelklick auf die gelb unterlegten Eingangsgrößen oder die blau unterlegten Ausgangsgrößen, im Bild 4 mit X, V, Y benannt, gelangt man in den „Membership Function Editor“. In diesem Editor können die Zugehörigkeitsfunktionen zwischen numerischen und linguistischen Variablen für Ein- und Ausgänge festgelegt werden. Mit einem weiteren Doppelklick auf das weiß unterlegte Rechteck wird der „Rule Editor“ geöffnet. Hier werden die Zugehörigkeitsfunktionen mit logischen Operatoren zu einer Regelbasis verknüpft. Weitere Hinweise zur Justierung und Einstellung erfolgen später zu den Erläuterungen der Optimierung. Nachdem die „Hindernis.fiz“ in den „Fuzzy Logic Designer“ geladen worden ist werden über die Reiter „File -> Export -> To Workspace“ die Spezifikation des Fuzzy-Reglers als

Workspace-Variable gespeichert. Nun erhält das Simulink-Modell des Hindernisses Zugriff auf die eingestellten Parameter des Fuzzy-Regler Entwurfs. Anschließend wird die Workspace-Variable über das „Block Parameters“ Fenster des FLC-Blocks im Unterpunkt „FIS matrix“ eingebunden. Hierbei ist zu beachten das die Bezeichnung der Workspace-Variable mit dem eingegebenen Namen übereinstimmt.

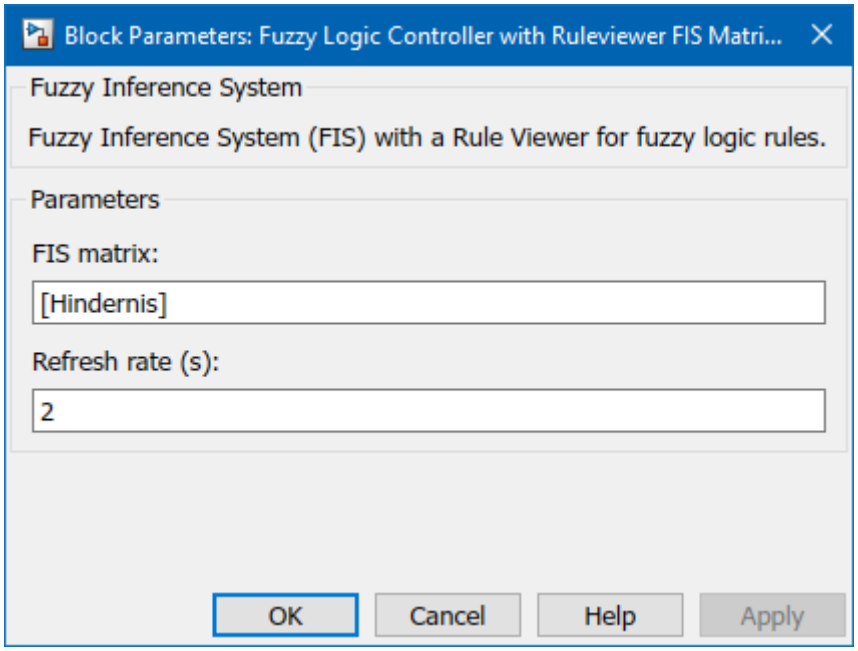

Abbildung 5: Block Parameter Fenster mit eingebundener Hindernis Workspace-Variable

Der FLC-Block berechnet nun auf Grundlage der fuzzifizierten Eingangsgrößen X0 und V0, sowie der auf Grundlage der eingestellten Regelbasis und Inferenz die defuzzifizierte Ausgangsgröße Y, die Bremskraft. Die Ausgangsgröße Y wird im Anschluss von einem Saturation-Block auf minimal 0% und maximal 100% begrenzt. Aufgrund der eingestellten Ausgangsgrößen des Fuzzy-Reglers welche von 0% bis 100% reichen ist diese Begrenzung nicht notwendig, da sie keinerlei Auswirkung hat. Im Anschluss liegt die Ausgangsgröße Y an einem „Transfer Function“ Block an. Dieser ist als PT1-Glied gestaltet und wandelt die prozentuale Bremskraft in die physikalische Bremskraft in Newton um. Da für den Regelkreis und insbesondere für die Ermittlung der Regeldifferenzen die Geschwindigkeit und die Strecke bzw. Position benötigt werden, wird die Bremskraft mit einem Gain-Block mit den Wert $\frac{1}{1000}$ multipliziert. Die Grundlage liefern die Newtonschen Gesetzte und die daraus resultierende Gleichung: $Beschleunigung = \frac{Bremskraft}{Masse}$. Das Ergebnis ist die Beschleunigung des Fahrzeugs die während des Bremsvorgangs auftritt. Der folgende Sum-

Block imitiert durch die Rückführung der Geschwindigkeit und durch die Multiplikation mittels Gain-Blocks die in der Realität auftretende Reibung. Um die Geschwindigkeit aus der Beschleunigung zu ermitteln wird ein Integrator-Block verwendet. Die Integration der Beschleunigung liefert die Geschwindigkeit nach der Gleichung $v = \int a(t)\, dt$. Gefolgt von einem weiteren Gain-Block, der die Simulation durch Multiplikation mit 5 beschleunigt und einem Saturation-Block welcher die Geschwindigkeit auf realistische Werte von 0 $\frac{km}{h}$ bis 230 $\frac{km}{h}$ begrenzt. Ein weiterer Pfad führt nach der Geschwindigkeitsintegration zur Wegintegration. Der Blockaufbau zur Ermittlung des Weges bzw. der Position erfolgt sinngemäß wie die Ermittlung der Geschwindigkeit. Einzige Unterschiede sind im Gain-Block der den Wert 2 besitzt und im Saturation-Block mit der Begrenzung von 0m bis 1000m. Im Anschluss an die grau gekennzeichneten Positions- und Geschwindigkeitsarealen werden die beiden Regelgrößen Geschwindigkeit und Position zurück zu den anfangs erwähnten Sum-Blöken zur Regeldifferenz Bildung geführt. Die Ergebnisse werden mithilfe von drei Scope-Blöcken geplottet und analysiert.

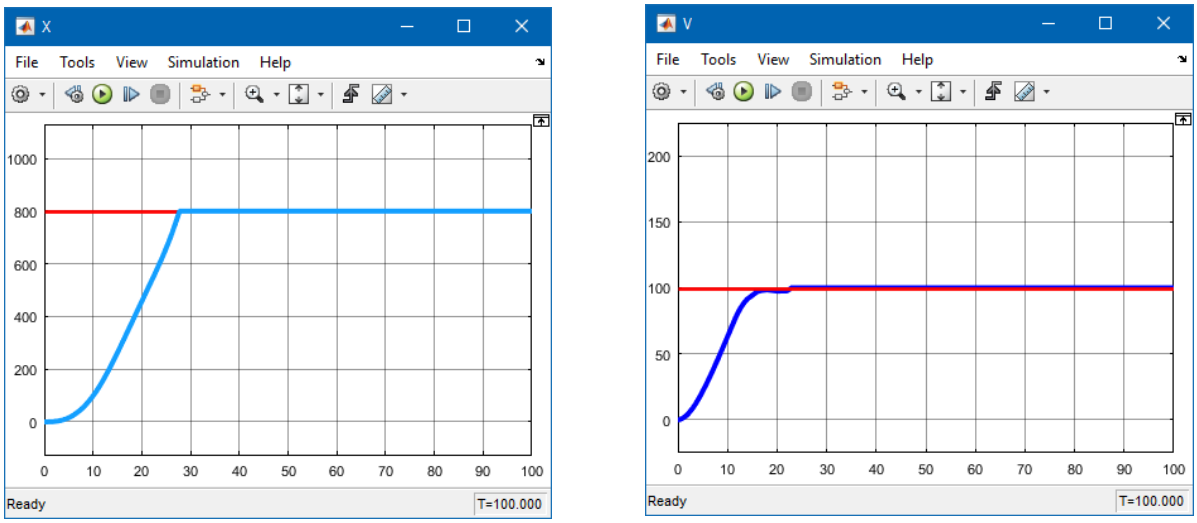

Abbildung 6: Scope-Plots der simulierten Werte X und V für X0=800 m und V0=100 km/h

Mittels plotten verschiedener Werte für X0 und V0 kann der Regelkreis bzw. das Modell und seine Funktion analysiert werden. Zur Analyse werden neun Versuchsreihen mit jeweils niedrigen, mittleren und hohen V0 und X0 durchgeführt. Dabei treten folgende Fehler auf. Bei mittleren und weiten Abständen zum Hindernis in Kombination mit geringer Geschwindigkeit ist die ausgegebene Bremskraft immer 0%. Es kommt folglich zu keiner Geschwindigkeitsveränderung. Bei hoher und mittlerer Geschwindigkeit und geringen Abstand beträgt das Bremsverhalten bei Kollision

ebenfalls nur 100%. Das Bremsverhalten einer Kollision sollte sich jedoch deutlich von der maximalen Bremskraft des Fahrzeugs unterscheiden. Ein drittes Fehlverhalten besteht in einer Positionsänderung trotz einer Geschwindigkeit von $0\frac{km}{h}$. Dieses Verhalten tritt bei mittleren Geschwindigkeiten sowie mittleren und weiten Abständen und auch bei niedrigen Abständen und niedrigen Geschwindigkeiten ein. Das Fahrzeug kommt erst einige Sekunden später zum Stillstand trotz nicht vorhandener Geschwindigkeit. Als Lösungsversuch wird der Kontenpunkt hinter der Geschwindigkeitsintegration verschoben. Da die Positionsintegration direkt nach der Geschwindigkeitsintegration erfolgt wirken der Gain- und Saturation-Block nicht für die Positionsintegration. Das bedeutet das der Weg aus einer anderen Geschwindigkeit integriert wird. Die gedachte Lösung des Problems besteht in der Verschiebung des Kontenpunkts zur Positionsintegration hinter den Geschwindigkeit Saturation-Block. Durch die Umstellung resultiert jedoch ein abruptes Bremsverhalten und das eigentliche Problem der Positionsänderung verschlimmert sich so, dass die Position noch länger bis zum Stillstand benötigt. Der Lösungsversuch wird somit verworfen. Das zweite Problem wird mittels einer Optimierung des Fuzzy-Reglers bzw. der „Hindernis.fiz“ gelöst. Dazu wird eine neue Zugehörigkeitsfunktion im „Membership Function Editor“ zur Eingangsvariable X und Ausgangsvariable Y hinzugefügt.

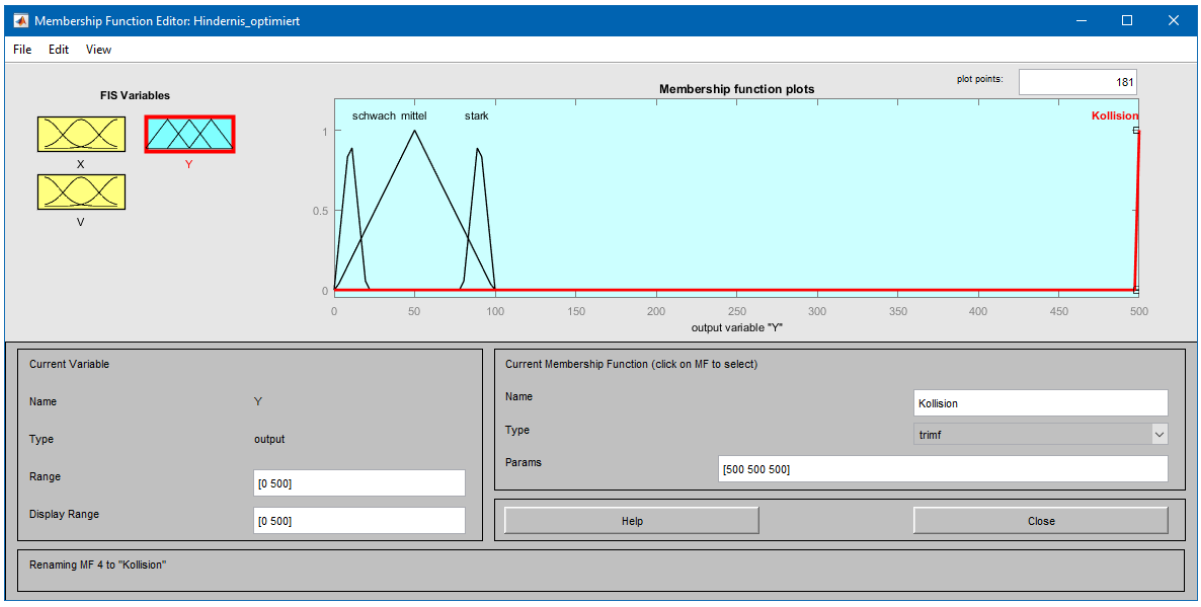

Abbildung 7: Anpassen der Zugehörigkeitsfunktionen via Membership Funktion Editor der "Hindernis_optimiert.fis" zur beseitigung der Fehler

Zuerst wird die Zugehörigkeitsfunktion „nahe“ auf die Parameter [1 1 85.5 369] ergänzt. Dann wird über die Reiter „Edit -> Add Custom MF“ eine neue trigonometrische

Zugehörigkeitsfunktion mit den Namen „Null“ und den Parametern [-500 0 0] erstellt. Das selbe verfahren wird für die Ausgangsgröße Y angewendet. Vorher wird jedoch die „Range“ mit den Parametern [0 500] versehen. Die neue Zugehörigkeitsfunktion erhält den Namen „Kollision“ und die Parameter [500 500 500]. Zusätzlich werden die Zugehörigkeitsfunktionen „schwach“ und „stark“ angepasst. Sie stellen nun Dreiecke mit den Parametern [0 10 20] und [80 90 100]. Diese bewirkt eine feinere Abstufung der Bremskraft. Des Weiteren wird durch die Anpassung der Zugehörigkeitsfunktion „schwach“ der Fehler der 0% Bremskraft bei zu niedriger Geschwindigkeit eliminiert. Abschließend müssen die neuerstellten Zugehörigkeitsfunktionen im „Rule Editor“ nach Abbildung 8 miteinander verknüpft werden.

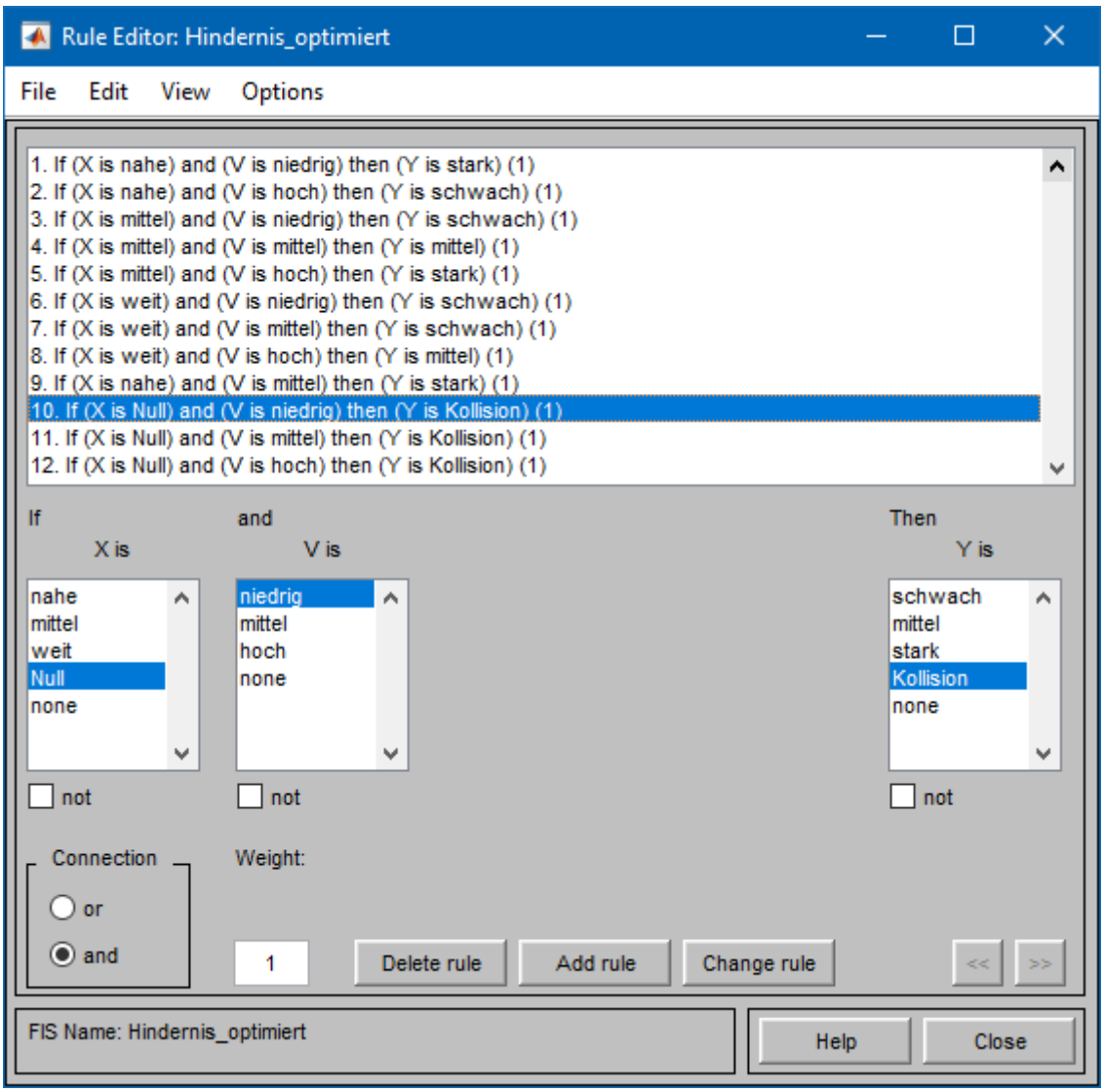

Abbildung 8: Die neuerstellten Regeln 10 bis 12 der optimierte "Hindernis.fiz" im Rule Editor

Im Anschluss werden die neuen Fuzzy-Regler Parameter als Workspace-Variable exportiert. Die Bremskraft von 500% wird nur im Falle einer Kollision ausgegeben. Als mögliche Optimierung wäre eine weitere Unterteilung der Kollisionsbremskräfte entsprechend den drei unterschiedlichen Geschwindigkeitsgruppen zu betrachten. Auch wenn dies eher dem realen Verhalten einer Kollision entspricht wird aus Gründen der Übersichtlichkeit auf diese Maßnahme für die Studienarbeit verzichtet. Damit sich die Bremskraft von 500% entfalten kann muss der Saturation-Block zur Bremskraftbegrenzung entfernt werden. Wie bereits erwähnt ist diese Begrenzung aufgrund der Funktionsweise des FLC-Blocks ohnehin überflüssig.

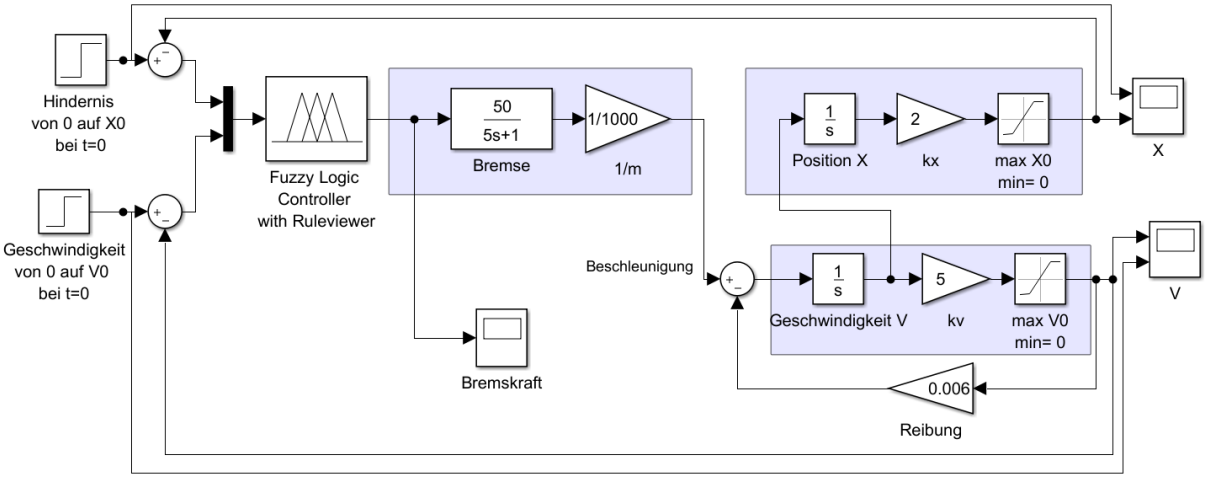

Abbildung 9: Optimierter und angepasster Regelkreis der "Hindernis.slx" Datei

Die durchgeführten Änderungen der Regelstrecke sind in der Abbildung 9 zu sehen. Signifikante Verbesserungen im Vergleich zu den Plots der originalen Dateien aus Abbildung 6 sind das verbesserte Bremsverhalten sowie das Bremsen bei niedriger Geschwindigkeit und weitem oder mittleren Abstand. Das Bremsverhalten wurde in zwei Aspekten verbessert. Zum einem ist der Geschwindigkeitsabfall bei nicht erreichen des Hindernisses und vorzeitigen Stillstand des Fahrzeugs wesentlich geringer was in der Realität den Fahrkomfort für Insassen des Fahrzeugs verbessern würde. Zum anderen wird auch bei niedrigen Geschwindigkeiten und weiten oder mittleren Abständen eine Bremskraft generiert. Diese ist mit 10% relativ hoch dimensioniert und sollte bei einer weiterführenden Studienarbeit angepasst werden. Trotz Optimierung und Anpassungen konnte das Problem der weiterlaufenden Positionsanzeige nur geringfügig verbessert werden. Das Problem besteht weiterhin.

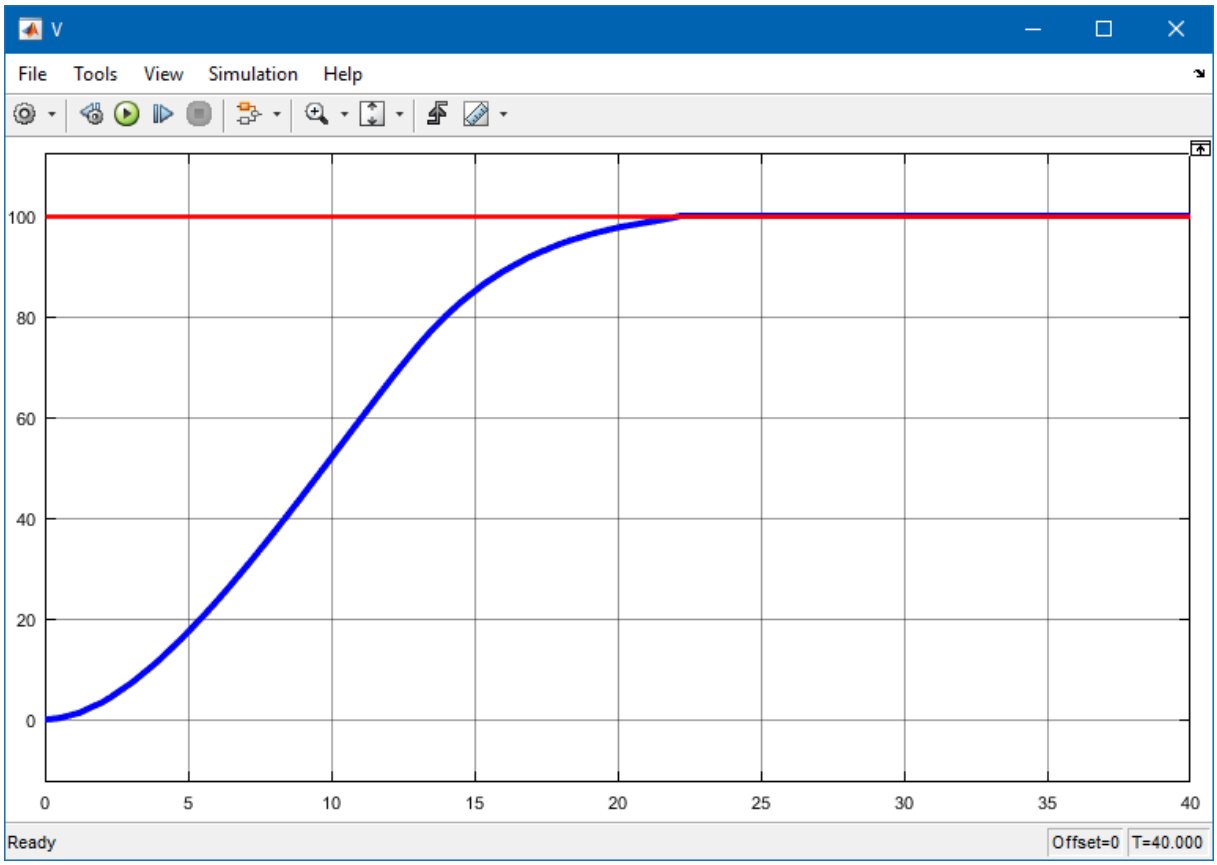

Abbildung 10: Scope-Plot des optimierten Regelkreises für den Wert V für X0=800 m und V0=100 km/h

Alle Dateien, Optimierungen und Anpassungen sind im Anhangsverzeichnis der beigelegten DVD unter dem Pfad MATLAB/Hindernis mit der Kennzeichnung „opt“ zu finden.

3.2 Regelkreis Abstand

Bereits im Kapitel 3.1 erklärte Schaltblöcke und Funktionen werden bei diesem Modell nicht noch einmal erläutert. Der Regelkreis bzw. das Modell Hindernis setzt sich zusammen aus der „Abstand_Fuzzy.slx“ und der dazugehöriger „Abstand.fis“. Anders als beim zuvor beschriebenen Modell sind in diesem Simulink-Modell zwei Fahrzeuge und dessen Verhalten zu simulieren. Die Fahrzeuge sind in ein vorausfahrendes und ein hinterherfahrendes zu unterteilen. Das sogenannte Folgeauto soll dem Vorderauto in einem eingestellten Abstand von 300m folgen. Dieser Abstand soll mittels Gas- und Bremspedalbetätigung erreicht und gehalten werden. Bei einem zu großen Abstand soll das Folgeauto beschleunigen und bei einem zu kleinen Abstand soll es bremsen. Zunächst wird der Regelkreis nach bekannten Muster beginnend von links nach rechts

beschrieben. Der Sollabstand Block gilt als Führungsgröße und bildet über eine Subtraktion mit dem Abstand zum Vorderauto die Regeldifferenz.

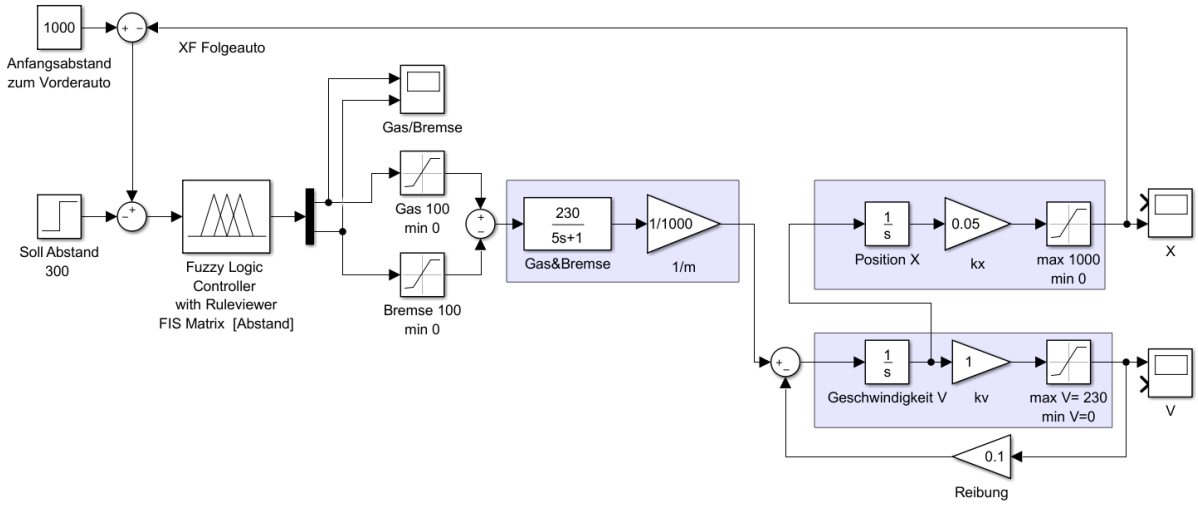

Abbildung 11: unverändertes Abstand Modell der bereitgestellten „Abstand_Fuzzy.slx“ Datei, dargestellt in Simulink

Die Regeldifferenz des Abstands ist die einzige Eingangsgröße des FLC-Blocks. Als Ausgangsgrößen sind Gas, die Gaspedalbetätigung in Prozent, und Bremse, die Bremspedalbetätigung in Prozent, gewählt. Diese werden als zweidimensionaler Vektor an einen Demut-Block, welcher einen zweidimensionalen Vektor in zwei eindimensionale Vektoren zerlegt, ausgegeben. Im Anschluss werden jeweils beide Ausgangsgrößen über jeweils einen Saturation-Block auf 0% bis 100% begrenzt. Mittels subtrahierenden Sum-Block wird das betätigen des Gaspedals als positiver Wert und das betätigen des Bremspedals als negativer Wert ausgelegt. Dies wird benötigt um die resultierende Kraft eine positive bzw. negative Richtung zu verleihen. Die Umwandlung des prozentualen Werts in die Kraft erfolgt durch das im Anschluss folgende PT1-Glied. Anschließend folgt ein sinngemäß identischer Aufbau wie im Regelkreis Hindernis. Ausnahmen sind die unterschiedlichen Parameter und eine einzige Rückführung. Eine Änderung der gegebene Gain-Block Parameter wird während der Bearbeitung nicht vorgenommen. Grund ist die fehlende Erfahrung der Auswirkungen auf das nachgebildete Fahrzeugmodell in Hinblick auf ein der Realität nachempfundenes Verhalten. Abschließend wird die zurückgeführte Position vom Anfangsabstand des Vorderautos abgezogen und bildet den resultierenden Abstand zum Vorderauto. Der Anfangsbestand wird mittels Constant-Block als konstanter Wert implementiert. Auch in diesem Regelkreis sind drei Scope-Blöcke enthalten. Bevor ein

erster Vergleichsplot simuliert wird, ist der Regelkreis anzupassen, da er nach aktuellem Entwurf bereits Probleme aufzeigt. Das erste Problem besteht in dem Verhalten des Vorderautos. Da dieses einem konstanten Wert zugrunde liegt ist es nur ein Hindernis welches in einer vorgegebenen Entfernung steht. Das Verhalten des Vorderautos ist durch ein auf Bewegung basierendes Verhalten zu ersetzen. Dazu wird der Constant-Block mit der Bezeichnung „Anfangsabstand zum Vorderauto“ entfernt und durch den in Abbildung 12 dargestellten Entwurf ersetzt.

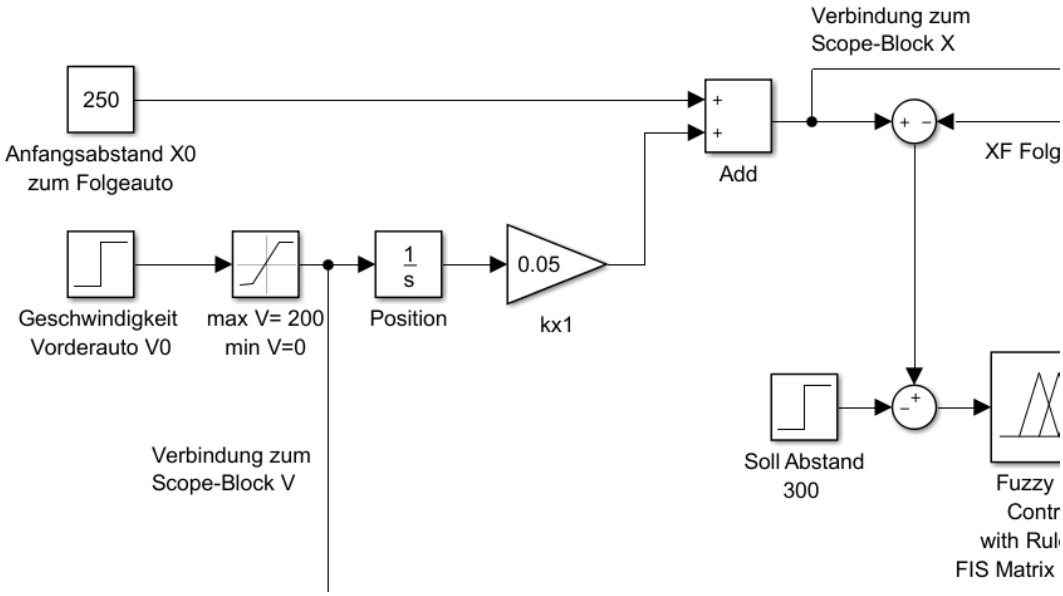

Abbildung 12: Fahrzeugmodell des Vorderautos

Dieser Entwurf beinhaltet zum einen die über die Step-Block Parameter einstellbare Geschwindigkeit, eine Geschwindigkeitsbegrenzung des Vorderautos auf $200\frac{km}{h}$ sowie einen über den Constant-Block einstellbaren Anfangsabstand zum Folgeauto. Des Weiteren können nun der Scope-Block X und V sinnvoll mit dem Vorderauto verbunden werden. Das Scope X vergleicht den zurückgelegten Weg des Folgeautos mit dem des Vorderautos und das Scope V vergleicht die Geschwindigkeiten beider Fahrzeugmodelle. Aus Gründen der Übersichtlichkeit wird ein Subsystem mit den Eingängen X0 und V0 und den Ausgängen Geschwindigkeit und zurückgelegte Strecke des Vorderautos erstellt. Nun kann die „Abstand.fiz“ nach bekannten Vorgehen implementiert werden. Für die erste Simulation wird ein X0 von $800m$ und V0 von $80\frac{km}{h}$ gewählt. Während der ersten Simulation kann es dazu kommen, dass die diese unvollendet bei stehen bleibt. Um dies zu vermeiden sind Einstellungen über das „Model Configuration Parameters“ Menü, erreichbar mittels Tastenkombination

Strg+E, zu tätigen. Unter dem Reiter Solver muss bei dem Punkt „Solver options“ der Type „Fixes-step“ eingestellt werden. Die „Fixed-step size“ wird auf den Wert 0.1 festgelegt.

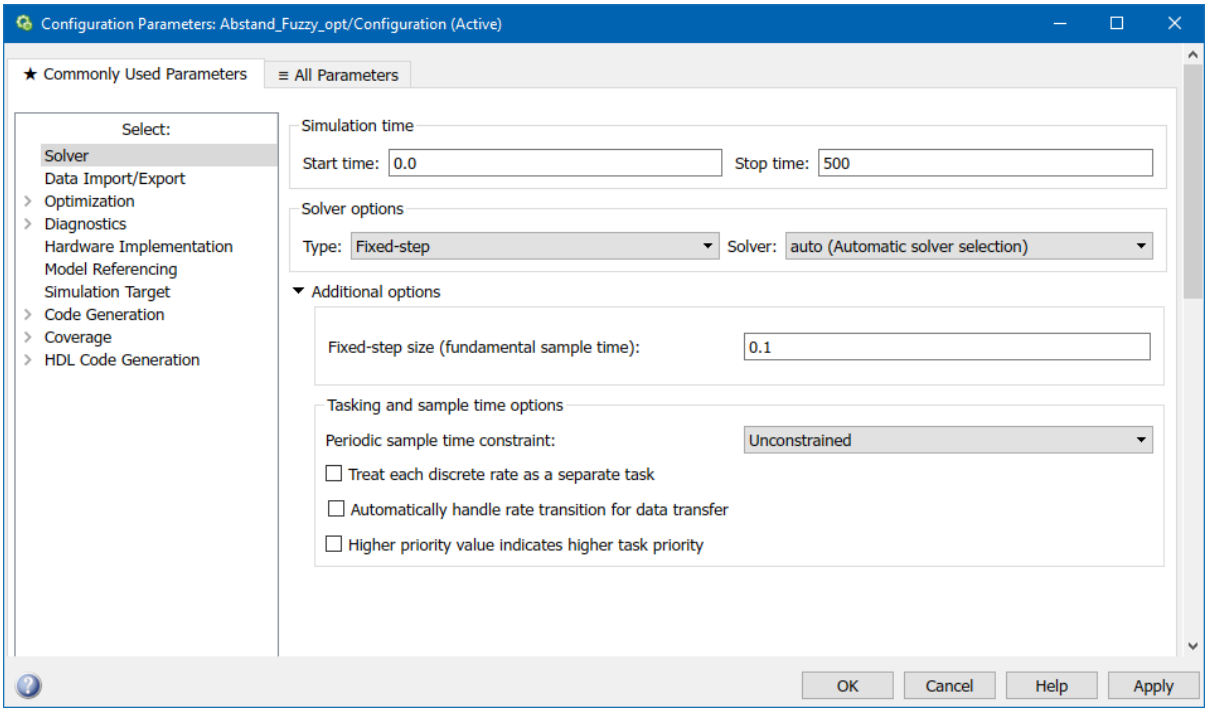

Abbildung 13: Configuration Parameters Fenster mit den empfohlenen Einstellungen des Solvers

Ein weiteres Hindernis einer erfolgreichen Simulation ist die Begrenzung der Position mittels Saturation-Block. Da nun beide Fahrzeugmodelle kontinuierlich Strecke zurücklegen verhindert die Begrenzung eine erfolgreiche und verwertbare Simulation.

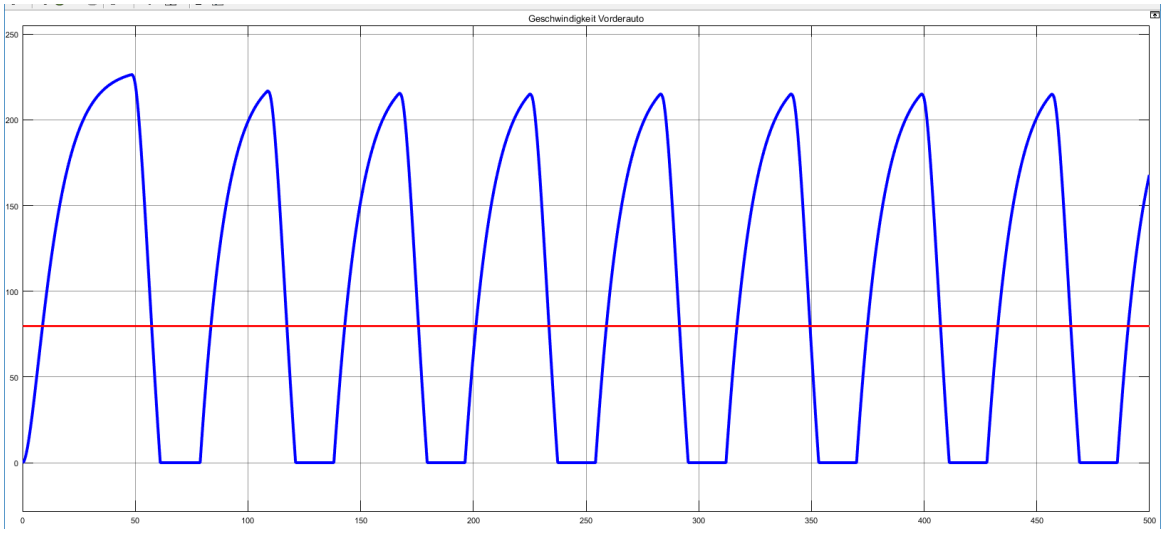

Abbildung 14: Vergleich der Geschwindigkeiten von Vorderauto (rot) und Folgeauto (blau) für die Werte X0 = 800 m und V0 = 80 km/h

Bei Betrachtung der ersten MIL Simulation und den dazugehörigen Plots werden zwei weitere Probleme ersichtlich. Zum einen gibt es eine konstante Abweichung des Anfangsabstands um 300m und zum anderen ist das Verhalten des Schwingkreises nicht stabil. Die Abweichung des Anfangsabstandes resultiert aus der Führungsgröße des Step-Blocks „Sollabstand“. Anders als angenommen und anders als in üblichen Regelkreisen bewirkt die Regeldifferenz an dieser Stelle keinen Vergleich von Führungsgröße und Regelgröße, sondern ein konstantes minimieren des Abstands beider Fahrzeuge von einander um 300 m. Step- und Sum-Block werden dementsprechend entfernt. Der Sollabstand wird mittels Optimierung des Fuzzy-Logik in das System eingeführt. Nachteil an dieser Methode ist das für eine Änderung des Sollabstands die gesamte Fuzzy-Logik angepasst bzw. die Zugehörigkeitsfunktionen verändert werden müssen. Das zweite Problem besteht in den aktuellen Zugehörigkeitsfunktionen und der dazugehörigen Regelbasis.

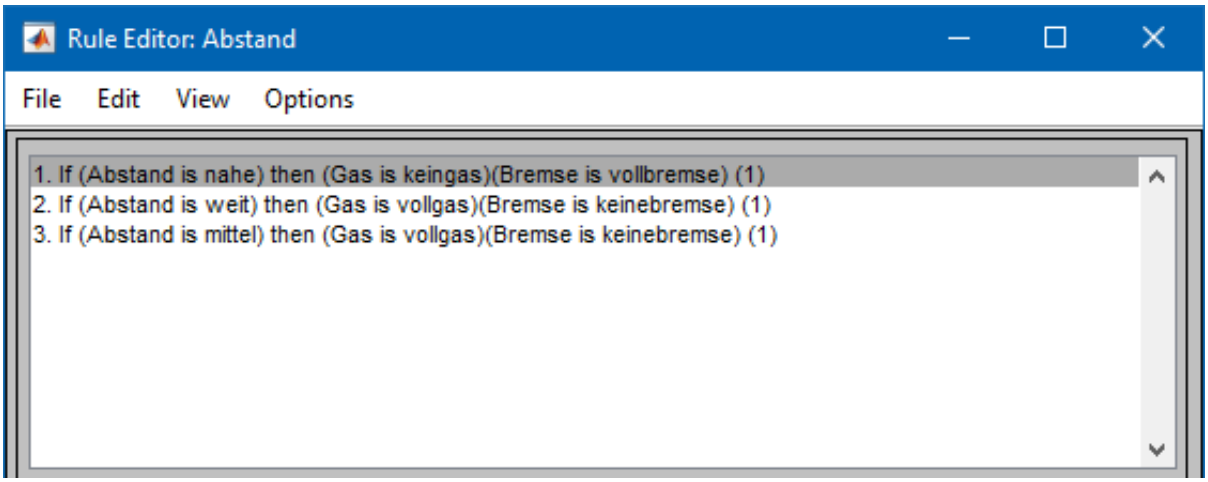

Abbildung 15: Rule Editor mit den Regeln der bereitgestellten „Abstand.fiz“

Zwischen den linguistischen Variablen „nahe“ und „mittel“ liegt der Wert 300m. Dieser kann jedoch nie eingeregelt werden. Beim Erreichen eines Abstandes der der Menge „nahe“ zugeordnet wird, erfolgt eine Vollbremsung. Beim Erreichen eines Abstandes der der Menge „mittel“ zugeordnet wird erfolgt Vollgas. Das bedeutet das der gegebene Sollabstand nie eingehalten werden kann. Des Weiteren ist zu beachten das das Gaspedal betätigt werden muss um eine Geschwindigkeit zu halten. Demzufolge muss der Fuzzy-Regler für eine bestimmte Geschwindigkeit eingestellt werden oder man nimmt eine Differenz vom Sollabstand in Kauf, kann dafür aber auf verschiedene Geschwindigkeiten des Vorderautos reagieren. Aus diesen Überlegungen heraus wird eine neue Fuzzy-Logik für den Regelkreis entworfen. Basis

der neuen Logik ist die Zugehörigkeitsfunktion „perfekt“ der Eingangsgröße „Abstand“. Diese linguistische Menge „perfekt“ beinhaltet einen Wertebereich um den Sollabstand von 300m. In dieser soll vorerst weder Gas- noch Bremspedal betätigt werden. Durch diesen betätigungsfreien Raum wird das schwingende Verhalten reduziert, jedoch nicht vollständig unterbunden. Da das Gaspedal betätigt sein muss um eine Geschwindigkeit aufrechtzuerhalten wird die Mitte der „perfekt“ Funktion nicht auf 300 gelegt, sondern etwas nach links verschoben. Dies verringert die Differenz zum Sollabstand, kann jedoch dazu führen das bei geringen Geschwindigkeiten der Sollabstand unterschritten wird. Da bei höheren Geschwindigkeiten die Abweichung des Sollabstandes, aufgrund der benötigten Gaspedalbetätigung um ein vielfaches größer ist, ist dies ein akzeptabler Kompromiss.

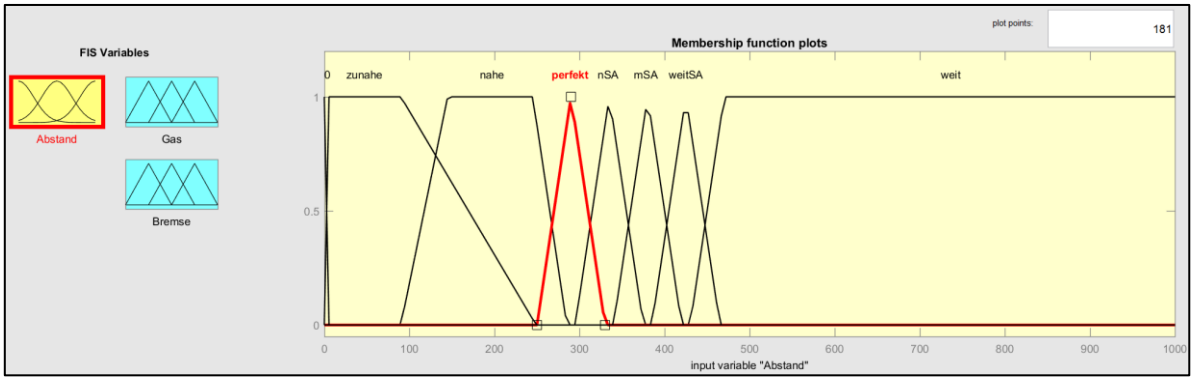

Abbildung 16: Zugehörigkeitsfunktionen der Eingangsgrößen der neu erstellten Fuzzy-Logik

Die Werte der „perfekt“ Zugehörigkeitsfunktion sind auf [240 290 330] zu setzen. Der große Wertebereich wird im späteren Verlauf im Zusammenhang mit den benachbarten Zugehörigkeitsfunktionen und der Aufstellung der Regelbasis erklärt. Des Weiteren werden nun Zugehörigkeitsfunktionen zum Beschleunigen im Falle eines Abstandes größer als 300m und Funktionen zum Bremsen für Abstände kleiner als der 300m benötigt. Die Anordnung des Entwurfs ist in der Abbildung 16 dargestellt. Die Abstände sind wie folgt definiert. Der Bereich „0“ umfasst einen Wertebereich -1000 bis 0. Dieser Bereich ist so gewählt das im Falle einer Kollision bzw. eines Überholvorgangs und dem daraus resultierenden negativen Abstand zwischen den Fahrzeugen die Fuzzy-Logik weiterhin funktioniert und den Simulationsvorgang nicht unterbricht. Wird dies nicht beachtet so verlässt die Simulation den Wirkbereich des Fuzzy-Reglers und es kommt zu Fehlermeldungen bzw. zum Stillstand der Simulation. Anschließend folgen die Bereiche „zunahe“ und „nahe“. In diesen Bereichen ist der Sollabstand überschritten. Daraus resultierend müssen diese Bereiche mit brem-

senden Funktionen verknüpft werden. Im Anschluss folgen die Bereiche „nSA“, „mSA“ und „weitSA“. Diese sind stark überlappend angeordnet um verschiedene Gaspedalbetätigungsgrade zu ermöglichen. Sichtlich wird dies nach der Verknüpfung von Ein- und Ausgangsgrößen im Rahmen der Regelbildung. Der letzte Bereich mit der Bezeichnung „weit“ reicht von 430 bis 1e+15. Auch hier ist darauf zu achten das die Simulation den Wertebereich des Fuzzy-Reglers nicht überschreitet. Des Weiteren werden die zwei Ausgangsgrößen „Gas“ und „Bremse“ mit Zugehörigkeitsfunktionen versehen. Für die Größe „Gas“ werden fünf Singletones-Funktionen mit den Werten 0% (keingas), 25% (1/4gas), 50% (halbgas), 75% (3/4gas) und 100% (vollgas) gewählt. Die Zugehörigkeitsfunktionen der Größe „Bremse“ werden mit den Singletones-Funktionen 0% (keinebremse), 20% (mittelbremse), 100% (vollbremse) und 200% (Kollision). Auf Grundlage der Erkenntnisse des Modells Hindernis und des ersten Abstand-Plots des Modells Abstand wird der Wert für eine mittlere Bremsung nur auf 20% gelegt, da höhere Werte in ein zu starkes Bremsverhalten resultieren und ein erhöhtes Schwingverhalten mit sich bringt. Im nächsten Schritt werden die Zugehörigkeitsfunktionen der Ein- und Ausgangsgrößen mittels „Rule Editor“ miteinander verknüpft.

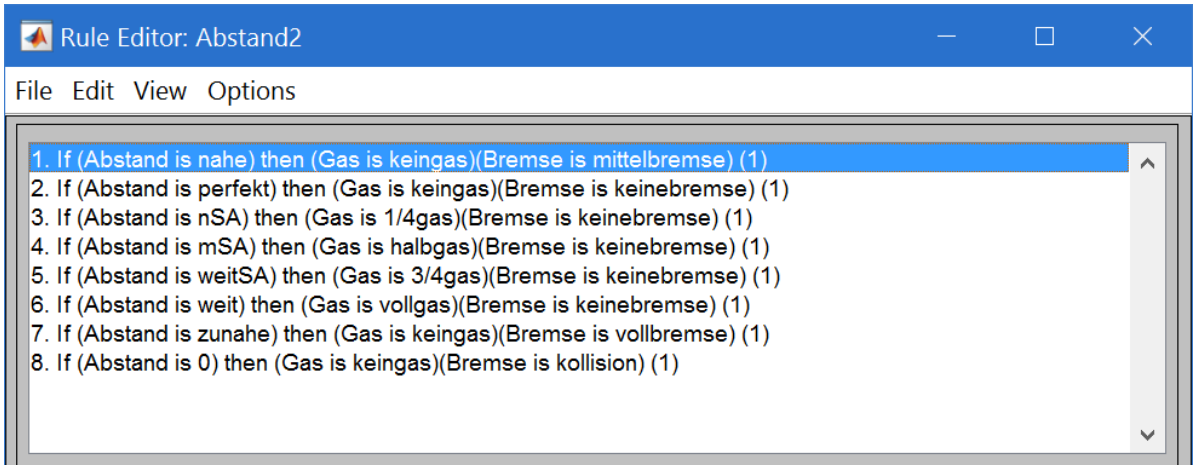

Abbildung 17: Rule Editor mit den Regeln der optimierten „Abstand_opt.fiz“

Ein weiterer Grund der Überlappung und des angesprochenen Bereichs der Zugehörigkeitsfunktion „perfekt“ ist die Kombination mehrere Ausgangsgrößen. Somit werden auch Werte zwischen den festgelegten Ausgabegrößen berechnet, vergleiche Abbildung 18. Durch den betätigungsfreien Bereich der Menge „perfekt“ wird ein gleichzeitiges Betätigen von Gas und Bremse unterbunden. Die Überlappung ermöglicht zusätzlich einen sanfteren Übergang vom Brems- und Beschleunig-

ungsvorgang, da die Ausgangsgrößen nun stufenlos sind. Des Weiteren werden durch die gewählte Anordnung maximal zwei Mengen der Eingangsgrößen verknüpft.

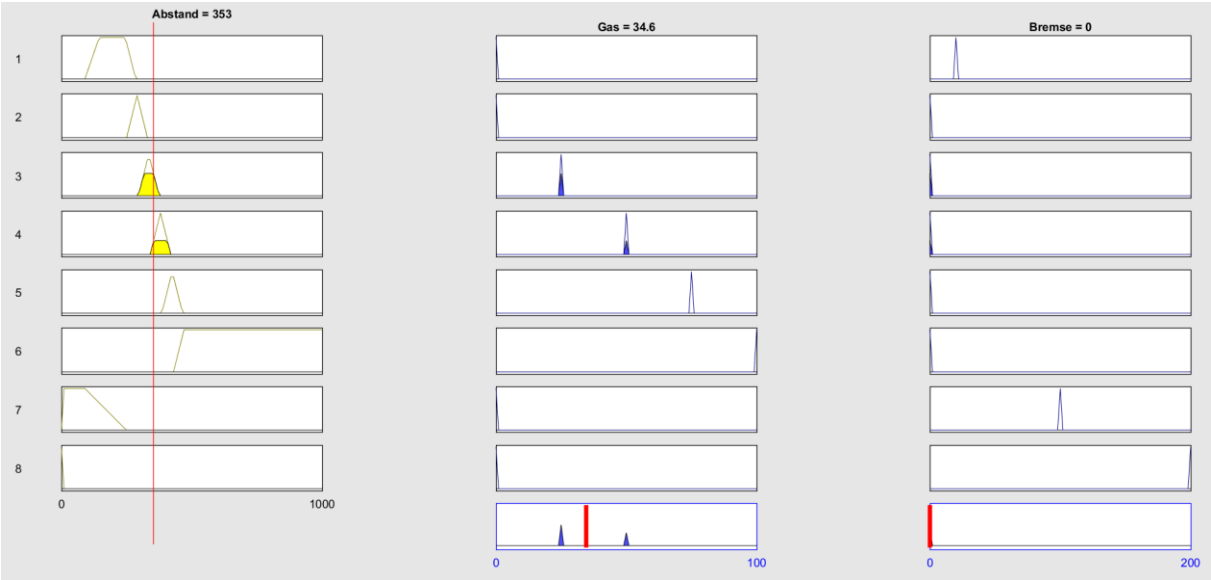

Abbildung 18: Rule Viewer mit eingestellten Beispielwerten und sichtbare Verknüpfung der Zugehörigkeitsfunktionen

Bei einer größeren Anzahl von Verknüpfungen häufen sich Simulationsfehler und Unterbrechungen. Das Einschwingverhalten der Abstandsregelung ist abhängig von dem gewählten Anfangsabstand und der gewählten Geschwindigkeit des Vorderautos. Mehrere Versuche die Fuzzy-Logik diesbezüglich zu optimieren resultieren in einer Verbesserung auf der einen Seite, z.B. niedrige Geschwindigkeit des Vorderautos bei geringem Abstand, aber auch in eine Verschlechterung bei geänderten X0 und V0 Parametern. Eine weitere versuchte Überlegung ist die breite oder Form der Zugehörigkeitsfunktionen zu modifizieren. Das Resultat ist jedoch identisch. Als Schlussfolgerung ist festzuhalten das ein ändern der Form der Zugehörigkeitsfunktionen nur bei festen X0 und V0 Parametern für das Vorderauto Erfolg versprechen. Werden die Werte des Vorderautos jedoch durch nicht änderbare ersetzt so ist es sinnvoller die Fuzzy-Logik an diese Werte anzupassen. Der Regler ist dann allerdings nur für einen bestimmten Fall benutzbar und somit weniger für das generelle halten eines Abstandes zweier Fahrzeuge zueinander geeignet. Deshalb wird der bereits genannte Kompromiss der Sollabstandsabweichung mit zunehmender Geschwindigkeit in Kauf genommen. Projiziert man dieses Verhalten in eine Reale Umgebung so ist größerer Abstand zum Voraus fahrenden Fahrzeug bei steigender Geschwindigkeit vertretbar. Je nach Geschäftssinn und Marketingstrategie kann dies

sogar als Vorteil verkauft werden. Wenn der Sollabstand als Mindestabstand oder Mindestsicherheitsabstand auslegt ist, dann ist dieses Problem gänzlich behoben. Ansonsten ist das Abstands Modell optimiert und angepasst. Das Folgeauto hält einen sich einpendelnden stabilen Abstand zum Vorderauto. Auch die Geschwindigkeit des Folgeautos passt sich die Geschwindigkeit des Vorderautos an.

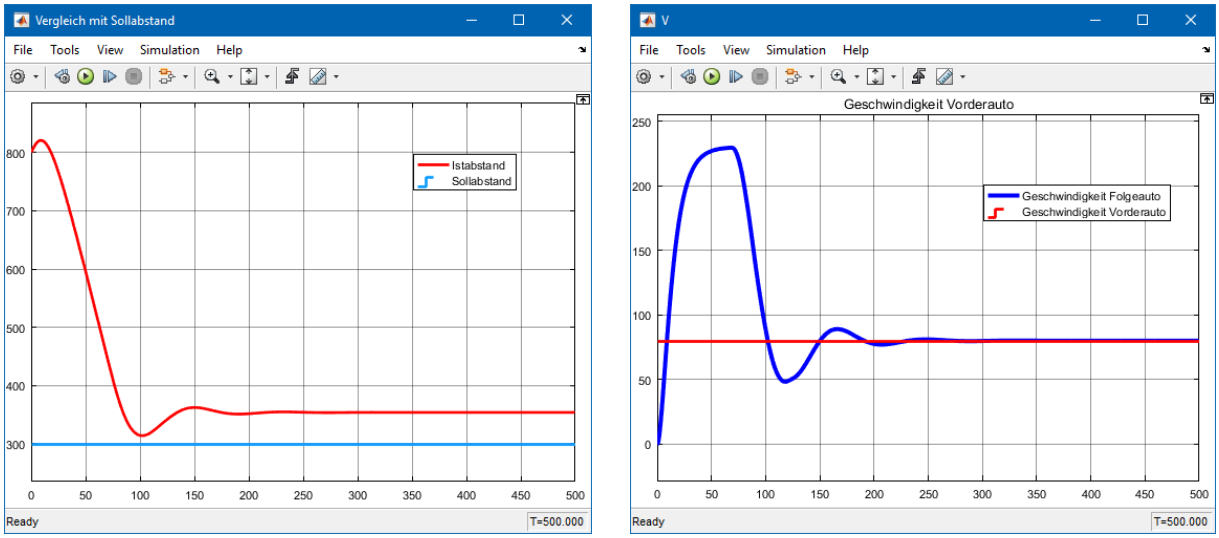

Abbildung 19: links: der Vergleich des Sollabstandes (blau) mit dem Istabstand (rot), rechts: der Geschwindigkeitsvergleich von Vorder- (rot) und Folgeauto (blau) für die Werte X0 = 800 m und V0 = 80 km/h

Alle Dateien, Optimierungen und Anpassungen sind im Anhangsverzeichnis der beigelegten DVD unter dem Pfad MATLAB/Abstand mit der Kennzeichnung „opt“ zu finden.

3.3 Wahl des Regelkreises

Die Wahl des Regelkreises bzw. des Modells für die weiterführende Bearbeitung fällt auf das Abstands Modell. Der Hauptgrund für diese Wahl ist das trotz vorgenommenen Anpassungen und Optimierungen das Problem der weiterlaufenden Positionsanzeige fortbesteht. Resultierend ist dieses Modell weiter von der Realität entfernt als das Abstands Modell. Positive Aspekte die für das Abstands Modell sprechen sind das der Sachverhalt ohne schwerwiegende Fehler zu simulieren ist, das es in Bezug auf das das Thema Industrie 4.0 und Mixed Reality das ansprechendere Szenario ist und das deutlich mehr Eigenleistung in der Anpassung und Optimierung des Regelkreises steckt. Der Optimierte Regelkreis „Abstand“ ist in der Abbildung 20 ersichtlich.

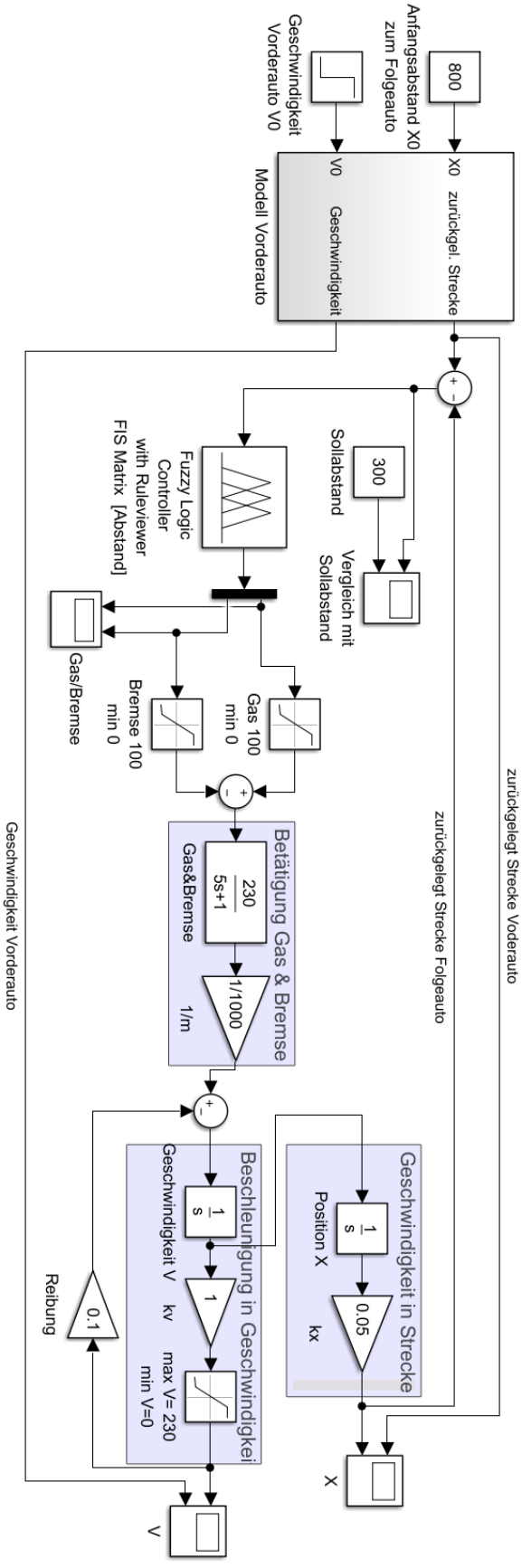

Abbildung 20: Ansicht des optimierten Regelkreises „Abstand“

4. Erstellen des SIL Controllers

Nach dem die MIL Simulation des Regelkreises bereits während der Optimierungsvorgänge zum Einsatz gekommen ist wird nun der nächste Entwicklungsschritt des Modells zur Mixed Reality durchgeführt. Wie bereits im Kapitel 2.3 erläutert, ist mittels SIL Simulation ein ausführbarer C-Code für die später verwendete Hardware zu erzeugen. Diese wird jedoch während der SIL-Simulation noch auf dem Entwicklungsrechner verwendet. Dafür ist es notwendig den bisherigen FLC-Bock in ausführbaren Code zu wandeln und einen SIL-Fuzzy-Controller-Block zu erstellen. Der SIL-Block wird benötigt um MIL und SIL Ergebnisse miteinander zu vergleichen und bei Abweichungen eine Analyse zur Problembeseitigung durchführen zu können. Als Anleitung dient das von Herrn Prof. Dr.-Ing. Zacher bereitgestellte Dokument „Sotware-in-the-Loop“ [10]. Achtung, bei allen nachfolgenden Arbeitsschritten darf der Ort und der Pfad der Dateien bzw. die Verzeichnisse nicht geändert werden. MATLAB erstellt während der SIL- und PIL-Simulationen ortsfeste Verzeichnisse. Die darin enthaltenen und erstellten Dateien sind nur in diesen Verzeichnissen funktionsfähig. Das gleiche gilt für die spätere PIL-Simulation. Als Erstes wird das optimierte Abstands Modell in Simulink geladen. Dann wird die „Abstand_opt.fiz“ geladen und implementiert. Es ist drauf zu achten das die Datei als „Abstand“ in den Workspace zu exportiert ist. Zunächst sind die Einstellungen der bereitgestellten Anleitung (vgl.[10], S. 3) in den „Model Configuration Parameters“ zu übernehmen. Dies betrifft die „Solver options“ und die „Fixed-step size“. Im Anschluss wird das Simulink-Model und die darin enthaltenen Signale und Glieder nach der Anleitung digitalisiert bzw. diskretisiert. Über den Reiter „Analysis -> Control Design -> Model Discretizer“ wird der „Simulink Model Discretizer“ aufgerufen. In diesem wird die „Sample time“ auf 0.1 gestellt und anschließend der „S->Z“ Button betätigt. (vgl.[10], S. 4). Die Digitalisierung des Modells wurde durchgeführt. Als nächstes wird der Fuzzy-Block in ein Subsystem mit der Bezeichnung „Controller“ umgewandelt. Anschließend sind mehrere Einstellungen in den „Configuration Parameters“ zu tätigen. Diese sind der Anleitung auf Seite 6 bis 8 und Seite 17 bis 21 zu entnehmen. Nach dem die Einstellungen angepasst worden sind wird der Simulationsmodus von „Normal“ in „SIL“ geändert. Der Controller wird mit einem Mausklick markiert und mittels „Build Selected Subsystem“ in einen verwertbaren Code kompiliert. (vgl.[10], S. 9f.). Das aufpoppende Fenster „Build code for Substem: Contoller“ wird mit „build“ bestätigt. Die Tabellen in

diesem Fenster sind leer. Sollte es zu Fehlermeldungen kommen ist darauf zu achten das alle in diesem Kapitel erwähnten Arbeitsschritte im gleichen Ordner und in der gleichen slx-Datei ausgeführt worden sind. Das Erstellen des Codes und die Simulation sind erfolgreich. Es wurde keine Fehlermeldung oder Warnung vom „Diagnostic Viewer“ ausgegeben. Im verwendeten Verzeichnis befinden sich nun zwei neuerstellte Ordner und eine Controller.exe. Allerdings ist der für den folgenden Vergleich notwendige SIL-Controller-Block nicht erstellt worden. Nach den in der Anleitung auf der Seite 22 enthaltenen Befehlen wird nun über das „Command Window“ MATLABs versucht den SIL-Block manuell zu erzeugen. Zuerst sind die zuvor erstellten Ordner aus dem Verzeichnis zu löschen. Danach sind die Befehle aus Abbildung 21 in das „Command Window“ einzugeben. Der Befehl „model“ wird mit den Namen des Regelkreises initialisiert. Gefolgt von dem Befehl „set_param (model,'CreateSILPILBlock','SIL')“. Abschließend wird mit dem Befehl „rtwbuild([model '/Controller'])“ (Leerzeichen nach model) der C-Code generiert. Das Wort „Controller“ stellt dabei das Subsystem des Fuzzy-Reglers da.

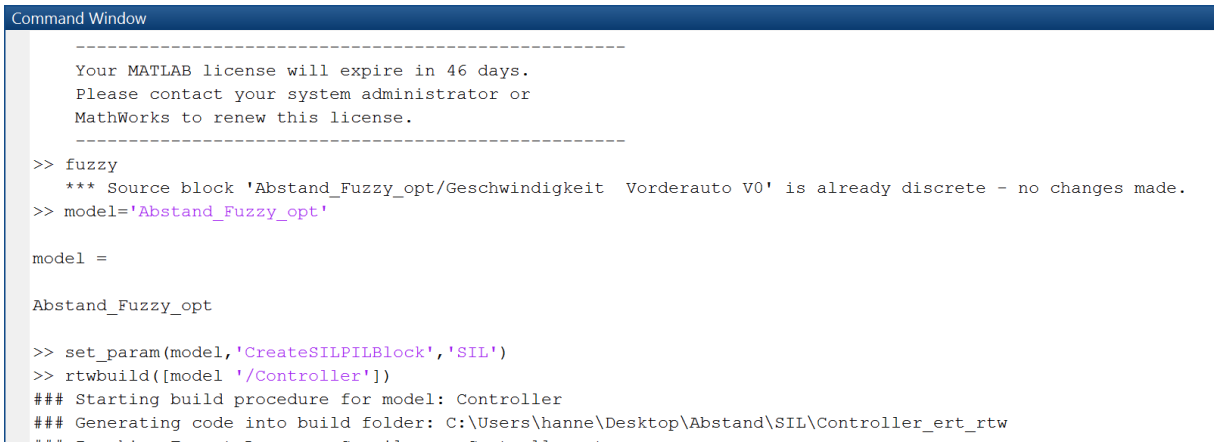

Abbildung 21: Darstellung der manuellen SIL-Block Erzeugung im Command Window

Der Kompilierungsprozess startet nun erneut und öffnet nach Beendigung ein neues „unnamed“ Simulink Fenster mit enthaltenem SIL-Block. Dieses wird als „SIL_Block.slx“ gespeichert. Alternativ zur manuellen Erzeugung ist im „Configuration Parameters“ Menü unter dem Reiter „All Parameter“ bei der Einstellung „Create block“ die Option „MIL“ auszuwählen und der ersten Beschreibung zu folgen. Anschließend wir das Subsystem „Controller“ durch den erstellten SIL-Block ersetzt und mittels „Normaler“ Simulation getestet. Es kann ggf. zu einer Treiberwarnung seitens des Windows Betriebssystems kommen. Diese ist mit Ok zu bestätigen. Die Simulation ist erfolgreich beendet. Es werden ggf. zwei Warnungen aufgrund des nicht verbundenen

Subsystems Controller ausgegeben. Diese können vernachlässigen werden. Im Anschluss erfolgt der Vergleich von MIL und SIL Fuzzy-Regler. Dazu muss die bestehende Schaltung modifiziert werden um beide Controller mittels Plot miteinander vergleichen zu können. Die Änderungen des Regelkreises sind in Abbildung 22 dargestellt.

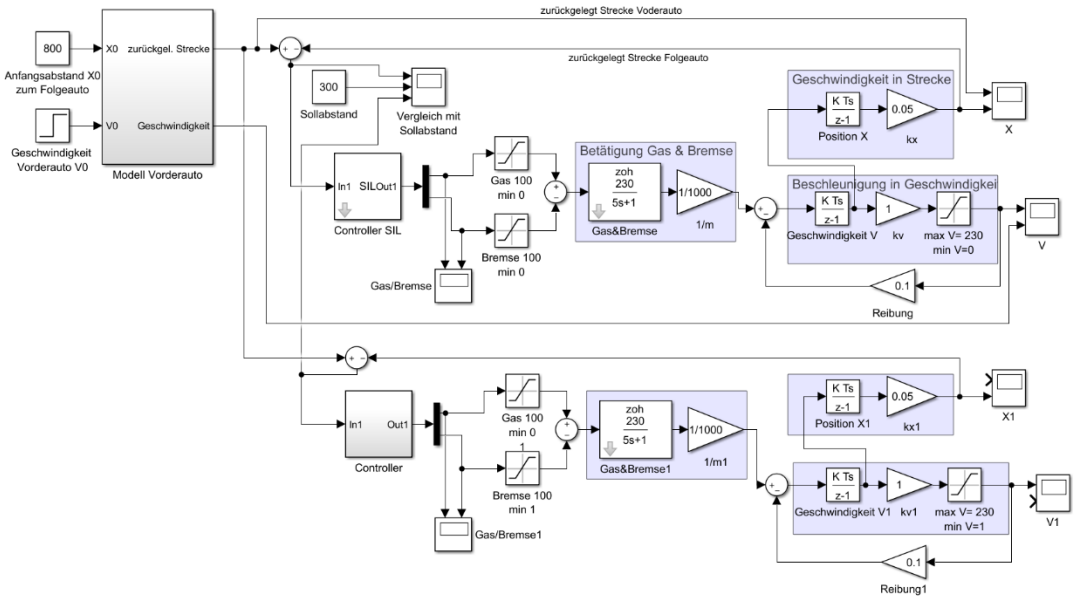

Abbildung 22: Simulink Modell für den Verlgeich von MIL- und SIL-Simulation

Zum Vergleich wird die Simulation nun im „Normal“ Modus gestartet. Als Vergleichswert können Geschwindigkeit, Position oder Sollabstand in Betracht gezogen werden. In der Abbildung 23 ist der Plot des Sollabstands vergleichend dargestellt.

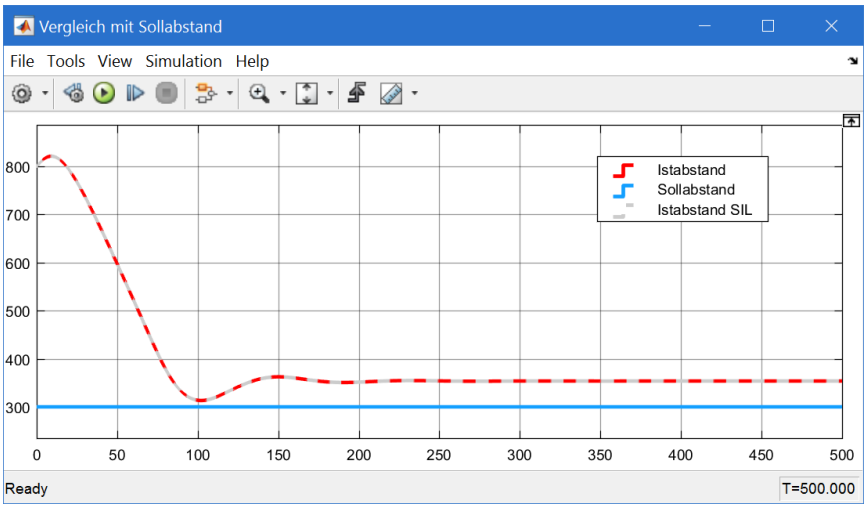

Abbildung 23: Vergleich des Sollabstandswert mit den Istabstandswerten von MIL- und SIL-Simulation für die Werte X0 = 800 m und V0 = 80 km/h

Da die Graphen des MIL und SIL Regelkreises identisch sind und keine Abweichung vorliegt ist bereits das bestmöglichste Ergebnis für den SIL-Block erzeugt worden. Es bedarf keinerlei weiterer Analysen oder Anpassungen des SIL-Controllers.

5. Erstellen des PIL Controllers & Programmieren des Mikroprozessors

Die letzte Entwicklungsstufe dieser Studienarbeit besteht in der Erstellung des PIL Controllers und der damit verbundenen Programmierung des STM32F Discovery Boards als Mikroprozessor. Folglich wird der Fuzzy-Regler auf eine Hardware ausgelagert und im Anschluss mit dem virtuellen Fahrzeugmodell als Regelkreis simuliert. Ab diesen Zeitpunkt kann man das Modell als einen Entwurf der Mixed Reality bezeichnen. Als Vorbereitung für die Erstellung eines PIL-Controllers sind die erforderlichen Treiber des STM32F4 Discovery Bords auf den Entwicklungsrechner zu installieren. Dazu wurde die Anleitung „Tutorial zu Installation für Board STM32F4-Discovery mit MATLAB®/Simulink®“ [11] verwendet. Das STM32F4-Discovery Board wurde auf zwei mit Windows 10 (64 bit) betriebenen Entwicklungsrechnern direkt beim Anschließen erkannt. Ein Installieren der Treiber nach Kapitel 1 der Anleitung ist somit nicht nötig. Anschließend wird das „Embedded Coder Support Package for STMicroelectronics Discovery Boards” installiert. Aufgrund der in der Anleitung verwendeten Version von MATLAB 2015b unterscheidet sich das „Get Hardware Support Packeges“ Menü und die Installation deutlich.

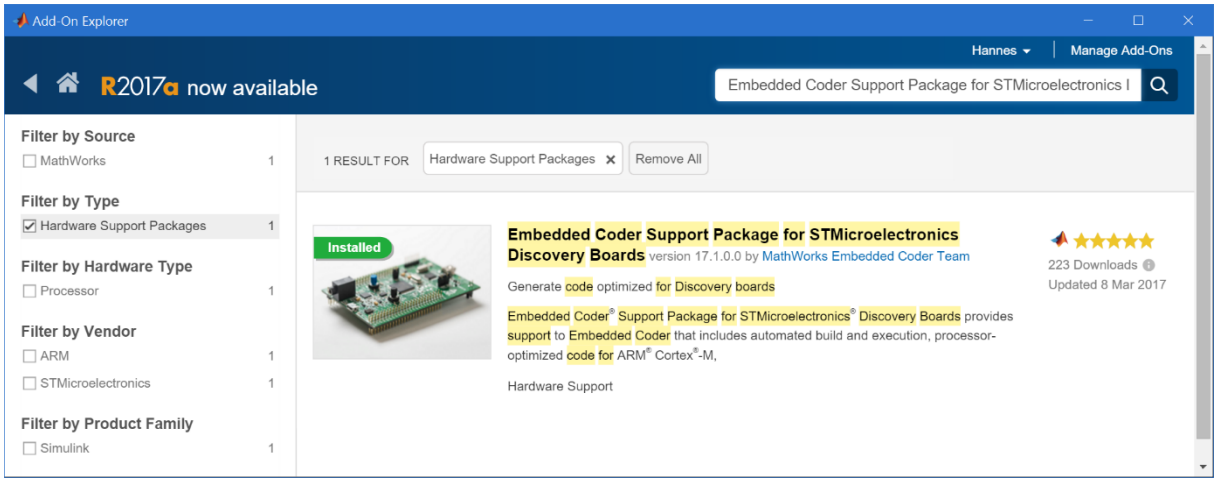

Abbildung 24: Neues Menü zur Verwaltung und Instalation von Add-Ons in MATLAB R2016b

In die Suchleiste ist der Name des gewünschten Support Package einzugeben. Anschließend wird das treffende Ergebnis mit einem Klick ausgewählt und über den „Install“ Button installiert. Download und Installation erfolgt in einem separaten Fenster automatisch. Ein Klicken durch mehrere Menüs wie in der Anleitung beschrieben ist nicht mehr nötig. Nach dem Herunterladen und Installieren der Support Package wird die Frage gestellt ob man das Support Package jetzt oder später konfigurieren möchte. Diese Frage wird mit „SETUP NOW“ bestätigt. Im Folgenden Fenster wählt man das

„STMicroelectronics Microcontrollers (Embedded Coder)“ und klickt auf „Next“. Woraufhin das in Abbildung 25 dargestellte Fenster erscheint.

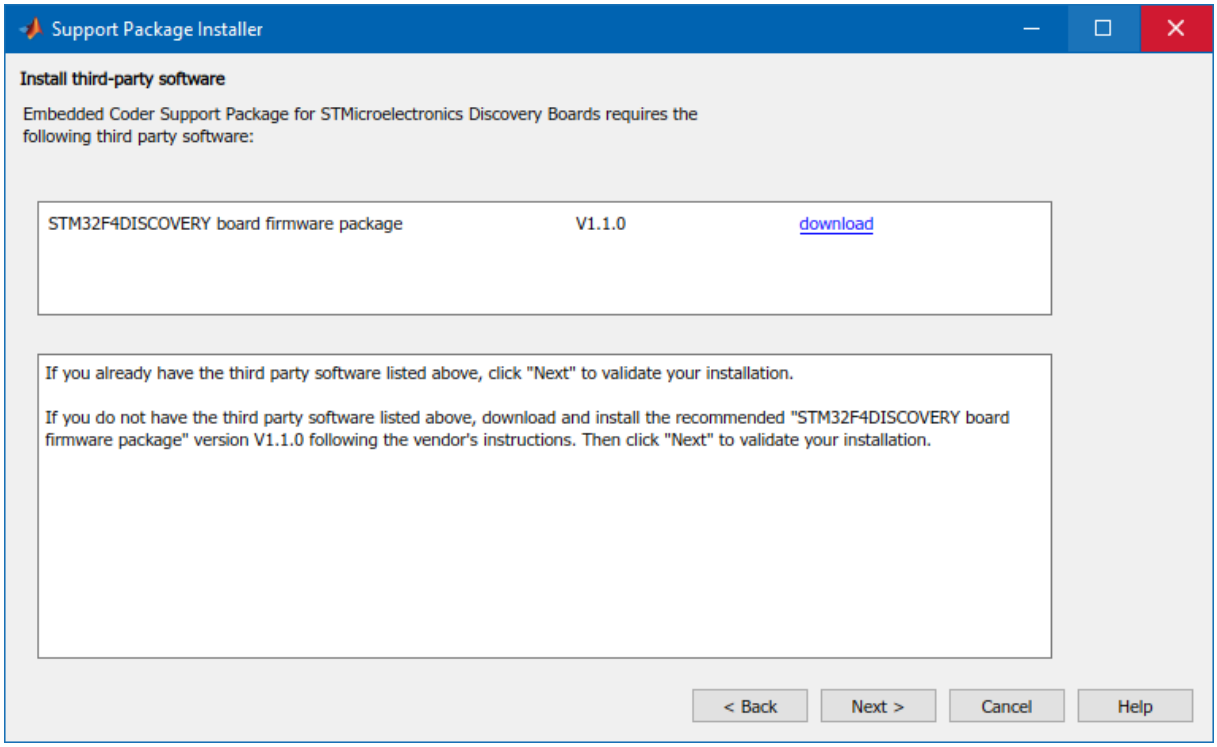

Abbildung 25: Konfigurieren des Embedded Coder Support Package for STMicroelectronics Discovery Boards

Zunächst ist das „firmware package“ herunterzuladen und zu entpacken. Dazu ist es erforderlich sich bei der Firma ST zu registrieren. Im nächsten Schritt muss das „firmware package“ validiert werden. Alle Schritte bis auf das Herunterladen und Entpacken erfolgen in MATLAB. Das Herunterladen der CMSIS Software von ARM Cortex laut Anleitung ist ebenfalls nicht mehr nötig. Nachdem die Treiber installiert worden sind werden diese auf Funktionsfähigkeit mit der Übung „Board STM32F4-Discovery mit Embedded Coder Discovery von MATLAB®/Simulink®“ [12] getestet. Die Übung wird dabei eins zu eins nachprogrammiert. Dabei sind keine Fehler aufgetreten und das Programm funktioniert wie in der Anleitung beschrieben. Da nun die Treiber installiert und funktionsfähig sind wird die ursprüngliche „Abstand_Fuzzy_opt.slx“ Datei in einem neuen Ordner, z.B. mit der Bezeichnung PIL, kopiert. Danach wird die Datei in Simulink geladen und die „Abstand_opt.fiz“ wird implementiert. Anschließend sind alle Einstellungen und Schritte der SIL-Simulation bis zum Erstellen des SIL-Blocks zu weiderholen. Das Controller Subsystem wird jedoch noch nicht kompiliert. Zunächst müssen weitere Einstellung im „Modell Configuration Parameters“ Menü getätigt werden. Zu beachten ist das das STM32F4-Discovery mit dem

Entwicklungsrechner verbunden ist. Unter dem Reiter „Hardware Implementation“ ist als „Hardware board“ das STM32F4-Discovery einzustellen. Des Weiteren ist unter den Reiter „Code Generation“ bei den „Toolchain settings“ „GNU Tools for ARM Embedded Processors“ zu wählen. Dann ist das Subsystem mit der Bezeichnung „Controller“ auszuwählen und im PIL Modus mittels „Deploy selected Subsystem to Hardware“ der Fuzzy-Controller auf das STM32F4-Discovery Board zu übertragen. Die Verfahrensweis ist nun wieder identisch mit der SIL-Simulation. Bei erfolgreicher Übertragung leuchtet das LD1 Lämpchen des STM32F4 grün anstatt rot. Obwohl die PIL-Simulation erfolgreich und ohne Fehler oder Warnungen abgeschlossen ist, ist ähnlich wie bei der SIL-Simulation kein PIL-Block entstanden. Das Problem wird manuell mit angepassten Befehl „set_param(model,'CreateSILPILBlock' ,PIL)“ gelöst. Alternativ ist im „Configuration Parameters“ Menü unter dem Reiter „All Parameter“ bei der Einstellung „Create block“ die Option „PIL“ auszuwählen und der Schritt der „Deploy selected Subsystem to Hardware“ erneut auszuführen. Anschließend wird Subsystem durch den PIL-Controller ersetz und mittels Simulation im Modus „Normal“ getestet. Die Simulation mit verbundenem STM32F4 dauert mehrere Minuten. Das LD1 Lämpchen des STM32F4 Discovery Board blinkt während dessen abwechselnd rot und grün. Die PIL-Simulation wird nun mit der MIL-Simulation verglichen. Dazu wird das Simulink Modell des SIL-Vergleichs übernommen und angepasst. Die Anpassung betreffen vor allem die Einstellungen im „Configuration Parameters“ Menü. Zusätzlich wird der SIL-Blocks durch den PIL-Block ausgetauscht. In Abbildung 26 ist das Ergebnis des Vergleichs dargestellt.

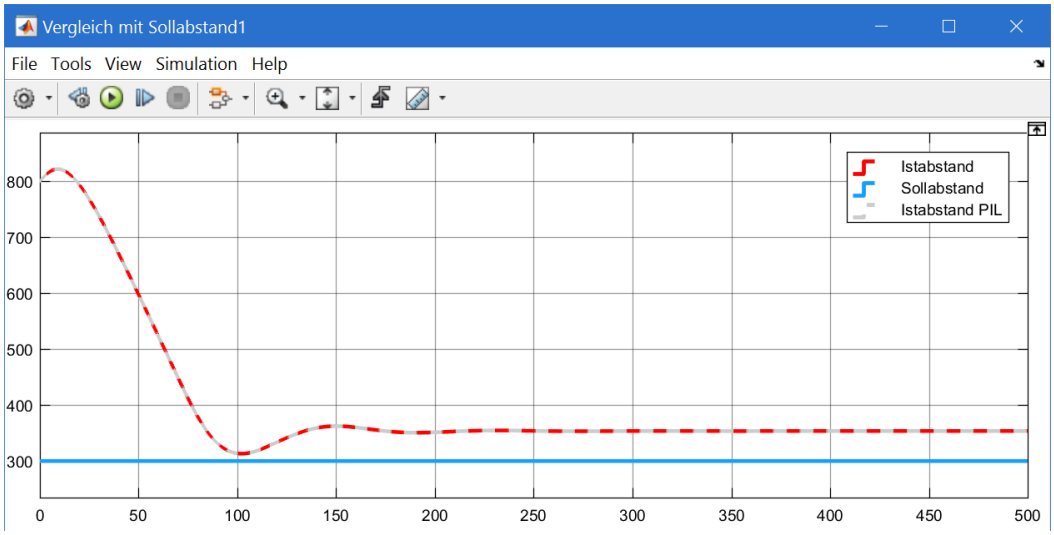

Abbildung 26: Vergleich des Sollabstandswert mit den Istabstandswerten von MIL- und PIL-Simulation für die Werte X0 = 800 m und V0 = 80 km/h

Auch hier sind die Graphen des MIL und PIL Regelkreises identisch und weisen keine Abweichung auf. Es bedarf also keinerlei weiterer Analysen oder Anpassungen des PIL-Controllers.

6. Zusammenfassende Betrachtung der Ergebnisse

Die Studienarbeit ist mit der erfolgreichen PIL-Simulation bzw. mit einem funktionierenden Entwurf der Mixed Reality und dem abschließenden Vergleich mit dem MIL-Fuzzy-Regler beendet. Auf Grundlage des zweiten Absatzes der Studienarbeit und des gesamt geschilderten Arbeitsablaufs wurde ein Eindruck der Arbeitsweiseweise und Entwicklung in der künftigen Industrie 4.0 dargestellt. Beide bereitgestellten Simulink Modelle wurden überarbeitet und optimiert. Eines der beiden Simulink Modell wurde über MIL-, SIL- und PIL-Simulation in einen funktionierenden Regelkreis aus Hardware- bzw. Mikroprozessor-Controller und virtuellen Fahrzeugmodell überführt. Entstandene Probleme und Fehler wurden dokumentiert und durch ebenfalls dokumentierte Lösungsvorschläge ergänzt. Nicht alle Probleme konnten gänzlich gelöst werden. Dazu zählen die fortlaufende Positionsänderung ohne vorhandene Geschwindigkeit des Hindernis Modells sowie die parameterabhängige, unregelmäßige Veränderung der Sollabstandsdifferenz des Modells Abstand welche sich bei zunehmender Geschwindigkeit vergrößert. Mit Verweis auf die ungelösten Probleme wurden ebenfalls kritische Aspekte aufgezeigt die eine Nutzung des Fuzzy-Reglers ohne weitere Elemente für das Abstandsszenario und dessen Anforderungen als suboptimal darstellen. Ein Grund ist unter anderem die fehlende Flexibilität der Fuzzy-Logik in Bezug auf variable Sollabstands-, Geschwindigkeits- und Positionswerte. Mittels neuen Betrachtungswinkel wurde versucht diese Defizite zu minimieren. Dazu wurde ansatzweise demonstriert das eine differenzierte Betrachtung mit Hinblick auf reale Sachverhalte sinnvoll gegenüber der reinen Auswertung bzw. dem Vergleichen von Zahlen, Werten und Graphen sein kann. Insbesondere sind hier das fließende Bremsverhalten des Hindernis Modells und die mögliche Auslegung des Sollabstandes als Sicherheitsabstand des Modells Abstand zu nennen. Trotz virtuell abgebildeter Realität und einfach durchzuführenden Simulationen, darf die reale Welt nicht außer Acht gelassen werden, da ein realer Sachverhalt seltenst vollständig in ein virtuelles Modell überführt werden kann. Die neue Betrachtungsweise löst jedoch nicht das Problem des Hindernis Modells. Zusätzlich wurden bereits bestehende und in der Studienarbeit verwendete Anleitungen mit aktuellen Vorgehensweisen für eine neuere Programmversion MATLABS ergänzt und um einige Anweisungen und Hinweise erweitert.

7. Literatur- und Quellenverzeichnis

[1] Bundesministerium für Bildung und Forschung, „www.bmbf.de," Bundesministerium für Bildung und Forschung, [Online]. Available: https://www.bmbf.de/de/zukunftsprojekt-industrie-4-0-848.html. [Zugriff am 09. März 2017].

[2] R. Drath, „Industrie 4.0 – eine Einführung," in *open automation (Ausgabe 3/14)*, VDE Verlag, 2014.

[3] H. Hirsch-Kreinsen, P. Ittermann und J. Niehaus, „Einleitung: Digitalisierung industrieller Arbeit," in *Digitalisierung industrieller Arbeit: Die Vision Industrie 4.0 und ihre sozialen Herausforderungen*, Baden-Baden, Nomos, 2016.

[4] Deutsches Institut für Normung, „www.din.de," DIN Deutsches Institut für Normung e. V., [Online]. Available: http://www.din.de/de/forschung-und-innovation/industrie4-0/was-ist-industrie-4-0. [Zugriff am 09. März 2017].

[5] S. Zacher und M. Reuter, Regelungstechnik für Ingenieure, Wiesbaden: Springer Vieweg, 2014.

[6] S. Zacher, Regelungstechnik 2, Mannheim: Vorlesungsskript, 2016.

[7] dSPACE, „www.dspace.com," dSPACE GmbH, [Online]. Available: https://www.dspace.com/de/gmb/home/products/sw/pcgs/targetli/simulation.cfm. [Zugriff am 12. März 2017].

[8] Wikipedia, „de.wikipedia.org," 17. Januar 2017. [Online]. Available: https://de.wikipedia.org/wiki/Hardware_in_the_Loop. [Zugriff am 12. März 2017].

[9] Plexim, „www.plexim.com," Plexim GmbH, [Online]. Available: https://www.plexim.com/de/plecs/pil. [Zugriff am 12. März 2017].

[10] S. Zacher, „Sotware-in-the-Loop, SIL.pdf," [Online]. Available: http://www.zacher-automation.de/Download-for-Students/. [Zugriff am 28. Jannuar 2017].

[11] S. Zacher, „Tutorial zu Installation für Board STM32F4-Discovery mit MATLAB®/Simulink®, Installationen_STM32F4.pdf," [Online]. Available: http://www.zacher-automation.de/Download-for-Students/. [Zugriff am 28. Februar 2017].

[12] S. Zacher, „Board STM32F4-Discovery mit Embedded Coder Discovery von MATLAB®/Simulink®", MBSE_z_demo_STM32F4.pdf," [Online]. Available: http://www.zacher-automation.de/Download-for-Students/. [Zugriff am 28 Februar 2017].