

# Fuzzygesteuerte U-Bahn Simulation

Antonia Wagner      Daniel Buth  
Johannes Müller

Fachbereich Angewandte Informatik der Hochschule Fulda

Modul *Soft Computing*  
Prof. Dr. Oleg Taraszwow

Betreuer: Lukas Militz

Wintersemester 2010/11



# Contents

<b>1 Projektrahmen</b>	<b>3</b>
1.1 Aufgabenstellung . . . . .	3
1.2 Projektrahmen . . . . .	3
<b>2 Vorüberlegungen</b>	<b>4</b>
2.1 Szenario . . . . .	4
2.2 Fuzzy-Regler . . . . .	4
2.2.1 Eingangsvariablen . . . . .	4
2.2.2 Ausgangsvariablen . . . . .	6
<b>3 Umsetzung</b>	<b>7</b>
3.1 Fuzzy-Toolbox . . . . .	7
3.2 Applet . . . . .	7
3.2.1 Dynamikberechnungen . . . . .	9
3.2.2 Fahrzeugparameter . . . . .	10
<b>4 Fazit</b>	<b>11</b>

## Abstract

Diese Dokumentation beschreibt das Projekt *FUZ-Bahn*, das wir im Praktikum zum Modul *Soft Computing* im Wintersemester 2010/11 am Fachbereich Angewandte Informatik der Hochschule Fulda durchführten unter Betreuung von Lukas Militz. Zum Projektergebnis gehören eine fisi-Datei der Fuzzy-Toolbox von Matlab sowie ein lauffähiges Java-Applet zu Visualisierung. Alle Ergebnisse inklusive Quellcode sowie diese Dokumentation sind über den Link <https://github.com/straight-shoota/FUZ-Bahn> auf der Online-Plattform Github zum Download verfügbar.

# 1 Projektrahmen

## 1.1 Aufgabenstellung

Unsere Aufgabenstellung war es, im Rahmen des Praktikums zur Lehrveranstaltung *Soft Computing* bei Prof. Dr. Oleg Taraszow einen Fuzzy-Regler zu erstellen. Als Werkzeug stand uns dazu die *Fuzzy-Toolbox* der Software *Matlab* zur Verfügung. Optionale Bonusaufgabe war die Erstellung eines Java-Applets zur Visualisierung.

- Eingangsvariablen: mindestens 3, mindestens 15 Terme
- Ausgangsvariablen: mindestens 2, mindestens 8 Terme
- Regeln: mindestens 25

Figure 1: Projektvorgaben

## 1.2 Projektrahmen

Bei Bewegungen kommen an vielen Stellen unscharfe Werte vor wie beispielsweise Geschwindigkeit, Entfernung und Veränderungen dieser Parameter. Daher wollten wir unser Projekt im Bereich Verkehr ansiedeln. Eine der ersten größere Anwendungen eines Fuzzy-Reglers in der Praxis die automatisierte Anfahr-, Beschleunigungs- und Bremssteuerung der U-Bahn Sendai in Japan und wir entschieden uns, ein ähnliches Konzept umzusetzen.



Figure 2: Sendai Subway

Quelle: Wikimedia Commons, hochgeladen von Benutzer Jet-0

[http://commons.wikimedia.org/wiki/File:Sendai\\_subway\\_1014\\_20081021.jpg](http://commons.wikimedia.org/wiki/File:Sendai_subway_1014_20081021.jpg)

Unser Projekt sollte die Fahr- und Bremssteuerung eines schienengebundenen Verkehrsmittels wie eine U-Bahn mittels Fuzzy-Logic umsetzen. Im Gegen-

satz zum Pionierprojekt der U-Bahn Sendai wollten wir jedoch eine komplette Fahrzeugsteuerung mittels Fuzzy-Regler erreichen.

Durch unseren Regler sollte das gesteuerte Fahrzeug in der Lage sein, aus dem Stillstand anzufahren und bis zu einer bestimmten, festgesetzten Geschwindigkeit zu beschleunigen und diese zu halten. Bei auf der Strecke voraus liegenden Wegpunkten wie eine Haltestelle oder ein Signal mit Haltbegriff soll ein Bremsvorgang eingeleitet werden, der das Fahrzeug möglichst sanft und punktgenau zum Stehen bringt.

Ebenso sollte es möglich sein, eine beliebige Geschwindigkeit erreichen und halten zu können. Dies kann beispielsweise bei einem langsameren vorausfahrenden Fahrzeug nötig sein. Auch die Einrichtung einer Langsamfahrstrecke kann ein solches Verhalten erfordern.

Zusätzlich zur eigentlichen Projektaufgabe haben wir uns zudem als Ziel gesetzt, als Bonusaufgabe eine Anwendung unseres Reglers als Java-Applet zu realisieren.

## 2 Vorüberlegungen

### 2.1 Szenario

Um das Projekt überschaubar zu halten, beschränkten wir unser Szenario auf den ganz einfachen Fall eines linearen, ebenen Streckenverlaufs ohne Abzweigungen oder sonstige andersgeartete Einflüsse. Auf dieser Strecke können sich einzige Hindernisse wie Haltestellen, vorausfahrende Züge oder Geschwindigkeitsbegrenzungen befinden. Das Fahrzeug sollte nur in der Lage sein, vorwärts zu fahren, sowie Beschleunigungs- und Bremsvorgänge auszuführen. Ein Fahrtrichtungswechsel ist nicht vorgesehen.

### 2.2 Fuzzy-Regler

#### 2.2.1 Eingangsvariablen

Als Eingangsvariablen wählten wir zunächst die einfachen Fahrzeugparameter der aktuellen Geschwindigkeit sowie die Entfernung zum nächsten Hindernis. Diese Variablen allein würden schon genügen, um ein einfaches Stop-and-Go-Verhalten zu erreichen, bei dem das Fahrzeug an jedem Punkt auf seinem Weg anhält um dann ggf. wieder neu anzufahren.

Dies reichte uns allerdings nicht aus, da auch die Geschwindigkeitskomponente des Zielobjektes berücksichtigt werden sollte, wenn es sich um einen vorausfahrenden Fahrzeug handelt oder eine Geschwindigkeitsbegrenzung erreicht werden soll. Die Verarbeitung der konkreten Geschwindigkeit des Zielobjekts im Regler würde allerdings zu einer unüberschaubaren Fülle an Kombinationen von aktueller und zu erreichender Geschwindigkeit führen. Daher führten wir stattdessen den Geschwindigkeitsunterschied als Differenz von Zielgeschwindigkeit und aktueller Fahrzeuggeschwindigkeit als Variable ein.

**Wertebereiche** Die aktuelle Geschwindigkeit beträgt mindestens 0  $km/h$ . Als zulässige Streckenhöchstgeschwindigkeit legten wir U-Bahn-typische 80  $km/h$  fest, um auch leichte Überläufe erlauben zu können, vergrößerten wir den Wertebereich allerdings noch um ein gutes Stück auf 120  $km/h$ .



Figure 3: Terme der Eingangsvariable Geschwindigkeit

Den Wertebereich für die Entfernung des Ziels legten wir fest auf den Bereich von 0 – 1500 m wodurch ein genügend weiter Ausblick auf die vor dem Zug liegende Strecke gewährleistet ist.

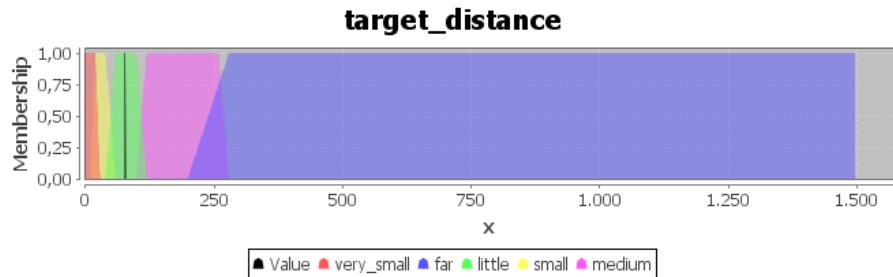


Figure 4: Terme der Eingangsvariable Zielentfernung

Analog zur aktuellen Geschwindigkeit liegt die Geschwindigkeitsdifferenz zwischen der Höchstgeschwindigkeit und dem betragsgleichen negativen Wert.

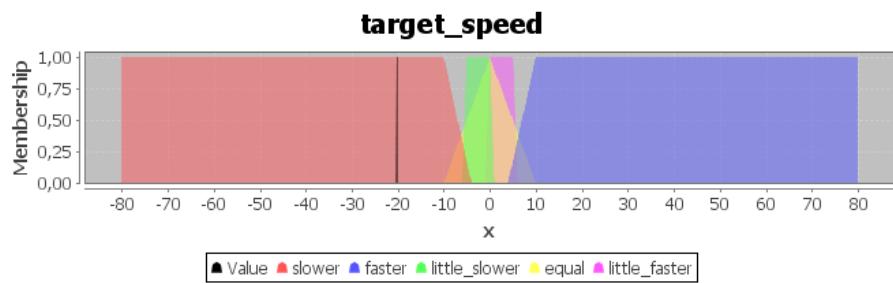


Figure 5: Terme der Eingangsvariable Geschwindigkeitsdifferenz

### 2.2.2 Ausgangsvariablen

Auf die Geschwindigkeit beziehungsweise die Beschleunigungsvorgänge eines elektrisch betriebenen Schienenfahrzeugs haben zwei Komponenten wesentlichen Einfluss: Der Fahrmotor sowie die Betriebsbremse (Druckluftbremse).

**Zum Beschleunigen** wird die Leistung des Fahrmotors erhöht, wodurch elektrische Energie in Bewegungsenergie umgesetzt wird. Um eine Geschwindigkeit beizubehalten (Nullbeschleunigung) genügt es, den Fahrmotor für eine geringe Leistungsabgabe anzusteuern, da lediglich die durch die Bewegung verlorene Reibungsverluste ausgeglichen werden müssen. Wenn der Motor in den Leerlauf schaltet, sinkt die Geschwindigkeit kontinuierlich ab.

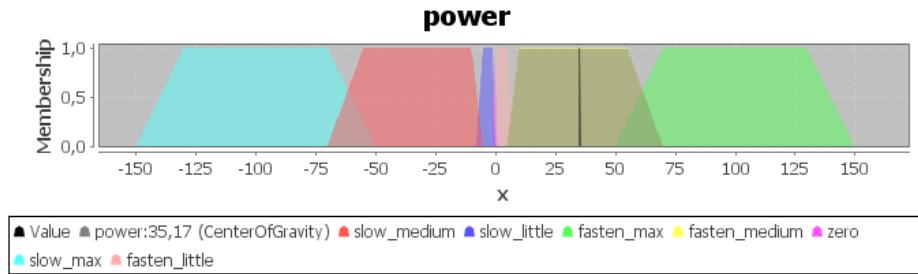


Figure 6: Terme der Ausgangsvariable Motorleistung

**Beim Bremsen** (negative Beschleunigung) wirkt einerseits die Betriebsbremse, die je nach gewünschter Bremsstärke durch druckluftbetriebene Bremszyylinder eine Bremskraft auf die Radachsen bewirkt. Im Regelbetrieb wird die zur Verfügung stehende Bremskraft nur zu einem Bruchteil beansprucht, volle Leistung wird nur bei einer Notbremsung abgerufen.

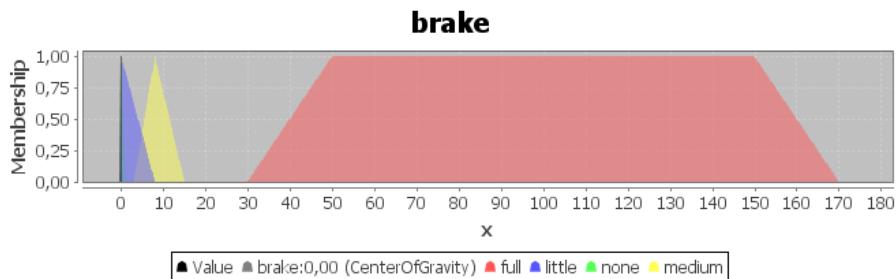


Figure 7: Terme der Ausgangsvariable Bremsstärke

Zudem kommt beim Bremsen eine Besonderheit des Elektromotors zum Tragen: Die elektrische Bremswirkung, die eintritt, wenn ein Elektromotor in den Generatorbetrieb umgeschaltet wird. Dieser Vorteil einer stufenlosen und verschleißarmen Bremswirkung wird in der Praxis bei nahezu allen elektrischen

Triebfahrzeugen eingesetzt, erreicht allerdings keine so starke Bremsbeschleunigung wir eine mechanische Bremse, weshalb diese stets zusätzlich vorhanden sein muss und bei stärkeren Bremsvorgängen additiv zugeschaltet wird. Auf in der Praxis oftmals zusätzlich verwendete weitere Bremssysteme wurde im Zuge der Vereinfachung verzichtet.

**Wertebereiche** Als Wertebereiche verwendeten wir für die Motorsteuerung den Bereich  $-100\%$  und  $+100\%$  während die Bremsstärke zwischen  $0\%$  und  $100\%$  liegt.

Damit waren die Projektvorgaben (siehe Figure 1) erfüllt.

### 3 Umsetzung

#### 3.1 Fuzzy-Toolbox

Die Fuzzy-Toolbox in MATLAB ist ein Werkzeug, um einen Regler zu erstellen und in einer graphischen Übersicht sehen zu können, ob in einer bestimmten Situation eine (bestimmte) Regel greift. Also ob ein erwarteter Output entsprechend dem Input erfolgt. In einer graphischen Umgebung können so Regeln erstellt und editiert, sowie das Greifen der Regeln überprüft werden. In den folgenden Bildern sind die einzelnen Teilbereiche der Toolbox zu sehen, mit der die Regeln für den Controller erstellt wurden.

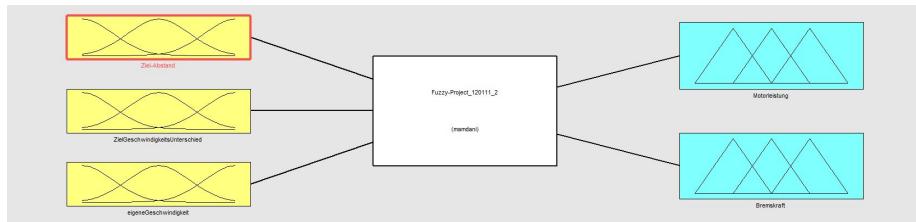


Figure 8: Hauptansicht Fuzzy-Toolbox

Die Regeln, die in MATLAB erstellt wurden und dann per Hand in das fcl-Format überführt wurden, waren mit dem Applet allerdings nicht lauffähig, sodass wir die Regeln erneut erstellten. Das zweite Mal verwendeten wir gleich das fcl-Format, um die Kompatibilität mit dem Applet gewährleisten und gleich testen zu können. Diese Regeln waren dann auch lauffähig.

#### 3.2 Applet

Das Applet setzt die theoretischen Vorgaben in Programmcode und visuell um. Dabei werden die angegebenen Ausgangsvariablen überwacht und entsprechend der Regeln auf die gegebene Situation reagiert.

Ein Zug, der zu dicht auf seinen Vordermann auffährt, bremst ab und überwacht den Abstand, bis wieder beschleunigt werden kann.

Nähert sich der Zug einer Station, so bremst er rechtzeitig ab und hält an der Bahnsteigkante an.

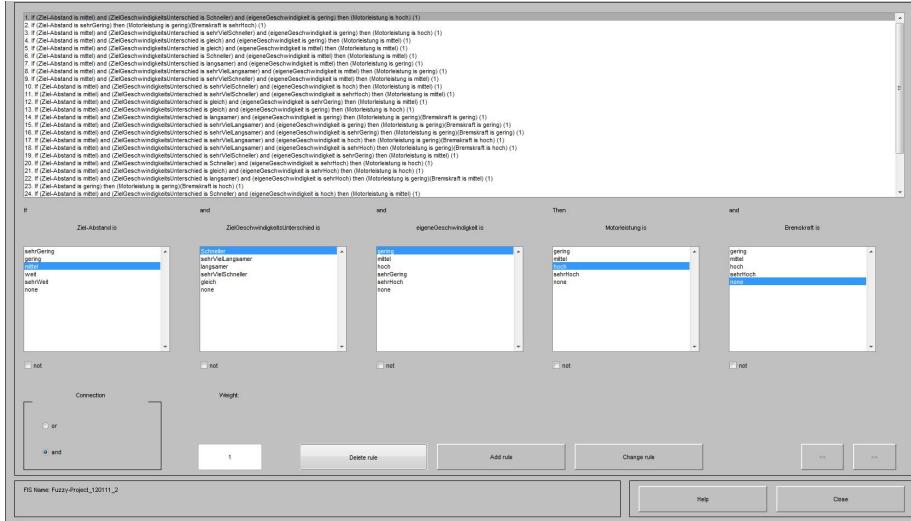


Figure 9: Regelansicht Fuzzy-Toolbox

- Zur Fuzzy-Steuerung dient die Open-Source-Bibliothek jFuzzyLogic (<http://jfuzzylogic.sourceforge.net>), ferner werden die Diagramme von der ebenso unter der GPL lizenzierten Chart-Bibliothek JFreeChart (<http://www.jfree.org/jfreechart/>).

Die graphischen Oberfläche ist mit Swing realisiert und in zwei Bereiche unterteilt. Der obere Bereich stellt graphisch die Situation dar. Der untere Teil zeigt die Werte der Eingangsvariablen und die zugehörigen Werte der Ausgangsvariablen, sowie das Szenario.

Das Szenario ist fest vorgeschrrieben. Es beinhaltet eine Station, an der der Zug anhält. Des Weiteren gibt es Abschnitte, die eine Geschwindigkeitsbegrenzung des fahrenden Zugs vorsehen. Zusätzlich ist implementiert, dass zu einem vorraufahrenden Zuge ein Sicherheitsabstand eingehalten wird.

Der Zustand eines Beispiel-Szenario kann man im folgenden Bild sehen:

Die erste Reihe sind die Eingangsvariablen als Messwerte verschiedener Sensoren. In der zweiten Reihe befinden sich die Ausgangsvariablen, die der Fuzzy-Regler anhand der vorgegebenen Regeln bestimmt. Die simulierten Züge werden alle vom Fuzzy-Regler gesteuert, die Anzeigen gehören jeweils zum rot markierten "Train No. 1". Neben den Ausgangsvariablen ist das Szenario als Liste dargestellt. Haltestellen und die Abschnitte mit Geschwindigkeitsbegrenzung sind im Programm als feste Teststrecke vorgegeben. Wenn ein Zug eine Haltestelle oder einen Langsamfahrstrecke passiert, werden die Elemente, abhängig der Nähe zum Zug, entsprechend in der Liste "Positions on track" verschoben.

Die Steuerung des Applets selbst wird in der Kopfleiste vorgenommen. Dort kann es gestartet, angehalten und zurückgesetzt werden. Außerdem kann die Simulationsgeschwindigkeit hier geregelt werden.

Innerhalb der Simulation hält der Zug entsprechend Abstand zu dem voran fahrenden Fahrzeug. Je nach Geschwindigkeitsunterschied wird entsprechend auf die Situation reagiert. Ist der vordere Zug langsamer, bremst die Bahn, ist

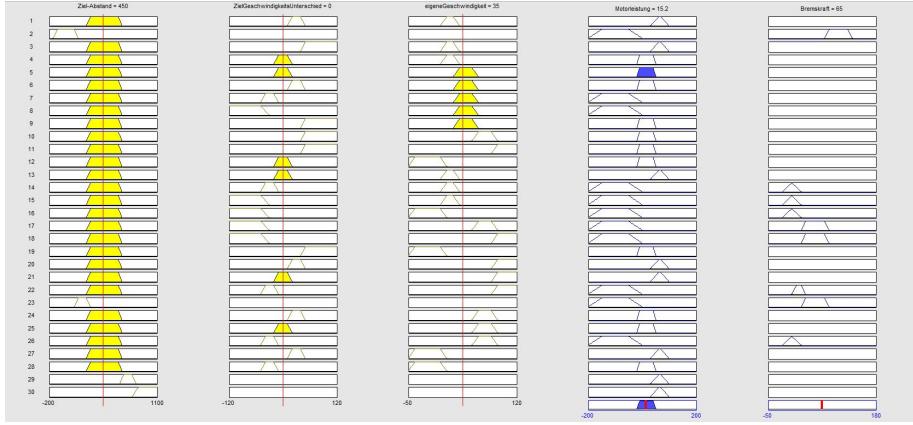


Figure 10: Regelansicht Fuzzy-Toolbox, graphisch

er schneller, kann ggf. beschleunigt werden, sofern keine Geschwindigkeitsbegrenzung in unmittelbarer Nähe vorliegt.

Diese theoretischen Vorgaben sind im Applet umgesetzt und der Regler hält sich daran. Dabei spielt es ebenfalls keine Rolle, wie viele Züge sich auf der Strecke befinden, der Regler erkennt dies und steuert alle Züge entsprechend der Situation.

Ist eine Geschwindigkeitsbegrenzung in unmittelbarer Nähe, gibt der Regler Signale an Bremse und Motor des Zugs, damit dieser entsprechend abbremsst und an der Marke der Geschwindigkeitsbegrenzung die vorgegebene Geschwindigkeit, beispielsweise 50 km/h, einhält.

Befindet sich eine Haltestelle in Reichweite des Zuges, regelt der Controller das Abbremsen und Anhalten des Zuges und nachfolgend das erneute Anfahren und das Verhalten des Zuges auf den weiteren Streckenabschnitten.

Demzufolge sind alle wichtigen Eigenschaften, die eine einfache U-Bahn simulation haben muss, erfüllt.

### 3.2.1 Dynamikberechnungen

$$E_{kin} = \frac{1}{2}mv^2 \quad (1)$$

$$\Delta E = \Delta t * \Delta E \quad (2)$$

#### Geschwindigkeit abhängig von Zeit und Energieveränderung

$$E_{kin}(t_1) = E_{kin}(t_2) + \Delta t * P_{eff} \quad (3)$$

$$v_1 = \sqrt{v_0^2 + 2 \Delta t \frac{P_{eff}}{m}} \quad (4)$$

**Fahrwiderstände** Zur Berechnung der Fahrwiderstände wurden jeweils die entsprechenden Grundformeln, aber stark vereinfacht eingesetzt.



Figure 11: Screenshot der Java-Oberfläche zum Fuzzy-Regler: Zug hält an Station A

$$W_F = W_R + W_L \quad (5)$$

Der Fahrwiderstand des Zuges besteht aus dem Rollwiderstand auf ebener und gerader Fahrbahn und dem Luftwiderstand.

### Rollwiderstand

$$W_R = c_w * m * g \quad (6)$$

mit  $c_w \approx 0.0015$

### Luftwiderstand

$$W_L = \frac{p_{Luft}}{2} * c_w * A * v^2 \quad (7)$$

mit  $A \approx 10m^2$  und  $p_{Luft} \approx 1.2$

**Streckenwiderstände** konnten komplett außer acht gelassen werden, da unser Szenario nur eine gerade, ebene, linear trassierte Strecke vorsieht, und daher keine Steigungs-, Neigungs- oder Krümmungswiderstände auftreten.

### Beschleunigungswiderstand

$$W_B = m * a \quad (8)$$

### Traktionsleistung

$$P_{eff} = \sum W * v \quad (9)$$

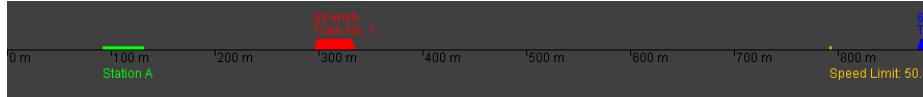


Figure 12: Ansicht der Gleisdarstellung

### 3.2.2 Fahrzeugparameter

Um die Berechnungen mit realistischen Werten durchführen zu können, erhielt unser Testfahrzeug typische Fahreigenschaften eines U-Bahn-Fahrzeugs. Insbesondere lehnten wir uns an die *Baureihe B* der U-Bahn München, die bei der U-Bahn Nürnberg als Dt2 auch im Mischbetrieb mit automatisch gesteuerten Triebfahrzeugen verkehrt, und die *Baureihe H* der U-Bahn Berlin an.

Eigenschaft	Wert
Masse	80 t
Länge	40 m
Höchstgeschwindigkeit	80 km/h
Nennleistung	780 kW
maximale Bremskraft	100 kN

Figure 13: Datenblatt unserer Testfahrzeugs

## 4 Fazit

Die Verwendung des in Matlab erzeugten Reglers erwies sich als schwieriges Unterfangen. Trotz Konvertierung in das von jFuzzyLogic verwendete FCL-Format brachte der Regler in der Java-Umgebung nicht die erwarteten Ergebnisse. Als Resultat blieb uns nichts anderes übrig, als die Regeln neu zu entwickeln und direkt in der Simulation live zu testen. Letztendlich kamen wir dabei dann sogar mit weniger Regeln gut zurecht. Um die Projektvorgaben (siehe Figure 1) jedoch einzuhalten, haben wir auch unsere ursprüngliche fis-Datei beibehalten. Die Zusammenarbeit in der Gruppe hat gut funktioniert, es ist in unseren Augen ein vernünftiges Ergebnis entstanden. Von der Fuzzy-Toolbox waren wir allerdings ein wenig enttäuscht, da sie nicht sehr übersichtlich und schlecht bedienbar ist.

Die Dokumentation in der Gruppe war interessant; da es sich um eine wissenschaftliche Arbeit handelt, haben wir uns entschlossen, sie in Vorbereitung auf kommende Aufgaben wir die Bachelorthesis in L<sup>A</sup>T<sub>E</sub>X zu schreiben, was auch gut geklappt hat.

Das Hauptproblem, auf das wir gestoßen sind, war, wie oben bereits genannt, die Tatsache, dass sich die vorgesehene Datei für den Fuzzy-Regler nicht verwenden ließ und wir mit den Regeln von vorne beginnen mussten.

Die Programmierung und Visualisierung hingegen waren nicht sehr problematisch, wenngleich nichts destotrotz auch aufwändig.